2023

The Application of Data Analytics Technologies for the Predictive Maintenance of Industrial Facilities in Internet of Things (IoT) Environments

Rieger, Thomas

http://hdl.handle.net/10026.1/20259

http://dx.doi.org/10.24382/632
University of Plymouth

UNIVERSITY OF PLYMOUTH

The Application of Data Analytics Technologies for the Predictive Maintenance of Industrial Facilities in Internet of Things (IoT) Environments

by

Thomas Rieger

A thesis submitted to University of Plymouth in partial fulfilment for the degree of

Doctor of Philosophy

School of Engineering, Computing and Mathematics

January 2023

Acknowledgements

I would like to thank my supervisory team for their guidance in the conduct of this PhD thesis. In particular, I would like to thank them for their active support, the numerous fruitful discussions and for always motivating me to carry out the work in a structured and targeted manner.

Prof Stefanie Regier	Department of Computer Science and Business Computer Science University of Applied Sciences Karlsruhe
Prof Ingo Stengel	Department of Computer Science and Business Computer Science University of Applied Sciences Karlsruhe
Prof Nathan Clarke	Professor of Cyber Security & Digital Forensics University of Plymouth

Author's declaration

At no time during the registration for the degree of Doctor of Philosophy has the author been registered for any other University award.

No part of the work submitted for this research degree forms part of any other degree either at University of Plymouth or any other establishment.

Presentations at conferences:

- Collaborative European Research Conference CERC2016, Cork Institute of Technology – Cork, Ireland, 23 - 24 September 2016, www.cerc–conference.eu
- SEIN 2016 Frankfurt University of Applied Sciences - Frankfurt am Main, July 18, 2016
- Collaborative European Research Conference CERC 2019, Darmstadt, Germany, March 29-30, 2019, https://www.cerc-conf.eu/archive/cerc2019/

This thesis has a word count of 70353.

Date: 19.08.2022

[illegible]

Thomas Rieger

Abstract

The Application of Data Analytics Technologies for the Predictive Maintenance of Industrial Facilities in Internet of Things (IoT) Environments

by Thomas Rieger

In industrial production environments, the maintenance of equipment has a decisive influence on costs and on the plannability of production capacities. In particular, unplanned failures during production times cause high costs, unplanned downtimes and possibly additional collateral damage. Predictive Maintenance starts here and tries to predict a possible failure and its cause so early that its prevention can be prepared and carried out in time. In order to be able to predict malfunctions and failures, the industrial plant with its characteristics, as well as wear and ageing processes, must be modelled. Such modelling can be done by replicating its physical properties. However, this is very complex and requires enormous expert knowledge about the plant and about wear and ageing processes of each individual component. Neural networks and machine learning make it possible to train such models using data and offer an alternative, especially when very complex and non-linear behaviour is evident.

In order for models to make predictions, as much data as possible about the condition of a plant and its environment and production planning data is needed. In Industrial Internet of Things (IIoT) environments, the amount of available data is constantly increasing. Intelligent sensors and highly interconnected production facilities produce a steady stream of data. The sheer volume of data, but also the steady stream in which data is transmitted, place high demands on the data processing systems. If a participating system wants to perform live analyses on the incoming data streams, it must be able to process the incoming data at least as fast as the continuous data stream delivers it. If this is not the case, the system falls further and further behind in processing and thus in its analyses. This also applies to Predictive Maintenance systems, especially if they use complex and computationally intensive machine learning models. If sufficiently scalable hardware resources are available, this may not be a problem at first. However, if this is not the case or if the processing takes place on decentralised units with limited hardware resources (e.g. edge devices), the runtime behaviour and resource requirements of the type of neural network used can become an important criterion.

This thesis addresses Predictive Maintenance systems in IIoT environments using neural networks and Deep Learning, where the runtime behaviour and the resource requirements are relevant. The question is whether it is possible to achieve better runtimes with similarly result quality using a new type of neural network. The focus is on reducing the complexity of the network and improving its parallelisability. Inspired by projects in which complexity was distributed to less complex neural subnetworks by upstream measures, two hypotheses presented in this thesis emerged: a) the distribution of complexity into simpler subnetworks leads to faster processing overall, despite the overhead this creates, and b) if a neural cell has a deeper internal structure, this leads to a less complex network. Within the framework of a qualitative study, an overall impression of Predictive Maintenance applications in IIoT environments using neural networks was developed. Based on the findings, a novel model layout was developed named Sliced Long Short-Term Memory Neural Network (SlicedLSTM). The SlicedLSTM implements the assumptions made in the aforementioned hypotheses in its inner model architecture.

Within the framework of a quantitative study, the runtime behaviour of the SlicedLSTM was compared with that of a reference model in the form of laboratory tests. The study uses synthetically generated data from a NASA project to predict failures of modules of aircraft gas turbines. The dataset contains 1,414 multivariate time series with 104,897 samples of test data and 160,360 samples of training data.

As a result, it could be proven for the specific application and the data used that the SlicedLSTM delivers faster processing times with similar result accuracy and thus clearly outperforms the reference model in this respect. The hypotheses about the influence of complexity in the internal structure of the neuronal cells were confirmed by the study carried out in the context of this thesis.

Table of Contents

List of Figures

List of Tables

Glossary of Terms

ABOD	Angle-Based Outlier Detection
ABSAD	Angle-Based Subspace Anomaly Detection
AI	Artificial Intelligence
ANN	Artificial Neural Network
AR	Augmented Reality
ARIMA	Autoregressive Integrated Moving Average
ARMA	Autoregressive Moving Average
ARMAX	Autoregressive Moving Average with Exogenous Inputs
AutoML	Automated Machine Learning
Bi-LSTM	Bi-directional Long Short-Term Memory Neural Network
CBM	Condition Based Maintenance
CEP	Complex Event Processing
CIFG	Coupled Input and Forget Gate LSTM
CMS	Condition Monitoring System
CPS	Cyber-physical Systems
CVA	Canonical Variety Analysis
CW-RNN	Clockwork RNN
DGLSTM	Depth-Gated LSTM
DWNN	Dynamic Wavelet Neural Networks
EAM	Enterprise Asset Management System
ED-FDS	Event-driven Fault Detection System
ELM	Extreme Learning Machines
ERP	Enterprise Resource Planning System
ES	Expert Systems
ESMMR	Exploratory Sequential Mixed Method Research
ESP	Event Stream Processing
EWPCA	Exponentially Weighted Principal Component Analysis
FDS	Fault Detection System
FFT	Fast Fourier Transformation
FL	Fuzzy Logic
FLD	Fisher Linear Discriminant
FMEA	Failure Mode and Effects Analysis
FNN	Feedforward Neural Network
GMM	Gaussian Mixture Model
grConv	Gated Recursive Convolutional Neural Network
GrLSTM	Grid Long Short-Term Memory
GRNN	General Regression Neural Network
GRU	Gated Reccurent Unit
H2M	Human to Machine Communication
HDSF	Hadoop Distributed File System
HHT	Hilbert–Huang Transform
HMI	Human Machine Interface

HMM	Hidden Markov Model
HSMM	Hidden Semi-Markov Model
I4.0	Industry 4.0
IIoT	Industrial Internet of Things
IoT	Internet of Things
IT	Information Technology
KF	Kalman Filter
KPCA	Kernel Principal Component Analysis
LCC	Life-Cycle-Cost
LDA	Linear Discriminant Analysis
LLE	Locally Linear Embedding
LOF	Local Outlier Factor
LSTM	Long Short-Term Memory
LVQ	Learning Vector Quantization
M2H	Machine to Human Communication
M2M	Machine to Machine Communication
MAAS	Manufacturing as a Service
MES	Manufacturing Execution System
ML	Machine Learning
MLP	Multilayer Perceptron Neural Network
MM	Match Matrix
MMR	Mixed MethodsResearch
MPP	Massively Parallel Processing
MTTF	Mean Time to Failure
NER	Named Entity Recognition
NF	Neuro-Fuzzy Systems
NLP	Natural Language Processing
NN	Neural Network
OLAP	Online Analytical Processing
OLTP	Online Transactional Processing
OPC	Object Linking and Embedding for Process Control
OPC UA	OPC Unified Architecture
OpenML-CC18	OpenML Curated Classification Benchmarking Suite 2018
PABSAD	Primitive Angle-Based Subspace Anomaly Detection
PCA	Static and Dynamic Principal Components Analysis
PdM	Predictive Maintenance
PF	Particle Filter
PHM	Proportional Hazards Modelling
PLC	Programmable Logic Controller (also called SPS)
PLOF	Primitive Local Outlier Factor
PLS	Partial Least Square
PM	Preventive Maintenance
PNN	Polynomial Neural Networks
QDA	Quadratic Discriminant Analysis
PVA	Peak Validation Accuracy
QUAL+QUAN	Qualitative Dominant Mixed Methods Research

QUAN+QUAL	Quantitative Dominant Mixed Methods Research
RBF	Radial Basis Function
RCFA	Root Cause Failure Analysis
RCM	Reliability Centered Maintenance
RDBMS	Relational Database Management Systems
RN	Recurrent Networks
RNN	Recurrent Neural Networks
RPCA	Recursive Principal Component Analysis
RUL	Remaining Useful Lifetime
RWNN	Recurrent Wavelet Neural Network
SCADA	Supervisory Control and Data Acquisition System
SCRN	Structurally Constrained Recurrent Network
SlicedLSTM	Sliced Long Short-Term Memory Neural Network
SIS	Safety Instrumented System
SOFM	Self-organizing Feature Maps
SOM	Self-organizing Maps
SPS	Programmable Logic Controller (also called PLC)
SRN	Simple Reccurent Network
SVDD	Support Vector Data Description
SVM	Support Vector Machine
SWLOF	Sliding Window Local Outlier Factor
SWPCA	Sliding Window Principal Component Analysis
TCO	Total Cost of Ownership
TDNN	Time Delay Neural Networks
TK-RNN	Temporal-Kernel Recurrent Neural Network
TPM	Total Productive Maintenance
TS-LSTM	Two-Stream Long Short-Term Memory
TSP	Time Series Predictions
TSPM	Time Series Prediction Models
TTF	Time to Failure
TW-CNN	Two-way Convolutional Neural Netzworks
WEA	Wind turbines
WPE	Wavelet Packet Energies

1 Introduction

This thesis is about the use of Deep Learning with neural networks for Predictive Maintenance of industrial plants in IIoT environments. The focus is on time-critical applications and the question of how to improve runtime behaviour to reduce latency and deliver immediate results. The processing of complex data streams, such as complex sensor data from IIoT environments, represent the input data. In this thesis, the novel model architecture Sliced Long Short-Term Memory Neural Network (SlicedLSTM) is developed and compared to the established model in this application domain. Which models are currently the reference in Predictive Maintenance (PdM) applications for IIoT environments will be determined in a qualitative study. The improvement in runtime achieved by the novel model approach SlicedLSTM is to be investigated in a quantitative study using a laboratory test.

1.1 Motivation

Software systems for the analysis of large amounts of data are applied in more and more areas. The subject of Predictive Maintenance (PdM) using data analytics methods is present in industrial manufacturing processes, especially in combination with Industrial Internet of Things (IIoT) environments. Such applications are often based on established analytical techniques using distributed storage and processing frameworks. Analytical applications often run on purely batch-oriented frameworks and can suffer from hours of delays (Marz, et al., 2015). If an online system receives continuous data streams of large volumes, the runtime behaviour becomes an important factor. An application must then be able to process data at least as fast as it is continuously received in the data stream. If an application is not able to do this, it will continuously fall behind in the temporal processing of its data.

Neural networks and Deep Learning are being used more and more in the field of PdM, which will also be shown in the qualitative study conducted within this thesis. An important reason for this is that neural networks are able to deal with unknowns and to learn themselves. When using mathematical models to describe the degradation of a plant, all physical conditions in regard to the behaviour of wear and tear must be known and described in the model. Due to increasingly complex plants in the industrial environment, but especially due to the increasing connectivity and more and more intelligent sensors, this already difficult task will become increasingly difficult to manage in the future. In addition, very deep expert knowledge is required for this. Neural networks can offer advantages here, but depending on their complexity, they also place higher demands on the required computing capacity. If sufficient resources (network, hardware) are available to the application, this may not be a problem at first. If this is not the case, fast and resource-efficient neural network models are to be preferred. Depending on the application case, a lower result quality of the neural network could be acceptable in return for better runtime behaviour. Particularly in strongly connected IIoT environments with decentralised and autonomous participants, there is a need to be able to carry out analyses locally on decentralised units (edge computing). Such decentralised units usually have limited hardware resources.

In order to further promote the use of neural networks in the application area of PdM and for the processing of time-critical data streams on systems with limited hardware resources, the runtime behaviour of neural networks will first be investigated in this thesis. The focus is on the model type that is most frequently used for PdM applications. Based on this, a new type of model architecture will be developed, which focuses on shorter runtimes with similar quality of the results. A model comparison with the standard model will demonstrate the runtime improvements achieved with the new model approach.

1.2 Research Question

To develop adequate research questions, formulate clear objectives or questions which can be clearly understood and answered within the scope of the work (Dodgson, 2020). In section 1.1 it was explained that the motivation of this thesis is to find a novel model approach for a neural network that outperforms the current standard model in terms of runtime behaviour. To this end, it is first necessary to find out which model represents the current standard (RQ1). The question must be clarified as to how the runtime behaviour of existing models, and in particular that of the standard model, is presented and whether there is actually a need for improvement in the application area of PdM applications in IIoT environments (RQ3). To this end, the circumstances and framework conditions of this area of application must be examined and taken into account when clarifying the question of a need for improvement (RQ2). Finally, it must be clarified what a new approach could look like and whether the desired improvement has actually been achieved (RQ4).

Table 1 lists the research questions. The research questions RQ1 to RQ3 are to be answered by means of qualitative research. The answers to research questions RQ1 to RQ3 can be found in section 5.5. Research question RQ4 will be answered in chapter 6, especially in section 6.4.

RQ1:	Which neural network is most commonly used for Predictive Maintenance (PdM) applications and thus forms the current standard?
RQ2:	What are the characteristics and requirements of modern data stream processing infrastructures and software systems for predictive systems?
RQ3:	How well do the currently used neural networks meet runtime requirements for time-critical PdM applications?
RQ4:	How can a new type of neural network be constructed so that it achieves measurably better runtime behaviour with comparable quality of the results achieved compared to the current standard model?

Table 1 - Research questions

1.3 Content of this Thesis

This thesis starts with a look at the background of the topic. The importance of maintenance and corresponding maintenance strategies in the industrial environment and especially Predictive Maintenance in industrial IoT environments are explained. Reasons for the necessity of time-critical processing and the special features of data streams are also outlined. Details on Deep Learning in the IIoT environment and on Long Short-Term Memory Neural Network (LSTM) conclude the background chapter.

Subsequently, related work in the subject area is discussed. The chapter on related work is divided into the main topics of Predictive Maintenance strategies, performance in predictive systems, fast predictions using Deep Learning and optimisation approaches for LSTM.

In this thesis, the novel model architecture Sliced Long Short-Term Memory Neural Network (SlicedLSTM) will be developed and compared with the established models in this application area. Which models are currently the reference in PdM applications for IIoT environments will be determined in the form of a qualitative study. The improvement in runtime achieved by the novel SlicedLSTM model approach will then be investigated in a subsequent quantitative study using a laboratory test. Chapter 4 describes the basic research design of this thesis. It explains why the procedure of qualitative study and subsequent quantitative study was chosen for this thesis and how the study design is structured.

In the following chapters, the preparation, implementation and evaluation of the qualitative and quantitative study are described in detail. The qualitative study takes the form of explorative expert interviews with a small group of eight selected experts. The aim of the qualitative study is to develop a general picture of the topic area, to show possibilities, to give insights and to find ideas.

Before conducting the quantitative study, a detailed theoretical consideration of the newly developed model architecture of the SlicedLSTM is given in chapter 6. The quantitative study is then conducted in the form of laboratory tests. The study uses synthetically generated data from a NASA project to predict failures of modules of aircraft gas turbines. The dataset contains 1,414 multivariate time series with 104,897 samples of test data and 160,360 samples of training data.

The result of the study proves that the new model architecture SlicedLSTM clearly outperforms the reference model Standard LSTM in the area of runtime. The study thus provides proof of the theoretical assumptions under the defined conditions. With the SlicedLSTM, a novel model architecture of neural cells is created for application areas in which runtime behaviour and processing time of the neural network have a higher priority than its maximum result accuracy. This thesis concludes with a summary and evaluation of the results obtained, as well as an outlook on subsequent research and application possibilities of the SlicedLSTM.

2 Background

This chapter presents the relevant background information for this thesis. It starts with the general topic of maintenance and an explanation of the established maintenance strategies. The different maintenance strategies are explained by using an example from the paper production. After that, Predictive Maintenance in general and its role in modern industrial Internet of Things environments (IIoT) will be discussed. This is followed by a section regarding time critical processing, why it is needed and what are the current techniques. The significance of the complexity of data streams in the context of this thesis is then explained. This chapter closes with the topic of using neural networks for PdM applications.

2.1 Maintenance Strategies

In literature, four basic strategies to organize maintenance work are usually mentioned: Corrective Maintenance, Preventive Maintenance, Predictive Maintenance and Prescriptive Maintenance (Frederiksson, et al., 2012). The enormous added value of the correct maintenance strategy can be demonstrated by an example from paper production. A Fourdrinier machine used for paper production consists of many sub-machines which are closely connected to one another in a production process. Figure 1 shows this schematically (Egmason, 2010).

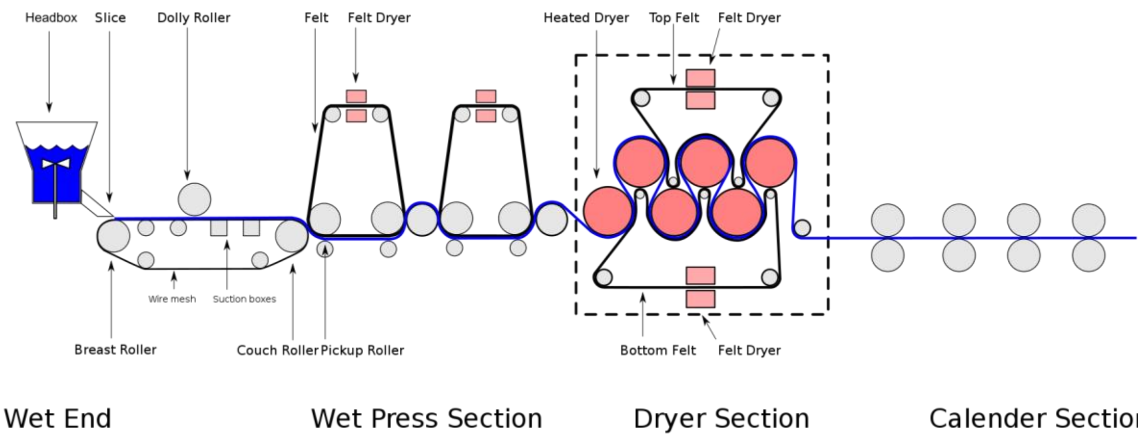


Figure 1 – Paper machine (Fourdrinier) as an application example (Egmason, 2010)

Modern paper machines are often 100 to 200 meters long and about 15 meters high. They usually consist of many subsystems with typically more than a hundred guide rolls for wires and felts as well as for the paper web and a large number of drying cylinders (VDW, 2022). The subsystems are strung together in a fixed process chain in the paper machine. The failure of a component of a subsystem in this process chain stops the entire paper machine and can lead to capital consequential errors and damage in the other subsystems. In addition, wet material (waste paper or pulp dissolved in water (VDP, 2022)) is processed in a paper machine from source material. If there is a fault in one of the machine's subsystems and this interrupts processing in the entire process chain, the wet material in the subsystems can harden. Resulting consequential damage and the expense of cleaning and restoring the functionality of the entire system can cause great expense. The example of the paper machine was chosen here because it very impressively illustrates the effects of a failure of a machine or a subsystem in a process chain of industrial production.

If Corrective Maintenance is the only maintenance strategy, it will only react in case of an error. Maintenance personnel, i.e. expensive specialists, must always be available. Spare parts must be stored, especially if they are very specific and difficult to procure parts. In the above example of a Fourdrinier machine, in the event of a fault, the coil (this is what the produced paper roll is called) produced is incomplete and therefore scrap. In addition, unprocessed material that is still in the machine could overflow and harden. Other parts of the system may be damaged by the fault, resulting in follow-up costs and further downtime due to subsequent repairs and cleaning.

With Preventive Maintenance, unscheduled failures are intended to be avoided through regular maintenance. For this, knowledge of the lifetime of the individual parts of the system is necessary. Maintenance is planned based on this knowledge to provide sufficient time for personnel planning and disposition of required individual parts. However, since Preventive Maintenance is based on assumptions and experience, unplanned failures can nevertheless occur. Furthermore, regularly unnecessary parts are exchanged. In the paper machine example, maintenance intervals are preferably planned for non-productive times. The entire system can be properly shut down during this time. Production losses and consequential damage can be avoided. Scheduling minimizes downtime to the time required for maintenance.

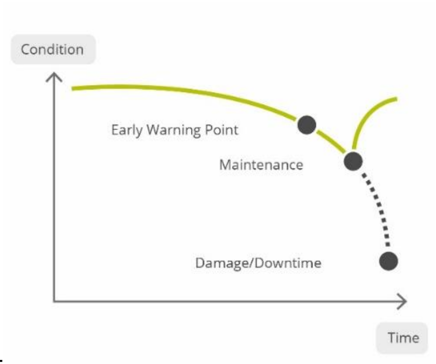
Predictive Maintenance attempts to predict failures. The predictions are based on accumulated knowledge and the current conditions of the machine. The aim is to avoid breakdowns by a timely prediction and maximizing the service life at the same time. The advantages mentioned in Preventive Maintenance also apply here. To this, additional potential savings are achieved by maximizing service life. In the example of the Fourdrinier machine, Predictive Maintenance can minimise failures and thus also the enormous risk of consequential damage.

Prescriptive Maintenance combines methods of Descriptive Analytics and Predictive Analytics. Descriptive Analytics tries to describe why certain things have happened. The goal is not only to correct errors, but also to avoid them. For example, design faults or weak points in a system can be permanently eliminated through targeted measures.

2.2 Predictive Maintenance

In Standard EN 13306:2001 PdM is defined as a part of the Condition Based Maintenance (CBM) system. Relying on data collected by a CBM system, data analytics are added to predict important indicators. Those indicators are dedicated to productivity aspects, like Remaining Useful Lifetime (RUL), Mean Time to Failure (MTTF) or on-time delivery, as a consequence of a predictable operational reliability. Apart from this, product quality, safe working environments, system safety, increase of maintenance effectiveness, optimizing maintenance effort, reduction of inspection and repair-induced failures are also mentioned as important indicators (Heng, et al., 2009). Overall, the primary objective is the reduction of life cycle costs (Schmidt, et al., 2015).

With PdM, maintenance costs have the potential to be reduced. Scheduled and preventive maintenance is much cheaper than unscheduled and reactive repair. The availability of production capacities becomes more reliable (eoda, 2014). Because an expensive and risky reactive maintenance strategy is avoidable by applying preventive and predictive strategies, more and more companies improve their maintenance organisation by implementing PdM systems. In addition to the usual targets like Remaining Useful Life (RUL) and Mean Time to Failure (MTTF), also strategic and economic indicators have been introduced like Total Productive Maintenance (TPM) (Fredriksson, et al., 2012).



4

Figure 2 - Finding the Early Warning Point (Hagenberg, 2017)

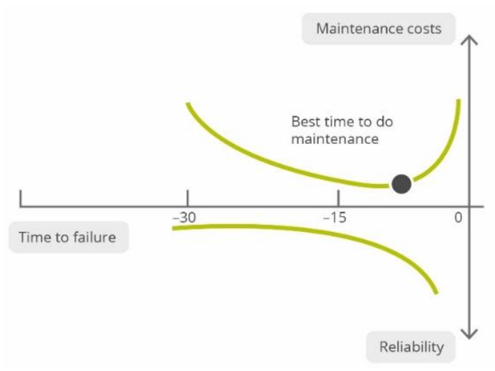


Figure 3 – Best time to do maintenance, taken from (Peng, et al., 2010)

The main purpose of PdM is to avoid unscheduled outages of machines and plants by anticipating a failure before its possible occurrence. In principle, maintenance should be carried out at the latest possible time, in order to avoid breakdowns and maximizing the service life at the same time. Scheduled service during normal working hours should be the result. Figure 2 illustrates this. According to a paper published by the Software Competence Center Hagenberg the *Early Warning Point* is the lead time necessary to plan the maintenance, including needed specialized personnel and material procurement (Hagenberg, 2017). As illustrated in Figure 3, the maintenance costs are also taken into account as they were expected to grow rapidly while degradation (Peng, et al., 2010).

In literature, PdM is often assigned to the topic of Reliability Centered Maintenance (RCM) (see Figure 4) or as a subcategory of Preventive Maintenance (PM) and Condition Based Maintenance (CBM) (see Figure 5).

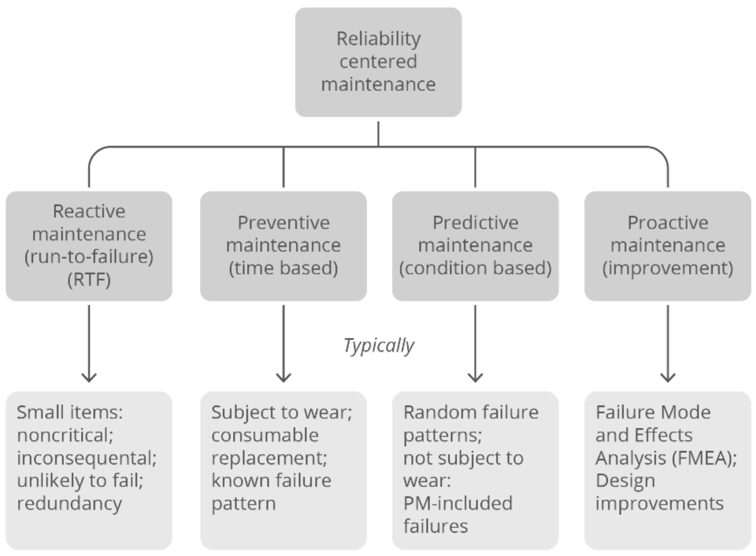

Reliability centered maintenance
Reactive maintenance (run-to-failure) (RTF)
Preventive maintenance (time based)
Predictive maintenance (condition based)
Proactive maintenance (improvement)
Typically
Small items: noncritical; inconsequental; unlikely to fail; redundancy
Subject to wear; consumable replacement; known failure pattern
Random failure patterns; not subject to wear: PM-included failures
Failure Mode and Effects Analysis (FMEA); Design improvements

Figure 4 -Common applications of maintenance strategies for RCM (NASA, 2008a)

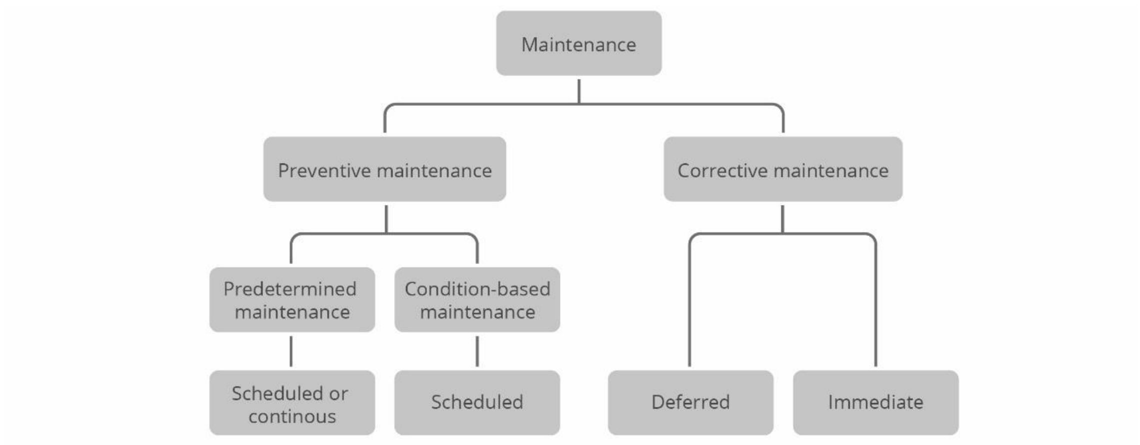


Figure 5 - Maintenance overview (Straub, 2012)

Usually, PdM systems assume that the failure behaviour is basically predictable. The prerequisite for this, however, would be that ageing and wear processes are deterministic and predominantly linear. However, this does not correspond to their behaviour in reality. Ageing and wear are often not deterministic and linear. In addition, other environmental influences play a role, such as ambient conditions, process deviations, interactions with other systems, e.g. (Kothamasu, et al., 2006).

2.3 Predictive Maintenance in IIoT

From IT perspective, industrial automation systems are about managing systems and processes. Communication is focused on the process-, production-, and product data. According to Marz et. al. (2015), storing raw data and especially sensor data for later analysis has not been a priority. Reasons for this include the classic structure of traditional automation environments and the strict separation into system levels that prevails there. Furthermore, there was previously no need for raw data at the higher system levels. It is only through data analysis applications that raw data is required at the higher system levels. (Marz, et al., 2015).

In traditional industrial environments, following the automation pyramid (Figure 6), sensors usually send their data over a fieldbus to a controller. According to Åkerman (2018), a hierarchical structure with monolithic systems and non-standardized communication protocols is one of the usual characteristics of such environments (Åkerman, 2018). The full and unprocessed sensor data, hereinafter referred to as raw data, occur only on the field level and the process level. From the perspective of raw data this traditional approach usually does not provide any “transparency” through the layers of the automation pyramid (Bauer, et al., 2017). This creates a barrier for the raw data at field and process level, blocking their path to the upper levels of the automation pyramid.

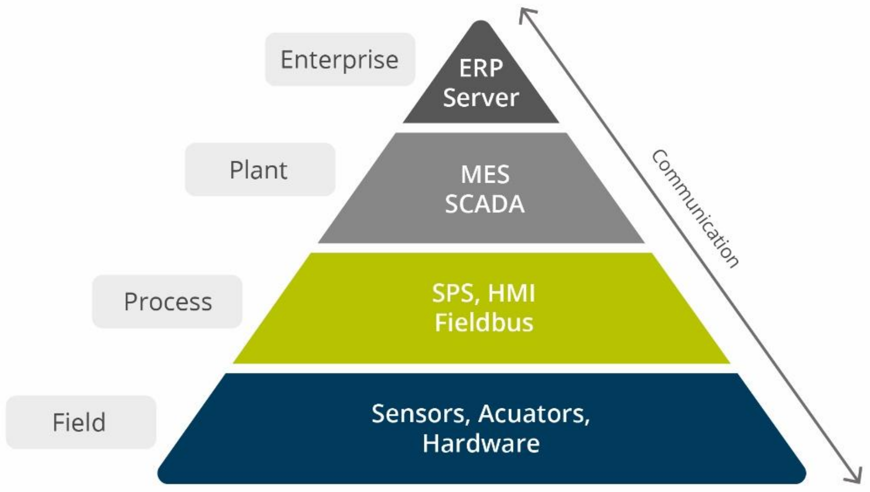


Figure 6 - The traditional Automation Pyramid following (Bauer, et al., 2017)

The term fieldbus used in Figure 6 at the process level stands for the part of the industrial computer networks through which the distributed units of the real-time controllers communicate with each other. SPS (also called PLC) stands for programmable logic controller and is a computer-based unit that is used to control or regulate a machine or system. A SPS virtually replaces hard-wired control logic with its programmable memory modules. HMI stands for Human Machine Interface. The term refers to digital displays that are used in the context of controlling and monitoring machines. The term MES is used to describe Manufacturing Execution Systems. These are production control systems for tracking and documenting the conversion of raw materials into finished products. Supervisory Control and Data Acquisition (SCADA) systems are used to control and acquire data from equipment in a standardised exchange format. The Enterprise-Resource-Planning (ERP) system contains the entire data and process chains of a company in all areas, such as finance, human resources, manufacturing, supply chain, services, procurement and more (SAP.com, 2022).

Systems are more and more equipped with sensors delivering an ever-growing number of measurements of different types. Measurements are positions, temperatures, pressures, vibrations, flow rates, e.g. Sensors typically generate a continuous stream of data (Krawczyk, et al., 2015). The high volume and the high complexity of data put massive demands on existing data processing techniques, which will be described in section 2.5 (Zhang, et al., 2017a).

In an IIoT environment, and in particular with Industry 4.0 (I4.0), intelligent devices are connected directly to the Internet and make their raw data available there. The collected (raw) data is for example sent to a central service, which can be a cloud service. As explained at the beginning of this section, the rigid hierarchy of the traditional automation pyramid blocks the path of raw data to the higher levels. However, applications for data analysis, and thus also Predictive Maintenance systems, are located at the higher levels and ideally need the raw data (Corbett, 2022). But breaking up the existing architecture of the automation pyramid and thus making raw data available at the higher system levels would mean having to change the established system architecture of an entire industrial company, which has usually grown over decades. The technological and economic effort required for this would be enormous and would significantly inhibit the spread of IIoT and I4.0. For this reason, modern IIoT environments typically take the path of connecting the components of the field level directly to the higher system levels via an additional network connection (Corbett, 2022). In that sense, sensor data bypasses the traditional automation pyramid.

Continuous streams of data with high volume and high velocity are therefore available to high level IT-Systems. Connecting these devices in IIoT environments makes it possible to store and analyse this data. As a result, the use of data analytics becomes possible at the higher levels of the automation pyramid. For maintenance purposes, it is recommended to take into account all data of the machine and its environment that are available and contribute to the subject of maintenance (hereinafter also referred to as relevant maintenance data). This comprises condition-monitoring data, feedback from routine inspections, failure data, maintenance resources, work orders, overhaul and refurbishment plans, e.g. (Zhang, et al., 2014). In addition to sensors, relevant maintenance data also comes from systems like Enterprise Asset Management (EAM) systems, Enterprise Resource Planning (ERP) systems, Condition monitoring Systems (CMS), Supervisory Control and Data Acquisition (SCADA) systems or Safety Instrumented Systems (SIS) (Zhang, 2016c).

EAM systems contain all asset management data. In addition to investment and cost planning, as well as life cycle and risk management of fixed assets, an EAM can also provide maintenance management data according to ISO 55000 (ISO 55000, 2022). The ERP system provides versatile data that are relevant for the maintenance of plants, such as data from production planning or the procurement process. A CMS system is used for the permanent monitoring of plants. By measuring physical variables, the condition of plants and machines is continuously recorded and analysed. If CMS systems are already installed, they form a basis on which Predictive Maintenance systems can be built. This is explained in more detail in section 5.5 of this thesis. While CMS systems only monitor, a SCADA system also serves to control and collect data from plants and thus provides additional data that can also be relevant for Predictive Maintenance (Ahmed, et al., 2008). SIS systems focus exclusively on safety-critical processes and plants (IEC 61511-1, 2016). If Predictive Maintenance is carried out for such plants, the data from the SIS system is also relevant.

2.4 Time critical Processing

The process of collecting data from production, interpreting it and processing it to predict possible failures is very complex and places high demands on the underlying IT infrastructure. If this process is not fast enough, this leads to short-term planning and immediate actions in maintenance and repair. In IIoT and I4.0 systems the representation of data used by systems like CMB or PdM is often defined in the form of a virtual representation or virtual clone/twin of physical components of the factory. This virtual representation is running in parallel to the physical factory. A digital twin is the image of its physical 'asset' in the real factory and mimics its functionality, behaviour and communication. Especially in the context of Industry 4.0, digital twins in combination with the so-called management shell are a core concept for mapping real systems into the digital world. The digital twin allows the simulation, control and improvement of its real twin in a virtual environment (Singh, et al., 2021). The drivers of the virtual representation are continuous streams of raw data of the factory systems and sensors. If data is not transferred and processed in a reasonable time, the virtual representation will lag behind the physical process. With respect to a PdM system this could lead to unintentional reactive maintenance because of insufficient lead time to plan the maintenance tasks (Bauer, et al., 2017).

In order to meet the high time requirements, various procedures and strategies are known. Frequently, dimension reduction techniques that reduce the complexity of data in a preliminary step are used. In machine learning, dimensionality is the number of attributes, features and input variables in the data. Dimensionality reduction is a method of reducing the number of these attributes, features and input variables in a data set. However, in order not to distort the meaning of the data, it is important that the reduction method preserves as much context as possible in the original data. The reduced complexity makes processing easier and faster (Lee, et al., 2014). Various strategies are used to reduce complexity. Thus, certain approaches, such as clustering or sliding windows, aim at reducing data to be processed by different forms of partitioning.

Another approach tries to evaluate if the failure of a certain component of a machine has a critical impact. Complex predictions are then only performed for those critical components. Figure 7 shows such an approach. The downtime caused by the failure of a component is shown as a function of the failure frequency. The aim is to define the most critical components in terms of impact and cost. According to Lee et al. (2014) the use of PdM is only recommended for components with low failure frequency and high downtime. The components that have a high failure frequency and cause high downtime must be eliminated "by design". All components that cause low downtime should be treated with standard approaches of Reactive and Preventive Maintenance. The respective definition of the limits as well as the classification of each component must be done by an expert, according to the circumstances and the desired objectives (Lee, et al., 2014). This has to be done for each application case as every physical machine is different. No generally accepted definition is possible. With this approach, the amount of data to be processed and the effort required to implement predictive analytics applications can be significantly reduced (Lee, et al., 2014). In addition to concentrating only on critical components, there are different approaches in which the prediction analytics is only performed in case of certain events. Events are generated in this case for defined constellations in the continuous stream of sensor signals and represent a relevant event for predictive analytics applications. What is considered relevant is manually defined by Data Scientist in so-called event schemas for sensor data changes. (Ait-Alla, et al., 2015).

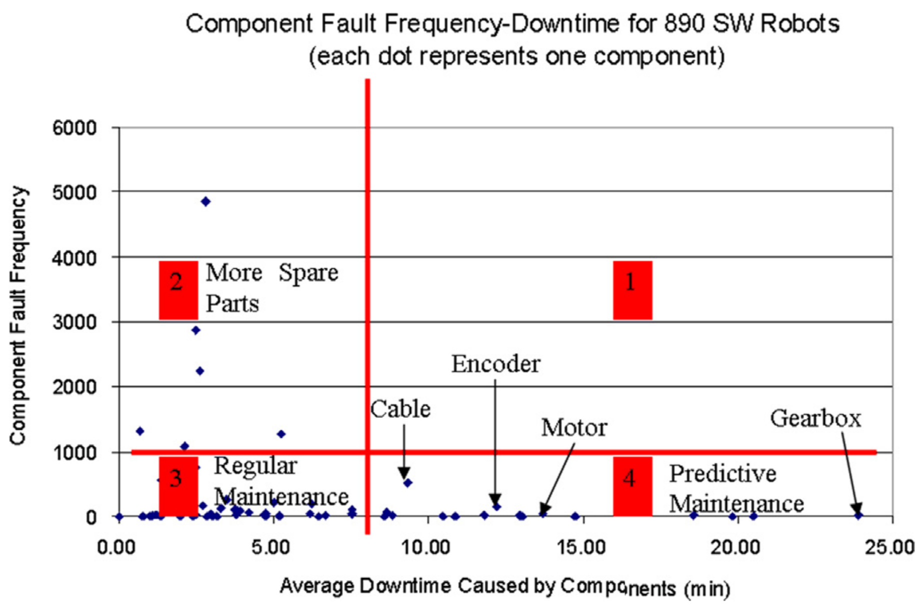


Figure 7 - Four quadrant chart for identifying critical components, taken from (Lee, et al., 2014)

In order to be able to process large amounts of data with complex and sophisticated algorithms fast, attempts are often made to divide a processing step into many small processing steps. This segmentation is intended to ensure that the individual steps can be processed in parallel on several computers / nodes (scale out). One approach is the vertical division of the monolithic processing sequence into individual steps in the form of a pipeline. These individual processing steps must be very simple and independent. The connection between the individual processing steps is performed exclusively via data (intermediate data), whereby the input data of a processing step corresponding to the output data of the previous processing step. When using pipelines, data must be parallelisable and distributable. The processing logic (algorithms) must be divisible into independent individual steps (Orenstein, et al., 2015).

2.5 Complexity of Data Streams

Performing analytics on data streams demands special considerations on the processing of the analytic algorithms. The types of complexities of data streams that could be potentially problematic in the context of this thesis are as follows:

- High volume (both the number of features and the total amount of data), high velocity and high variety of data (see section 2.5.1)
- High complexity caused by
    - High dimensionality (see section 2.5.2)
    - Non-stationary or evolving data (see section 2.5.3)
    - Non-linear data (see section 2.5.4)

This section begins with a general consideration of the different types of complexity and their relevance to this thesis. In the following sections 2.5.1 to 2.5.4, the relevant types are explained in detail.

In literature, complexity is often characterized by the three Vs, high volume, velocity and variety. High volume refers to the total amount of data occurring over time. Velocity means the speed of processing. If data streams are present, velocity also refers to the frequency with which data arrives for processing. The meaning of variety in this context refers to manifold data sources and data contexts (Montgomery, 2014). Beyer et al. (2011) adds a “c” to denote complexity in the meaning of variety, high dimensionality, non-stationary and non-linearity data (Beyer, et al., 2011). Zhang (2016c) also subsumes dimensionality, non-stationary and nonlinearity under the term of complexity. In addition, Zhang (2016c) assigns those characteristics in particular to maintenance data (Zhang, 2016c). In his paper *Massive Data Analysis: Tasks, Tools, Applications, and Challenges* Pusala et al. (2016) extends the characterizing features of data streams to the 5V’s volume, variety, velocity, variability and veracity. With variability primary the effect of concept drifts is meant. Veracity addresses the quality of the processed data by means of relevance for analytics, integrity and right balance (Pusala, et al., 2016). Each single complexity has an impact on the applicability of the algorithms as well as the underlying IT-infrastructure and must therefore be taken into account when designing data analytics systems. How high the impact of one of the characteristics is to a system, cannot be answered generally. This strongly depends on the specific application case and can vary widely between individual applications (Zhang, 2016c).

Depending on the use case, immediate answers can be expected from the algorithms. In his review paper about data stream classification and data analytics Krawczyk et al. (2015) notes that with algorithms not adapted to process streaming data of huge size, data processing is not possible even with new and developed computer technologies. They also add that it’s not only the size but the increasing complexity of data streams which demands new types of algorithms, as well as distributed and parallel computing methods. It is also mentioned that adaptive models, able to improve dynamically on evolving data streams, are indispensable (Krawczyk, et al., 2015).

2.5.1 Volume, velocity and variety

In IIoT environments the number of connected devices and intelligent sensors is increasing tremendously. This leads to (raw) data available of high volume. Data streams evolve when data is generated constantly at a high rate, e.g. sensors measuring constantly and with high frequency. In IIoT environments, diverse sources of data are typically connected. The structure of data can be highly heterogeneous and vary between structured, unstructured and poly-structured data. Examples of the latter most are plain text, log files, images and image streams from monitoring systems, audio and video files and much more (Zhang, 2016c).

Data streams have become an inherent feature in industrial environments. High velocity and time-critical processing leads to the fact that algorithms performing analytics must have low computing complexity, low memory usage and a transient nature (Zhang, 2016c). Techniques for processing data streams are already established and developing more and more. Also there are many algorithms already optimised to the demands of stream processing (Zhang, 2016c).

2.5.2 High dimensionality

High dimensionality is one of the main complexities of data analytic systems in general, often referred to as the curse of dimensionality (Verleysen, et al., 2005). It becomes even more relevant when processing data streams (Chen, et al., 2014). To cope with data streams, the complexity and therefore the dimensionality of data should be as low as possible (Zhang, 2016c).

Each feature of a dataset represents a dimension. For sufficient training of the model, the data of a data set must represent the relationships of all relevant combinations of all features to each other. Only in this way can the model sufficiently train the possible combinations. Adding another feature to a dataset creates another possible combination of all existing features with the newly added one. Adding another dimension to a dataset thus exponentially increases the amount of training and test data required to adequately train the model. Accordingly, the processing effort and processing time for the model also increase sharply (Karanam, 2021).

If a data set contains more features than the information contained in the data requires, there is a risk of massive overfitting of the model to be trained. The result quality of the model decreases rapidly despite the high training effort. As already explained, the amount of data and features is constantly increasing, especially in IIoT environments. The approach of collecting everything that is available can become a problem for the usability of machine learning. Dimensionality reduction methods are therefore gaining in importance (Yiu, 2019).

In this thesis, a newly developed model architecture is compared with a reference model. All comparisons are made with the same data. Adverse effects due to the dimensionality of the data therefore affect both models in the same way and thus have no influence on the relative model comparison. The complexity caused by high dimensionality is therefore not considered relevant in the context of this thesis and is therefore not considered further.

2.5.3 Non-stationary or evolving data

In streaming environments with non-stationary (or evolving) data, it is imperative that algorithms must be able to detect concept drifts and recover from them. Concept drifts, also including concept shifts and recurrence, are caused by changes in data generating entities over time. For example, in case of a real person, this could be the changing interest and personal growth over time. In case of IIoT, concept drifts can be a result of direct determinants like ageing processes of equipment or materials as well as indirect determinants like ambient temperature or humidity. Autonomous and self-organizing industrial IoT-Environments are also causes of concept-drifts (Shaker, 2016).

Detecting concept drifts is a major challenge, since data in streams can usually be scanned only once or only a very few times (Shaker, 2016). To allow adequate reactions to concept drifts different techniques are known. For example, with the so called weighting, recent data is given more impact on the forecast than older data (Zhang, 2016c). As a result, changed behaviour is learned faster by a neural network. But this leads to the fact that recent data is considered more relevant than older data. For this reason, fading functions were developed, putting an adjustable weighting factor to newer and older data when updating the model. Additionally sliding window techniques are also quite common, always exchanging the oldest sample by the newest one (Zhang, 2016c), (Jeng, 2010). The ability to recover autonomously from concept deviations becomes even more important when analyses are done online with an adaptive modelling approach on infinite data streams (Shaker, et al., 2013).

In the further course of this thesis, the runtime improvements of the new model approach developed in this thesis are demonstrated by means of a laboratory test. Data used for this test is anonymized. The reason for using anonymised data and the implications for this thesis are explained on page 59 of section 4.4.3. Due to anonymisation, it is not known whether concept drifts are present in the test data. As in the case of high dimensionality, possible effects due to concept drifts affect both models compared in this thesis in the same way and thus have no influence on the relative model comparison. Therefore, no action is taken in this thesis to respond to concept drifts.

2.5.4 Non-linear data

The relationship between the values of data (features) can be highly nonlinear in practice. Non-linear data induces additional complexity and extra efforts in the definition of an appropriate model (Göb, 2014). In practice, attempts are often made to avoid non-linearity by approximating relationships between individual features. This so called linear approximation is only feasible in less complex situations and not appropriate if data is complex (Alippi, et al., 2014), (Zhang, 2016c). In complex situations, where non-linear data are a matter of fact, more sophisticated approaches are necessary to avoid underfitting problems. Depending on data available and the chosen training set and in addition with the complexity of non-linear data, underfitting and overfitting problems play an even greater role.

2.6 Deep Learning in IIoT

Neural networks and Deep Learning play a dominant role in the field of Predictive Maintenance (PdM) of industrial plants. Sensors provide permanent data streams over long periods of time. Neural networks are able to handle data streams comprising dependencies in data over time and store relevant information over time. Recurrent Neural Networks (RNN) have this ability, but only for short term dependencies (Cho, et al., 2014a), (Cho, et al., 2014b), (Ciresan, et al., 2012a). Long Short-Term Memory (LSTM), as a variant of RNN, is able to handle long term dependencies without the memory requirement growing steadily with time (Hochreiter, et al., 1997), especially if they learn to forget (Gers, et al., 1999).

This section starts with a short introduction into Deep Learning (DL) and Artificial Neural Networks (ANN) applied in IIoT environments. A classification of different DL methods mentioned for the use in industry und IoT will then be provided. The classification will be done based on the theoretic approaches, application areas and strength and weaknesses in regard to the demands of Predictive Maintenance (PdM) in industrial IoT (IIoT) environments.

Deep Learning (DL) can be defined as a subcategory of Machine Learning (ML) whereas Machine Learning is a segment in the field of Artificial Intelligence (AI). DL itself is often defined as a class of optimised Artificial Neural Networks (ANN) comprising numerous layers (hidden layers). The high number of layers and neurons allow the abstraction of more complex problems and support further characteristics like the ability for unsupervised learning or automatic feature extraction (Lee, et al., 2015). The basic idea behind an ANN is to imitate the biological neural network in mammalian brains.

Components of an ANN are neurons (in ANNs also called nodes) and connections between those nodes. The nodes are organized in layers producing non-linear output data based on the input data. The connections between the nodes transfer the output of one node to the input of another node. Weights assigned to each connection determine the relevance of the transferred signal. As in biological neural networks the output signal of a neuron (node) is ruled by a threshold function. To set up an ANN all weights have to be set to an initial value. By training the network those weights are adjusted in a holistic way following a defined learning rate to achieve a valid and balanced network. ANNs are known for more than 50 years and various ways have been developed since. Due to the stochastic nature of neural networks, their setup and initialisation is usually done by simple poring (Devcoons, 2016), (Chatfield, et al., 2014).

In Mohammadi et al. (2018) the following DL models are listed for the use in IoT application: Auto-encoder (AE), Recurrent Neural Network (RNN), Restricted Boltzmann Machine (RBN), Deep Belief Network (DBN), Long Short-Term Memory (LSTM), Convolutional Neural Network (CNN), Variational Auto-encoder (VAE), Generative Adversarial Network (GAN) and Ladder Net. The DL models are categorized by Mohhammadi et al. (2018) into the three main groups of generative approaches (AE, RBM, DBN, VAE), discriminative approaches (RNN, LSTM, CNN) and hybrid (GAN, Ladder Net) as a combination of the two approaches mentioned before. This categorisation mainly refers to the underlying learning method whereas generative approaches basically follow the principle of unsupervised learning and discriminative approaches follow the principle supervised learning. Beside the definition of the required number of layers the underlying learning method is a decisive factor for the selection of a DL approach. The categorization in generative and discriminative approaches chosen by Mohammadi et al. (2018) can be fundamentally found in many other works (Mohammadi, et al., 2018). Different DL models are also categorized by their suitability in IoT applications. The relevant characteristics mentioned are the ability to work with (partially) unlabelled data, the magnitude of needed training dataset, dimensionality reduction abilities, the ability to deal with noisy data and time series data and their general performance classification (Mohammadi, et al., 2018). For the reduction of high dimensional data and to cope with unlabelled data the combination of RNN with Deep Belief Networks (DBN) and Auto-encoders (AE) is recommended. If the system is meant to make predictions like in PdM systems, DBN and AEs are often used as an upfront layer providing classified data to a subsequent RNN (Mohammadi, et al., 2018).

Especially when the data develops sequentially, RNNs are recommended, as they show competitive results here. However, if the data also contains long-term dependencies, RNNs are not recommendable because they can only remember previous states for a short time (Song, et al., 2016). Song et al. (2016) describes how to handle sequential data streams from human mobility and transportation transition models containing long term dependencies. The described solution is a combination of simple RNN and LSTM in the form of a specialized RNN architecture. Besides the ability to handle long term dependencies the LMST also adds labelling and predictive functionality to that combination. The LSTM and its numerous variants are mentioned as the most frequently used neural network for handling data streams or time series with long-term dependencies (such as certain behaviours or wear and tear of machines) (Song, et al., 2016). This is also confirmed in many other works (Mocanu, et al., 2016), (Gensler, et al., 2016), (Ibrahim, et al., 2016).

In his paper *IoT Data Analytics Using Deep Learning* Xie et al. (2017) describes how to select the right ANN to archive predictions from data streams and time-series data. To retrieve trends and predictions and also validate those trends and predictions in parallel by anomaly detection, a combination of LSTM with Naive Bayes models is proposed. The LSTM produces the predictions on data streams whereas the Naive Bayes model is responsible for anomaly detection performed on the results of the LSTM (Xie, et al., 2017). This paper also reflects on the fact that Simple Feedforward Artificial Neural Networks (FNN) like Single-layer Perceptron (SLP) and Multi-layer Perceptron (MLP) using standard backpropagation (BP) for training are often not a recommendable choice because they do not perform well in complex situations and on data streams especially with long-term dependencies. This is especially true when the aim of the model is to predict future events or trends. Data streams and time-series data usually have dependencies over time. Such dependencies are typical for IoT data and provide relevant insights. In simple ANNs data moves straight through the layers with the assumption that input data is independent from output data. Because of this, there is no way to remember previous input and output states (previous results). This is particularly disadvantageous when previous data is linked to current data. Using RNN instead can archive better results in data streams and time-series data. Since the connections between nodes in an RNN take the form of sequences or loops, it is possible to remember previous states, but only a few. Therefore, only short-term dependencies are recognized. Because of this Mocanu et al. (2016) recommends the application of LSTM in complex IoT environments to recognize long-term dependencies in data. LSTM are a variant of RNN introducing gated memory units. Those memory units can remember important previous states and forget the unimportant ones (Mocanu, et al., 2016).

Another paper in the field of energy management also emphasizes the very powerful forecasting abilities of DL. The application of Auto-encoder (AE) and LSTM is described for predicting the power generation of solar systems. The result quality reached by a combination of AE and LSTM (Auto-LSTM) is compared to other neural networks (namely MLP) as well as to a physical model (Gensler, et al., 2016). The benchmark data is taken from 21 real solar power plants. The following measurements are taken as benchmarks: average root-mean-square deviation (RMSD), average mean absolute error (MAE), average absolute deviation (Abs. Dev.), average BIAS and average correlation. The measured results demonstrate that all ANN- and DL-based models show far better results than the physical model. Among all ANN and DL-based models, Auto-LSTM is the best choice in this specific scenario with the specific data.

2.7 Long Short-Term Memory (LSTM)

Three evolutionary steps led to the current standard architecture of an LSTM (Graves, et al., 2005). The introduction of the forget gate meant that an LSTM is also capable of forgetting and can thus also deal with infinite data streams as input data (Gers, et al., 1999). The introduction of peepholes improved the ability of an LSTM not only to remember information over a long period of time, but also to take into account the temporal occurrence and its frequency (Gers, et al., 2000). For use in image processing, an LSTM was extended by the convolutional technique (Convolutional LSTM) (Cho, et al., 2014b), (Chiu, et al., 2016). In the following, the first two evolutionary optimisation steps for LSTM are described. Convolutional LSTM is not considered, as it is specialised in image processing and thus does not represent a use case in the context of PdM and the thesis presented here.

One of the first and most fundamental optimisations of LSTM was the introduction of forget gates by Gers, Schmidhuber and Cummins (Gers, et al., 1999). In the paper *Learning to Forget: Continual Prediction with LSTM*, it is explained that the introduction of LSTM by Hochreiter and Schmidhuber (Hochreiter, et al., 1997) was an enormous advance in that an LSTM is also able to learn long term dependencies in data and take them into account. Recurrent Neural Networks (RNN) are also able to handle dependencies in data over time, but only over relatively short periods of time. The paper states that a well-dimensioned RNN can remember relevant dependencies over a maximum of 5-10 epochs. Beyond that, remembered information is lost, even if it is still relevant. An LSTM, on the other hand, does not have this limitation. In addition to the internal layer for generating the output values, an LSTM has two additional layers that manage the memory (cell state) of the LSTM. One of these two layers decides which of the current input values (input vector comprising input data and hidden data) should be remembered. The second layer determines a weighting vector which, multiplied by the output of the first layer, determines the relevance of each value for the cell state (how strongly a value should be remembered). The input vector generated in this way is added to the cell state of the last epoch and results in the new cell state. In other words, these two layers learn which information will be needed again later and how important it is. Dependencies between samples over 1,000 epochs and more are no longer a problem. The memory requirement is ultimately determined only by the amount of relevant information. The temporal aspect has no influence on the memory requirement (Hochreiter, et al., 1997).

One problem with the original version of the LSTM according to Hochreiter and Schmidhuber (Hochreiter, et al., 1997) is that with data sets over time and with continuous data streams, the cell state can become very large. This is due to the fact that the LSTM can only decide what should be remembered from the current input and how relevant the different entries in the memory are. There is no way to forget values once they have been memorised. In the experiments conducted by Hochreiter and Schmidhuber (Hochreiter, et al., 1997), several datasets with a limited amount of data were used. At the beginning of each new epoch, the cell states were automatically reset. With continuous data streams, this implicit reset of the cell states does not happen since data arrive infinitely, or at least continuously over a very long period of time. Therefore, the idea was developed to introduce another internal layer inside the LSTM that decides what can be forgotten. This led to the introduction of the forget gate, which is now an integral part of a standard LSTM (Gers, et al., 1999), (Graves, et al., 2005). The forget gate enables an LSTM to deal with data sets over very long periods of time and also with unlimited data streams. The values that are no longer relevant are reset in the memory. In other words, the forget gate learns what can be forgotten and ensures that it is forgotten. This prevents the cell state from increasing continuously over time. In contrast to the input gate, the forget gate consists of only one layer (Gers, et al., 1999).

Figure 8 shows the inner architecture of a LSTM with forget gate, input gate, tanh gate and output gate. The cell state is indicated by the letter **c**, where $\mathbf{c_{t-1}}$ is the cell state before and $\mathbf{c_t}$ is the cell state after it was updated in the current processing step. The hidden states $\mathbf{h_{t-1}}$ and $\mathbf{h_t}$ are accordingly. The letter **x** stands for the current input data of the LSTM cell at the current time **t** ($\mathbf{x_t}$). The activation functions used for the layers of the LSTM are the sigmoid function (σ) and the tangent hyperbolicus function (tanh), shown in orange rectangles. The green circles determine the component-wise multiplication and addition of the gate results into the cell state and the hidden state.

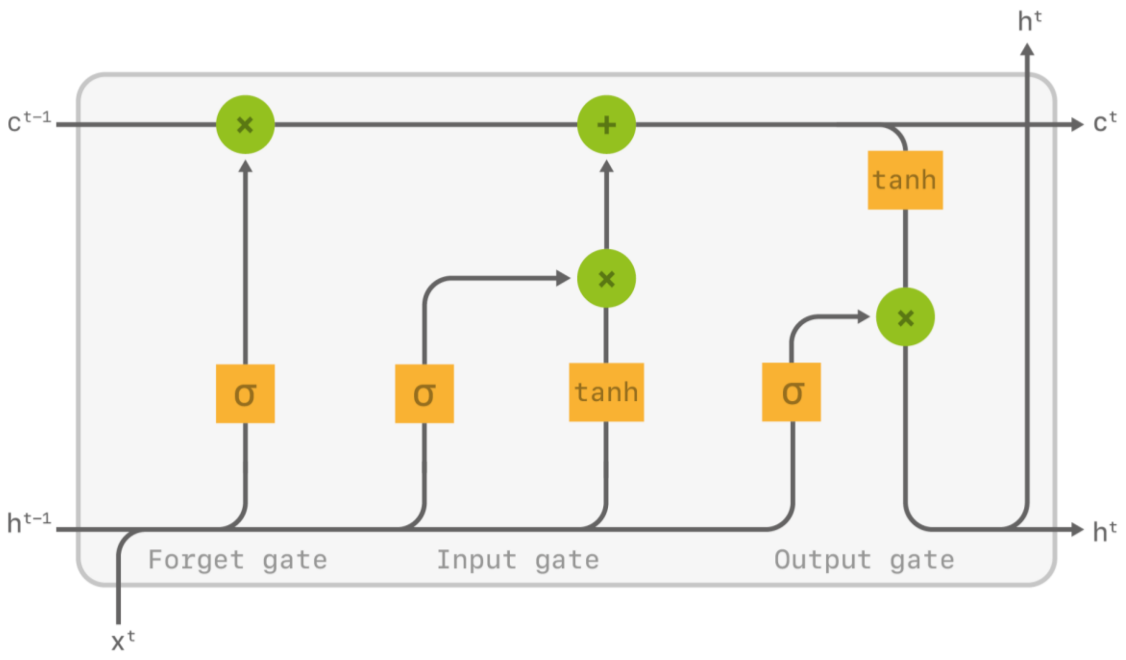


Figure 8 – Standard LSTM according to (Hochreiter, et al., 1997)

Another basic optimisation approach for LSTMs is described in the paper *Recurrent Nets that Time and Count* (Gers, et al., 2000). LSTMs are generally described in this paper as the best available solution when dependencies in data over time are relevant. In contrast to classical RNNs, an LSTM can handle long term dependencies and thus solve tasks that a standard RNN cannot. However, the question of whether an LSTM is also able to draw conclusions purely from the duration of the temporal intervals of certain results and the frequency of occurrence remains unresolved. In a series of tests, it was investigated whether an LSTM is able to measure the temporal intervals of certain events over long periods of time and count their occurrence. Furthermore, it was investigated whether an LSTM can predict temporal intervals. The dataset used for this test shows a spike in data values every 50 cycles. Nothing worth mentioning happens in the 49 cycles between two spikes. The investigations were carried out with a standard LSTM with forget gate (Gers, et al., 1999) and with an adapted LSTM. In the adapted LSTM, the forget gate and input gate were additionally connected to the stored cell state (memory state before the present input was processed). The output gate was additionally connected to the new cell state (memory state after processing the present input). Each of these three connections was also assigned a bias. The three introduced connections were called peephole connections. Figure 9 shows the inner architecture of a Peephole LSTM.

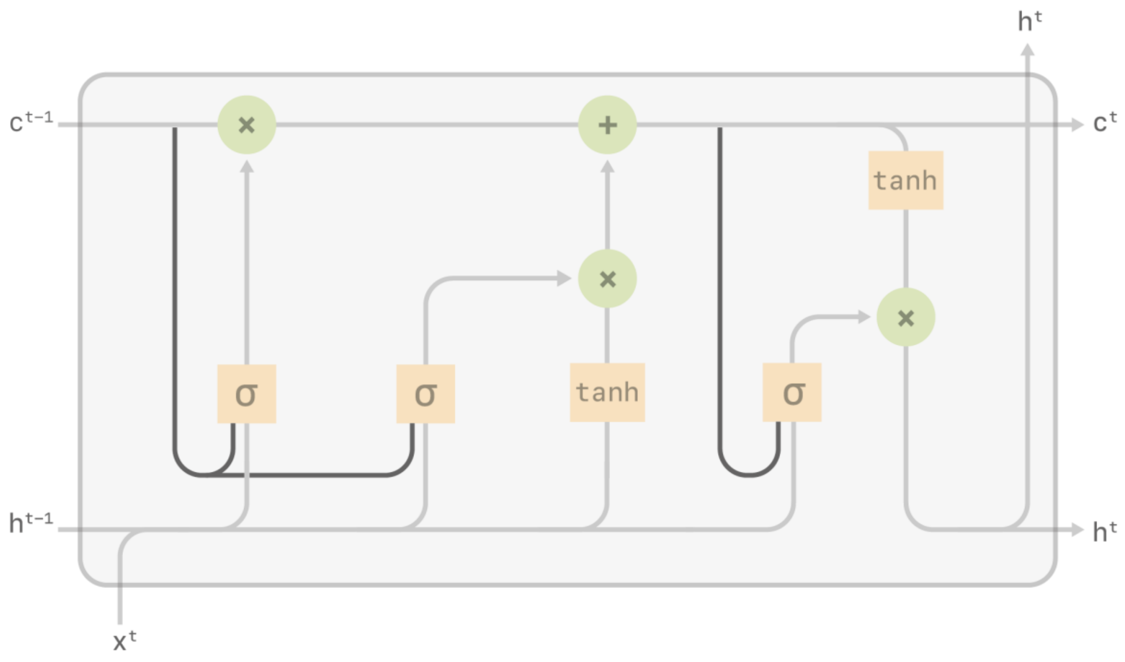


Figure 9 - Peephole LSTM according to (Gers, et al., 2000)

In a standard LSTM, the forget gate and the input gate only receive the hidden state (result) of the last processing and the current input values. Through the peephole connection to the cell state, these two gates now also receive access to the entire information stored in the cell state and its relevance.

Likewise, when determining the current result (hidden state), the new cell state with the newly remembered information and without the just forgotten information is also taken into account. This is also referred to as letting the gate layers look at the cell state. Hence the name peepholes (Gers, et al., 2000).

As a result, the peepholes significantly improve the property of an LSTM to handle temporal relationships and count the number of occurrences of events in a temporal context. The results of the investigations described in the paper *Recurrent Nets that Time and Count* prove this (Gers, et al., 2000). To store the temporal aspects of the measured signal and their frequency over time, two different strategies were used in the tests. Firstly, the values were increased at each occurrence in such a way that the temporal intervals could be determined from that value. The second approach led to oscillating cell states whose temporal courses could be derived from their oscillation. Both approaches led to comparable results (Gers, et al., 2000). It should be mentioned that the overhead of the peephole connections is considered very low in the context of the overall complexity of the LSTM cell. The complexity of the network is only increased by three additional connections, each with a bias. Adding the cell states increases the input vectors accordingly, which in itself does not increase the complexity of the network but does increase the number of vector operations (Gers, et al., 2000). Peephole Connections for LSTM have established itself as a form of optimisation in many application areas. For example, multi-dimensional LSTM with peephole connections are used in computer vision, video processing or medical imaging. In a test with video streams containing X-ray data from air cargo, the introduction of peephole connections led to significant improvements in the detection of recurring, temporally related events (Graves, et al., 2013a).

3 Related Work

In this chapter, the related research most relevant to this thesis is presented. The literature was selected according to its relevance to this thesis. Over all, the literature chosen covers a detailed discussion about the topics mentioned. Especially papers describing prediction algorithms using neural networks and novel approaches to improve established models have been chosen. This chapter is divided into four sections. The first section explains basic strategies for PdM that are important in the context of this thesis. The second section focuses on the issue of performance, followed by a section on the use of Deep Learning methods. The last section of this chapter deals with recent work discussing optimisation approaches specifically for the LSTM model, which emerged in the qualitative study as the most important model for current PdM applications with neural networks.

3.1 Predictive Maintenance Strategies

This section discusses related work on the basic classification and the strategies used in the field of maintenance and especially PdM. The maintenance goals are a result of those strategies. The Remaining Useful Lifetime (RUL) is one such goals. As will be explained later, RUL is used as the result value in this thesis and thus draws on the findings from related work described in this section.

In Lee et al. (2014) the selection of the appropriate maintenance strategy is carried out on the basis of the complexity and uncertainty of the underlying system and its data. Uncertainty includes measurement errors, incomplete data and the shortage of expert knowledge as well as errors caused by the methods used. Figure 10 illustrates this. Accordingly, Condition Based Maintenance (CBM) can be used for systems that are deterministic and stationary/static and also have meaningful data with low dimensionality. Reliability Centered Maintenance (RCM) approaches are suitable for probabilistic systems. PdM is seen in both of these fields and thus comprises both CBM and RCM (Lee, et al., 2014).

Similarly to Lee et al. (2014), Sikorska et al. (2011) assigns PdM to Diagnostic (e.g. CBM) and Prognostic (e.g. RCM), describing the prognostic as highly dependent on the diagnostic. The main difference according to Sikorska et al. (2011) is that diagnostic is mainly used for the identification and analysis of errors, whereas prognostic tries to predict errors in time before they occur. Therefore, diagnostic is retrospective in nature, while prognostic is prospective in nature (Sikorska, et al., 2011). In the paper *Prognostic modelling options for remaining useful life estimation by industry* four groups are defined to classify the relevant methods for making predictions (Sikorska, et al., 2011). In this analysis, the forecast is based exclusively on the prediction of the Remaining Useful Lifetime (RUL). The described groups are shown in Figure 11.

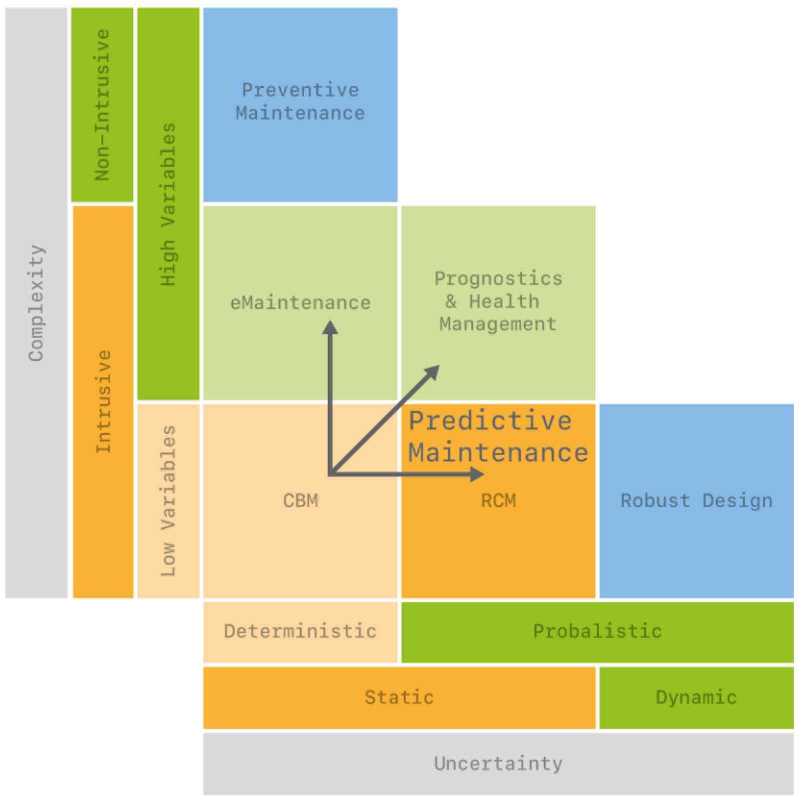


Figure 10 - Maintenance transformation map (Lee, et al., 2014)

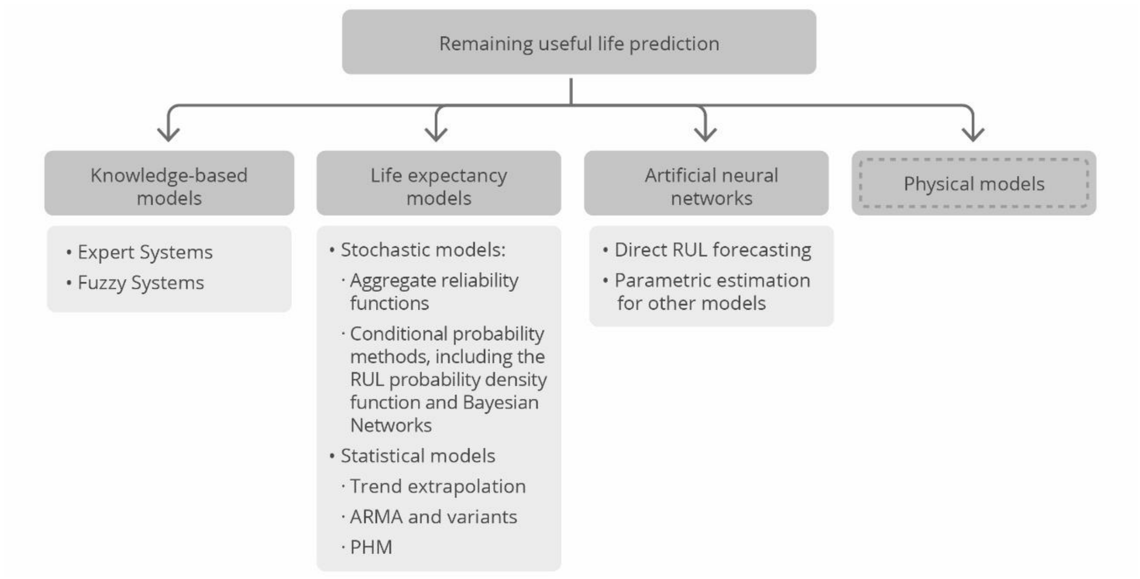

Remaining useful life prediction
Knowledge-based models
Life expectancy models
Artificial neural networks
Physical models
• Expert Systems
• Fuzzy Systems
• Stochastic models:
· Aggregate reliability functions
· Conditional probability methods, including the RUL probability density function and Bayesian Networks
• Statistical models
· Trend extrapolation
· ARMA and variants
· PHM
• Direct RUL forecasting
• Parametric estimation for other models

Figure 11 - Groups of RUL methods, summarised from (Sikorska, et al., 2011)

Knowledge-based models compare the degree of conformity of a current pattern with known, classified patterns. Life expectancy models, on the other hand, consider each individual relevant component of a system separately and determine the probability of failure under various operating conditions. Artificial Neural Networks (ANN) are known to mimic the functioning of the mammalian brain in learning processes. The learning process is mainly based on observing and evaluating the impact of calculated predictions rather than on expert knowledge. In contrast, physical models calculate wear or ageing processes over the entire life cycle of a system, based on mathematical functions. Physical models require exact expert knowledge of material properties and physical processes. They have to be specifically developed for each application case and cannot be adapted. For this reason, physical models are hardly used for prediction systems and PdM and are therefore not taken into account (Sikorska, et al., 2011).

For the prediction of RUL in PdM-Systems Sikorska et al. (2011) especially recommends methods from the area of Artificial Neural Networks (ANN). This is due to the fact that ANNs handle complex and non-linear data very well. This also applies if data available for training is limited. ANNs do not necessarily require comprehensive expert knowledge and can also exploit limited or selective expert knowledge. This is helpful because fewer and fewer proven experts with in-depth technical and process knowledge are available in industry (Sikorska, et al., 2011).

As examples of established and widely used ANN methods Sikorska et al. (2011) mentions Multi-Layer Perceptron (MLP), Radial Basis Function (RBF) Network and General Regression Neural Network (GRNN). Simple recurrent networks (SRN) are also mentioned because they are able to remember previous results, at least for a short time. LSTMs are mentioned best when long-term dependencies are relevant. In addition to data analytics and pattern recognition, Recurrent Networks are also suitable for reducing complexity through clustering (Sikorska, et al., 2011).

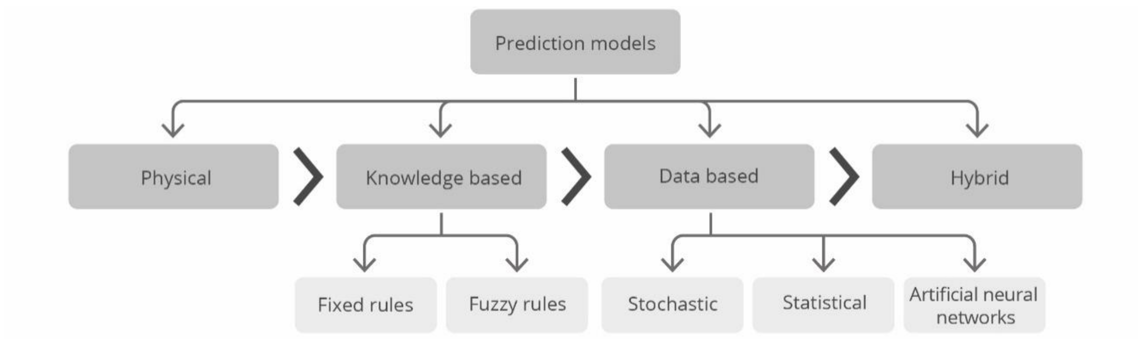


Figure 12 - Prediction models classification, taken from (Schmidt, et al., 2015)

In Schmidt et al. (2015) a similar grouping of the methods for PdM is carried out as in Sikorska et al. (2011). Figure 12 shows a diagram that summarises the categories of PdM as proposed in the paper *Predictive Maintenance: Literature Review and Future Trends* (Schmidt, et al., 2015).

Unlike Sikorska et al. (2011), Schmidt et al. (2015) defines the class of data-based models to which he assigns stochastic models, statistical models and ANNs. In addition, it is pointed out that model combinations are often used in real applications. In Figure 12, method combinations are indicated by the fourth group Hybrid. Knowledge-based models are described as expert systems comprising mainly fixed rules. Historical data is only used for data-driven models. ANNs are particularly recommended when there is no in-depth expert knowledge of the system and the underlying process, but knowledge is available in the form of historical data (Schmidt, et al., 2015). However, Schmidt et al. (2015) considers that an acceptable quality of the predictions can only be achieved if sufficient training data is available, whereas in Sikorska et al. (2011) applicability even with incomplete training data was mentioned.

Statistical methods are mostly used if sufficient information about the system is already available. According to Schmidt et al. (2015), statistical methods are often used when CBM systems already exist, and their knowledge can be reused. Common statistical methods for PdM include trend extrapolation, regression methods, Support Vector Machines (SVM), as well as Autoregressive Moving Average (ARMA) and Autoregressive Moving Average with Exogenous Inputs (ARMAX) from the autoregressive models (Schmidt, et al., 2015).

In addition to the utilisation of context data, knowledge management and the selection of a systematic approach, Schmidt et al. (2015) agrees with Lee et al. (2014) that dealing with "uncertainty" is one of the major challenges of a PdM system. The sources for "uncertainty" include measurement errors, missing data and a lack of expert knowledge, as well as conceptual or implementation errors in applied methods. Here neural networks have advantages (Schmidt, et al., 2015), (Lee, et al., 2014).

In Heng et al. (2009) the various models are also arranged quite similarly. Figure 13 summarises the arrangement described. Data-driven models are also given greatest importance since they can develop and continuously improve models on the basis of collected (and historical) data. Moreover, comprehensive expert knowledge of the system and the process is not a must for all variants of data-driven models (Heng, et al., 2009).

A decisive factor for the successful use of Preventive Maintenance as a maintenance strategy is the knowledge of how long the durability, and thus the serviceability, of individual components of a system are. With this knowledge, optimum maintenance intervals can be determined. Only if the required knowledge is available, the strategy is implementable. However, there is still a risk of unplanned breakdowns, for example due to material failure, because no analysis of current machine data in the sense of condition monitoring is included in this maintenance strategy. Freitag et al. (2015)

accordingly proposes a strategy in which Preventive Maintenance is chosen as the main strategy, but maintenance intervals are continuously adjusted dynamically by using Predictive Maintenance (Freitag, et al., 2015).

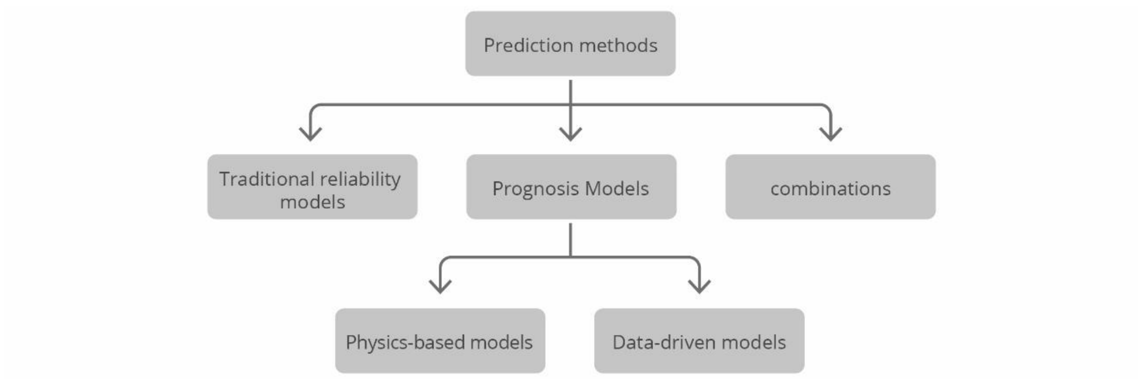


Figure 13 - Classification of Prediction Methods following (Heng, et al., 2009)

Data-driven models are subdivided by Peng et al. (2010) into statistical approaches and AI approaches. In contrast to other statements in the literature Peng et al. (2010) refers to ANNs as the only practicable approach for complex, high-dimensional, dynamic and non-linear data (Peng, et al., 2010).

3.2 Performance in Predictive Systems

This section discusses related work that specifically addresses runtime behaviour and performance issues in traditional predictive applications. Strategies and mechanisms for reducing the processing effort and complexity represent the basic approach of all related work explained below.

Since the cost of setting up and operating a predictive system such as PdM is high, many papers like Tran et al. (2022) or Lee et al. (2014) suggests that PdM should only be performed for the most critical components and that less critical components should be treated with less complex procedures. This applies in particular to limited computational resources and when rapid response times are required (time-sensitive systems). To define the most critical components, all components in the system that could potentially fail and whose failure should be prevented by timely maintenance are to be identified. In the next step, all components are evaluated according to their failure frequency and the expected downtime due to their failure. Only for components with low failure frequency and high downtime is the use of PdM procedures recommended. The components that have a high failure frequency and cause high downtime must be eliminated "by design". All components that cause low downtime should be treated with standard approaches of Reactive Maintenance and Preventive Maintenance. The respective limits for classification into high or low downtime must be determined by an expert, depending on the circumstances and the desired objectives (Tran, et al., 2022), (Lee, et al., 2014).

Algorithms suitable for PdM, extract from (Lee et al. 2014)	**Complexity / Dimensionali**	**Non-stationary / evolving**	**Non-linearity**	**Adaptive**	**Real-time-enabled**	**Comments**
Wavelet Packet Energies (WPE)		x	x	x		Very powerful for non-stationary signal analysis; not easy to handle
Hilbert–Huang Transform (HHT)	x	x	x	x		Adaptive and unsupervised method for non-stationary/nonlinear signals; high computational load
Principal Component Analysis (PCA)	x					Reduces dimensionality implicitly; occasional poor performance; requires linear data
Fisher Linear Discriminant (FLD)	x					Reduces dimensionality implicitly; stores superordinate relationships; requires linear data
Gaussian Process Regression/Prediction				x		Can learn and adapt; only suitable for Gaussian likelihood
Particle Filter (PF)			x			For non-linear/non-Gaussian data; high accuracy; very high computation costs
Kalman Filter (KF)				x		Corrective by each measurement; only works with linear systems and Gaussian noise
Feature map pattern matching (Selforganizing Maps)	x					Unsupervised learning methods; lack of standard algorithm
Bayesian Networks	x					Reduces dimensionality implicitly; intensive training and domain knowledge necessary
Neural Network (NN / ANN)	x	x	x	x		Adaptive, suitable for complex, non-linear and unstable systems; no standards, high computational costs
Autoregressive Moving Average (ARMA)	x			x		For linear time-invariant systems; utilizes historical data; limited when non-linear and complex
Fuzzy Logic (FL)	x			x		For complex/unknown systems; could be unprecise under specific conditions
Match Matrix (MM)	x	x	x	x		Better longterm predictions; needs sufficient historical data (including degradation data)
Support Vector Machine (SVM)	x				x	Efficient for large datasets and real-time analysis; the selection of an appropriate kernel method is needed
Hidden Markov Model (HMM)		x	x	x		For dynamic systems and high dimensionality; complex modelling is needed

Table 2 - Relevant algorithms for PdM, according to (Lee, et al., 2014)

This approach aims on reducing the complexity and therefore the effort involved in calculating predictions in an upstream step. Optimisation in the calculation of the predictions by adapting the algorithms or the underlying IT processes is not considered (Tran, et al., 2022), (Lee, et al., 2014). The algorithms considered as relevant for PdM by Lee et al. (2014) are listed in the Table 2. According to that table, approximately 67% of these algorithms are suitable for data of high complexity and high dimensionality. Almost the same number has adaptive abilities and can react dynamically to changes in data. In contrast, only about 40% of the algorithms are suitable for non-linear data and only about 33% are able to react adequately to concept-drifts. Table 2 shows that methods based on neural networks are well suited in 4 of the 5 categories. Only for real-time applications Lee et al. (2014) does not consider neural networks to be suitable (Lee, et al., 2014).

The strategy of reducing complexity in an upstream step was found very frequently in the literature reviewed. However, the proposed methods used to reduce complexity differ significantly from one another. Decomposition is another popular strategy, especially for processing data streams, but could lead to insights based only on local dependencies. (Krawczyk, et al., 2015).

In the paper Real-time fault detection for advanced maintenance of sustainable technical systems Ait-Alla et al. (2015) describes how real-time requirements can be achieved in the area of fault detection. Error detection methods, like methods for predicting future errors, are very complex and therefore not in themselves suitable for time-critical applications. Ait-Alla et al. (2015) explains how this problem can be solved by an additional pre-processing step. Other approaches attempt to reduce the complexity of data. In Ait-Alla et al. (2015) however, the pre-processing step acts as a sort of filter. By using Complex Event Processing (CEP) methods, events are generated on the incoming data. The elaborate fault detection algorithms are only triggered when specific events occur. Depending on the event type, only the relevant data is included. The generation of events is based on a dynamic event scheme, which is created with the help of data mining methods and adapts dynamically in parallel to the event processing. Ait-Alla et al. (2015) describes this method as an event-driven fault detection system (ED-FDS). This approach could also be interesting for time-critical PdM applications, especially in combination with other approaches like the concentration on critical components as mentioned in Lee et al. (2015) and Freitag et al. (2015) (Ait-Alla, et al., 2015), (Lee, et al., 2015), (Freitag, et al., 2015). Other event-driven approaches similar to the ED-FDS can also be found in the literature, such as focusing on specific time intervals instead of data-related events (Pan, et al., 2021).

As with Lee et al. (2014), Freitag et al. (2015) also try to assess the individual components of a system by how critical the failure of a component is. Depending on this classification, less critical components are monitored very simple by means of threshold values, while for critical components, diagnostics and predictions are performed as a combination of condition monitoring and predictive analytics. In the case of critical components, Freitag et al. (2015) draws a further distinction between those whose wear and tear or aging course are known and should ideally be linear and those whose course is unknown or non-deterministic. For the first group, Fourier transformation, Wavelet analysis and Weibull distributions are mentioned as possible methods. The use of neural networks is recommended for the second group (Freitag, et al., 2015).

As Ait-Alla et al. (2015) explains in his paper, the approach of the event-driven fault detection system (ED-FDS) also has disadvantages. These are, in particular, the additional effort that the pre-processing approach requires. The event rules and the system for event-driven fault detection are set up and parameterised manually in the current ED-FDS. It can be assumed, however, that the parameterisation of the error detection is not only necessary initially during the system setup, but continuously during the entire operation. In his paper, Ait-Alla et al. (2015) mentions that an automatic and continuous updating of the event rules is required. However, Ait-Alla et al. (2015) makes no statement in his paper about how this should be done. The paper only points out that the approach of the ED-FDS is limited to the manually predefined event rules and does not address event rules beyond these. In the practical application of the ED-FDS, this probably represents a further limitation of this approach.

In his paper, Ait-Alla et al. (2015) also states that the event-driven fault detection system can detect changes in real time. However, the requirements for a system environment regarding real-time processing of event-driven fault detection are not explained.

As already explained in this section, the widely used decomposition approach has the serious disadvantage that it can lead to findings that are only based on local dependencies. Since the ED-FDS does not reduce the data itself, but the processing frequency (only for certain events), the ED-FDS should not be affected by this disadvantage. In the context of the ED-FDS, it would be interesting if there were further studies on this.

In his work *Big Data Analytics for Fault Detection and its Application in Maintenance* Zhang (2016c) covers the problematic caused by data streams and exemplifies the significance of high complexity (namely volume and velocity, high dimensionality, non-stationary and non-linearity), combined with the fact, that the volume of heterogeneous data from arbitrary sources is “exploding” in industrial systems (Zhang, 2016c). In the paper *An Angle-based Subspace Anomaly Detection Approach to High-dimensional Data: With an Application to Industrial Fault Detection* Zhang et al. (2016a) introduces the Angle-Based Outlier Detection (ABOD) algorithm for high-dimensional spaces. In a corresponding paper, focusing on fault detection in data streams, Zhang (2016c) recommends a Principal Component Analysis (PCA) algorithm. The initial intent of PCA is dimensionality reduction, but it’s also widely used in practice for anomaly detection. Following Zhang (2016c) conventional PCA algorithms are not adaptive and therefore inadequate for performing analytics on data streams. For that reason, a recursive PCA approach (RPCA) was introduced updating the PCA model recursively when new data is available. Because the PCA was designed to work in a batch-oriented system and always performs on the entire data sets, it treats all data with the same priority when updating the model. To further improve the PCA/RPCA the idea was taken into account that in data streams with varying context over time recent data could be more relevant than older data. For this reason fading functions were developed. As an example, an exponentially weighted PCA (EWPCA) was mentioned, putting an adjustable weighting factor to newer and older data when recursively updating the model. Sliding window PCA (SWPCA) was mentioned, using a window of fixed size and trying to exchange the oldest sample by the newest one identified as normal (Zhang, 2016c), (Zhang, et al., 2016a).

In the paper *Sliding Window-based Fault Detection from High-dimensional Data Streams* Zhang et al. (2017a) developed a new approach for analysing data streams by extending an Angle-Based Subspace Anomaly Detection (ABSAD) approach with sliding window functionality. Additionally, the process of fault detection was divided into two steps to improve parallelization and scalability. To measure the efficiency of the adapted ABSAD approach, measurements were conducted by applying different established approaches beside the adapted ABSAD approach on a synthetic data set. The approaches chosen by Zhang et al. (2017a) are primitive ABSAD, primitive LOF and sliding window LOF. The synthetic data was loaded in a stream fashion and divided into normal and faulty samples whereby parts of the normal samples change to faulty samples over time (concept drift). The evaluation of the binary classification was done by measuring the degree of improvement of the so-called Type I error and Type II error. The Type I error is defined as the false positive rate quantifying the quality by which the faulty samples are determined as such. The Type II error (false negative) quantifies the failure to detect a faulty sample. The result of this research presented in Zhang et al. (2017a) shows that the Type I error could be lowered significantly in all cases, whereas Type II error improvements could only be achieved for the LOF-based approaches. Even if the research stated in Zhang et al. (2017a) is dedicated to fault detection it was mentioned that this could also be true for predictions (Zhang, et al., 2017a).

It would have been interesting to know what share of the results is accounted for by the recursive approach of PCA alone. This is not clear from the papers Zhang et al. (2016a), Zhang (2016c) and Zhang et al. (2017a). Through the combination with established sliding window techniques, it is not clear what share the recursive PCA has in the results achieved by the ABOD/ABSAD, RPCA and SWPCA algorithms. The recursive processing of the PCA algorithm presumably leads to a significantly more complex system structure. It can be assumed that the application of ABOD/ABSAD, RPCA and SWPCA is complex, making their comprehensibility and applicability more difficult. One result of the study conducted in this thesis is that very simple algorithms are often used in practical applications. Existing disadvantages of simple algorithms are often accepted in order to achieve an overall setup that is still manageable and comprehensible (see section 5.5). However, the algorithms ABOD/ABSAD, RPCA and SWPCA described in the papers Zhang et al. (2016a), Zhang (2016c) and Zhang et al. (2017a) increase the overall complexity of a setup. It would have been interesting to see how Zang assesses this. However, no statement is made on this in the two papers listed.

In the paper *Classifying Data Streams with Skewed Class Distributions and Concept Drifts* Gao et al. (2008) states, that for processing high-speed data streams granularity-based techniques like specialized sampling methods have been developed. Regarding this Gao et al. (2008) states that reduction techniques are more or less the standard when high volumes of data have to be processed in real-time. This is applies even more if data is of high dimensionality. There are only a few studies about processing high dimensional data in real-time. This is a gap that needs to be closed by further research (Gao, et al., 2008).

In summary, Table 3 lists all the related work discussed in this fundamental section with the respective methods.

Reference	**Methods to improve Performance**	**Relevance**
(Tran, et al., 2022), (Lee, et al., 2014)	Concentration on relevant methods, PdM only for the most critical components	Complexity reduction
(Pan, et al., 2021), (Ait-Alla, et al., 2015)	Event-driven filtering of incoming data in a pre-processing step, parallel processing, concentration on critical components	Complexity reduction
(Zhang, et al., 2017a), (Zhang, 2016c), (Zhang, et al., 2016a)	Decomposition of data by fading functions, Sliding window for data streams	Complexity reduction Reducing incoming data
(Krawczyk, et al., 2015).	Decomposition of data	Complexity reduction Split incoming data, parallel processing of parts of data
(Gao, et al., 2008), (Tenenbaum, et al., 2000), (Roweis, et al., 2000).	Granularity-based and sampling techniques, reduction techniques in general	Complexity reduction Reducing incoming data

Table 3 – List of the discussed related work

As shown in this section, the applied algorithms are very diverse and there is no general rule for applying specific algorithms. When performance is relevant, reduction techniques can be regarded as the common approach in the viewed literature. ANN's are mentioned when high complexity and uncertainty is present. Research related to this thesis regarding Deep Learning and ANNs in the field of PdM is discussed in the next sections.

3.3 Fast Predictions using DL

PdM applications can benefit from DL, especially when it comes to high complex, non-linear and unlabeled (unknown) data. Especially with PdM applications being used in connected smart factories, low latency predictions are essential. For this reason, processing time is becoming increasingly important and must therefore be taken into account when constructing deep networks and selecting the neuronal cell types used. This section provides an analysis of related literature applying Deep Learning (DL) techniques and Artificial Neural Networks (ANN) in the field of industrial IoT (IIoT) to produce fast predictions of maintenance issues.

In many IoT applications, fast processing of the incoming data is essential. For example, in a PdM system high latency could lead to unintentional reactive maintenance because of insufficient lead time to plan the maintenance tasks (Bauer, et al., 2017). This is particularly the case when data arrives in continuous data streams. How fast the processing needs to be, strongly depends on the application case, the incoming data and the complexity of the task. According to Rippel et al. (2017) in micro manufacturing systems, where vast volumes of micro parts are manufactured with high speed, the term real-time means microseconds. In his paper he shows that with systems for fault detection and PdM the rejection rate of the manufactured micro parts decrease by increasing processing speed (Rippel, et al., 2017). In other scenarios, the term of real-time can mean seconds, minutes or hours. For example, in PdM Applications for offshore wind turbines the frequency with which data is available is mostly minutes and hours (Freitag, et al., 2015).

In his paper *Metro Density Prediction with Recurrent Neural Network on Streaming CDR Data* Liang et al (2016) describes the implementation of a real-time public transportation crowd prediction system using a weight-sharing recurrent neural network in combination with parallel streaming analytical programming (Liang, et al., 2016). Fast response time to emergent situations (e.g. entrance records in metro stations combined with telecommunication data) demand real-time analysis. The use of a powerful neural network model with strong learning capability offers a wide range of new insights but contrast with the need for fast response time. The way to meet this goal is described in the paper in three steps: a) adopting a RNN model to improve its ability to work on data streams, b) implement strategies for parallelization of RNNs and c) the use of parallel streaming analytical algorithms over a cloud-based stream processing platform. In the project described each metro station is modelled by an independent RNN. Shared layers are introduced to share weights from stations which are in similar situations (e.g. downtown stations during rush hour) and across several models dynamically (Liang, et al., 2016).

Especially with streams of sensor data, LSTM and the Gated Reccurent Unit (GRU) provide better performance than other models. Such Sensor data is dominating in most PdM applications (Pusala, et al., 2016). In order to be able to develop and permanently adapt models on massive data comprising the behaviour of people and their spatial and temporal attributes together with transportation capacities, fast processing and fast learning approaches are essential. Song et al. (2016) describes a multi-task deep LSTM learning architecture. The basic idea of this concept is not to use a joint feature vector, but various LSTM tasks separated by their domain (e.g. respectively a separate task for mobility and transportation mode prediction). This architecture performs parallel learning whereas the results are aggregated depending on the intended insights (Song, et al., 2016).

Lee et al (2015) mentions that because of the demands for real-time processing, the organization of layers and connections have changed. Fully connected networks where each node of a layer is connected to all nodes of the subsequent layer can handle complex problems but also demand a lot computing power (Lee, et al., 2015). Dropout those connections not really influencing the result is a strategy to reduce the complexity of a DL network, and therefore its computing demand, without decreasing result quality in a relevant manner. Besides dropout also max pooling layers, batch normalization and transfer learning are mentioned as additional strategies for performance optimization (Lee, et al., 2015).

Despite all the mentioned papers discussing performance enhancements and real-time abilities of DL models, Canziani et al. (2016) considers that highest accuracy still stands over all in mostly all current DL projects (Canziani, et al., 2016). In his paper *An Analysis of Deep Neural Network Models for practical Applications* he argues that numerous DL approaches described in literature are simply not suitable for practical use. This is for example because of their long processing time or excessive power consumption. In his paper he demands to spend more attention to performance issues because they are key factors in practical DL applications. The paper compares 14 different specific DL projects like AlexNet or GoogLeNet by comparing their accuracy, memory footprint, parameters, operations count, inference time, and power consumption. The paper shows that a small increase in accuracy lead to an enormous increase in computational power and computation time. The paper recommends defining a maximum energy consumption for each DL project and adjusting the quality of results accordingly (Canziani, et al., 2016).

This section has shown that the use of DL in IoT and PdM is a vital topic in industry. Many different applications are in use in practice and are constantly being developed and improved. Frequently reported are combinations of different DL models to combine different advantages and strengths in one application. Also, the need for real-time processing of complex data and data streams has been demonstrated in certain application scenarios. This include in particular applications for predictions such as PdM. In order to increase the real-time capability, concepts of parallel DL networks using a final aggregation layer, or intermediate layers for the reduction of complexity are frequently used. Although many activities can be observed in the area of real-time processing of DL models, there are also critical voices criticizing the absolute focus on result quality and calling for a greater focus on performance and lighter applications suitable for practical use. Almost all reports agree that a lot of research is still needed in this area.

3.4 Optimisation Approaches for LSTM

In this section we will look at related work that deals specifically with optimizing the performance of LSTMs. The application area considered is not limited to PdM applications. LSTM optimisations from other application areas are also explained, as these also provide inspiration for the construction of the SlicedLSTM. Even though hardware-related optimisations represent the majority of the work found, they are not considered here because hardware optimization is not in the focus of this thesis. Hardware optimisations accelerate the processing of software. In the field of machine learning and neural networks, hardware optimisations often focus on parallel processing on Graphics Processing Units (GPU) (Stollenga, et al., 2015). All software-side optimisations for LSTM described in this section can benefit from hardware optimisations as additional optimisation potential.

The optimisation approaches listed here as related work are all software based. They can be divided into two basic categories: optimisation of the training performance and optimisation of the runtime performance. Optimisations in training an LSTM are mainly related to mitigating the problem of vanishing and exploding gradients (Hochreiter, et al., 1997). The runtime optimisations presented here, on the other hand, are based on reducing the complexity of the network or the amount of processing at a given processing step. This section ends with a tabular representation and classification of the related work discussed.

In the paper *Parallel Multi-Dimensional LSTM, With Application to Fast Biomedical Volumetric Image Segmentation* Stollenga et al. (2015) describes an optimization approach for LSTM to reduce model complexity and thereby optimise the performance of Multi-Dimensional LSTM (MD-LSTM) (Stollenga, et al., 2015). The tests relate to the processing of 2D and 3D images and videos. The task of the LSTM is to segment foreground from background pixel by pixel in the 2D and 3D image data. Multi-dimensional means that all layers of a neural network are connected to each other. In a two-dimensional data space (2D image or video), for example, each cell (pixel) has two connections, one to the cell on the right and one to the cell below it. Each cell processes one pixel in the image. Using the information from the neighbouring pixels, edges can be detected and thus the currently processed pixel can be segmented. The segmentation in the example here is into pixels that belong to the foreground or background of the image. The approach of multi-dimensional NN for processing 2D and 3D volumetric data sources is a well-known method in image processing. Mostly, Convolutional Neural Networks (CNN) are used. The method with multi-dimensional CNNs is described in detail in the paper *Multi-Dimensional Recurrent Neural Networks* (Graves, et al., 2013a) and in the paper *ImageNet Classification with Deep Convolutional Networks* (Krizhevsky, et al., 2012), among others. One application of this method is in the medical field, such as in the processing of computer tomography images in neurology (Wang, et al., 2015), (Ciresan, et al., 2012a), (Liu, et al., 2014). The adaptation of this method using LSTMs instead of CNNs has also been done. The significant advantage of LSTM in contrast to CNN is that a multi-dimensional LTSM can perceive the entire spatio-temporal context of each pixel in a few passes through all pixels due to its cell state (Byeon, et al., 2015).

Even though there are advantages of using LSTM instead of CNN as a multidimensional mesh, they can hardly be parallelised on a GPU due to their structure. Therefore, the novel approach of Pyramidal Multi-Dimensional LSTM (PyraMiD-LSTM) was developed based on MD-LSTM. The idea is to reduce the number of neighbours in volumetric data, i.e. the number of connections in the mesh that a data point (pixel) has. This is done by recursively determining the environment of a data point (pixel) no longer for the horizontal (pixel on the right) and vertical (pixel below) neighbours, but only for the respective diagonal neighbour (always from left to right). The recursive evaluation of the context thus takes place at a 45° angle. In a 2D image, this means that the context of a pixel no longer includes the pixel to the right and below, but only the one pixel diagonally to the right. The processing of the context information of a pixel in a two-dimensional data space therefore no longer takes place in the form of a rectangular grid, but in a triangular form. In a 3D image, the processing is accordingly no longer in a cubic form, but in a pyramidal form. If one uses one LSTM for each pixel to be evaluated in a 3D image, one needs at least 8 LSTMs to fill a cubic shape and 6 LSTMs to fill a pyramidal shape. For any given dimension d, an MD-LSTM thus requires 2^d LSTMs, whereas a Pyra-LSTM requires only 2d LSTMs. In other words, an MD-LSTM grows exponentially with its dimension, whereas a Pyra-LSTM grows only linearly. The larger such a mesh is, the more pronounced the effect of the Pyra-LSTM. If fewer LSTMs are required to process data of the same dimension, the number of required computational operations in each processing step is reduced and thus the complexity of the mesh as a whole. In addition, the pyramidal structure is mentioned to be much more suitable for parallel processing on GPU hardware (Stollenga, et al., 2015). The fact that this leads to a measurable improvement in performance compared to a standard MD-LSTM was proven by means of test series. The tests were carried out with two different datasets with 3D image data from the medical field. The first dataset comprises neurological images of a fruit fly, which are to be segmented into their individual structures (Cardona, et al., 2010). The second dataset includes data from Magnetic Resonance (MR) brain images, as well as an associated evaluation framework for brain image segmentation in 3-tesla magnetic resonance imaging (3T MRI) as a reference (Mendrik, et al., 2015).

The PyraMiD-LSTM optimises the previous MD-LSTM in two areas. On the one hand, the pyramidal arrangement of the LSTM mesh reduces the number of LSTM cells required, the more so as the mesh becomes more complex. On the other hand, the pyramidal arrangement can be better parallelised and thus optimises the processing time on a GPU (Stollenga, et al., 2015).

However, the Pyra-LSTM approach can only be applied to volumetric data sources where the context of a data point depends on its direct neighbours. For 2D and 3D image and video data, the segmentation into foreground or background (and thus the detection of edges) can be determined by evaluating the pixels surrounding that pixel. In contrast, the thesis presented here relates to the analysis of sensor data from industrial plants with neural networks. Such sensor data are to be used to draw conclusions about wear and special events in a plant and to initiate maintenance measures at an early stage. The arrangement and sequence of data in the input vector of such a system do not allow any conclusions to be drawn a priori. The relationship "direct neighbour" does not have a firmly defined context, as is the case with image data. Therefore, multi-dimensional approaches are generally not suitable for PdM applications. The Pyra-LSTM is not applicable for PdM and thus does not represent an optimisation concept in this area.

A model inspired by the LSTM model but with a much simpler structure is the Gated Recurrent Unit (GRU) (Lu, et al., 2017). In the paper *Learning Phrase Representations using RNN Encoder-Decoder*, Cho et al. (2014a) describes a newly developed approach for translating texts (Cho, et al., 2014a). The model consists internally of two neural networks (encoder and decoder). The encoder divides a text into sequences of fixed length, which are then translated directly into the target language by the decoder. This novel model layout thus combines the two steps of encoding and decoding in one cell. With texts that are to be translated from English into French, Cho et al. (2014a) achieved significantly better translation results than with a standard RNN (Cho, et al., 2014a). Based on the internal architecture of an LSTM, he then extended the RNNs used in the encoders and decoders by two gates in a further step. This resulted in a new type of model layout, which is called a gated recurrent unit (GRU) (Cho, et al., 2014a). Unlike the LSTM, however, the GRU uses only the hidden state as internal memory and does not introduce an additional cell state. With each processing, the hidden state of the previous processing ($\mathbf{h_{t-1}}$) is transferred to a new hidden state ($\mathbf{h_t}$). The GRU manages the hidden state with the help of two gates, the update gate and the reset gate. The update gate decides how strongly the hidden state $\mathbf{h_{t-1}}$ and the results of the current processing step (candidate states) are to influence the new hidden state $\mathbf{h_t}$. The Reset Gate deletes entries from the hidden state $\mathbf{h_{t-1}}$ beforehand. This achieves an effect similar to that of the LSTM. The GRU is thus also able to decide which previous information should continue to be remembered, which new information is relevant enough to be remembered and which should be forgotten (Cho, et al., 2014a).

In his test series, Cho et al. (2014a) compares the newly introduced RNN encoder-decoder based on standard RNN and with a version using GRUs (Cho, et al., 2014a). An existing series of tests comparing the use of neural networks in the field of language translation with established approaches and demonstrating the advantages of neural networks in this field was used as the basic test setup (Kalchbrenner, et al., 2013), (Sutskever, et al., 2014). The tests by Cho et al. (2014a) clearly show that the results are many times better when using the GRU as opposed to RNN. This confirms the effectiveness of the concept adapted from the LSTM of controlling the memory of an RNN via gates. The GRU thus takes the idea of the LSTM but implements it in a much simpler way. This simplified internal structure compared to the LSTM makes the GRU easier to handle and train. In certain use cases, such as the translation of texts, the GRU achieves very comparable results despite its simpler structure (Cho, et al., 2014a). In another paper, Cho et al. (2014b) introduces a next optimisation step for his GRU approach, in which he extends the GRU with convolutional techniques. The resulting Gated Recursive Convolutional Neural Network (grConv) is again compared to the RNN-based encoder-decoder model. In the paper *On the Properties of Neural Machine Translation: Encoder-Decoder Approaches* the test series again consists of text translations from English to French. As a result, it was found that the already confirmed gate functions are essential for comparable results. In addition, the grConv shows comparable results in learning grammatical structures in the texts independently (Cho, et al., 2014b).

That the introduction of gates brings a decisive improvement in contrast to simple RNNs is also described by Bahdanau et al. (2015) in his paper *Neural Machine Translation by Jointly Learning to Align and Translate* (Bahdanau, et al., 2015). The research conducted in this paper also relates to the translation of texts. Gated Recurrent Units, i.e. GRU and LSTM, clearly outperform comparable approaches without gates in this use case (Bahdanau, et al., 2015).

Whether GRU and LSTM also outperform RNNs without gating units in other application areas is investigated by Chung et al. (2014) in his paper *Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling* (Chung, et al., 2014). His research is based on the assumption that deep neural networks have the potential to replace classical approaches in almost all application areas (Sutskever, et al., 2014), (Bahdanau, et al., 2015). Chung et al. (2014) mentions that almost all studies that make this statement in the results were not carried out with simple neural networks (like FFNN), but with recurrent neural networks. Very often, models were used that can deal with long term dependencies. The paper of Chung et al. (2014) describes an empirical evaluation based on tests with different data sets. The test data are three data sets from the field of polyphonic music and two data sets with raw speech data. These test data were processed for comparison with a standard RNN, with a GRU and with an LSTM. The focus was on how the different approaches can deal with sequential data (data streams) and how well they are able to notice connections over long temporal distances in data. As a result, it was clear that the models with gated units outperformed those without. This was particularly noticeable in the processing of speech data. The comparison of GRU and LSTM, on the other hand, showed no clear winner. Both models performed slightly differently in different scenarios. In certain scenarios, the much simpler GRU showed equivalent or even better results than the LSTM. However, Chung et al. (2014) clearly states in his paper that he did not investigate further the causes for when LSTM or GRU performed better and cannot provide a reason or rule for this. The aim of his study was to show that GRU and LSTM outperform classical RNNs in areas other than text translation. In general, however, it can be stated that GRU and LSTM perform similarly well in most cases. However, GRU, which has a much simpler structure, also has the advantage of being easier to use and required significantly less CPU time in the tests (Chung, et al., 2014).

GRU has been considered in great detail here. This is justified by the fact that GRU is now a very popular and widespread form of gated RNNs (Cho, et al., 2014a), (Cho, et al., 2014b), (Chung, et al., 2014), (Jozefowicz, et al., 2015) (Yao, et al., 2015). Moreover, the GRU impressively demonstrates that performance benefits can be achieved by reducing the complexity of the inner architecture of a model based on LSTM (Chung, et al., 2014). This insight contributed to the basic idea for this thesis.

In the paper *LSTM: A Search Space Odyssey* Greff et. al. (2017) gives an overview of known optimisation options for LSTMs and compares their performance using an extensive test series (Greff, et al., 2017). The comparison was made with a test setup that was kept as simple as possible with three differently structured datasets from different application areas (acoustic modelling, handwriting recognition, and polyphonic music modelling). A total of 5,400 experimental runs were carried out with all variants and all datasets. According to the author, the tests would require about 15 years of CPU time on a standard PC hardware and can thus be classified as one of the most extensive test series in this environment. The measured results are exclusively relative values and serve the empirical comparison of the different variants. A standard LSTM (Graves, et al., 2005) serves as the starting point for all comparisons. In accordance with the numerous variants of LSTMs that have arisen through adjustments to the gates or the internal structures and links, eight variants of an LSTM were generated, to each of which only one adjustment was made in order to isolate the resulting effect. For this reason alone, the test results are not comparable in absolute terms, but only relatively within the study. Well-known optimisation approaches for LSTM, such as the introduction of a forget gate (Gers, et al., 1999), the use of peephole connections (Gers, et al., 2000) or gated recurrent units in neural networks (Chung, et al., 2014) form the basic idea for the eight adapted variants that are compared here. The tested variants were labelled NIG (No Input Gate), NFG (No Forget Gate), NOG (No Output Gate), NIAF (No Input Activation Function), NOAF (No Output Activation Function), CIFG (Coupled Input and Forget Gate), NP (No Peepholes) and FGR (Full Gate Recurrence) (Greff, et al., 2017).

As a result of the comparative values, it was found that the standard LSTM used as a baseline showed competitive performance on all three datasets. For the variants NIG, NOG, NIAF, CIFG, NP and FRG, no significantly better performance values of a particular variant were evident in the direct comparison. Removing the forget gate (NFG) or the output activation function (NOAF), on the other hand, led to a significant deterioration in performance. It should also be mentioned that the connection of the input gate with the forget gate (CIFG) led to a significant simplification of the LSTM network, without leading to worsened performance values in terms of accuracy of the network. For complex tasks, it is conceivable that this variant could outperform due to its simpler structure (Greff, et al., 2017).

LSTM is reported as the most common form of RNNs. Gated Recurrent Unit (GRU) Neural Network, as a more recent optimisation form for RNN, is becoming increasingly popular (Jozefowicz, et al., 2015). In the paper *An Empirical Exploration of Recurrent Network Architectures* the motivation of the authors was described as searching for a novel architecture that outperforms both LSTM and GRU. The authors pose the fundamental question of whether LSTM and GRU are indeed the perfect architectures, or whether there are unknown architectures that perform significantly better. In addition, the question was raised whether the existing architecture of an LSTM is already optimal and whether all components of an LSTM are actually necessary. In addition to new architectures, experiments were also conducted with LSTM variants in which parts of the LSTM were omitted (Jozefowicz, et al., 2015). Furthermore, an LSTM version was tested in which the forget gate was initialised with a bias of 1 (Gers, et al., 2000).

In the study conducted, a wide variety of RNN architectures were programmatically generated and tested (Jozefowicz, et al., 2015). Each variant generated in this way had to prove a required minimum quality in basic test runs. If this was achieved, more extensive test runs were carried out with three different datasets. The datasets addressed very different areas in which the performance of each variant was measured. One dataset included numeric values and arithmetic tasks. Another dataset included XML fragments that needed to be completed correctly by the RNN. The third dataset contained polyphonic music data sets that had to be transcribed. In total, more than ten thousand RNN architectures were generated and tested in this test. Based on the results, a top 100 list was created, which lists the best variants with the respective measurement results in the three different datasets. In addition, different variants of an LSTM were tested, as well as a GRU (Jozefowicz, et al., 2015). The LSTM variants included a standard LSTM (Gers, et al., 1999), an LSTM-f without forget gate, an LSTM-i without input gate, an LSTM-o without output gate, and an LSTM-b where the bias of the forget gate was initialised to 1. The architecture of the tested GRU corresponded to the currently common structure (Cho, et al., 2014a), (Chung, et al., 2014).

All five variants of the LSTM and the GRU achieved top rankings in the top 100 list. The top three ranked architectures besides the LSTMs and the GRU showed comparable results with strengths and weaknesses in different areas. Notably, these top 3 architectures were all very similar to the GRU. The comparison of the different LSTM variants showed that the forget gate and the input gate are indispensable components of an LSTM. Removing the output gate, on the other hand, had little influence on the measurement results (Jozefowicz, et al., 2015). These results were also confirmed in other studies (Greff, et al., 2017). It was particularly emphasised that initialising the bias of the forget gate with a positive bias of 1 or greater significantly improved the training performance of the LSTM. In general, this should be done every time an LSTM is used (Jozefowicz, et al., 2015), (Gers, et al., 2000).

To obtain a temporal context in data, the bidirectional approach is already known from standard RNNs. In the bidirectional approach one network goes through data from the beginning to the end (standard forward path) of all network layers and another goes backwards through all the layers from the end to the beginning (backward path). The result at a certain point in time is always determined from the output of both networks (Schuster, et al., 1997). If one uses an LSTM in this form instead of an RNN, the ability of the network to establish connections in data over a long period of time is added. The backward path through the network layers means that not only information from the past but also from the future will be considered (Graves, et al., 2013b). Bi-LSTM are therefore always used when the greatest possible context is required. Natural Language Processing (NLP) represents such an area of application. Without the backward pass, models applied to Text and Speech Recognition or Named Entity Recognition (NER) have a limited context that often makes it impossible to solve the task at hand (Graves, et al., 2013b). A typical application example is the completion of cloze texts, where often not only the beginning of the sentence but also the end of the sentence after the cloze must be known in order to determine the missing word. Figure 14 shows such an example.

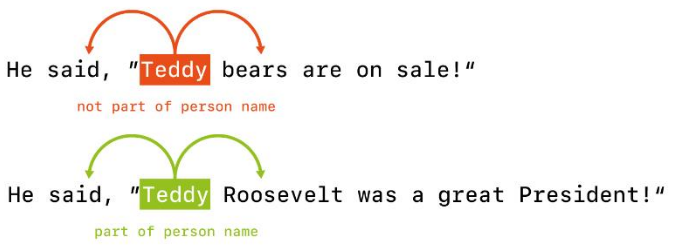

He said, "Teddy bears are on sale!"
not part of person name
He said, "Teddy Roosevelt was a great President!"
part of person name

Figure 14 - Cloze text example, taken from (Aggarwal, 2019)

In the field of NER, Chiu et al. (2016) also uses a Bi-LSTM to recognise proper names and to classify them (e.g. person, company, ...). In the paper *Named Entity Recognition with Bidirectional LSTM-CNNs* he described the application and highlights the advantages of the extended context of a Bi-LSTM on the basis of extensive test series (Chiu, et al., 2016). His work is based on a series of tests using a standard LSTM for NER (Hammerton, 2003), with the results of which he compares his approach. Another practical use case of Bi-LSTM is described in the paper *Predicting the Objective and Priority of Issue Reports in Software Repositories*. Here, descriptions of issues are used to make a multiclass classification into different categories (bug, improvement, ...) and predict their importance. Here, too, the Bi-LSTM shows better results than a standard LSTM due to its extended context (Izadi, et al., 2021). Besides NLP and NER, anomaly detection is also one of the known application areas of Bi-LSTM (Lee, et al., 2021).

A major limitation for the application of Bi-LSTM is that the entire dataset must be known. This is usually the case while training. However, if a trained network is fed with data streams in which data arrive sequentially on a continuous basis, only data up to the current point in time are available. Because of this, Bi-LSTM cannot be used on streaming data for PdM applications in IIoT (Chiu, et al., 2016).

Another optimisation approach found in literature is to connect the Input Gate and the Forget Gate. The forget gate usually receives the inverted value of the input gate (f=1-i with f=forget gate and i=input gate and sigmoid as activation function) and is often referred to as Coupled Input and Forget Gate (CIFG) (van der Westhuizen, et al., 2018). CIFG shows competitive performance in some special application areas of NLP or NER. However, no general advantages are attributed to the CIFG, although in many cases it does not worsen the results compared to the standard LSTM (Greff, et al., 2017).

Another optimisation approach groups the cells of the layers of a multi-layer LSTM (stacked LSTM), in order to execute them at different frequencies (Fernandez, et al., 2007), (Graves, et al., 2009), (Graves, et al., 2013a). The grouping criterion is the duration of the temporal dependencies that the cells are to process. The groups formed are assigned a repeat rate, which determines at which periods the cells of this group are active. In recurrent processing, not all but only the cells of the network assigned to the current period are processed in one processing step. For example, the cells of a group with a repeat rate of 2 are only processed in every second period. This is also referred to as operating subsets of cells at different rates (Koutník, et al., 2014). This significantly reduces the overall processing effort. Subsets that tend to remember long-term relationships are operated at a low speed, subsets for short-term relationships at a high speed. This optimisation approach is called Clockwork RNN (CW-RNN) (Koutník, et al., 2014). In the paper *A Clockwork RNN* this setup is described and tested using a series of tests with audio signals for recognising spoken words. In this application scenario, better results were obtained than with an RNN and LSTM. By reducing the number of processing steps per period, a significant performance gain was also observed. With a minimum repeat rate greater than 1, it was also possible to eliminate noise in data and thus implicitly add a kind of low-pass filter (Koutník, et al., 2014). A similar method, in which parts of the RNN network are omitted in a processing step, is the Temporal Kernel Recurrent Neural Network (TK-RNN) (Sutskever, et al., 2009). As with CW-RNN, parts of the network are executed in different frequencies. Unlike the CW-RNN, the TK-RNN controls the temporal clocking via an additional connection of each neuron to itself. These connections have a decreasing weighting over time. Improvements in performance have also been achieved with the TK-RNN. The TK-RNN's ability to handle dependencies over longer periods of time is reported to be roughly equivalent to the standard LSTM. However, compared to the CW-RNN, the TK-RNN creates more complexity in the network and thus overhead (Sutskever, et al., 2009).

Another approach in which activations and thus updates of the inner cells are controlled by a clock at a certain frequency is phased LSTM (Neil, et al., 2016). In CW-RNN and TK-RNN, the inner clock is there to make updates to the inner memory in relation to dependencies in data over time. For the periods in between, when there is no relevant information in data for a particular effect, the relevant cells remain inactive. This reduces the overhead of the network at runtime and reduces typical problems in training very long-term relationships, such as vanishing or exploding gradients (Koutník, et al., 2014), (Sutskever, et al., 2009). The phased LSTM approach is particularly aimed at application areas in which continuous data streams are processed. This type of data is especially found in sensor data in industrial applications and IIoT. If a network receives information from many different sensors, these sensors typically deliver their data in different frequencies based on their sampling rate. In a standard LSTM, all inner units are updated every cycle. A standard LSTM must therefore set its processing cycle to the shortest data cycle and thus to the highest frequency. This results in many unnecessary processing steps. CW-RNN and TK-RNN improve this by being able to classify neurons into groups of different speeds. A current trend, especially in modern IIoT environments, is event-based communication. Events are usually data-driven. Sensors only deliver information when there is a reason to do so. In contrast to the permanent transmission of the sensor signal in a fixed frequency, the communication effort is significantly reduced with the event-based approach. In order to be able to decide which data might be of interest, when and for whom, sensors need intelligence, context and self-sufficiency (Uffelmann, et al., 2021). Phased LSTM is not only able to form groups of different speeds, but also to handle individual asynchrony in data. For this purpose, the phased LSTM contains an additional time gate. Through the time gate, the internal memory of the cell is cyclically updated, whereby the cell is able to learn the asynchronous behaviour in data (Neil, et al., 2016). In test series with synthetically generated asynchronous sensor data, it was demonstrated that the phased LSTM is able to adapt to the different temporal behaviour of data in a data stream at a very fine granular level. The tests showed a 95% reduction in the number of internal processing steps at runtime for a phased LSTM cell compared to a standard LSTM cell, given optimal data structures. Despite this dramatic reduction, the accuracy of phased LSTM was often even better compared to standard LSTM (Neil, et al., 2016). The reduced number of internal updates further improves the ability of the phased LSTM to handle very long relationships in data. The effect here is roughly comparable to the application of the concept of leaky integrator neurons, which is used for example in Structurally Constrained Recurrent Network (SCRN). The optimisation here refers to the structure of the deep network and not to a single cell (Mikolov, et al., 2015).

In the optimisation approaches listed so far, gated connections were often introduced within a cell of an RNN. These connections and gates served to remember and forget information and thus to manage the internal memory (Gers, et al., 1999), (Hochreiter, et al., 1997), (Graves, et al., 2005), (Greff, et al., 2017). In addition, such internal connections were introduced in order to be able to include information stored by the cell in its current processing step. Peephole connections represent an example of such connections (Gers, et al., 2000). Connecting cells across layers of a deep neural network is another approach that has led to different optimisation models. Here, the focus is not on a single cell and its internal structure, but on a network of many stacked and interwoven cells (deep neural network) (Fernandez, et al., 2007), (Graves, et al., 2009), (Graves, et al., 2013a). In a so called Depth-Gated LSTM (DGLSTM), the cell state of one LSTM is connected to the cell state of the LSTM above it in the network. The connection has a gate and optionally an additional activation function (Yao, et al., 2015). With these connections, relevant long-term information is stored not only within the cells, but also exchanged across the layers of a network. If a standard LSTM only receives the output of the underlying layer as input ($\mathbf{x_t^{L+1}} = \mathbf{h_t^L}$, with x=input and h=hidden state at time t and layer L), a DGLSTM additionally receives memory information of the underlying layer ($\mathbf{c_t^L}$ **->** $\mathbf{c_t^{L+1}}$, where "->" stands for a gated connection with optional activation function) (Yao, et al., 2015). In the paper Depth-Gated LSTM, a DGLSTM is compared with an LSTM and a GRU. One of the test series was done with Chinese text to be translated into English. A comparison test annotated the syntactic structure of a text (Penn TreeBank). In both test series, the DGLSTM outperformed the other two models (Yao, et al., 2015).

The Highway Network approach is similar to the DGLSTM. With the Highway Network, in addition to the connection to the memory cell, another connection is established between the output of the cell below and the input of the cell above. Test series with image data have shown that the highway network approach can be advantageous for very deep neural networks (Srivastava, et al., 2015). A similar approach is also being pursued by Kalchbrenner et al. (2015) with the Grid Long Short-Term Memory (GrLSTM) (Kalchbrenner, et al., 2015). In contrast to the aforementioned approaches with stacked LSTMs, a multidimensional arrangement of the cells is chosen here in order to achieve a deep architecture. The approach is thus based on the model of the Multi-Dimensional LSTM (MD-LSTM) (Stollenga, et al., 2015) and extends it, as with the DGLSTM and GrLSTM, by several gated connections. The gated connections also create additional connections across the layers of the network and thus strengthen the network's ability to train and process spatiotemporal dimensions in data. Tests with algorithmic tasks and language translations showed better results than with a standard LSTM in the cases tested. Tests with sensor data were not carried out in the papers mentioned (Srivastava, et al., 2015).

With the so-called Structurally Constrained Recurrent Network (SCRN), Mikolov et al. (2015) propose an optimisation approach based on a simple RNN (SRN) that can nevertheless handle data streams with long term dependencies. Due to the much simpler structure of an SRN compared to an LSTM, the complexity of the resulting network is reduced. This is particularly advantageous in cases where the size of a network or the available hardware resources are limited. With an SRN, significantly more complex tasks can be learned and processed in such cases than is the case with an LSTM. In order to be able to use an SRN for dealing with longer-term contexts in data, it is extended by a second hidden layer. This second layer is called the context layer (Mikolov, et al., 2015). Since the hidden layer is always completely updated with each recurring processing step in an SRN, it is only able to remember dependencies over a maximum of 5-10 processing steps (Gers, et al., 1999). Thus, it only remembers the short-term dependencies in data. The context layer added in SCRN, on the other hand, is updated much less frequently and is thus able to store long-term dependencies. The idea of SCRN is based on the concept of leaky integrator neurons. The SCRM is significantly less complex and thus easier to handle and faster to process (Mikolov, et al., 2015). In an empirical comparison with data from the field of language modelling, similarly results were achieved with SCRN than with LSTM. This approach also achieves the effect of being able to store long-term correlations through slower or weaker updates of the hidden layer (Jaeger, et al., 2007). For a comparison with the SCRN, test results of a test series were used in which an SRN was extended by leaky integrations (Bengio, et al., 2012). The comparison test with the same data confirmed significantly better results for the SCRN (Mikolov, et al., 2015).

In his paper *The unreasonable Effectiveness of the Forget Gate* van der Westhuizen et al. (2018) mentions that many existing optimisation approaches for LSTM further improve the properties of an LSTM, but in doing so again significantly increase the complexity of the inner structure of the cells (van der Westhuizen, et al., 2018). In this category, he includes the Tensorized LSTM (He, et al., 2017), the concept of Sequential Neural Models with Stochastic Layers (Fraccaro, et al., 2016), a mechanism called Zoneout to influence activations (Krueger, et al., 2017), or the application of variational methods for RNN and LSTM (Graves, 2011), (van der Westhuizen, et al., 2018). Since performance aspects are in the foreground in this thesis, approaches that lead to higher complexity will not be explored further.

In addition to optimising the internal architecture of LSTMs, some performance optimisation approaches focus on arranging and processing neural networks in parallel. In this case, the input data is divided into several subsets in a preliminary step. Each network receives only one subset at a time. The networks are processed in parallel. The results are combined in a final aggregation layer.

One frequent application of this approach is the processing of video streams for activity recognition. The video data is divided into different classes. Video data is split according to spatial and temporal information in the image data and forms two input classes from this. The spatial data contain the actual image data in the classical sense. The temporal data includes the optical flow of the video data, that is the differences in value between the individual pixels of successive video frames. After data has been divided into spatial and temporal information, it is then loaded into two separate, parallel neural networks. Particularly with video data, which is usually very extensive and also represents a continuous stream of data, high-performance processing is crucial in many applications. The approach is usually referred to as two-way deep neural networks. Significant performance improvements are reported by the application of this approach (Jin, et al., 2022), (Ertan, 2021), (Bouaziz, et al., 2017), (Zhang, et al., 2017b), (Ding, et al., 2021). Another strategy of segmentation of video data divides the input vector into global and local image information. Global image information is image content that describes the overall context of the image or represents main components. Local image contents, on the other hand, are detailed aspects in the form of a subset of a global feature (Huang, et al., 2016). In the field of mobility and traffic forecasting, input data is often also segmented by location data (e.g. underground/bus/train stations) and context data on the mode of travel (e.g. walking, cycling, traveling by car or train) (Liang, et al., 2016), (Song, et al., 2016).

In addition to the field of image processing, Wang et. al 2019 describes the application of parallel LSTM for performance optimisation of forecasts for electric power generators and energy consumption. He also divides data into two groups, which are processed in parallel LSTMs. The first group represents the forecast data for decentralised energy producers (mainly wind power and photovoltaic). The second group models the interaction between source (energy producer) and load (energy demand) based on user-driven behavioural data (Wang, et al., 2019).

Because of its memory cells LSTM models are predestined if data comprises long-term dependencies. If data structure allows the separation of single entities with their specific behaviour as well as the formation of groups of entities, it could be then possible to process each entity and every group with its own neural network. This opens up parallel processing possibilities of the single neural networks. Normally each single and parallel processed neural network provides its result to an aggregation layer aggregating all outputs to an overall result. In his paper *A Hierarchical Deep Temporal Model for Group Activity Recognition* Ibrahim et al. (2016) describes how to recognize situations in a volleyball match. One LSTM model per player predicts the behaviour of this player, remembering his previous behaviour in the match (long-term dependencies). Each single situation of the match is then modelled as a group of the players. The LSTMs are hierarchically ordered where the LSTM models of all involved players are subordinated to a scene. The scenes and the players behaviour is extracted based on images using CNN (Ibrahim, et al., 2016). This basic idea of segmentation and parallel processing in less complex subnets inspired the development of the SlicedLSTM.

However, there is also criticism of the approach of splitting input data into multiple sub-packages for parallel processing. Also based on the use case of activity recognition in video data, Ma et al. (2018) state that by splitting the video data into spatial and temporal subsets and executing them in parallel networks, the correlations between data of the parallel networks are not taken into account. As a result, information is lost in data. In the present case, this concerns temporal information on the separated image information. In addition, it is criticised that current projects mostly use two-way Convolutional Neural Networks (TW-CNN). CNNs are not able to recognise long-term dependencies in data. This results in further disadvantages besides the loss of temporal correlations due to the segmentation of the video data. With continuous video data, the temporal sequences of events, as well as the knowledge of earlier events in the video stream, do have relevance and must be considered. Ma et al., (2018) therefore recommends the use of LSTM instead of CNN. By further segmenting the spatial data in terms of their temporal dimension into individual time buckets, the temporal references are brought back into the spatial data. The result is called Two-Stream Long Short-Term Memory (TS-LSTM). Like all two-way approaches mentioned here, the TS-LSTM does not represent a model optimisation. As with the aforementioned approaches, the optimisation also lies in the entire system structure and not in the neural network itself. With the TS-LSTM, the input data is also segmented manually or semi-automatically in a preliminary step in order to then be processed in parallel on standard neural networks (here LSTM) and finally aggregated (Ma, et al., 2018).

Goal	**Approach**	**Method**	**Reference**
Improve context and long-term dependencies	bidirectional processing	BI-LSTM	(Chiu, et al., 2016), (Graves, et al., 2013b), (Izadi, et al., 2021), (Lee, et al., 2021), (Chiu, et al., 2016)
	special cell/layer structures	Depth-Gated LSTM	(Yao, et al., 2015)
		GrLSTM	(Kalchbrenner, et al., 2015)
Reduction of the processing effort through grouping	only process required parts	Phased LSTM	(Neil, et al., 2016)
		SCRN	(Mikolov, et al., 2015)
		TK-RNN	(Sutskever, et al., 2009)
	special cell/layer structures	CW-RNN	(Koutník, et al., 2014)
		Multi-layer LSTM/Stacked LSTM	(Fernandez, et al., 2007)
	special layers	SCRN	(van der Westhuizen, et al., 2018)
Reduction of the processing effort in general	special layers	Zoneout	(Krueger, et al., 2017)
	split data set	TS-LSTM	(Ma, et al., 2018)
		TW-Deep Neural Networks	(Jin, et al., 2022), (Ertan, 2021), (Bouaziz, et al., 2017), (Zhang, et al., 2017b), (Ding, et al., 2021), (Huang, et al., 2016), (Liang, et al., 2016), (Song, et al., 2016), (Wang, et al., 2019)
Reduction of the processing effort by reduced model complexity	special cell/layer structures	Deep MD-LSTM/CNN	(Krizhevsky, et al., 2012), (Wang, et al., 2015), (Ciresan, et al., 2012a), (Liu, et al., 2014), (Byeon, et al., 2015)
		Multi-Dimensional LSTM	(Graves, et al., 2013a)
	special gating	grConv	(Cho, et al., 2014b)
		GRU	(Lu, et al., 2017), (Chung, et al., 2014)
		LSTM and Variants	(Greff, et al., 2017), (Jozefowicz, et al., 2015)

		RNN- and GRU-based Encoder-Decoder	(Cho, et al., 2014a)
Reduction of the processing effort by simplified net structure	special cell/layer structures	PyraMiD-LSTM	(Stollenga, et al., 2015), (Cardona, et al., 2010), (Mendrik, et al., 2015)
	special gating	CIFG	(van der Westhuizen, et al., 2018), (Greff, et al., 2017)
	special layers	SCRN	(Mikolov, et al., 2015)
		SNMSL	(Fraccaro, et al., 2016)
		Tensorized LSTM	(He, et al., 2017)

Table 4 - Summary of Optimisation Approaches for LSTM

All the approaches discussed in this section are summarized in Table 4. The approach, the applied methods and the according references are listed by the respective goals regarding the optimisation of an LSTM.

In summary, it can be said that the reduction of complexity is the basic idea in the majority of the papers presented here. Some of the papers presented here reduce complexity by splitting the data into independent subpackets, which then additionally favour their parallel processing. In other papers, the complexity is reduced by internal machanisms for switching off subnetworks for certain processing steps. The basic idea presented in those papers and the various approaches to implementing complexity reduction inspired the idea of SlicedLSTM.

4 Research Methodology

This chapter describes the selection of the research methodology used and its characteristics. For this thesis, a Mixed Method Research (MMR) approach with a sequential arrangement of a qualitative and a quantitative study is chosen. The qualitative study is used to get a close picture about the use of neural networks in PdM systems. The subsequent quantitative study aims to prove the runtime improvements archived by the novel model developed within this thesis compared to the established standard model in PdM applications. The design, setup, implementation and evaluation of the qualitative study is described in chapter 5. Section 6.3 addresses accordingly with the quantitative study within the MMR study design.

4.1 Fundamental Research Design

The motivation behind this research is to develop a novel approach of a neural network model that will better meet time-critical requirements in Predictive Maintenance (PdM) applications. This new model will be based on the model that experts consider to be the state of the art, or the standard model in current PdM applications. This standard model will be determined with the help of research. Additionally, the goal of the marked research is to find out which models are generally common in the field of Predictive Maintenance and IIoT environments and how their performance can be evaluated and compared. Using the strengths and weaknesses of the identified models, it is necessary to determine for which data and application areas these models were more or less suitable. In doing so, it is also necessary to consider the special requirements of data stream processing in IIoT environments. Based on the results of the research, the model that has proven to be the most suitable at present and probably also in the future will be used as the standard model for further consideration in this thesis. Most suitable in this context means, above all, that this model is the most popular and most frequently used model in current research and industrial projects. This most suitable model will then be used as a basis for further optimisation in this thesis. The determination of what constitutes an expert in the context of the qualitative study conducted in this thesis is explained in detail in section 5.1 (Selection of Experts).

Optimisation here refers exclusively to the modification or extension of the standard model (here synonymously also referred to as standard model) for software-side performance enhancement. Optimisations on the hardware side are not considered in this thesis. To receive significant and comparable results, it is necessary to determine a defined and limited application scenario, which will be considered exclusively. This will also to be determined through research in this thesis. Optimisation in the sense of the present thesis is then given, if the new model approach lead either to improved time-sensitive behaviour (reduced response times, reduced resource requirements) without affecting the quality of the results, or improved quality of the results with similar time behaviour. An important question to be determined in this thesis therefore is how to measure and evaluate the results of different models and approaches in a representative way. The evaluation of the new model in comparison to the standard model is done with the help of quantitative research as described later on.

The research methodology in this thesis therefore requires a two-step approach. The first step is to show, on the basis of expert opinions, which models are used in PdM applications and how their relevance is assessed by the experts. The task of qualitative research is to show possibilities, give insights and find ideas (Johnson, et al., 2008), (Lichtman, 2006). Qualitative research is therefore the right method for this step. The research methodologies used in this thesis are consistent with those of qualitative research. Thus, the intended study goals of identifying opportunities, providing insights and finding ideas coincide with those of qualitative research. The target groups of qualitative research methodology are smaller groups that are randomly selected. In the study conducted here, it is a group of eight people. They are referred to as experts. The definition of the term expert is given in section 5.1 of this thesis. Another characteristic of qualitative research methodology is that the studies focus on a set of issues as a whole and not on specific details. This is also in line with the requirements of the study conducted here, in which an overall picture of the use of neural networks in the environment of Predictive Maintenance systems is to be shown. One of the typical methods for data collection in qualitative studies is the collection in text form, audio or images. In the study within the scope of this thesis, the statements of the experts are documented on the basis of an interview guide and validated by means of an audio recording of the interview. The evaluation of the results of the interviews conducted in the context of this thesis is carried out by identifying interrelated topics and similar statements in the form of common motifs. This approach to data evaluation is consistent with the methodology of qualitative studies. Similarly, the documentation of the study findings in the form of a narrative report with contextual description and direct quotes from the study participants aligns with qualitative research methodology (Johnson, et al., 2008), (Lichtman, 2006). Table 5 summarises the features of the research methodology.

In the second step of the study design of this thesis, the performance of a newly developed model approach compared to the standard model will be evaluated. This has to be done with measurable, quantitative methods, taking into account the specified scope of application and the defined framework conditions. The tasks of a quantitative study are to prove hypotheses, to look at their cause and effect in detail and to make predictions (Johnson, et al., 2008), (Lichtman, 2006). In contrast to qualitative studies, quantitative studies collect data, for example in the form of numbers and statistics. In the study described here, these are specifically measured values and their progressions, which are recorded within the framework of the laboratory tests. The data collection method used in the study described here of collecting quantitative data based on precise measurements with structured and validated data collection instruments is in line with the requirements of a quantitative research methodology (Johnson, et al., 2008), (Lichtman, 2006).

The fundamental research design for the primary research of this thesis is therefore based on the Mixed Method Research (MMR) approach and is structured into both qualitative and quantitative research activities. MMR is defined as a method of research in which researchers can combine quantitative and qualitative techniques in a single study (Ngulube, 2015). For this work, the MMR approach is ideal as it requires both a qualitative study to determine what experts consider relevant and what is considered standard practice, and a quantitative analysis in the form of measurements of the results of a laboratory experiment. The MMR approach and its application in this thesis is described in more detail in the following sections 4.2, 4.3 and 4.4.

4.2 The Mixed Method Approach

Until the 1990s, quantitative and qualitative research processes were two opposing approaches within the scientific discussion as alternatives for action with incompatible paradigms. (Kuß, et al., 2012). The purists in both camps defended passionately their respective paradigm as being the only suitable primary research method. The focus has always been on the differences of the two methods (Ngulube, 2015). However, this somewhat uncompromising attitude has been changing for quite some time. For many years now, qualitative and quantitative approaches have no longer been seen as irreconcilable opposites that cannot be combined, but rather as complementary methods that both have their own strengths and specific relevance for certain areas of application (Creswell, 2014). The earlier emotional debates about the method have given way to a process of factual consideration and constructive engagement with the respective other approach (Hafsa, 2019).

As a result of these developments, the mixed methods research approach has come to the fore as a research procedure. Mixed method research is defined as a method of research in which researchers mix or combine quantitative and qualitative techniques, methods, approaches, concepts or languages within a single study. This represents an attempt to combine the strengths of both approaches and to offset the relevant weaknesses (Creswell, 2014), (Ngulube, 2015), (Hafsa, 2019). The following Table 5 shows a comparison of the qualitative and quantitative research methods based on different criteria. The content in Table 5 was summarized by Xavier University (Xavier Unversitiy Library , 2016) based on the work of Johnson and Lichtman (Johnson, et al., 2008), (Lichtman, 2006).

Criteria	**Qualitative Research**	**Quantitative Research**
Purpose	Show possibilities, provide insights, find ideas	Test hypotheses, look at cause & effect & make predictions.
Group Studied	Smaller & not randomly selected	Larger & randomly selected
Variables	Study of the whole, not variables.	Specific variables studied
Type of Data Collected	Words, images, or objects	Numbers and statistics
Form of Data Collected	Qualitative data such as open-ended responses, interviews, participant observations, field notes & reflections	Quantitative data based on precise measurements using structured & validated data-collection instruments
Type of Data Analysis	Identify patterns, features, themes, common motifs, e.g.	Identify statistical relationships
Objectivity, subjectivity and representativeness	Subjectivity is expected, psychological representativeness	Objectivity is critical, statistical representativeness
Results	Particular or specialized findings that are less generalizable	More generalizable findings that can be applied to other populations
Scientific Method	Exploratory or bottom–up, the researcher generates a new hypothesis from data collected	Confirmatory or top-down, the researcher tests the hypothesis and theory with data
View of Human Behaviour	Dynamic, situational, social and personal	Regular & predictable
Most Common Research Objectives	Explore, discover, & construct	Describe, explain, & predict
Focus	Wide-angle lens; examines the breadth and depth of phenomena.	Narrow-angle lens; tests specific hypotheses.
Final Report	Narrative report with contextual description and direct quotations from research participants	Statistical report

Table 5 - Qualitative versus quantitative research, summarized by (Xavier Unversitiy Library , 2016), originally taken from (Johnson, et al., 2008) and (Lichtman, 2006)

It is often postulated that the mixed method approach should be viewed as an independent approach in addition to purely qualitative and quantitative methods, and that, just as with the other paradigms, it represents a superior approach under certain framework conditions (Ngulube, 2015).

As described in the following in this section exploratory interviews are conducted as the qualitative research method within this thesis. Here, the focus is placed on an extensive and complete gathering of thematically-relevant information in the form of problem-oriented expert discussions. The qualitative research will then be followed by a subsequent quantitative evaluation in terms of laboratory tests and evaluations. In the sense of a mixed method approach, this leads to a sequential progression of the two methods. With regard to the underlying typology of MMR, the approach chosen corresponds to the procedure of conducting a qualitative study (QAL), which then forms the basis for a subsequent quantitative study (QUAN). This is also referred to as the QUAL=>QUAN approach (Ngulube, 2015). Figure 15 illustrates this.

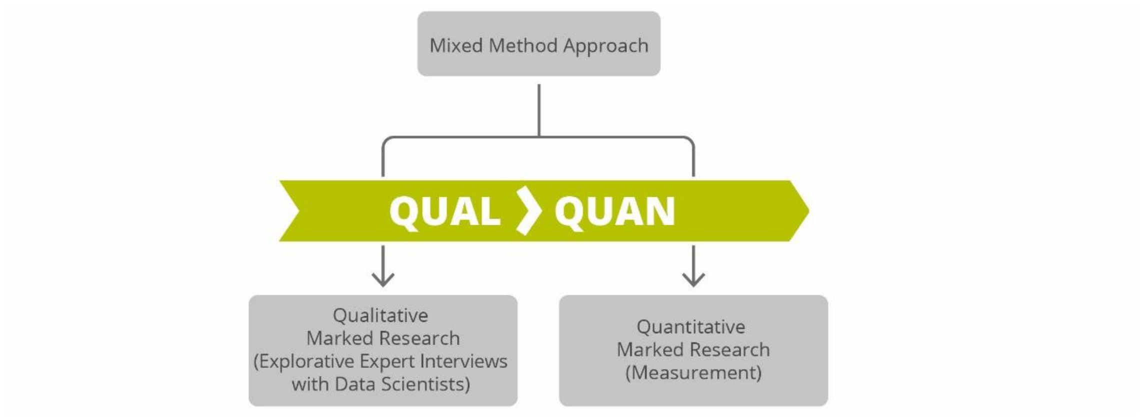

Mixed Method Approach
QUAL
QUAN
Qualitative Marked Research (Explorative Expert Interviews with Data Scientists)
Quantitative Marked Research (Measurement)

Figure 15 - Research design following the Mixed Method approach according to (Ngulube, 2015)

The selection of the QAL=>QAN approach in this thesis was based on a number of considerations. Apart from the conviction of the general strengths of the approach, thematically-specific requirements, like the ability to sequentially combine a qualitative study with a quantitative test, also spoke in favour of the mixed method approach. Thematically-specific difficulties, include novelty and complexity, result in the requirement of specific expert knowledge and the emergence of a high degree of freedom amongst expert opinion. In order not to influence the respondents in their opinion regarding the most suitable method, the respondents had to be given the greatest possible freedom to contribute their own knowledge and experience. The high information requirement of the subject area as well as the goal of identifying personal experience, induced an exploratory and qualitative approach with problem-centred interview techniques.

Another indicator for a qualitative approach was the fact that a large number of systems and software suites already exist in the field of predictive analytics and the higher-level discipline of machine learning (hereinafter referred to as tools). Due to the rapidly increasing popularity of predictive analytics in a variety of fields, such tools increasingly focus on user types hereinafter referred to as “business users”. Business users are not necessarily data science specialists who want to deal with the pure analytic methods and algorithms. Such users rather expect modern tools to feature comfortable and intuitive user interfaces, wizards and automated help functions which, for example, automatically recommend suitable methods on the basis of data or apply them implicitly (MacLennan, 2012). Results are processed by such tools automatically and presented in easy-to-understand forms, most often graphs. Business users are therefore often not aware of the predictive methods used under the bonnet of their tool, because they do not need this knowledge for their activities either. For this reason, it is necessary to determine the knowledge and practical experience of data science experts in this area by means of expert interviews. For the qualitative study within this thesis, business users were also admitted as experts, as they are not expected to have in-depth knowledge, but important information from practical application.

However, Häder (2015) warned against using the results of qualitative studies as the sole basis for analysis or decisions. Especially in the case of novel and complex topics, as well as highly heterogeneous target groups, Häder (2015) points out that the results of qualitative explorative research are usually difficult to generalise. This problem opposes the justified interest in obtaining an adequate picture of a sufficiently large group of persons (Häder, 2015). Although qualitative methods cannot replace a representative survey, they are very important in providing information as an upstream instrument. Qualitative research is thus an indispensable part. Without the characteristics and advantages of qualitative studies, many research projects would not be feasible. Quantitative studies, which build on qualitative studies, concretise explorative results and subject them to verification by a more representative whole (Häder, 2015).

It can be assumed that the so called group of business users usually cannot make reliable statements about the methods used, since the question of method selection does not usually arise with the tools used by this group. It can also be assumed that this group has little interest in the underlying methods, even if the selected methods are visible in their tool. In addition, Predictive Maintenance and streaming analytics is subject to further specific requirements and framework conditions, which are only known to experts in this industrial application field. From this argumentation, it can be deduced that a possibly planned subsequent quantitative study on a representative sample of well-trained experts would exceed the possibilities of this thesis due to the recruitment effort alone. How the experts interviewed in the course of the qualitative study were selected and recruited is described in section 5.1 of this thesis. How the content of the interviews was evaluated is described in detail in section 5.4.

Within the MMR approach, a distinction is often made as to whether more weight is given to the qualitative or the quantitative part in the study design. Figure 16 shows a representation of the three basic research methodologies as well as transitional forms between them (Johnson, et al., 2007). Here, the pure mixed approach represents the middle of the three basic variants. The purely qualitative and quantitative methods each form the opposite ends and are to be regarded as equivalent within a study design. Qualitative Mixed (or Qualitative Dominant) and Quantitative Mixed (or Quantitative Dominant) approaches are presented in between, as subtypes in the MMR study design. In these subtypes a certain method dominates the course of the study. In Qualitative Dominant Mixed Methods Research (QUAL+QUAN), a qualitative study forms the starting point. The subsequent quantitative study then provides supplementary data or confirms qualitative statements (Johnson, et al., 2007).

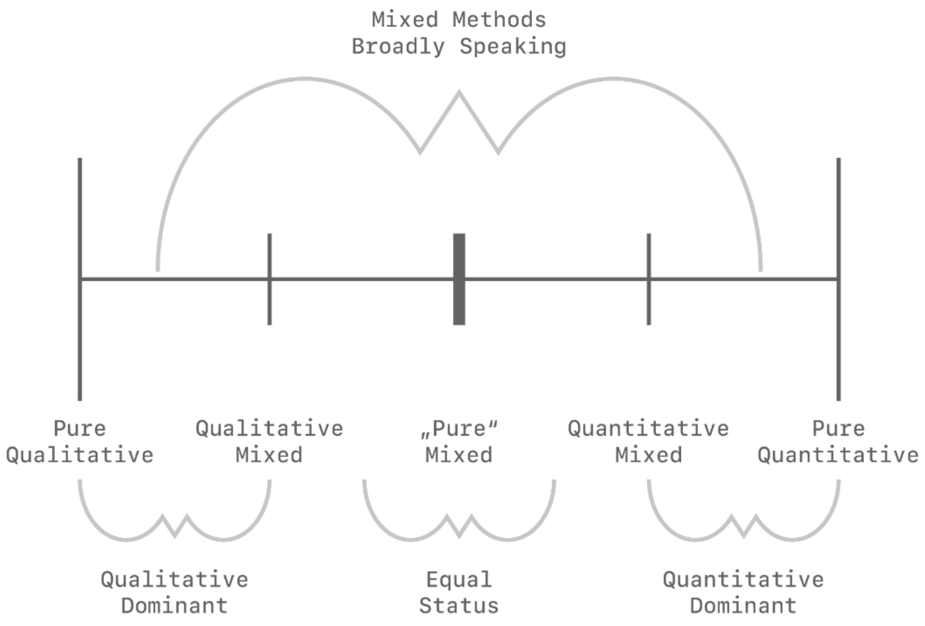


Figure 16 - Three Major Research Paradigms, Including Subtypes, taken from (Johnson, et al., 2007)

In this thesis, a qualitative study is conducted to find out the standard model in modern PdM applications for IIoT environments. A new model approach is then defined based on the standard model. In a subsequent quantitative study, the improvements in processing time of the new model are compared with the standard model. This quantitative study consists of measurements and evaluations in a laboratory environment. According to subtypes of MMR defined by Johnson et al. (2007), the study design of this thesis can be classified as Qualitative Dominant Mixed Methods Research (QUAL+QUAN).

A further classification into subtypes of MMR methods is made according to the order in which qualitative and quantitative methods are used. With the term Convergent Parallel Mixed Methods, Chreswell (2014) and Hafsa (2019) define a study design in which qualitative and quantitative methods are mixed up in one study. In contrast, conducting separate and sequential studies are referred to as sequential MMR approaches. The term Exploratory Sequential Mixed Method Research is used by Chreswell (2014) and Hafsa (2019) to describe a study design in which the opinions of the study participants are first ascertained using a qualitative study. The results of the qualitative study decisively define and structure the content of the subsequent quantitative study. Expert groups in particular are considered suitable as study participants. The results of the subsequent quantitative study is classified as often numerical in nature and taken from narrowly limited data, such as measurements and test results (Creswell, 2014), (Hafsa, 2019). The study design chosen in this thesis, using MMR in the form of a preceding qualitative expert study and a quantitative study based on it, is in line with the definition of Exploratory Sequential Mixed Method Research (ESMMR).

4.3 Expert Interviews as a Qualitative Procedure

According to Bogner et al. (2014), the expert interview is very popular as a qualitative survey tool within social research, even though, among the followers of qualitative research, it is not universally accepted as an independent acquisition method (Bogner, et al., 2014). This circumstance, inter alia makes it noticeable, that the expert interview is frequently mentioned only briefly, or not at all, in relevant basic literature. In particular, the analysis of such interviews is little dealt with. The literature discusses the question of how to gain appropriate access to the field and how to conduct an interview (Finkbeiner, 2016).

One reason for this is the fact that the expert interviews are comparatively less theoretically and methodologically examined and supported. There are different definitions for the concept according to experts. Furthermore, there is criticism that the interviewer often fails to meet the criteria of openness and not influencing the experts (Bogner, et al., 2014).

Some authors rate the slight methodological foundation as uncritical. Although methodological concepts can provide support for specific research projects, their specific implementation requires an adaptation of the methodological specifications to the respective research context (Kassner, et al., 2002). This results in highly divergent interaction structures during the interview, which is why it is impossible to develop a committed model for expert interviews (Bogner, et al., 2014). Instead, there are a large number of targeted approaches, which must primarily be adapted to the research context (Kassner, et al., 2002).

In the present case, the implementation of expert interviews was obvious for two reasons. On one hand, the aim of the qualitative marked research is to obtain a comprehensive picture of the opinions of experts. On the other hand, it is also important to gather different opinions and experiences from experts on Predictive Analytics and the specific requirements for methods in the application area of Predictive Maintenance (PdM) in IIoT environments. Furthermore, expert knowledge of the evaluation and validation of models is important. This knowledge will become very important for the subsequent evaluation.

In contrast to focus groups, the expert interviews allow different experts to be addressed in different ways and to develop guidelines from interview to interview based on feedback. The basic idea of expert interviews in qualitative research is to determine an idea or overall picture from the knowledge of interviewed experts, their opinions and hints. The expert interview is therefore not an instrument to develop generalisable statements (Flick, et al., 2004 S. 203-206). For this reason, small samples are also sufficient. The moment a repetition of a statement occurs in the form of an analogous statement, this is already sufficient for the formation of an idea or an overall picture. Group sizes of less than ten participants are common in expert interviews (Newington, et al., 2014). Apart from concerns regarding to the incompatibility of the various experts, the comparatively reduced organisational effort demonstrated the benefit of conducting expert interviews. In the study conducted as part of this thesis, a total of seven experts were interviewed, five of them from the group of experts and two from the group of professionals.

According to the convergent parallel mixed-methods research design chosen in this thesis, the qualitative study is the first step of the studies conducted separately and consecutively.

4.4 Experimental Study as a Quantitative Procedure

In the sections 4.1 and 4.2, the research methodology MMR and its subtype ESMMR were selected as the underlying research design. The qualitative study in chapter 5 develops a picture of how current PdM applications in IIoT environments are structured and which of the models can be called the standard. Building on the results of the qualitative study, a quantitative study will be conducted. Based on the standard model selected, a new model approach will be developed within the framework of this thesis. The aim of the new model approach is to improve the runtime behaviour and its suitability for time-critical PdM applications compared to the model defined as standard. The quantitative study serves to prove the potential for improvement achieved by the new model. Based on the research question RQ4 "*How can a new type of neural network be constructed so that it achieves measurably better runtime behaviour with comparable quality of the results achieved compared to the current standard model?"* (see section 1.2) two hypothesis are formulated to be proven by quantitative marked research:

- Hypothesis 1: If the input data of a neural cell is divided into several simpler sub-layers, the neural cell can process data faster overall despite the resulting overhead and with comparable quality of results.
- Hypothesis 2: If a neural cell has a deeper inner structure, this leads to less complex networks.

In order to be able to carry out a quantitative evaluation, information must either be directly available numerically or collected in such a way that it can be converted into a numerical representation. Already in the study design, care must be taken that all aspects that are to be part of the quantitative evaluation fulfil this criterion. Possible data collection methods are observations, measurements, self-report procedures (e.g. questionnaires) or tests. It is also recommended to use similar studies and their results as a comparison in order to avoid errors. The study design should at least answer the following questions: which collection method is used, how is the selected collection method applied and how are the results validated (Duckett, 2021).

4.4.1 Collection Method used

In this thesis, a newly developed model approach of an artificial neural network (hereinafter referred to as new Model) is to be compared with the model approach considered as standard (hereinafter referred to as Standard Model). The aim is to prove that the newly developed model requires shorter runtimes than the standard model. The new model should achieve the same level of quality of outcomes as the standard model. This is to be proven with the quantitative study. It can be clearly deduced that measurements are chosen as data collection method. The values to be measured are the runtimes and the achieved level of quality of outcomes of the two models to be compared. The exact definition of quality of outcomes of a model will be done in the definition of the experimental study in chapter 6.

4.4.2 Application of the Collection Method

Measuring the runtimes of software does not make any special demands on the measurement setup. The software components used and the underlying operating system provide established instruments for this purpose, such as time stamps. Ultimately, start and end times must be recorded under always the same conditions and the resulting runtime calculated.

4.4.3 Validation and Comparability of Results

The data collection method (measurements) and its application (time measurement via timestamp) are thus defined. Now it remains to be defined how the measured results are validated. Duckett (2021) summarizes the statements regarding the term validity made by Shadish et al. (2002) as follows: *"We use the term validity to refer to the immediate truth of a conclusion. When we say that something is valid, we are making a judgement about the extent to which the relevant evidence supports that conclusion as true or correct."* (Duckett, 2021 S. 457ff). The validity of a study and its results can be divided into the groups of internal validity, external validity, statistical conclusion validity and construct validity. Internal validity of a study refers to the validity of conclusions based on causal relationships between the experiment and its outcome. External validity of studies refers to the generalisability of the results of a study. This is often justified by their cause-effect relationship. Statistical conclusion validity refers to the validity of statements made about the results of an experiment. The construct validity of a study often refers to a psychological process or a characteristic of individual or group-related differences (Duckett, 2021). Based on this typification, the quantitative study carried out in the context of this thesis can be assigned to the type of statistical conclusion validity.

In the field of Deep Learning with artificial neural networks, no approach exists for measurement, evaluation and benchmarking which can be classified as standard and generally valid (Märtens, et al., 2019). In the case of classifications (binary classification is used in the laboratory test in this work) cross-validation methods are mainly used to evaluate performance, predictive ability and model accuracy. Cross-validation methods apply a so-called Train-Test-Split, in which data is divided into subsets for training and testing. There are various strategies for the split, such as simple-, stratified-, group-, leave-one-out- or n-fold-cross-validation, subsampling, the holdout-method and many more (Krawczyk, et al., 2015). However, cross-validation is only possible if validation data is available containing the classification results. In many projects, however, this is not the case, which is why synthetically generated data is often used. The vast majority of known and publicly available studies ultimately rely on the subjective judgement of the respective experts. Typical expert statements are "the model performed well under the conditions...", or "in many cases model A outperformed model B". There are very few attempts to define standards for the comparability and benchmarking of Deep Learning projects with artificial neural networks. One example is the OpenML Curated Classification Benchmarking Suite 2018 (OpenML-CC18). This platform initially started as an exchange platform where scientists can upload and share their projects (Bischl, et al., 2021 S. 10). The OpenML-CC18 platform was described by Bischl et al. (2021) as the first practical usable benchmarking suite (Bischl, et al., 2021 S. 1). There are no general valid standards for the comparability and benchmarking of Deep Learning projects with artificial neural networks (Krawczyk, et al., 2015), (Chung, et al., 2014), (Heng, et al., 2009). Heng et al. (2009) consider it one of the challenges of Deep Learning that must be overcome in order to be able to compare and evaluate different methods at all (Heng, et al., 2009).

On of the main hurdles is to find an appropriate layer configuration (in the following referred to as model setup) and the fact that an evaluation can only be made specifically for a certain use case with its specific data (Heng, et al., 2009). There are no fixed rules for the model selection and the setup of the dimensions of a network and its hyperparameters. According to the current state of the art, this is done empirically by the expert. Dimensions and hyperparameters, such as the learning rate, are simply determined through systematic trial and error and the experience of the data scientist (Märtens, et al., 2019).

In the experimental study defined here it is meant to use cross validation as validation method and a synthetically generated dataset comprising sufficient classification data. For each of the two model types to be compared a neural network with a layer configuration that is considered suitable has to be defined. The layer configuration determines the dimension of the inner layers of the neural network using either the newly developed model or the standard model as cells. In machine learning, it is state of the art that layer configurations are defined empirically by trial and error (Jeyakumar, et al., 2020).

Since it will probably not be possible to find a layer configuration that provides exactly the same conditions for both models, measurements should be carried out with several different configurations. In addition, several runs are to be carried out for each selected configuration. This should compensate for disturbing influences and identify outliers. The minimum required system processes of the operating system and the hardware of the test computer in particular are to be mentioned as disturbing influences.

The data used are decisive for the results of the study. Only with comprehensive and sufficiently classified data with regard to the outcome class can meaningful and robust results be achieved. As explained, data from industrial projects are hardly available and usually anonymised. Anonymising data means that it is no longer possible to draw conclusions about the quality of the classifications made. However, this is not considered a disadvantage for the study in this thesis. If the classification data do not correspond to the real conditions of the underlying plant or machine under consideration, they are still suitable for the relative model comparison. The use of a synthetically generated and anonymised data set and the associated lack of verifiability of the classification information given in data therefore does not impair the validity of this study. Another requirement of the study design is random allocation to experimental and control groups. In the present study, this is the division of data into test and validation datasets. This should be taken into account when selecting data for this study.

In order to obtain comparable values for the quality of the results achieved, the respective expected result of a processing is required. The data set used for the tests must therefore be classified with regard to the expected values. If the expected result is known, the deviation from the respective result of a model can be easily calculated. In the quantitative study of this thesis, a binary classification task is to be solved. In PdM, the result of a processing step is typically assigned to one of the two time groups *before or equal to a certain time* and *after a certain time*. In section 2.2 of this thesis, the *Early Warning Point* (Hagenberg, 2017) and the *Best time to do maintenance* (Peng, et al., 2010) were described as such times in the context of different maintenance strategies.

The two models compared in this thesis provide as a class or type of result that is a numerical value, indicating the number of cycles remaining until failure. This class of result is assigned to one of the two result subclasses RUL <= 30 cycles and RUL > 30 cycles. The quality of the results is to be measured in terms of classification accuracy. In the field of machine learning, classification accuracy refers to a special type of quality measurement of classification tasks. Classification accuracy is calculated as the percentage of predictions made correctly by the model in relation to the total number of predictions made (Bu, et al., 2017), (Pranckevicius, et al., 2017).

$$accuracy = \frac{number\ of\ correct\ predictions}{total\ number\ of\ predictions}$$

However, there is also criticism of using classification accuracy alone as a percentage value for quality. In binary classifications, an evaluation of the classification results according to the four classes True Positive (TP), True Negative (TN), False Positive (FP) and False Negative (FN) is also common. Positive stands for the positive class defined in the binary classification (1), negative for the negative class (0). True and false indicate whether the respective class was predicted correctly or not. In this form, the prediction made by the model is not only evaluated as correct or incorrect. It is also determined in which of the two result classes how many correct or incorrect predictions lie. The evaluation in this procedure is usually carried out using a confusion matrix (Sokolova, et al., 2009)

$$accuracy = \frac{(\mathrm{TP} + \mathrm{TN})}{(\mathrm{TP} + \mathrm{FP} + \mathrm{FN} + \mathrm{TN})}$$

In addition to accuracy, other key figures such as precision can be determined with this form of evaluation. Precision is the number of TP instances in relation to all positively labelled instances:

$$precision = \frac{TP}{TP + FP}$$

However, determining the accuracy according to TP, TN, FP, FN only offers a better evaluation of the results if the result classes are unevenly distributed in data (no balanced dataset) and if the negative results FP and FN cause different effects in the underlying use case. Otherwise, the determination of the classification accuracy in the form of the percentage value is sufficient for a model evaluation (Sokolova, et al., 2009), (Bu, et al., 2017), (Pranckevicius, et al., 2017).

In this thesis, a standard model is to be compared with a newly developed model. Initially, we will only compare the accuracy of the predictions made by the models. The effect of a FP or FN prediction is not the subject of the investigation and depends on the application and the respective data. For the investigations within the scope of this thesis, therefore, only the simple classification accuracy in the form of the percentage of predictions made correctly by the model in relation to the total number of predictions made is used and considered to be a sufficiently comparable measure for the model comparison and evaluation. In the following, the term accuracy or classification accuracy will exclusively be used to refer to the simple classification accuracy of the binary classification task determined as a percentage.

Why the way in which the measurements of runtime and accuracy are considered valid and comparable under the conditions of the study design has thus been explained. It should be mentioned that the quantitative study aims to compare the newly developed model with the standard model. The comparison of the two models therefore has only relative significance. Absolute or generally valid values are not achieved within the framework of the study conducted in this thesis. When designing the quantitative study, it is therefore a priority to ensure that all comparisons are carried out under the same conditions (data, study setup, environment).

5 PdM in IIoT- a qualitative primary Research

This chapter describes the definition, preparation, implementation and evaluation of the qualitative study. The following steps of the study are described in detail below: Selection of Experts, creation of the Interview Guidelines, conducting the Interviews, evaluation of the Interviews and the summary of the results and implications. The Appendix chapter A contains all materials used for the preparation and implementation of the study.

The aim of this qualitative study is to obtain a comprehensive picture of predictive systems and PdM in the field of IIoT. The currently most relevant neural models for predictive systems and their application and framework conditions are to be found out on the basis of expert opinions. The results of the qualitative study form the basis for the further steps in this thesis. A new type of model will be developed on the basis of the insights and results gained here. This new type of model is intended to achieve increased performance at runtime, with the accuracy of the results being as comparable as possible. In the sense of the selected MMR approach and its subcategory Qualitative Dominant Mixed Methods Research (QUAL+QUAN) and Exploratory Sequential Mixed Method Research, proof of the effect of the novel model approach will be provided in the form of a subsequent quantitative study.

5.1 Selection of Experts

The selection of experts is highly relevant for qualitative surveys (Glende, et al., 2011). First of all, it is important to clarify who can be regarded as an expert within the framework of the respective research context. The most extensive definition of experts is based on the evidence that every human being is an expert in coping with his individual daily life (Bogner, et al., 2014). In this sense, for example, chronically ill patients can be regarded as experts on their disease since they have a wealth of experience (Silverman, 2016). According to Bogner et al. (2014), an expert has three different types of knowledge:

- Technical knowledge in the sense of specialist knowledge
- Process knowledge, which is acquired from practical experience in the context of his professional activities
- Interpretative knowledge in the sense of interpretation and explanation patterns

(Bogner, et al., 2014)

Experts can be the immediate target group of the study or represent a complementary action unit for the target group. In the latter case, the experts have the function of providing information on the action context of the affected target group (Finkbeiner, 2016). They may also refer to other relevant interlocutors with similar or competing opinions and provide contacts. Such experts also serve as nucleation points of insider knowledge and are interviewed in place of a large number of actors to be questioned (Bogner, et al., 2014). In this thesis, the persons described as ”business users” should have represented the complimentary user group for data scientists. However, business users do not meet the listed criteria of an expert. Therefore, in the present study, persons generally designated as experts were divided into two target groups, described in the following. Although those two target groups are less complementary to each other, they still have different views on the problem and meet the requirements of an expert.

Experts from the field of research and teaching functioned as one target group of experts in the survey. To distinguish these persons linguistically, this target group is referred to as “specialists” in the present study. Persons of this target group deal intensively with the relevant topics in this area. This target group is employed at colleges and universities, scientific institutes and research facilities. The target group therefore deals with the topic from a more theoretical and scientific point of view. Usually, practical experience is also gained in the context of application-oriented projects and project co-operations with companies.

The individual persons of this target group have different focal points within the relevant knowledge area, which usually result from their professional background. Specialists with a focus on data science were able to provide comprehensive information on predictive analytics, methods and algorithms. Specialists with an IT background have provided the most basic knowledge on streaming analytics and real-time systems. Specialists, whose technical background is in mechanical engineering and systems engineering, in particular provided knowledge about Predictive Maintenance processes and approaches, as well as about general questions such as the relevance of real-time for maintenance processes. All in all, all specialists demonstrated extensive expertise.

When sampling specialists, various criteria were taken into consideration. Due to the limited time frame, willingness to participate and time availability in terms of "convenience sampling" were the decisive criteria (Creswell, 2014). Conducting interviews by means of a web conference facilitated scheduling. In addition, web conferencing eliminated possible disadvantages from the strong geographic distribution of the specialists. For the people who were assigned to the target group of professionals, it was assumed that they are used to participating in web conferences due to their professional activities and that they have the necessary equipment. Therefore, web conferences should not represent an additional barrier for these individuals.

Furthermore, attention was given to ensure, that interviewed specialists had already published scientific articles in the relevant area and therefore have appeared as experts.

In the survey, a second target group is represented by practising experts. By differentiation, persons in this target group were referred to as "professionals" in this study. People in this target group also deal intensively with the relevant topics in this area. This target group is employed in companies and is actively involved in the implementation of real projects. Projects can be carried out both internally and on behalf of customers (data science consultant). The target group therefore deals with the topic from a more practical and application-oriented perspective.

As with the specialists, the professionals focus on various key subjects within the relevant knowledge area and possess comprehensive knowledge in the entire subject area. The sampling of the professionals also had to take into account the criteria of limited time frame, willingness to participate and availability as well as the strong geographic distribution of the participants. The interviews were also performed via web conference.

The existence of the required knowledge and practical experience was initially accepted on the basis of the position of the respective person within their company. In addition, the interview guidelines adapted for this target group included additional questions to assess their knowledge and practical experience.

For the recruitment of the experts, the characteristics of the two groups of experts already listed in this section were used. For the group "specialists" it was defined that they typically work in the field of research and teaching. Their typical place of work is therefore colleges, universities, scientific institutes and research facilities. The group of "professionals", on the other hand, typically work as data scientists in IT departments of private companies. The entire recruitment of experts for the qualitative study was done through the author's network and his contacts with people in research institutions and private companies.

5.2 Interview Guidelines

The complete interview guide could be found in Appendix A.1. In the following the considerations that led to this interview guide are explained.

Usually, expert interviews are conducted as guideline interviews and represent a partially structured interview (Silverman, 2016). In this type of interview, the pre-formulated questions of the interview guidelines are posed in an arbitrary order. Subjects arising during the individual interview process can be recorded at any time and explored in more detail if necessary (Atteslander, et al., 2010). Because of their depth, the wide range of topics covered and the comparatively minor influence of the interviewee through the use of unstructured or semi-structured questions, guideline interviews provide an abundance of valid revelations and valuable information (Silverman, 2016).

One single interview guideline with two slightly different strands was prepared as the basis for conducting the expert interviews. Specifically, additional questions were introduced for the professionals target group, which were intended to question the knowledge and the practical experience of the interviewee. The complete guideline questionnaire can be found in the Appendix of this thesis.

The careful design of the interview guide served to provide important intellectual support for the possible direction of the interview. Furthermore, a properly elaborated interview guideline is important for the creation of the complex interaction processes between interviewer and interviewee (Lietz, 2010).

At the outset, the questionnaires contain simple questions concerning technical background, knowledge and experience in the topic area. This is followed by questions about the attitudes and opinions of the interviewee regarding the selection of suitable methods under different factors, as well as their assessment of the relevance of certain topics in the subject area. Basically, the questions asked develop from general to specific and are assigned to three logical blocks. Personal data gathered from interviewers included the stated job title and the self-chosen classification as a data scientist, IT specialist or technical specialist. This happened at the end of the survey.

In the literature there are many references to typical challenges in the formulation of appropriate questions (Nunan, et al., 2013). According to the references, a simple choice of words was used without foreign-language expressions or highly abstract terms. The questions were formulated neutrally, so as not to provoke any specific answers. “Yes”-“No” questions were not used, not even as filter questions. Open questions were preferably used to motivate respondents towards free, uninfluenced narrative. In this way, it was intended that professionals, in particular, would provide information about habitual actions from daily life. Specific technical questions were formulated as concisely as possible.

5.3 Conducting the Interviews

This section describes how the interviews were practically conducted.

Based on many degrees of freedom, guideline-based surveys place high demands on the interviewer who, depending on the direction of the interview, has to decide ad-hoc on the order of the questions and, if appropriate, about their omission. It is also the responsibility of the interviewer to decide when to return to the guidelines after any deviation. This requires a high degree of sensitivity to the respective interview process and to the interviewee. In addition, the interviewer is always faced with the dilemma of balancing between time constraints and the generally, relatively high interest in information (Creswell, 2014).

As a general survey strategy, an attitude of friendly assurance has established itself as a normal form to be aimed at (Atteslander, et al., 2010). Supporting comments are also permitted, while the interviewer refrains from expressions of opinion in order not to influence the interviewees (Atteslander, et al., 2010). The creation of a pleasant atmosphere, corresponding to that of an everyday conversation, is recommended instead of the interview-type character of an artificial interrogation (Deppermann, 2013). Since the interviews were carried out via web conference, it was always ensured that the interviewer's camera was switched on and that the interviewee can see the interviewer throughout the conversation. The interviewee, on the other hand, was free to turn his camera on or off. In addition, the interview was not started directly, instead the conversation was initiated with small talk. As the interviewees were accustomed to taking part in web conferences in their professional life, it is assumed that this form of communication does not introduce factors that influence the conversation. For persons who never or rarely participate in web conferences, this new situation would have to be taken into account during the interview.

Despite all efforts, distortions and influences during the interview cannot be completely ruled out. Therefore, critical reflection, systematic control, and transparent discussion of such effects are required. Interference effects are known, for example, in surveys of specific age groups or target groups with similar characteristics and usually occur independently of the type of questioning. In addition, problems caused by expert and technical vocabulary, the frequent lack of imagination, diminishing capacity to concentrate, the feeling of being overwhelmed, incipient time pressure or strong digressions are often amplified by the open nature of general questioning (De Vaus, 2013). In the present study, the target group consisted exclusively of experts who are constantly dealing intensively with the subject area in their professional activities and are also accustomed to situations of intensive conversation via web conference. It is therefore assumed, that no relevant distortions and influencing factors needed to be considered during the preparation, execution or evaluation of the interviews.

When conducting interviews, it is useful to reflect upon the interview guidelines after each individual survey. In essence, this only led to minor content-related changes and to the change in the order of two questions.

5.4 Evaluation of the Interviews

This section describes how the results of the interviews were elaborated on the basis of the information given by the experts. This section gives only a summary. The steps of evaluation are described in all details in Appendix A.2.

As already described, there are few recommendations in literature with regard to the systematic evaluation and generation of information from expert interviews. Finkbeiner (2016) offers a recommendation for a model for the analytical evaluation of content. The order of content within individual interviews in this model is irrelevant for the evaluation. Instead, the emphasis is on the search for similar thematic units (Finkbeiner, 2016). The approach in this study was based on this. First, a text document with relevant thematic blocks was prepared on the basis of the questionnaire, before the interviews were conducted. If new issues arose during the interviews, these were supplemented accordingly in the document.

All statements of the experts interviewed were recorded in writing by the Interviewer in the form of a protocol. This protocol was the basis for the subsequent evaluation. It was possible to make audio recordings of all interviews. All interviewed experts gave their consent to this in advance of the interview. The audio recordings were evaluated immediately after each interview and only served to verify and, if necessary, complete the interviewer's protocol. The audio recordings were thus only a tool to ensure that all statements made by the expert in the interview were completely and correctly recorded in writing. The use of audio recordings only as a supporting element is often used in expert interviews (Al-Yateem, 2012). For this reason, no audio transcription was carried out for the audio recordings. The verification of the interview transcript on the basis of the audio recordings took the form of the interviewer listening to the audio recordings several times after the interview and, if necessary, adjusting the interview protocol accordingly. In accordance with the conditions under which the experts had agreed to the audio recording, the audio recordings were deleted after verification by the interviewer.

Important passages from the interviews could be paraphrased or quoted verbatim and assigned to the previously defined thematic blocks, and transcribed. The respective author was colour coded. For time reasons a complete transcription was waived. A summary of the expert interviews and the expert specific answers can be found in the Appendix (see section A.2).

After this step had been completed for all interviews, the text passages were reorganized and reduced to their essential components through the abstraction of common motifs. The originator of the respective statements remained identifiable (see section A.2.1). Based on this, superordinate statements or findings on the respective thematic blocks were collected in note form (see section A.2.2). These superordinate statements formed the outline of the following evaluation section, in which the respective findings were formulated in detail.

5.5 Results and Implications

This section summarizes the results and implications taken from the study. In the following the stated results and implications are ordered according to the topics indicated by the questions of the interviews guide. The interview guide can be found in Appendix A.1. The cumulative results of the expert interviews and their detailed evaluation can be found in section A.2.

Regarding the main subjects of concern in the area of predictive analytics, stream processing and IIoT, the experts opinion can be summarizes as following. It was stated by nearly all experts, that the current projects using neural networks are more likely to be the first steps in the form of simple systems. Contrary to the external impression which is obtained from the topics I4.0 und IIoT, industrial projects are often only at the beginning. In most projects online data are not available yet. The main reason for this is the not adequately developed IT infrastructure in many areas. In addition to this, industrial customers are often missing an exact idea of what knowledge is contained in their data and how to explore and utilize it.

In current maintenance systems, the goal is usually the detection of errors and the rapid initiation of measures. This is often based on Condition Monitoring (CBM) and Complex Event Processing (CEP). In addition, data analytics is also carried out with the aim of determining optimization potentials in the plants.

Regarding the significance of time-criticality and stream processing as well as the methods and algorithms intended to be optimal for data streams and real-time processing, the experts opinion can be summarizes as following. The experts agree that the current situation, which has been described with reference to the previous question, will change rapidly with the ongoing establishment of I4.0 and IoT in the industrial sector (IIoT). Predictive and highly adaptive online systems that can process data streams in real-time will be standard in the opinion of experts in the near future. Predictive Maintenance (PdM) will become an important application in the area of Predictive Analytics. Autonomous transport and conveyor systems as well as self-organizing production plants require intensive communication with people (Human to Machine Communication H2M, Machine to Human Communication M2H) and machines (Machine to Machine Communication M2M). With increasing use of highly complex sensor technology and high sophisticated image processing and pattern perception, the amount of data increases significantly. The statements of the groups “Specialists” and “Professionals”, defined in the underlying study design (section 5.1) were almost identical in this question.

Consensus opinion of the experts was that in near future the ability of highly adaptive online systems to process data streams in some kind of real-time will be a key factor. Because of the not yet been sufficiently developed communication infrastructure in industrial environments and the lack of experience in the field of Predictive Analytics, this is not yet the focus. Due to the rapidly advancing development in the area of I4.0 and IIOT, time-sensitive systems will be the normal case in the near future.

According to the experts, in current projects with time-related performance requirements mostly hardware optimization and reduction techniques to reduce the amount of data to be processed are used. The design of algorithms and models do not focus on optimisation in terms of time-related performance and parallel processing. Here, there is still a lack of extensive knowledge and experience to make algorithms faster and less resource-intensive. It was also mentioned that the adaptive learning process must also happen with high performance and a minimum of computational resources needed. There were no group-specific differences in the statements of the experts from the "Specialists" and "Professionals" groups in the question of the high relevance of time-related performance and stream processing capability.

The question which methods and algorithms are used most frequently by the experts has largely been answered in a consistent manner by the fact that this depends on the application case. Therefore, no general statements can be made. Certain methods and algorithms are, in principle, better or worse suited for a particular application. However, the real applications are usually more complex, no general rule can be derived from this. Usually, in real applications, several methods are combined to achieve specific properties.

Some experts commented that they select algorithms according to how easy they are to use and how well their internal operations are understandable. The experts are aware of the fact that this does not lead to optimal results. Many real industrial projects are still at an early stage. Therefore, it is mostly about carving out the knowledge contained in data and developing first, simple prediction systems. As a popular simple algorithm, PCA has often been called, including its combination with reduction techniques such as sliding windows (SWPCA).

Methods from the field of Machine Learning (ML) and Artificial Intelligence (AI) are increasingly gaining acceptance, according to the opinion of the surveyed experts. Additionally, the opinion of the majority of the surveyed experts is that Deep Learning represents the key future procedures. An important feature of ANNs is the ability to create models with incomplete and partial non-existent expert knowledge. ANNs are therefore also suitable for analysing unknown data in an evaluation phase and for determining suitable methods. By permanent learning, adaptive systems are achieved, whose quality can be enriched with additional expert knowledge. Due to their structure, neural networks are also preferable for parallel and distributed processing. However, ANNs are rapidly become very complex and processor intensive. The combination with reduction techniques and filters may be advisable.

According to the expert opinions, further knowledge about the application of neural networks in environments with time-relevant performance requirements and on data streams would be required. In particular, models are needed that can deal with long-term dependencies in data without creating a constantly growing demand for resources. Simple algorithms such as PCA and its know variants (e.g. RPCA, EWPCA, KPCA) showing competitive performance but they are not able to handle long term dependencies. LSTM was mentioned by nearly all of the experts as the de facto standard neural network model in PdM applications. In the statements on the algorithms used, there were no fundamental differences between the surveyed groups “Specialist” and “Professionals”.

Regarding the question about the tools used, the experts were divided in two groups, with only one of the surveyed experts being assigned to the second group. The first group deals intensively with the underlying methods and algorithms. This group develops PdM systems individually for every application. For the realization of the algorithms and for mathematical functions mostly open-source libraries are used. Data science tools are only used by this group to visualize data at the beginning of the project and to determine initial findings from data. The second group uses high-level data science tools that severely limit or even obscure the choice of methods as well as all optimisation steps for the user. This second group could therefore not make any statement on the suitability of the different methods. It was established that the tool used (in the specific case ThingWorx Analyse of the company PTC Inc.) decisively uses ANNs for predictions. Since only one of the seven experts intensively uses tools, no conclusions can be drawn regarding the groups “Specialists” and ”Professionals”, defined in the underlying student design (section 5.1).

Regarding the question about the most important goals of Predictive Maintenance, the experts opinion can be summarizes as following. As the main aim of PdM applications, the experts have always referred to the reduction of costs (Life-Cycle-Cost and Total Cost of Ownership). From the experts opinion it is clear, that unscheduled failures cause the highest costs. The prediction of potential failures is therefore central to a PdM application. The exact strategy of a PdM system depends strongly on the application case and cannot generally be answered. Other objectives of PdM systems are to ensure reliable production, maximize asset utilization to optimise yields and minimize reject rate. Group specific differences in the statements of the experts from the groups "Specialists" and "Professionals" did not exist in the question of the objectives of a PdM system.

In the interview guide the topic of complexity was divided into four questions. One general question and three specific questions about the different complexity factors dimensionality, evolving data and non-linearity (see Appendix A.1). Regarding the question about the significance of complexity factors the experts stated the following. In general, all the experts corresponded that the influence of the complexity factors known from data streams are massive. Since many PdM systems are not well developed yet, complexity factors are usually ignored. The resulting errors must be tolerated. The use of reduction techniques and filters was mentioned in view of the fact that complexity is largely a result of volume and speed. Dimensionality is not a relevant factor, as PdM systems usually only contain one-dimensional data (sensor data). With the use of reduction techniques, the required response times are still achieved in classical systems having no special real-time capabilities. However, the actual problem, the lack of real-time capability of the algorithms, is not solved. According to the concurring opinion of the experts, PdM systems have to be scalable and elastic to high loads in the future.

In the opinion of the experts, the automatic recognition of concept drifts is not performed in current PdM systems. Non-linear data represent a known complexity factor in PdM systems. The modelling of non-linear relationships using physical models can be very complex. Therefore, this complexity is also mostly ignored in such projects. Non-linear regions are blanked, or simply linearized. If models from the field of neural networks are used, the nonlinearity is learned by the neural network, provided that sufficient training data is available. These statements are in line with the characteristics of the various complexities listed in section 2.5 as a basis and their effects on the area of application dealt with in this thesis.

Regarding the questions about the significance of historical data the opinions of the experts can be summarized as following. According to the expert opinions, the availability of historical data is very important. The historical data, if available and valid, contains the entire knowledge about the industrial plant. Historical data is usually available in real-world projects. However, this is often faulty and has little significance. In particular, there is usually a lack of explicit error cases (unbalanced data). A critical consideration of historical data is therefore advisable. All interviewed experts thought that historical data should also be included in predictions. Historical data are usually processed incrementally in order to keep the required computing power to a minimum. However, incremental processing can lead to a loss of quality and to errors. According to the experts, the processing of historical raw data is desirable. Adaptive models should ideally be trained with historical raw data.

Regarding the questions about adaptive learning the opinions of the experts can be summarized as following. Adaptive models and adaptive learning were described as an essential property by all interviewed experts. In PdM systems very complex structures have to be modelled. The models, which are mostly simple in their initial stages, have to be constantly improved by permanent learning and adaptation. Only in this way valid models can be developed. A permanent learning and adaptation process is complex in implementation and requires high computing power. In online systems, models must automatically adapt themselves permanently. Automatic learning and adaptation processes must be scalable and react elastic. Automatic adaptation requires methods from the field of Machine Learning (ML), but also expert knowledge.

Regarding the question about the validation of results from PdM systems, the interviewed experts agreed, that no generally valid validation methods and benchmarks exist. Because of the very different application cases, no general procedures can be defined according to expert opinions. In real projects, a subjective evaluation of the results by experts is always carried out. For an evaluation, the real data is usually not sufficient since in real world cases the number of representative error cases is usually insufficient.

Table 6 summarises the textual results and statements once again. The column "Group-specific differences" indicates whether there are differences in the statements of the two groups Specialists and Professionals.

Topic	**Main statements**	**Group-specific differences**
Approaches for PdM	• Use of neural networks is not very common • Most projects are in the initial phase • Despite IIoT, industrial projects are often only at the beginning • Online data is often not available • Adequately developed IT infrastructure is often missing • Exact idea of what knowledge is contained in data and how to explore and utilize it is missing	No
Predictive Methods	• Often based on CBM and CEP • Data analytics is also carried out to determining optimization potentials	No
Processing Data Streams	• Systems must be able to process data streams in defined time/resources • Situation is changing with the ongoing establishment of I4.0 and IIoT	No

	• Predictive and highly adaptive online systems will become standard • The use of neural networks is becoming the standard • Highly complex sensor technology will increase the amount of data significantly	
Time criticality	• Highly adaptive online systems on data streams will become a key factor • In I4.0 and IIOT, time-sensitive systems will be the normal case in the near future • Hardware optimization and reduction techniques are the current standard • No algorithms focusing on optimisation in terms of time-related performance and parallel processing yet • Adaptive learning need high performance systems	No
Methods and Algorithms	• Strongly depends on the application case • No general statements can be made • Algorithms often selected according to how easy and understandable they are to use • Machine Learning is increasingly gaining acceptance • Deep Learning represents the key future approach • Models needed that can deal with long-term dependencies in data without creating a constantly growing demand for resources • LSTM is the de facto standard of neural networks in PdM applications	No
Tools used	Group Specialists: • Deal intensively with the underlying methods and algorithms • Develop PdM systems individually for every application • Mostly open-source libraries are used • Data science tools only used to visualize data and determine initial findings Group Professionals: • Intensive use of high-level data science tools	Yes
Most important goals of Predictive Maintenance	• Reduction of costs (Life-Cycle-Cost, Total Cost of Ownership) • Unscheduled failures cause the highest costs • Prediction of potential failures is central • Strategy of a PdM system depends strongly on the application case and cannot generally be answered	No
Complexity	• Influence of the complexity factors known from data streams are massive • Largely a result of volume and speed	No

	• Since many PdM systems are not well developed yet, complexity factors are usually ignored • Use of reduction techniques and filters • Lack of real-time capability of the algorithms	
Historical Data	• Availability of historical data is very important • Contains the entire knowledge about the industrial plant • Often faulty • Normally there is a lack of explicit error cases no group-specific differences	No
Adaptive Learning	• Essential property for PdM systems • PdM systems are very complex • Models for PdM are mostly simple in their initial stages	No
Validation of results from PdM Systems	• No universally valid validation methods and benchmark • No general procedure can be defined, because of the very different application cases • In real projects, validation is often manually done by the experts	No

Table 6 - Summary of the results and statements of the qualitative study

As the explanations in this section and their summary in Table 6 show, the results of the expert interviews provided extensive information on the questions of the qualitative study. The individual opinions differ only slightly from one another. Overall, the qualitative research fulfilled the goals of identifying the important methods for the given area of application, allocating advantages and disadvantages and ordering them according to relevance by employing the opinions of the questioned experts. The qualitative research made an important contribution to the selection of the relevant methods for this thesis.

The order of the topics listed in this section corresponds to the order of the questions in the interview guide. Because the interviews were conducted openly, the interviewees were able to talk freely. Therefore, the order of the questions in the interviews differed and not all questions were discussed in each interview.

Overall, the experts confirmed that PdM is one of the important areas in IIoT and I4.0 environments and is still under development in practise. Beside the fact, that current implementations are mostly simple, because of a lack of knowledge and experience in many areas, they also confirmed, that in future scenarios methods and systems are essential that are able to cope with high complexity as well as time-related performance demands. To handle highly complex data and ensure adaptive and flexible behaviour, accompanied by the ability to deal with uncertainty, the surveyed experts agree that Deep

Learning approaches and specifically Artificial Neural Networks (ANN) represent the key future procedures. From the results of the expert interviews, it can be derived, that there is a need for deeper research in the area of the application of ANN in PdM systems with time-related performance demands. Among many other issues, an important point mentioned by the experts is how to make ANNs better and easier to use for PdM applications, especially when streams of complex data have to be processed in time.

6 Approach for a performance-optimised Model

The aim of this thesis is to improve the use of neural networks for PdM applications in the field of IIoT. In the context of this thesis, the targeted improvement refers exclusively to reducing the processing time required by a neural network while maintaining the same quality of results. The frequently used term performance in relation to neural networks usually refers to the accuracy of the results achieved. Less often, performance also involves the processing effort and time of neural networks (Akwensi, et al., 2021), (Wick, et al., 2019). In this thesis, the terms processing time and runtime are used instead of performance to avoid misunderstandings. The processing time or runtime of a neural network comprises two parts, the processing time of the forward pass and the backward pass. The forward pass describes the processing of input data into a result by a neural network. The backward pass describes the process of weight adaption through backpropagation of the calculated error, also named learning. On the basis of a result, the network is run through backwards and adapted to the expected result on the basis of a determined deviation (Wu, et al., 2018), (Hochreiter, et al., 1997). The focus of this thesis is on the forward pass, since the goal is to improve the processing time of time-sensitive applications at their runtime. In the context of this thesis, however, it has already been shown that adaptive and self-learning systems are increasingly used in modern PdM applications (Krawczyk, et al., 2015), (Hu, et al., 2014), (Beyer, et al., 1999), (Shaker, et al., 2013), (Lee, et al., 2014), (Zhang, et al., 2016a), (Zhang, et al., 2016b). The backward pass is therefore also relevant for the use of such systems, since the system not only learns in an upstream training period, but also during its use and therefore passes through the backward path. For this reason, the backward pass is also measured in the evaluation described below and used to assess the results.

In the context of this thesis, the LSTM was determined to be the most frequently used neural network for PdM applications in the area of IIoT. The determination was made through intensive literature research and on the basis of a qualitative study with expert interviews (see chapters 5 and 3). The LSTM is therefore used as a reference for the comparative measurements and as a starting point for the optimisations in the further course of this thesis. Based on a standard LSTM, a novel approach for an improved LSTM will be developed in the following. This is done by changing the internal architecture and the underlying algorithms of the standard LSTM. The newly developed approach is called Sliced Long Short-Term Memory (SlicedLSTM) Neural Network. Within the framework of a quantitative study, a laboratory prototype with suitable measurement and validation procedures will be used to prove that the SlicedLSTM can bring advantages in the processing time of PdM applications in the area of IIoT and outperforms a standard LSTM in certain application scenarios.

This chapter starts with a summary of the previous findings in the context of this thesis. Afterwards, the basic idea that led to the idea of the SlicedLSTM is explained. The architecture and the algorithm of the SlicedLSTM is then described in the form of theoretical considerations in detail. The research design for the quantitative study proving the effect of the SlicedLSTM is then described. A proof of concept experiment is then used to get initial indications whether the approach of the SlicedLSTM can improve the runtime behaviour of the standard LSTM. Section 6.3 describes in detail the setup of the experimental Study, the selection of the data set, as well as the conduction of the laboratory tests. This chapter closes with an evaluation and a summary of the results of the quantitative investigation.

6.1 Summary of previous Findings

From the results of literature research summarizes in chapter 3 and the qualitative study which was summarized in section 5.5 of this thesis, it can be derived that PdM systems in industrial environments are in an even less developed state as would be expected based on the topics I4.0 and IoT. The objective of PdM systems is ultimately always the reduction of costs. Failures and unscheduled downtime cause the highest costs. Predicting failures and scheduling maintenance and repairs based on predictions is intended to avoid such unscheduled downtime.

As described in section 2.2, industry frequently still lacks deep knowledge about its own data and on predictive procedures. Even though the degree of automation in industrial environments is very high, complete communication infrastructures from the field level to the higher-level IT and cloud systems is rare. The direct streaming of raw data from the field into an IT or cloud system is far from being standard. A common solution to this problem is to connect the process level equipment with edge gateways installed in the field. The edge gateways have independent connectivity, such as cellular, which they use to connect the data side of the process level with applications at a higher level. The data is typically transmitted in data streams. The recipients of the data are often cloud systems for analysing and diagnosing data (Wang, et al., 2019). The unlimited connectivity and autonomous action and communication of things on the principle of an IoT landscape are still rare in industrial environments. Barriers here are technical reasons, such as a lack of standardisation, as well as certainly also security concerns of the companies (Uffelmann, et al., 2021). The process towards flexible and globally networked production infrastructures requires a rethinking and releasing of well-known procedures. Terms like "digital transformation" and "disruptive technologies" are therefore often called in the context of I4.0 and (I)IoT. Accordingly, the Gartner Hype Cycle for Emerging Tech 2022 also includes many topics that play a central role in IIoT. These are in particular Autonomic Systems, Digital Twin and Asset Shell, as well as Industry Cloud Platforms and of course Machine Learning (Gartner, 2022). This level of development was described in section 2.2 of this thesis and substantiated by the results of the study summarised in section 5.5.

In future industrial applications, a complete interconnection with corresponding communication infrastructures will have to be given. Intelligent sensors and actuators will not only provide comprehensive data, but also have the ability to interact smart and fast. This was substantiated by the results of the related research stated in chapter 3 of this thesis, as well as the qualitative study which was summarized in section 5.5. Additionally the demand for time-sensitive processing of data streams was also derived from the theoretic background described in sections 2.2, 2.4 and 2.5.

Among the pure application data, self-diagnosis data will increasingly being added. This will provide even more sophisticated information on the condition of the plants and possible wearing and aging processes to data analytics systems. The communication has to be increasingly standardized to enable an unhindered interaction between different systems ad hoc. See sections 2.1 to 2.5 for detailed description. Emerging and pioneering standards like IO-Link can be mentioned here. IO-Link is a vendor independent communication standard for sensors and actuators. It provides ad-hoc connectivity and automatic setup functionality. The IO-link community claims to be the first standardized IO technology worldwide (Uffelmann, et al., 2021).

An additional finding of the qualitative study was that PdM applications are currently mostly developed offline (see section 5.5). In offline analysis, runtime behaviour is usually not the most important issue. However, the experts questioned in the qualitative study agree that this will change dramatically in near future. Responsive systems that enable fast and less resource-intensive calculations will become the standard.

6.2 The SlicedLSTM

This section explains the idea behind the Sliced Long Short-Term Memory (SlicedLSTM) Neural Network as a new model architecture. The SlicedLSTM is based on the standard LSTM architecture and aims to improve processing time by keeping the accuracy of the generated results at the same level as the standard LSTM. First, the basic idea that leads to the SlicedLSTM is explained, followed by a detailed theoretical description of the model architecture. An initial PoC is carried out to give an indication of whether or not an impact is being achieved.

6.2.1 Basic Idea

In the context of the related work of this thesis, different approaches for optimising a standard LSTM with regard to its processing time were listed (see section 3.4). Those approaches can be divided into two main categories with regard to the underlying approach (see also Table 4). The first category optimises the layers of deep networks and the connections of the LSTM cells between the layers. The second category optimises the cell architecture and thus the internal structure of a single LSTM cell.

The approaches designed to optimise the layers and their interconnection showed measurable reductions in the processing time required, especially with an approach so-called grouping. Grouping in this context means that parts of the deep neural network, i.e. certain cells across several layers, are combined into groups. A certain group does not necessarily have to be run through every time the network is processed. Depending on the context or certain parameters, groups can also skip processing steps. This is done, for example, if the current input data does not concern a certain subnetwork. One example for this is grouping according to frequencies of the input signals. If a deep neural network processes a continuous data stream containing sensor data of different frequencies, not all data is relevant or available in each data sample. For example, if all data is transmitted at the frequency of the signal with the highest frequency, data with lower frequencies will not provide new sensor data between their cycles. Such lower frequency data therefore needs to be processed through the network less frequently than high frequency data. As a result, the processing of certain parts of a network is not necessarily required for every processing time (sample). The complexity of deep neural networks can therefore be reduced at certain processing times. Lower complexity leads to a reduction in the required processing time. Examples of this approach are the Phased LSTM (Neil, et al., 2016), SCRN (Mikolov, et al., 2015), TK-RNN (Sutskever, et al., 2009), CW-RNN (Koutník, et al., 2014) and various variants based on multi-layer or stacked LSTM (Fernandez, et al., 2007).

The concept of grouping aims exclusively at reducing the complexity of deep networks. The behaviour for complexity reduction is either based on temporal dependencies (e.g. frequency-driven (Neil, et al., 2016)), or data-driven, i.e. for specific input data or scenarios (like event-based (Sutskever, et al., 2009) or relationship-based (Koutník, et al., 2014)). At runtime, only parts of the deep neural network are processed. However, the network in its entirety has the full depth and complexity that is required for processing the overall task. During training, the entire deep neural network must be trained with all aspects of the entire data spectrum. The mechanisms for managing different network parts or groups are added as an overhead. At training time, this generates more training overhead than with a standard setup. Also, this form of optimisation does not improve the deep neural network's concurrency capability (Neil, et al., 2016), (Mikolov, et al., 2015), (Sutskever, et al., 2009), (Koutník, et al., 2014), (Fernandez, et al., 2007), (van der Westhuizen, et al., 2018).

Another approach to optimising deep neural networks includes measures to simplify the network structure. Models listed in section 3.4 of this thesis that follow this approach are PyraMiD-LSTM (Stollenga, et al., 2015), (Cardona, et al., 2010), (Mendrik, et al., 2015), CIFG (van der Westhuizen, et al., 2018), (Greff, et al., 2017), SCRN (Mikolov, et al., 2015), SNMSL (Fraccaro, et al., 2016) and Tensorized LSTM (He, et al., 2017). While the grouping approaches omit parts of a deep network at runtime, these approaches aim at limiting the structure of the network to the most necessary already by design. For example, special gates and layers are used for this purpose. The general reduction of the complexity of the deep network then also leads to less processing effort during training. In addition, less complex networks are also easier to set up and train (Stollenga, et al., 2015), (Cardona, et al., 2010), (Mendrik, et al., 2015), (van der Westhuizen, et al., 2018), (Greff, et al., 2017), (Mikolov, et al., 2015), (Fraccaro, et al., 2016), (He, et al., 2017).

The second category of approaches, which was explained in section 3.4 of this thesis, optimises the architecture and thus the internal structure of a single LSTM cell to reduce complexity. However, the cell itself is optimised and not the structure of the connection of the layers to each other. In an LSTM cell, the inner structure consists of four dense layers/gates: the input gate, the output gate with its tanh gate and the forget gate. In addition, there is the cell state as an internal memory. The processing steps within a single LSTM cell therefore cause computational effort. Since a deep network contains a very large number of cells, optimisations on the cell add up (Lee, et al., 2020) (Liang, et al., 2019). Also, optimisation on a cell type aims at reducing complexity rather than its ability to be processed in parallel. In section 3.4 of this thesis, the following variants were considered, which follow this optimisation approach: GRU (Lu, et al., 2017), (Chung, et al., 2014), grConv (Cho, et al., 2014b), several generated LSTM Variants in which individual gates are missing (Greff, et al., 2017), (Jozefowicz, et al., 2015), RNN- and GRU-based Encoder-Decoder (Cho, et al., 2014a).

All approaches explained in this section and described in detail in section 3.4 show that the complexity of a network has a decisive influence on its processing time. For example, PyraMiD-LSTM reduces the number of computing operations required in relation to the number of input data from exponential to linear (Stollenga, et al., 2015), (Cardona, et al., 2010), (Mendrik, et al., 2015). Reduced complexity also results in further advantages, such as simpler setup and training, as well as reduced impact of training effects such as the vanishing or exploding gradient problem (Hochreiter, et al., 1997), (Gers, et al., 1999).

The aim of this thesis is to improve the processing time of an LSTM network by model optimisation on the inner structure of an LSTM cell. For this purpose, a novel approach of an inner cell structure is developed and compared with a standard LSTM. In the introductory chapters of this thesis, the improvement of the parallelisability of a neural network cell was stated as the basic idea. If the inner processes of such a cell can be processed concurrently, this does not result in less computational effort, but ideally in a shorter overall processing time (Asanovic, et al., 2006). The considerations of related work in chapter 3 and especially with regard to the LSTM in section 3.4, have shown that the reduction of complexity is attributed the highest optimisation potential. All the approaches listed aim at least at this. During the research for this thesis no approaches that aim to change the internal structure of a neural network cell to better support concurrency were found.

However, the fact that parallel processing can lead to reduced processing times can be seen from other project work, also discussed in section 3.4. By dividing the input data into smaller, independent data packets, data could be processed in parallel in neural networks. The resulting concurrent processing leads to shorter processing times. In addition, the individual neural networks became less complex, since they only have to be dimensioned for processing a part of the data. This approach is basically similar to the approach of grouping deep networks, which has already been presented in this section. Dividing data into sub-packages and processing them in sub-networks thus supports parallelisability and reduces complexity. Section 3.4 of this thesis describes projects that follow this approach. Projects to be mentioned are using TS-LSTM (Ma, et al., 2018), as well as TW Deep Neural Networks (Jin, et al., 2022), (Ertan, 2021), (Bouaziz, et al., 2017), (Zhang, et al., 2017b), (Ding, et al., 2021), (Huang, et al., 2016), (Liang, et al., 2016), (Song, et al., 2016), (Wang, et al., 2019). However, all these projects do not describe optimisation of a deep neural network or the neural cells used, but project-specific system setups. In these project setups, data is divided manually or semi-automatically according to specific procedures in a preceding processing step. The smaller data packets are then processed in standard networks in parallel. Finally, the results of the individual networks are then combined into an overall result in another part of the system. A neural network that already includes a segmentation of the input data and parallelisation capability in its internal architecture could not be found in all the research carried out in the context of this thesis. This, together with the other approaches to complexity reduction listed in this section, led to the idea of the Sliced Long Short-Term Memory Neural Network (SlicedLSTM). The SlicedLSTM itself should be able to split the incoming data, process it in parallel layers and aggregate it in a final result layer. This results in a novel form of LSTM cells that combines the advantages of the existing approaches explained in this section. The SlicedLSTM should measurably reduce the processing time compared to a standard LSTM through concurrency and by reducing the complexity of the inner networks.

In the further course of this chapter, the basic idea of the SlicedLSTM will be continued. The inner structure and functioning of the SlicedLSTM will be explained and described theoretically and in all details. In particular, it will be explained how input data can be sliced into sub-packages by the SlicedLSTM. Data in a data set can have relationships of any relevance. If one splits data into individual packages, relationships can be lost. This could lead to a deterioration in the accuracy of the results provided by the network. The projects listed in section 3.4, which follow the approach of splitting data, do not make any statement on this. In these projects, data is split manually or semi-automatically according to certain procedures in an upstream processing step. How data is split is determined on the basis of existing expert knowledge. It is therefore assumed to be known which relationships exist in data and which relevance these have for the neural network. As a result, in the projects mentioned, data from different data packages have no or negligible relationships to each other. This is similar to the grouping approach. In order not to impair the accuracy of the network's results, however, the data splits or groups must also be largely independent of each other.

One requirement for the SlicedLSTM is that it should independently perform the slicing of the input data within its cell. Which strategy is used to split data in the construction of the SlicedLSTM is therefore an important question. The cell itself has no expert knowledge of the data and the current design of the cell does not allow this to be gained through training. From the projects listed in section 3.4, however, it is also not possible to deduce how strongly an unfavourably chosen data split affects the quality of the network. There was complete trust in the knowledge of the experts who carried out the division of data.

The SlicedLSTM receives the desired division into slices via an additional hyperparameter. The exact allocation of the input vector and the hidden vector to the slices must be determined. The investigations and tests to be carried out within this thesis are intended to show whether comparable accuracies can already be achieved with an empirically selected division of data. The comparison is always made against the standard LSTM. If no comparable accuracy can be achieved with a random split of data, the architecture of the SlicedLSTM must be extended by suitable measures.

From the explanations given in this chapter, it can already be deduced which values must be used for the comparative measurements. The primary goal of this thesis and the developed SlicedLSTM is to improve runtime of an LSTM. For the evaluation of the processing time at runtime, the time duration for the so-called forward pass is measured. This is the time that the network needs to determine a result from input data. As explained, reducing the complexity of a network or its cells also has a beneficial effect on the training of the network. Therefore, the processing time during training should also be measured. This processing time, referred to in the following as backward pass, comprises the time required to backpropagate training updates based on the model error. The model error is the deviation of a result generated by the net from the expected result. Other hyperparameters, such as the learning rate, are also included in the error calculation and adjustment of the weights. For the measurement of the backward pass, it is therefore also interesting to see how the SlicedLSTM behaves in comparison to the standard LSTM with different values for the hyperparameters. The requirement set for all comparisons in this thesis is that the accuracy of the results generated by the SlicedLSTM must remain comparable to those of the standard LSTM. In section 6.3.1 the term "comparable" will be defined precisely for this thesis. The achieved accuracy of the results of a network compared to the expected results is, besides the runtimes for forward and backward pass, the third relevant measured value in the context of the investigations carried out here.

The SlicedLSTM also entails a certain overhead. Therefore, at the time of the theoretical consideration, it is not ensured that the advantages of the approach also outweigh the disadvantages in practical application. Therefore, a simple proof of concept (PoC) will be conducted in this chapter (section 6.2.3) to find initial indications that the chosen approach of the SlicedLSTM is indeed capable of outperforming a standard LSTM. The following section 6.3 then describes the setup of a qualitative study to demonstrate the optimisation achieved by the SlicedLSTM compared to a standard LSTM. The design of the study is based on the research methodology described in chapter 4 and the quantitative study design described in section 4.4. Section 6.4 then summarise the results of the study and the overall results.

6.2.2 Theoretical Considerations

In this section, the inner structure and functioning of the SlicedLSTM is explained and described in all details. For this purpose, the architecture of the SlicedLSTM, which has been adapted in comparison to a standard LSTM, is shown with its additional split and connector layers. Then, using an example vector of input data, it is shown schematically how the individual data is split, processed and merged again in the inner layers of the SlicedLSTM. The section concludes with a mathematical definition of the SlicedLSTM in comparison to the standard LSTM. All examples given here use input data in the form of a one-dimensional data vector. As this thesis focuses on the application of PdM in industrial environments, a typical form of input data is sensor data in the form of one-dimensional data vectors (MathWorks, 2021). In other application areas, this may be different. For example, in the field of image processing, input data is typically in 2D or 3D data vectors. However, the concept of SlicedLSTM can be easily adapted to process input data for any dimensionality.

Figure 17 shows the schematic of a standard LSTM. According to the consideration chosen in this section, a standard LSTM receives the input data $\boldsymbol{x_t}$ as a one-dimensional data vector at a time t. The length of the vector is determined by the number of features. The input data $\boldsymbol{x_t}$ are concatenated with the hidden state of the preceding processing step $\boldsymbol{h_{t-1}}$ and thus represent the input data of the individual gates of an LSTM at time t. The meaning and functionality of the individual gates (forget gate, input gate, output gate) has already been explained in the basic section 2.7.

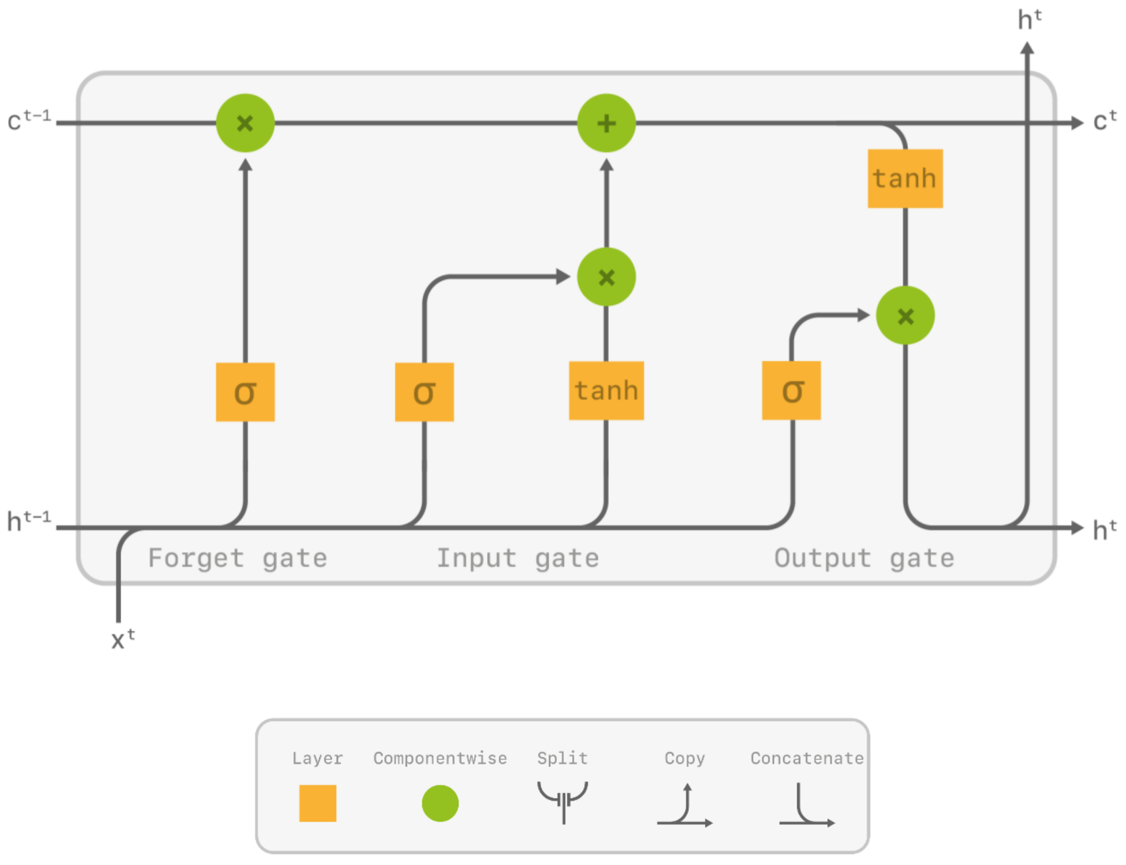


Figure 17 – Standard LSTM (Hochreiter, et al., 1997)

In the context of this thesis, approaches have already been explained that divide input data into partial packets in a manual pre-processing step, process these in parallel LSTMs and then merge them again into an overall result. The SlicedLSTM is intended to take up this approach. The splitting and thus the parallelisation achieved is to be carried out completely and automatically inside the LSTM by means of a novel model architecture. In addition to a standard LSTM, a SlicedLSTM therefore requires two additional layers. One additional inner layer is required to calculate the results of the splits. This layer is referred to as the SplitLayer in the following. A second inner layer connects the results of the SplitLayer to an overall result in the sense of a dense layer. This layer is referred to as the ConnectionLayer in the following. The interfaces of an LSTM cell for incoming and outgoing data shall remain unchanged from the standard LSTM. Figure 18 shows a SlicedLSTM with SplitLayer and ConnectionLayer and a split rate of 2.

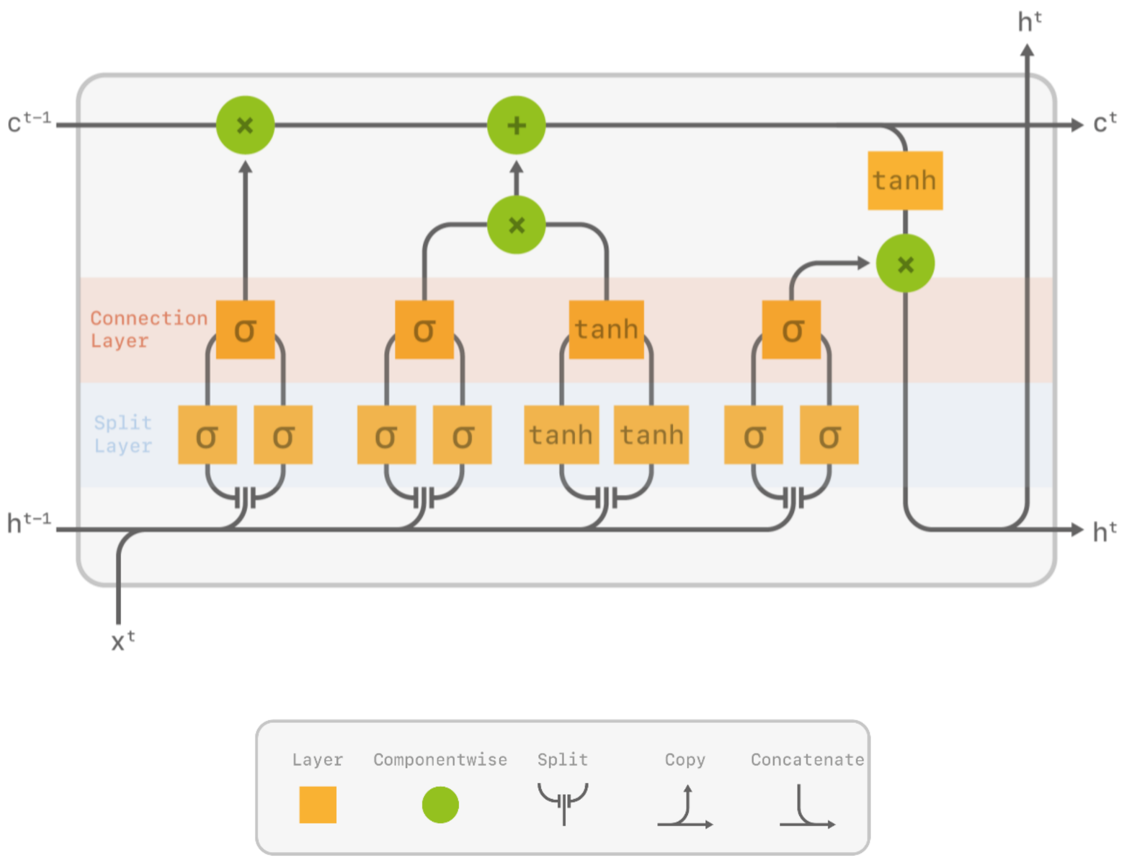


Figure 18 – SlicedLSTM with SplitLayer and ConnectionLayer and a split rate of 2

As shown in Figure 18, the additional SplitLayer splits $\boldsymbol{x_t}$ and $\boldsymbol{h_{t-1}}$ into parts according to the defined split distributions. The SplitLayer consists of one dense layer per split (gate) which processes its slice of hidden and input data as specified by the user when creating the SlicedLSTM model. The data processed in the SplitLayers are then combined again in the ConnectionLayers. The ConnectionLayer is a simple dense layer that fully connects all input values (output values of the SplitLayers) and thus generates the result value. The update of the cell state $\boldsymbol{c_t}$ and the generation of the output $\boldsymbol{h_t}$ is then the same as with the standard LSTM.

At this point, the underlying concept of SlicedLSTM is recalled once again. The subnetworks shown in Figure 18 (here two per gate) are to become significantly smaller and more easily processable in parallel through the splitting. In order to process the splits, however, additional components are required in the SlicedLSTM in the form of the SplitLayer. Whether and under what conditions the advantage of a SlicedLSTM exceeds its overhead is to be determined in the quantitative study of this thesis.

Based on the inner model architecture of a SlicedLSTM shown in Figure 18, Figure 19 shows the functionality of a SlicedLSTM from the perspective of the single features in the data vectors.

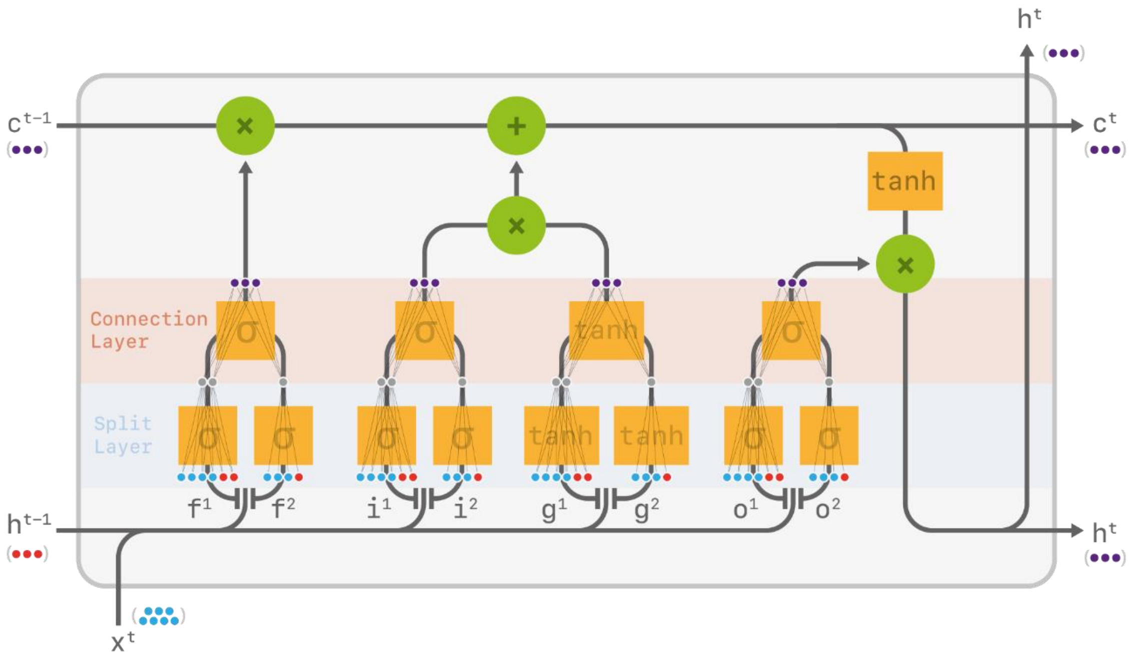


Figure 19 - SlicedLSTM from the perspective of the data vectors

In the example shown in Figure 19, the input vector ***x*** has a size of 7. The features of ***x*** are coloured in blue. The hidden size is 3. Features of the hidden vector are coloured red. In the SplitLayer, data is split into two slices. In this example, due to the user configuration, the respective left layers receive 4 values (blue dots) from the input vector and 2 values (red dots) from the hidden vector. The respective right layers receive 3 values (blue dots) from the input vector and one value (red dot) from the hidden vector. The left layers deliver two values and the right layers one value to the ConnectionLayer. In the ConnectionLayer, all values of all subnetworks are connected with each other (fully connected dense layer). As a result, the ConnectionLayer must deliver a data vector of size *hidden size*. As with the standard LSTM, this then goes into the cell state $\boldsymbol{c_t}$ or respectively the output $\boldsymbol{h_t}$.

To further explain the functioning of the SlicedLSTM, another example (hereafter referred to as example 2) will be explained. In example 2, the input vector comprises 6 features and the hidden size is set also to 6.

- The input data at processing time t is $\mathbf{x_t} = (\mathrm{x}_1 \quad \mathrm{x}_2 \quad \mathrm{x}_3 \quad \mathrm{x}_4 \quad \mathrm{x}_5 \quad \mathrm{x}_6)_t$
- The hidden data from the last processing step is $\mathbf{h_{t-1}} = (\mathrm{h}_1 \quad \mathrm{h}_2 \quad \mathrm{h}_3 \quad \mathrm{h}_4 \quad \mathrm{h}_5 \quad \mathrm{h}_6)_{t-1}$
- The split configuration for $\mathbf{x_t}$ an $\mathbf{h_{t-1}}$ in this example is empirically set to $[(3,2),(2,3),(1,1)]$
- Following the split configuration the input vector $\boldsymbol{x_t}$ is divided into the partial vectors $(\mathrm{x}_1 \quad \mathrm{x}_2 \quad \mathrm{x}_3)_t, (\mathrm{x}_4 \quad \mathrm{x}_5)_t, (\mathrm{x}_6)_t$
- The hidden vector $\boldsymbol{h_{t-1}}$ is divided into the partial vectors $(\mathrm{h}_1 \quad \mathrm{h}_2)_{t-1}, (\mathrm{h}_3 \quad \mathrm{h}_4 \quad \mathrm{h}_5)_{t-1}, (\mathrm{h}_6)_{t-1}$
- Then the resulting splits are
  - Split 1: $(\mathrm{x}_1 \quad \mathrm{x}_2 \quad \mathrm{x}_3 \quad \mathrm{h}_1 \quad \mathrm{h}_2)_t$
  - Split 2: $(\mathrm{x}_4 \quad \mathrm{x}_5 \quad \mathrm{h}_3 \quad \mathrm{h}_4 \quad \mathrm{h}_5)_t$
  - Split 3: $(\mathrm{x}_6 \quad \mathrm{h}_6)_t$

The division of the input and hidden vector in example 2 is deliberately chosen uncorrelated and unevenly. The hidden vector was also deliberately divided unevenly and its size is not based on the division of the input vector. The divisions chosen in this example are intended to indicate that at first sight of the test there is no rule for the division for the SlicedLSTM and that this can be done arbitrarily. Based on the dimensions of the input data and the hidden data, one could assume that the splitting made in example 2 is not advantageous. However, there are no reliable indications for this at the moment. From the approaches presented in section 3.4 of this thesis with a manual division of data based on expert knowledge, it can be deduced that there is no universally correct form of division of data. In all cases listed in section 3.4, the optimal division has no fixed metric but is determined by experts based on data and its inner relationships, as well as the use case.

Figure 20 shows the schematic representation of example 2 with the input nodes and output nodes of the three LSTM splits in SplitLayer and ConnectionLayer. The result vector of each SplitLayer always has the size of its share of the hidden vector. The output nodes of the SplitLayer are named with the letter S in Figure 20. In the final (dense) ConnectionLayer, the results of the subnetworks are concatenated into one result. The output nodes of the ConnectionLayer are named with the letter $\acute{S}$ in Figure 20.

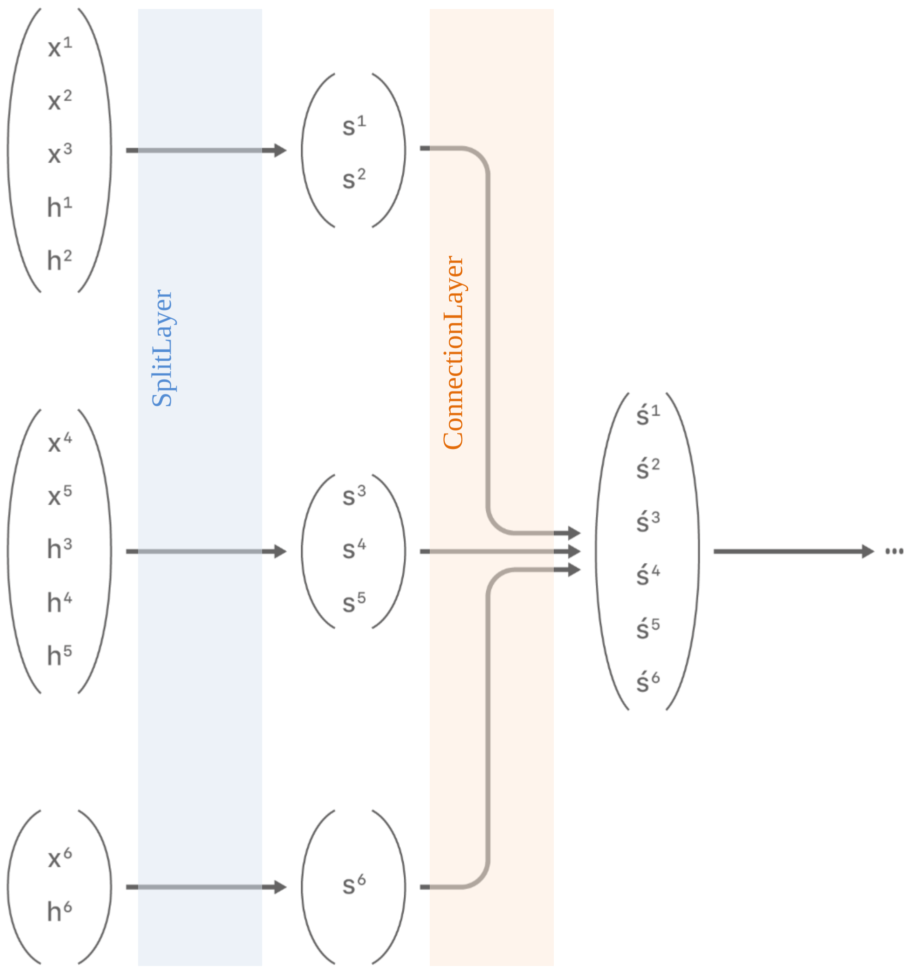


Figure 20 - Schematic Representation of the SplitLayer of Example 2[*]

* $((S^1 \quad S^2) \quad (S^3 \quad S^4 \quad S^5) \quad S^6)$ are the output nodes of the SplitLayer and $(\acute{S}^1 \quad \acute{S}^2 \quad \acute{S}^3 \quad \acute{S}^4 \quad \acute{S}^5 \quad \acute{S}^6)$ are the output nodes of the ConnectionLayer

Another example 3 is used to explain how data is processed in the individual layers and assumes an input vector with 3 features and a hidden size of 3. The input data at the beginning of the processing time *t* is thus $\mathbf{x}_t = (x1, x2, x3)_t$. The hidden data at the beginning of processing time *t* is $\boldsymbol{h}_{t-1} = (h1, h2, h3)_{t-1}$. In example 3, data is divided as follows: Split 1 *(x1, x2, h1, h2)* and Split 2 *(x3, h3)*. Slice 1 of the SplitLayer thus delivers two values to the result and slice 2 one value.

The following formulas of example 3 only show the forget gate. The procedures in the other gates are the same, except for the activation function of the tanh gate. The following formulas describe the processing of the two slices in the SplitLayer and their merging in the ConnectionLayer. In addition to the input data and the hidden data, the formulas also contain a weighting matrix and a bias vector. The weight matrix uses the letter *w* (SplitLayer) and *u* (ConnectionLayer). The bias vector uses the letter *b* (SplitLayer) and *p* (ConnectionLayer). The following formulas are intended to illustrate the processing of the individual values of the slices in the individual layers. Weights and biases are not relevant for this illustration and are only listed in the formulas for the sake of completeness. The meaning of weights and biases is described in detail later in this section (see Forget Gate at page 88).

SplitLayer:

$$\text{Slice 1: } \sigma\left((\mathrm{x}_1 \quad \mathrm{x}_2 \quad \mathrm{h}_1 \quad \mathrm{h}_2) \cdot \begin{pmatrix} \mathrm{w}_{11} & \mathrm{w}_{12} \\ \mathrm{w}_{21} & \mathrm{w}_{22} \\ \mathrm{w}_{31} & \mathrm{w}_{32} \\ \mathrm{w}_{41} & \mathrm{w}_{42} \end{pmatrix} + (\mathrm{b}_1 \quad \mathrm{b}_2) \right) = (\mathrm{s}_1 \quad \mathrm{s}_2)$$

$$\text{Slice 2: } \sigma\left((\mathrm{x}_3 \quad \mathrm{h}_3) \cdot \begin{pmatrix} w_1 \\ w_2 \end{pmatrix} + \mathrm{b}_3 \right) = (s_3)$$

The intermediate results *s₁*, *s₂* and *s₃* are then to be connected in the ConnectionLayer accordingly:

$$\sigma\left((\mathrm{s}_1 \quad \mathrm{s}_2 \quad \mathrm{s}_3) \cdot \begin{pmatrix} \mathrm{u}_{11} & \mathrm{u}_{12} & \mathrm{u}_{13} \\ \mathrm{u}_{21} & \mathrm{u}_{22} & \mathrm{u}_{23} \\ \mathrm{u}_{31} & \mathrm{u}_{32} & \mathrm{u}_{33} \end{pmatrix} + (\mathrm{p}_1 \quad \mathrm{p}_2 \quad \mathrm{p}_3) \right) = (\mathrm{f}_1 \quad \mathrm{f}_2 \quad \mathrm{f}_3)$$

The resulting vector at a specific time step $\boldsymbol{f^t} = (f_1 \quad f_2 \quad f_3)^t$ then updates the cell state $\mathbf{c^{t-1}}$ by vector-matrix multiplication. The procedure in the other gates is correspondingly.

Based on the three examples presented, the entire functionality of the SlicedLSTM will now be formally summarised. The SlicedLSTM will be compared with the standard LSTM. All formulas for the standard LSTM listed below are based on Hochreiter & Gers (Hochreiter, et al., 1997), (Gers, et al., 1999). In order to make the presentation of the operations in the individual gates and the comparison with those of the standard LSTM simpler, the division of the incoming vectors $\boldsymbol{x^t}$ and $\boldsymbol{h^{t-1}}$ into the individual slices is presented first. Then each gate is discussed individually and the comparison with the Standard LSTM is highlighted.

For the selected representation of the following formulas, it should be mentioned that index specifications are always subscripted. The time intervals that were previously displayed in subscript are then displayed in superscript. This form of representation was chosen by the author with the idea of preferably using subscripted values for frequently changing values. For simple values normal font is used. Vectors are indicated with a lower case letter in bold. Matrices are always indicated with upper case letters in bold.

Representation of the individual slices:

Let the input vector at time t with n features be $\mathbf{x}^{\boldsymbol{t}} = (\mathrm{x}_1, \dots, \mathrm{x}_\mathrm{n})^t$

Let the hidden vector with m features be $\mathbf{h}^{\boldsymbol{t-1}} = (\mathrm{h}_1, \dots, \mathrm{h}_\mathrm{m})^{t-1}$, where at time *t=0* the hidden vector is initialised with zeros $\boldsymbol{h}^{\mathbf{0}} = (0, 0, \dots, 0)$

Let a slice *s* be defined as a tuple of index intervals $s = (\alpha, \beta)$ with
$\alpha = [\alpha_1, \alpha_2] \subseteq [1, \mathrm{n}],\ \ \beta = [\beta_1, \beta_2] \subseteq [1, \mathrm{m}]$ whereby $(\alpha, \beta) \subset \mathbb{N}^2$ and $\mathrm{n}, \mathrm{m} \subset \mathbb{N}$

A set of slices $\mathcal{S} = \{\mathrm{s}^1, \dots, \mathrm{s}^k\}$ is valid for a SlicedLSTM cell if and only if there is a permutation $(s^1, s^2, \dots, s^k)$ of $\tilde{\mathcal{S}}$ that holds all of the following constraints:

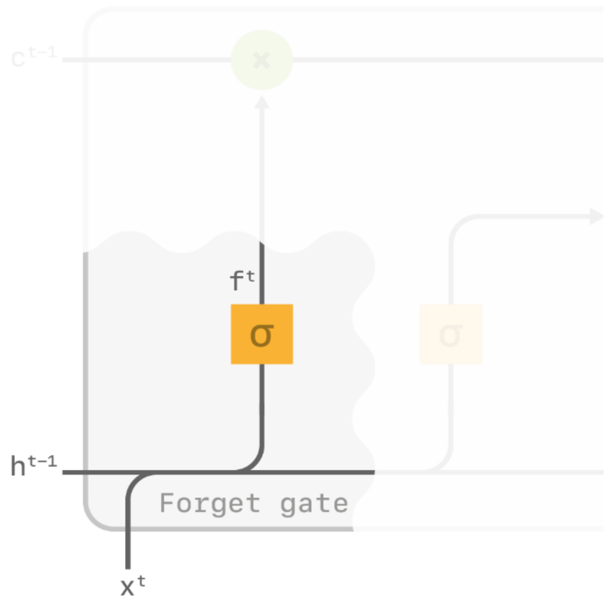
$s^1 = (\,[\,1,\ \alpha_2^1\,], [1, \beta_2^1]\,)$	First slice includes the first part of hidden and input vector.
$s^\mathrm{k} = \big(\,[\alpha_1^\mathrm{k}, \mathrm{n}], [\beta_1^\mathrm{k}, \mathrm{m}]\,\big)$	Last slice includes the last part of hidden and input vector where n is the size of the input vector and m the size of the hidden vector.
$\alpha_2^\mathrm{i} = \alpha_1^{\mathrm{i}+1} - 1$ whereby $\forall \mathrm{i} \in [1, \mathrm{n}) \subset \mathbb{N}$ and $\beta_2^\mathrm{i} = \beta_1^{\mathrm{i}+1} - 1$ whereby $\forall \mathrm{i} \in [1, \mathrm{m}) \subset \mathbb{N}$	Each slice connects without gaps to the previous slice (except the first slice)

Based on the indices defined in $\tilde{\mathcal{S}}$, $\boldsymbol{x_t}$ and $\boldsymbol{h_{t-1}}$ can be subdivided into disjoint and exhaustive partial vectors. These partial vectors are then processed in the SplitLayer of the SlicedLSTM in each gate. Let the concatenation of the partial vectors of $\boldsymbol{x_t}$ and $\boldsymbol{h_{t-1}}$ defined by a slice *i* then be defined as:

$$\mathrm{s}_\mathrm{i}^\mathrm{t} = \left[\mathrm{x}_{\alpha_1^\mathrm{i}}^\mathrm{t}, \mathrm{x}_{\alpha_1^\mathrm{i}+1}^\mathrm{t}, \dots, \mathrm{x}_{\alpha_2^\mathrm{i}}^\mathrm{t}, \mathrm{h}_{\beta_1^\mathrm{i}}^{\mathrm{t}-1}, \mathrm{h}_{\beta_1^\mathrm{i}+1}^{\mathrm{t}-1}, \dots, \mathrm{h}_{\beta_2^\mathrm{i}}^{\mathrm{t}-1}\right]$$

Looking at the forget gate:

The Forget Gate has the task of deciding which information from the cell status $\mathbf{c}_{t-1}$ is no longer needed or is classified as less important. For this purpose, the Forget Gate layer applies the sigmoid function as an activation function to the sum of the weighted input values $\boldsymbol{x}^t$ and $\boldsymbol{h}^{t-1}$. Thus, for each number in the result vector $\boldsymbol{f}_t$, a number between 0 and 1 is created. In the later component-wise multiplication of $\boldsymbol{f}_t$ with $\mathbf{c}_{t-1}$, $\boldsymbol{f}_t$ values close to zero cause cell state components to be forgotten. Respectively $\boldsymbol{f}_t$ values close to 1 have the effect that values remain in the cell state. All values in $\boldsymbol{f}_t$ greater than 0 and less than 1 change the importance of the memorised value accordingly. The illustrations in Figure 21 and Figure 22 show the processing and its mathematical representation in the Forget Gate for the Standard LSTM and for the SlicedLSTM.



$$\mathbf{f}^{\mathbf{t}} = \sigma\left([\mathbf{h}^{\mathbf{t-1}}, \mathbf{x}^{\mathbf{t}}] \cdot \boldsymbol{W}_{\boldsymbol{F}} + \mathbf{b}_{\boldsymbol{F}}\right)$$

(the subscript capital **F** stands for "forget gate")

(weight matrix and bias vector are not shown in the figures)

Figure 21 – Forget Gate of Standard LSTM

In the standard LSTM, the result vector $\boldsymbol{f}_t$ of the forget gate is obtained by the vector matrix product of the concatenated vectors $\boldsymbol{x}_t$ and $\boldsymbol{h}_{t-1}$ with the weighting specified for the forget gate. Therefore, a weight matrix must be defined for the forget gate of an LSTM, which will be denoted by the letter $\boldsymbol{W}$ (SplitLayer) and $\boldsymbol{U}$ (ConnectionLayer) in the following. It should be remembered from the notation already explained that in this thesis simple values are shown in lower case and normal font and matrices with a capital letter in bold type. In the following, the letter w or u refers to a single weight value and $\boldsymbol{W}$ or $\boldsymbol{U}$ to a matrix of weight values. A bias is also added to the result. The vector of biases is indicated in the following by the letter $\boldsymbol{b}$ (SplitLayer) and $\boldsymbol{p}$ (ConnectionLayer) since vectors are marked with a bold lower case letter in the context of this thesis. The sigmoid function serves as the activation function in the forget gate. Figure 21 illustrates this.

Weights, biases and activation functions are also present in the other gates of an LSTM accordingly. The processing methods for the gates of an LSTM described here correspond to the standard definition of an LSTM. These have already been explained in section 2.7 of this thesis and are based on the basic and still current definition of the LSTM by Hochreiter and Gers (Hochreiter, et al., 1997), (Gers, et al., 1999). The vectors listed below are always row vectors whose components are notated next to each other. For two vectors $\boldsymbol{x} = (x_1, \dots, x_n)$ and $\boldsymbol{y} = (y_1, \dots, y_m)$ the notation $[\boldsymbol{x}, \boldsymbol{y}]$ means the concatenation of the two vectors $[\boldsymbol{x}, \boldsymbol{y}] = (x_1, \dots, x_n, y_1, \dots y_m)$.

The processing of the input vector in the SlicedLSTM follows the same principle as in the standard LSTM, but is divided into SplitLayer and ConnectionLayer. In the SplitLayer, the individual partial vectors $\mathbf{s}_\mathbf{i}^\mathbf{t}$ for the slices *i* for *i* from 1 to k are processed by individual layers with sigmoid activation. The division into the partial vectors $\mathbf{s}_\mathbf{i}^\mathbf{t}$ was explained at the beginning of this section. The resulting *k* vectors then concatenated and processed in the ConnectionLayer to the result of the forget gate $\mathbf{f}^\mathbf{t}$. The processing in the SlicedLSTM layers is a vector-matrix multiplication with the weight matrix and a vector-addition with the bias vector of the respective dense layers. The sigmoid function again serves as an activation function. As with the standard LSTM, the length of $\mathbf{f}^\mathbf{t}$ corresponds to the length of $\boldsymbol{h}^{t-1}$ *and* $\mathbf{c}^{t-1}$.

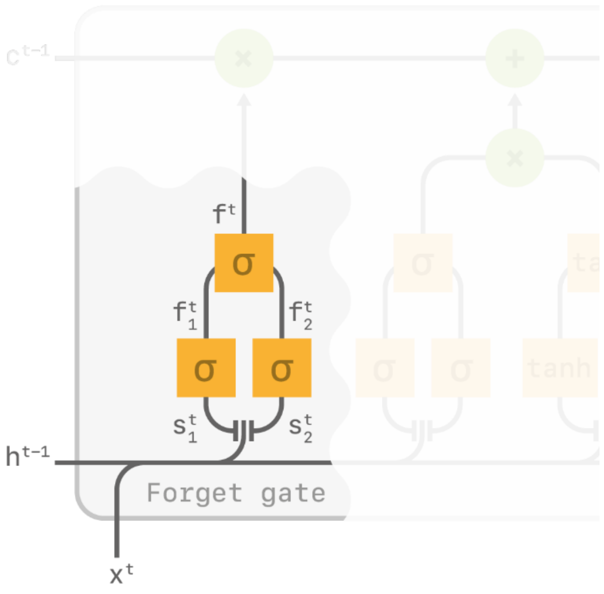


Figure 22 - Forget Gate of SlicedLSTM

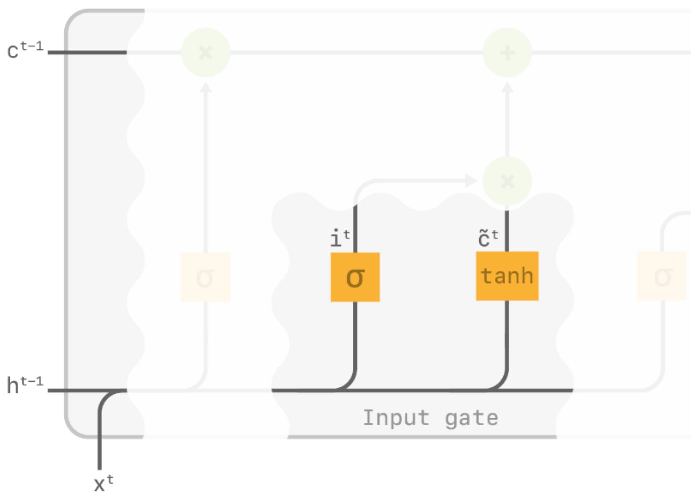
$$\forall \mathrm{i} \in [1, \mathrm{k}] \subset \mathbb{N}: \mathbf{f}_\mathbf{i}^\mathbf{t} = \sigma\left(\mathbf{s}_\mathbf{i}^\mathbf{t} \cdot \boldsymbol{W}_{\mathbf{Fi}} + \mathbf{b}_{\mathbf{Fi}}\right)$$

$$\mathbf{f}^\mathbf{t} = \sigma\left([\mathbf{f}_\mathbf{1}^\mathbf{t}, \mathbf{f}_\mathbf{2}^\mathbf{t}, \dots, \mathbf{f}_\mathbf{k}^\mathbf{t}] \cdot \boldsymbol{U}_{\boldsymbol{F}} + \boldsymbol{p}_{\boldsymbol{F}}\right)$$

(the subscript capital **F** stands for "forget gate")

Looking at the input gate:

The input gate decides which new information is considered important enough to be remembered for later processing cycles. To do this, the so-called input gate layer first decides which values of the current input vector are relevant. The sigmoid function again serves as the activation function. The result of this layer is denoted by $\boldsymbol{i^t}$ (see Figure 23). In the additional tanh layer, another vector $\mathbf{\tilde{c}^t}$ is generated with the so-called candidate values. Tangent hyperbolicus (tanh) is used as the activation function. This layer decides which of the values determined by the input gate should actually be included in the new cell state and with what intensity. This is done by pointwise multiplication of the two vectors $\boldsymbol{i^t}$ and $\mathbf{\tilde{c}^t}$. The resulting vector of the input gate is then added to the current cell state $\boldsymbol{c^t}$. Figure 23 illustrates the input gate of the standard LSTM.



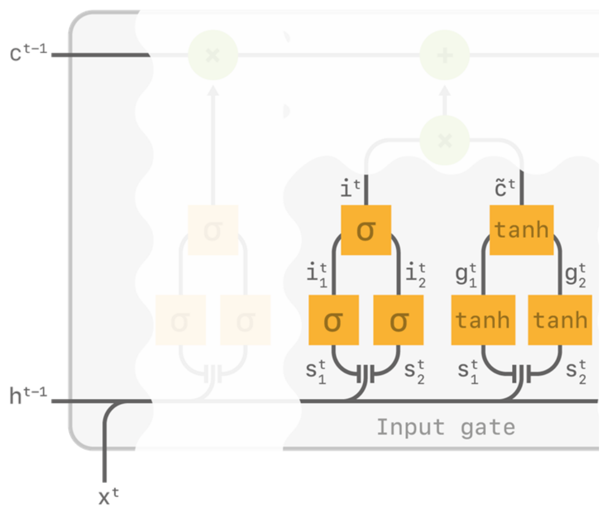
$$\mathbf{i^t} = \sigma\left([\mathbf{h^{t-1}}, \mathbf{x^t}] \cdot \boldsymbol{W_I} + \mathbf{b}_{\boldsymbol{I}}\right)$$

$$\mathbf{\tilde{c}^t} = \tanh([\mathbf{h^{t-1}}, \mathbf{x^t}] \cdot \boldsymbol{W_G} + \mathbf{b}_{\boldsymbol{G}})$$

(the subscript capital **I** stands for "input gate", the subscript capital **G** stands for "tanh gate")

Figure 23 – Input Gate of Standard LSTM

With SlicedLSTM, the processing in the input gate and tanh gate again takes place in SplitLayer and ConnectionLayer. The ConnectionLayer again processes the concatenated results of the sub-gates into $\mathbf{i^t}$ and $\boldsymbol{\tilde{c}^t}$. Figure 24 illustrates this. Since the letter t has already been assigned for the time indication, the tanh gate is designated in the following with the letter g.



$$\forall j \in [1,k] \subset \mathbb{N}: \mathbf{i}_j^t = \sigma\left(\mathbf{s}_j^t \cdot \boldsymbol{W}_{\mathbf{Ij}} + \mathbf{b}_{\mathbf{Ij}}\right)$$

$$\mathbf{i}^t = \sigma([\mathbf{i}_1^t, \mathbf{i}_2^t, \dots, \mathbf{i}_k^t] \cdot \boldsymbol{U}_I + \boldsymbol{p}_I)$$

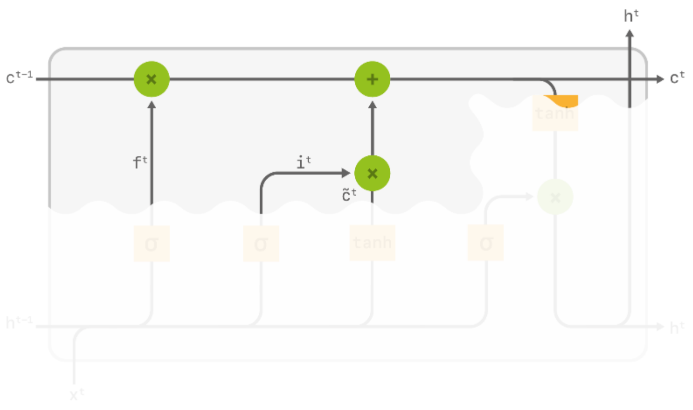
$$\forall i \in [1,k] \subset \mathbb{N}: \mathbf{g}_i^t = \tanh\left(\mathbf{s}_i^t \cdot \boldsymbol{W}_{\mathbf{Gi}} + \mathbf{b}_{\mathbf{Gi}}\right)$$

$$\widetilde{\mathbf{c}}^t = \tanh\left([\mathbf{g}_1^t, \mathbf{g}_2^t, \dots, \mathbf{g}_k^t] \cdot \boldsymbol{U}_G + \boldsymbol{p}_G\right)$$

(the subscript capital **I** stands for "input gate", the subscript capital **G** stands for "tanh gate")

Figure 24 – Input Gate of SlicedLSTM

The described processing steps in the forget gate and input gate determined which of the data stored in cell state $\boldsymbol{c}^{t-1}$ can be forgotten and which should be remembered. In the step illustrated in Figure 25, the results of the forget gate and input gate now enter the old cell state $\boldsymbol{c}^{t-1}$ and thus transfer it to the new cell state $\boldsymbol{c}^t$. This process is identical in the SlicedLSTM to that in the Standard LSTM.



$$\mathbf{c}^t = \mathbf{f}^t \odot \mathbf{c}^{t-1} + \mathbf{i}^t \odot \tilde{\mathbf{c}}^t$$

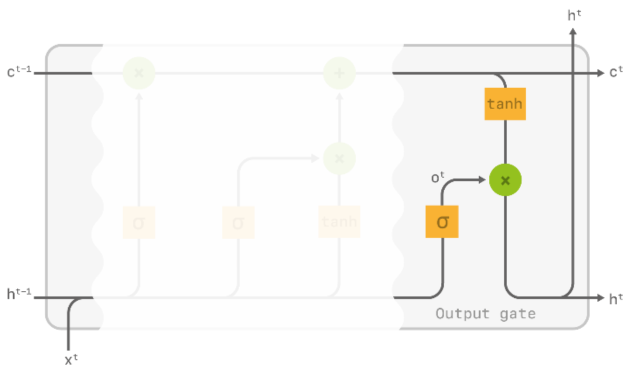
$\odot$ stands for the component-wise multiplication of vectors.

At time *t* = 1 a zero cell state is used for c^{t-1}: $\mathrm{c}^0 = (0, 0, \dots, 0)$

Figure 25 – Update the cell state of standard and SlicedLSTM

Looking at the output gate:

The last of the three gates of an LSTM is the output gate. It delivers the result of the current processing $\boldsymbol{h}^t$. The output gate first determines an output vector $\boldsymbol{o}^t$ on the basis of the input data. The activation function is again the sigmoid function. In addition, the current cell state is brought into the value range (-1 ,1) by means of the hyperbolic function *tanh*. The output vector $\boldsymbol{h}^t$ is then determined by component-wise multiplying the adjusted cell state by the output $\boldsymbol{o_t}$. Through this arrangement, what is memorised in the cell state influences the output determined on the basis of the input data at a specific processing step *t*. The following Figure 26 describes the function of the output gate of a standard LSTM.



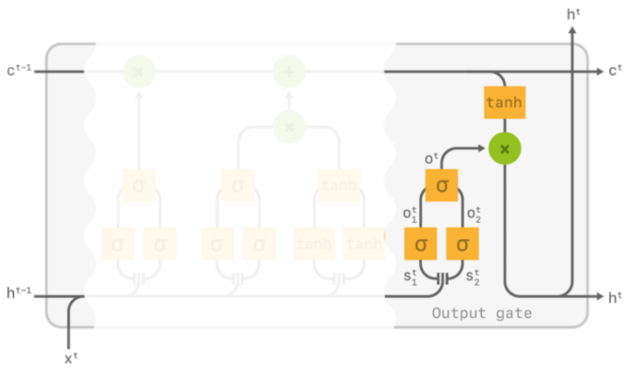
$$\mathbf{o^t} = \sigma\left([\mathbf{h^{t-1}}, \mathbf{x^t}] \cdot \boldsymbol{W_O} + \mathbf{b_{\boldsymbol{O}}}\right)$$

$$\mathbf{h^t} = \mathbf{o^t} \odot \tanh(\mathbf{c^t})$$

(The subscript of $\boldsymbol{W_O}$ ***and*** $\mathbf{b_{\boldsymbol{O}}}$ is an "O" and stands for output gate)

Figure 26 – Output Gate of Standard LSTM

In comparison, the SlicedLSTM again has the familiar division into two layers from the previously described gates. Figure 27 shows the output gate of a SlicedLSTM. The final determination of $\boldsymbol{h^t}$ for the SlicedLSTM is then identical to that of the standard LSTM.



$$\forall i \in [1, k] \subset \mathbb{N}: \mathbf{o_i^t} = \sigma\left(\mathbf{s_i^t} \cdot \boldsymbol{W}_{\mathbf{Oi}} + \mathbf{b_{Oi}}\right)$$

$$\mathbf{o^t} = \sigma\left([\mathbf{o_1^t}, \mathbf{o_2^t}, \dots, \mathbf{o_k^t}] \cdot \boldsymbol{U_O} + \boldsymbol{p_O}\right)$$

$$\mathbf{h^t} = \mathbf{o^t} \odot \tanh(\mathbf{c^t})$$

Figure 27 - Output Gate of SlicedLSTM

Finally, all the explanations of the formal, mathematical processes of a SlicedLSTM, which have been shown step by step and for each gate, are to be summarised once again in Table 7. The illustration again contrasts the SlicedLSTM with the standard LSTM.

SlicedLSTM	**Standard LSTM**
Forget gate	
$\forall i \in [1,k] \subset \mathbb{N}: \mathbf{f}_i^t = \sigma(\mathbf{s}_i^t \cdot \boldsymbol{W}_{Fi} + \mathbf{b}_{Fi})$ $\mathbf{f}^t = \sigma([\mathbf{f}_1^t, \mathbf{f}_2^t, \dots, \mathbf{f}_k^t] \cdot \boldsymbol{U}_F + \boldsymbol{p}_F)$	$\mathbf{f}^t = \sigma([\mathbf{h}^{t-1}, \mathbf{x}^t] \cdot \boldsymbol{W}_F + \mathbf{b}_F)$
Input gate	
$\forall j \in [1,k] \subset \mathbb{N}: \mathbf{i}_j^t = \sigma\left(\mathbf{s}_j^t \cdot \boldsymbol{W}_{Ij} + \mathbf{b}_{Ij}\right)$ $\mathbf{i}^t = \sigma([\mathbf{i}_1^t, \mathbf{i}_2^t, \dots, \mathbf{i}_k^t] \cdot \boldsymbol{U}_I + \boldsymbol{p}_I)$ $\forall i \in [1,k] \subset \mathbb{N}: \mathbf{g}_i^t = \tanh(\mathbf{s}_i^t \cdot \boldsymbol{W}_{Gi} + \mathbf{b}_{Gi})$ $\widetilde{\mathbf{c}}^t = \tanh([\mathbf{g}_1^t, \mathbf{g}_2^t, \dots, \mathbf{g}_k^t] \cdot \boldsymbol{U}_G + \boldsymbol{p}_G)$	$\mathbf{i}^t = \sigma([\mathbf{h}^{t-1}, \mathbf{x}^t] \cdot \boldsymbol{W}_I + \mathbf{b}_I)$ $\tilde{\mathbf{c}}^t = \tanh([\mathbf{h}^{t-1}, \mathbf{x}^t] \cdot \boldsymbol{U}_G + \boldsymbol{p}_G)$
Update cell state	
$\mathbf{c}^t = \mathbf{f}^t \odot \mathbf{c}^{t-1} + \mathbf{i}^t \odot \tilde{\mathbf{c}}^t$	$\mathbf{c}^t = \mathbf{f}^t \odot \mathbf{c}^{t-1} + \mathbf{i}^t \odot \tilde{\mathbf{c}}^t$
Output gate	
$\forall i \in [1,k] \subset \mathbb{N}: \boldsymbol{o}_i^t = \sigma(\boldsymbol{s}_i^t \cdot \boldsymbol{W}_{Oi} + \boldsymbol{b}_{Oi})$ $\boldsymbol{o}^t = \sigma([\boldsymbol{o}_1^t, \boldsymbol{o}_2^t, \dots, \boldsymbol{o}_k^t] \cdot \boldsymbol{U}_O + \boldsymbol{p}_O)$	$\boldsymbol{o}^t = \sigma([\boldsymbol{h}^{t-1}, \boldsymbol{x}^t] \cdot \boldsymbol{W}_O + \boldsymbol{b}_O)$
Output	
$\boldsymbol{h}^t = \boldsymbol{o}^t \odot tanh(\boldsymbol{c}^t)$	$\boldsymbol{h}^t = \boldsymbol{o}^t \odot tanh(\boldsymbol{c}^t)$

Table 7 – SlicedLSTM versus Standard LSTM[*]

[*] (The subscripts **F**, **I**, **G**, **O** on the weight matrices ($\boldsymbol{W}$ *and* $\boldsymbol{U}$) and the bias vectors $\boldsymbol{b}$ ***and*** $\boldsymbol{p}$ stand for F=forget gate, I=input gate, G = tanh gate and O= output gate)

6.2.3 Proof of Concept Experiment

The concept of the SlicedLSTM entails a certain overhead due to the division into parallel networks and their aggregation, which can have a negative effect on the processing time. Therefore, at the time of the theoretical consideration, it is not ensured that the advantages of the approach outweigh the resulting additional effort in practical application. A simple Proof of Concept (PoC) will be used to find initial indications that the chosen approach of the SlicedLSTM is indeed capable of outperforming a standard LSTM. More specifically, the aim of the PoC is to find out whether the underlying concept of SlicedLSTM of processing data in multiple small LSTM splits actually speeds up the computation time while maintaining a similar result quality. The idea being that LSTM splits can use information locality while processing data to achieve comparable prediction accuracy with overall smaller models, thus resulting in decreased computation (and training) time.

In the PoC we evaluate two neural networks, both using standard LSTMs. The first LSTM network uses a standard layer design and receives the entire input values. For the second LSTM network, the input data is manually halved in a preprocessing step and then processed by two independent and smaller LSTM networks. The first model is subsequently referred to as Normal LSTM Layout (NormalLSTM). The second model is subsequently referred to as Manually Split LSTM Layout (ManSplitLSTM). The PoC setup of the second network takes up the idea of splitting data into smaller, independent data packages. This approach and its potential for improvement was explained in detail in section 3.4 (Ma, et al., 2018), (Jin, et al., 2022), (Ertan, 2021), (Bouaziz, et al., 2017), (Zhang, et al., 2017b), (Ding, et al., 2021), (Huang, et al., 2016), (Liang, et al., 2016), (Song, et al., 2016), (Wang, et al., 2019).

It is worth noting that simply splitting data manually in a preliminary step and passing it in separate LSTM cells is not the research idea of this thesis. This is only done to investigate the approximate behaviour of the chosen idea in a first step and with very little implementation effort. It should also be mentioned that the architecture and the implemented dimensions of the layers and hidden layers of the two LSTM networks for this PoC were chosen randomly. This PoC therefore only serves as a relative comparison of the two approaches and does not represent any general validity. It is also assumed that the version of the LSTM network that uses split data benefits from taking information locality into account. However, the splitting of data based on content correlations is not considered in this experiment. The main reason for this is that the newly developed SlicedLSTM model will also work with arbitrary data splits.

The neural network with LSTM in this PoC is implemented with the Open Source Deep-Learning-Library Keras (Chollet, et al., 2022) and a TensorFlow backend (Google Brain, 2022). By using these libraries, the implementation effort for the PoC could be reduced. In addition, both libraries are widely used in numerous Deep Learning applications. It can therefore be assumed that the models provided by Keras and TensorFlow already have a high degree of optimisation. Furthermore, in order to make the PoC setup and its evaluation as efficient as possible, a data set was used that has complete classification data and has already been used in a well-documented PdM project. This should not only make the PoC easier, but also avoid errors in the test design, the test implementation, as well as in the preparation and application of data.

In order to make the PoC setup as simple as possible, a search for existing and publicly available PdM projects with available data was carried out. Criteria for the search were the availability of a complete project with comprehensive documentation and fully prepared data. Projects and data that were referenced by additional work or were also used in other projects were preferred, as these were assumed to have a higher validity. An authoritative source for the search was the paper *Machine Learning in Production, Application Areas and freely available Data Sets* (Krauß, et al., 2019). This paper includes a search for projects and publicly available data sets with a focus on production. The application area refers to industrial ML applications and in particular PdM. The summary of the results of this paper provides an overview with an assessment of their suitability by the authors.

Based on the above criteria, a project published by Microsoft was selected. The project work is entitled *Predictive Maintenance Template* and was published by the AzureML Team for Microsoft (AzureML Team, 2015). The project describes step by step how to create and use Predictive Maintenance models to predict equipment failures. All steps, from data preparation and feature engineering to deployment, are disclosed and described in detail. The entire data processing pipeline including the source code is made available on Github and is used as a reference for this PoC (AzureML Team, 2018). Data are synthetically generated data from a NASA project to predict failures using Remaining Useful Lifetime (RUL) as result class. This project looks at modules of aircraft gas turbines and tries to predict whether RUL is smaller than a specified threshold, indicating imminent failure. The data set is freely available and has been used in many other projects (Saxena, et al., 2008). Among others, the dataset was used in the Prognostics and Health Management (PHM) data competition at PHM'08 and was extensively used and validated in this competition (Jia, et al., 2018). The data is available through the NASA Prognostics Center of Excellence Data Repository (NASA, 2008b). At the time the dataset was selected, the only discernible drawback was that the individual data attributes (features) were anonymised. Thus, the meaning of the features is unknown. This form of anonymisation is present in all the freely available datasets considered in the selection made here. One reason for this could be that non-anonymised data can be used to draw conclusions about the architecture and functioning of a machine or system under consideration. Such conclusions usually affect the proprietary knowledge of a manufacturer. This also applies if data was generated synthetically for a system or machine (Krauß, et al., 2019). In this thesis, it is to be investigated whether a measurable improvement in computation time can already be achieved with a randomly selected division of data. The meaning of the individual features and their possible local connections to each other are not taken into account. For this reason, the anonymisation of data used is not a limitation in this application.

To determine the accuracy of the results, a binary classification of the RUL available in the data is used in this test. Whether the values for RUL given in the data are actually valid in practice cannot be proven. In particular, it should be mentioned again that this is synthetically generated data. However, this can be accepted since the PoC only make relative comparisons of two LSTM models. A closer look at the data is given in the laboratory tests in section 6.3.2.

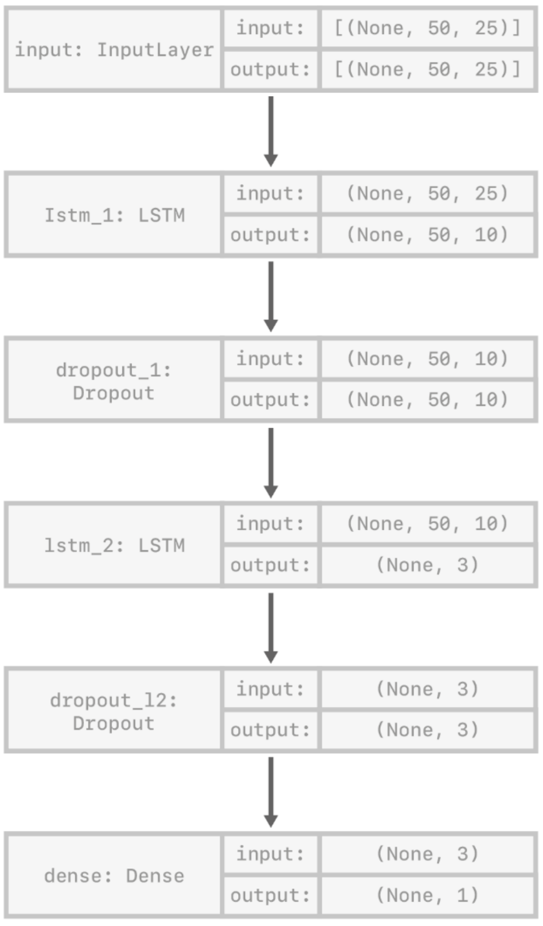


Figure 28 - NormalLSTM example with layers and dimensions

In the following the structure of the Normal LSTM Layout (NormalLSTM) and the manually split LSTM Layout (ManSplitLSTM) will be explained. When creating an LSTM, its dimension must be specified. Required dimension in Keras are only the size of the hidden state and the cell state. TensorFlow assumes a data stream. For processing by the LSTM, the data stream is divided into time frames with a fixed time span and thus a fixed number of time intervals (samples). The number of time intervals in a time frame is called sequence length. The number of time windows is called Batch Size. The Input Dimension determines the dimension of the input data, i.e. the number of features in data. In the test described here, we prepared data to a sequence length of 50 in one batch. The dimensions of the input data for both LSTM networks were set as (none, 50, 25). Thus, data from 50 consecutive time intervals are processed in this test, with each time interval having 25 features. The NormalLSTM network was empirically dimensioned in such a way that it has an LSTM layer with hidden size 10 below the input layer, followed by a dropout layer with dropout rate 0.2, another LSTM layer with hidden size 3, another dropout layer with dropout rate 0.2 and finally a dense output layer with 1 node output. The output values are in the range of [0, 1] indicating the binary classification in RUL <= 30 cycles or RUL > 30 cycles. Figure 28 shows the structure of the NormalLSTM.

The ManSplitLSTM has the same input layer dimension of (none, 50, 25). Below the input layer, the ManSplitLSTM has a division into two separate net paths. The division is done in this PoC with (50, 12) and (50, 13). For this purpose, the input data is split into two parts manually, with one side receiving 12 features and the other 13 features. The division is chosen arbitrarily and splits data as equally as possible. The further processing of the two data parts then takes place in two separate subnetworks with the same structure. The dimension of the subnetworks is chosen with an LSTM layer of hidden size 4, a dropout layer with dropout rate 0.2, another LSTM layer with hidden size 2, another dropout layer with dropout rate 0.2. In an additional Concatenation Layer, the results of the two subnetworks are aggregated by concatenating the respective results. The final dense output layer has 4 nodes, each of which represents the classification class RUL and produces the final result RUL. Figure 29 shows the structure of the ManSplitLSTM.

In the PoC, we measured and compared training time and validation accuracy for three runs per model. One run consisted of 50 training epochs. The number of repetitive runs and epochs was determined empirically. The tests were carried out on a standard Desktop PC with Ubuntu operating system. The detailed specification of the test PC can be found in section 6.3.1. All tests were carried out with the simulated data of aircraft gas turbines already described. The result of a processing run, as well as the classification data, provides a RUL as a number of cycles remaining until failure. The measurement of the accuracy of the results carried out in this test is based on the classification of the results into the two result classes RUL <= 30 cycles and RUL > 30 cycles. The task that the neural network has to solve thus corresponds to a binary classification. This approach was adopted from the Predictive Maintenance Template published by the AzureML Team for Microsoft which these PoC is based on (AzureML Team, 2015). The complete data processing pipeline is also taken from the Predictive Maintenance Template and its sources published on GitHub (AzureML Team, 2018). The reason why the training time and not the runtime was measured in the PoC is due to the Keras framework used in the PoC. Keras offers a ready-made and optimised LSTM which was used in this PoC. Since the Keras LSTM model provides the training time but not the forward time as a result parameter, training time was used in the PoC for reasons of simplicity.

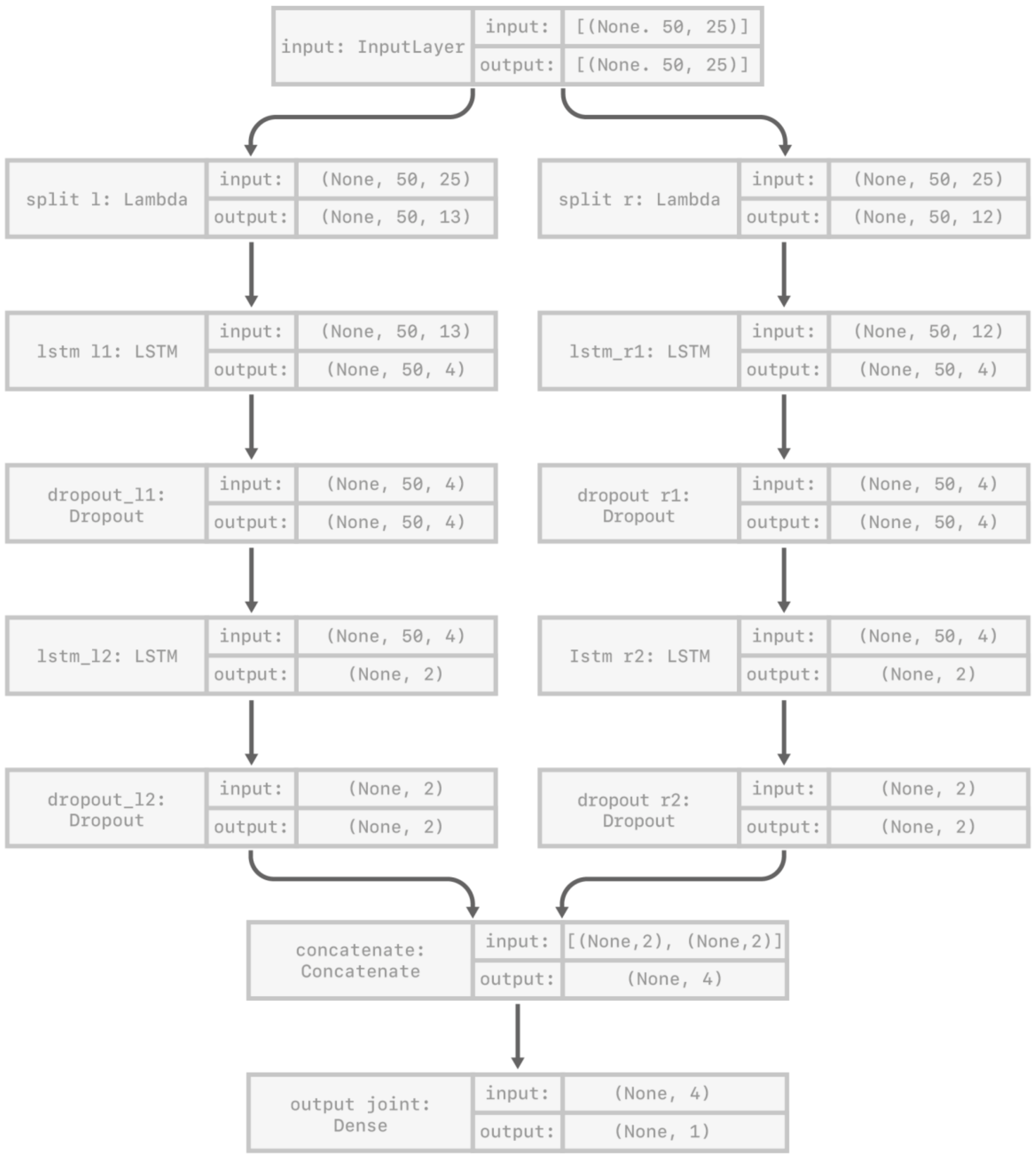


Figure 29 - ManSplitLSTM example with layers and dimensions

Figure 30 shows the results of the three runs per model with 50 training epochs each. On average, the ManSplitLSTM required a training time of 16 seconds per epoch. The NormalLSTM, on the other hand, required an average of 18.3 seconds per epoch. The required training time per epoch of the ManSplitLSTM is thus lower than that of the Standard LSTM in all three runs. Due to the rather randomly chosen experimental design, however, the improvement achieved can only be evaluated qualitatively in the sense of "comparably better" on the basis of the measured training times. Such a determination is also the aim of this PoC. The concrete measured values and the resulting improvement in training time of around 13% should not be used as a quantitative value. Figure 30 shows the measured result data in comparison.

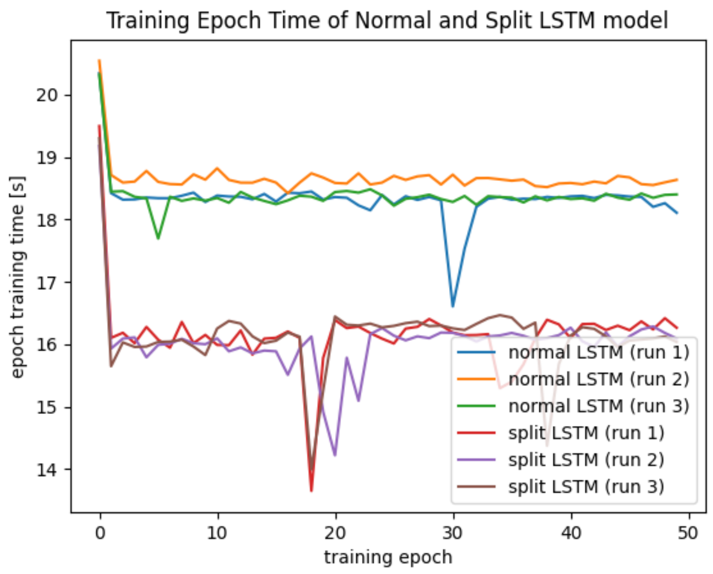


Figure 30 - Training Epoch Time of NormalLSTM and ManSplitLSTM

In the tests conducted, a slower increase in accuracy was observed for the ManSplitLSTM within the first 50 training epochs compared to the NormalLSTM. It is noticeable that the NormalLSTM model reaches a stable level of accuracy after about 20 epochs. This is significantly earlier than with the ManSplitLSTM. The measured accuracy of the ManSplitLSTM increases steadily throughout the training, but much more slowly than the NormalLSTM. After 50 epochs, the ManSplitLSTM does not quite reach the accuracy of the NormalLSTM. This could indicate that the ManSplitLSTM model needs more epochs to fully train. The randomly chosen split of data without taking into account possible correlations could be an explanation for the increased training requirement. Depending on the intended use, this effect could reduce the advantage gained by the reduced epoch training time in practical use.

Figure 31 shows the measured values for the achieved accuracies in comparison. At well over 90%, both models achieve a hight level of accuracy. Looking at similar PdM projects that use LSTM, an accuracy of over 90% can be considered a very competitive value (MathWorks, 2021). The PoC is only intended to provide initial indications and not exact measurement results. How high a possible loss of accuracy due to the splitting of data actually is, cannot be determined on the basis of this PoC. However, this is also not its task. The results of the PoC show that the accuracy of the ManSplitLSTM has the potential to reach a similar accuracy as the NormalLSTM after about 50 epochs but clearly outperforms it in runtime (about 16s compared to 18.3s which is 12,6% faster). From the course of the measurement curves shown in Figure 31 it can be assumed that the ManSplitLSTM can achieve even better accuracies through a higher number of training epochs. A well-chosen division of the data instead of the random division chosen here in the PoC also has the potential to further improve accuracy without affecting runtime.

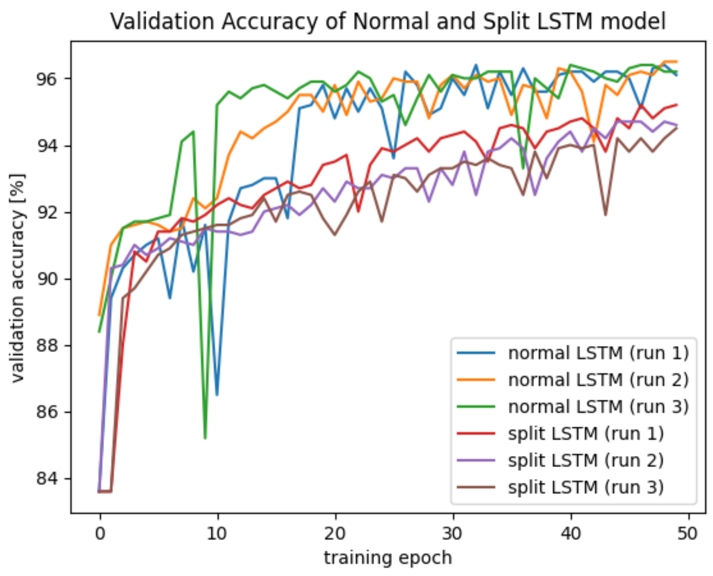


Figure 31 - Developing accuracy of NormalLSTM and ManSplitLSTM

The results obtained from the PoC can be evaluated as successful in relation to the goals of this thesis. The expectations placed on the approach of splitting data in less complex subnetworks were basically confirmed. It is worth mentioned that no explicit parallel processing of the subnetworks was implemented in the PoC. Doing this could further improve the results. In practice, however, the degree of improvement always depends on the use case, the structure chosen for the neural network and the underlying data. Motivated by the positive results of the PoC, the SlicedLSTM approach will be further pursued in the following sections and tested in a quantitative study.

6.3 Experimental Study

This section describes the setup of the quantitative research in the form of an experimental study. The aim of this study is to prove the optimisation achieved by the SlicedLSTM compared to a standard LSTM. The design of the study is based on the general research methodology described in chapter 4 and the quantitative study design described in section 4.4.

In the following, the experimental study and its results will be explained. The study is conducted using a laboratory test with a test setup explained in section 6.3.1. Section 6.3.2 explains the selection of data used. The following sections 6.3.3 and 6.3.4 then explain the details of implementation. The evaluation of the result is done in section 6.3.5.

6.3.1 Setup

The test setup consists of two deep neural networks, one with standard LSTM and one with SlicedLSTM as cell type. The data collection method is the measurement of processing times and accuracies. The meaning of the terms processing time and accuracy in the context of this thesis was defined in section 4.4. The test setup should be such that adjustments to the structure of the nets, the hyperparameter values and the underlying data can be easily made for different test runs. In the course of test execution, a certain number of test and repeat runs are required to obtain reliable results and to identify any outliers.

The basic task of the laboratory test is to determine processing time in relation to the accuracy achieved for the neural nets tested in the comparative test. The idea behind the SlicedLSTM is that the time for processing a given task can be shortened compared to the standard LSTM due to its inner model architecture. Because of the arbitrary division of data in the SlicedLSTM, it is expected that correlations in data will be lost. This will probably lead to a reduction in accuracy. In addition, a certain overhead is to be expected due to the division into parallel networks and their aggregation, which can have a negative effect on the processing time. The processing time measured with the help of the laboratory test in relation to the accuracy are on first hand to prove whether the SlicedLSTM basically offers an advantage over the standard LSTM. Using various test series, it will additionally be shown how the runtimes and accuracies of the SlicedLSTM behave under different conditions and which trade-offs are necessary when using it.

As already mentioned in the PoC, the prerequisite for all comparisons in this thesis is that the accuracy of the results generated with the SlicedLSTM must remain comparable to those of the standard LSTM. An accuracy at well over 90% seems to be standard when looking at similar PdM projects using LSTM (MathWorks, 2021). The test results of this series of tests will show that the loss of accuracy of the SlicedLSTM compared to the standard LSTM is in the range of a few percentage points. However, a precise definition of the term "comparable" will not be given in the context of this laboratory test. The laboratory test will show measurement results of processing time in relation to model accuracy. The assessment of how much loss of accuracy can be accepted for a given improvement in runtime is not intended to be part of this investigation. In the end, such a decision is up to the expert who makes the model selection for a specific project and knows the prevailing framework conditions and objectives there.

The laboratory tests are all carried out on a standard Desktop PC with the following specifications:

CPU	AMD Ryzen 5 3600
Motherboard	MSI B450 TOMAHAWK(MS-TC02)
Memory	32 GB DDR4 SDRAM
GPU	NVIDIA GeForce RTX 2060 Super 8GB
Drives	Samsung SSD 970 Plus 500 GB
Operating System	Ubuntu 21.04

All test runs were carried out on the CPU of the computer and not on its GPU. This is because the performance of the CPU can be better observed. In addition, the test code does not contain any special measures for scheduling on a GPU, which could possibly cause further disturbances in the measurement results. In all tests, only the test program ran on the test computer and thus had the full capacity available. To compensate for interference from operating system processes, each test run was repeated five times. Tests on different hardware were not carried out. Since this laboratory test only aims at relative comparisons of the models and no generally valid statements are made, this was also not considered necessary.

For a run of the experiment, the following steps must be carried out in the given order (see A.4).

1. Definition of the current model in torch_curent_model.py
    a. Change the layer structure in the corresponding class to create a new model structure.
    b. Adapt forward method to new layer structure
2. Adapt torch_lstm_experiment.py
    a. (line 352) set model_type: "reference-model" or "sliced-model".
    b. (line 353) set architecture_string: should reflect model layout
3. Execute Torch_lstm_experiment.py

6.3.2 Data Set

For the experimental study, the same data set is used as of the PoC. All of the data selection criterias described in section 6.2.3 are also valid for the experimental study. As with the PoC not only the data set but the entire data processing pipeline and the source code was included from the reference project (AzureML Team, 2018). Data is synthetically generated data from a NASA project to predict failures using Remaining Useful Lifetime (RUL) as result class.

The data are available in NASA's repository as zip-compressed text files and comprise 1,414 multivariate time series of varying lengths between 19 and 486 operating cycles (data samples). The data has already been divided by NASA into training and test subsets. Each time series in data represents one operational use of a turbine. The turbines are different, but all turbines are of the same type. The training and test data sets each comprise an equal number of 707 time series. Each row (data sample) in data corresponds to one operating cycle of the turbine within a time series. The test data comprise 104,897 samples, the training data 160,360 samples. Each column in data contains a different attribute or a different measured value. The data sets each comprise 26 numerical columns with the individual attributes. A space is used as a separator. The attributes were anonymised so that their meaning cannot be traced. NASA provides the following information: *Column 1: unit number, Column 2: time, in cycles, Column 3: operational setting 1, Column 4: operational setting 2, Column 5: operational setting 3, Column 6: sensor measurement 1, Column 7: sensor measurement 2, ..., Column 26: sensor measurement 21*.

Each turbine starts with a different degree of initial wear and manufacturing variations. However, the level of initial wear and manufacturing variations are unknown and thus represent an unknown in the learning process of the neural network. The initial state of a turbine in a time series must therefore be considered normal. There are three operating settings that have a significant influence on the turbine's performance. These settings are also included in data but are not recognisable due to the anonymisation of data. The data are also contaminated with sensor noise (NASA, 2008b).

Each turbine runs normally at the beginning of a time series and develops an error at some point during the series. In the training set, the error increases until the system fails (last operation cycle in the time series). In the test set, the time series ends some time before the system fails. The goal of a PdM application, or neural network, is to predict the number of operating cycles remaining before failure, hereinafter referred to as Remaining Useful Lifetime (RUL). The RUL determined for a time series of the test set thus corresponds to the number of operating cycles that the engine will continue to run after the last cycle present in data (Saxena, et al., 2008), (Aha, et al., 2017), (Jia, et al., 2018).

The accuracy of the results in the laboratory tests carried out in this thesis is determined by binary classification. The results are assigned to one of the two result classes RUL <= 30 cycles or RUL > 30 cycles. The NASA data set used provides a vector of true values for the remaining useful life (RUL) for each data sample in the test data. This is used for the binary classification and finally the calculation of the achieved accuracy. In terms of validating the results obtained, it is trusted that the data provided by NASA and already used in several projects (including the Prognostics and Health Management (PHM) data competition at PHM'08) are reliable (Jia, et al., 2018). However, even if the reported RUL cycles did not correspond to the real conditions of the underlying turbine type, data would still be suitable for the relative comparison of the two LSTM models Standard LSTM and SlicedLSTM with identical setups and data carried out in the context of this thesis.

Because the data set was already prepared in the Azure project mentioned, no additional data preparation was necessary and the complete data processing pipeline from this project is used.

Nevertheless, the preparation of data should be explained here, even though it was not carried out within this thesis. The following information is mainly taken from this project (AzureML Team, 2018).

The raw data are divided into 4 data sets. Each data set represents a different turbine mode. The files of the different turbine modes for training, testing and the associated RULs are each distinguished by a file suffix, as follows:

*_FD001
- 100 Trajectories (time series)
- Conditions: ONE (sea level)
- Fault Modes: ONE (High-pressure compressor (HPC) Degradation)

*_FD002
- 260 Trajectories
- Conditions: SIX
- Fault Modes: ONE (HPC Degradation)

*_FD003
- 100 Trajectories
- Conditions: ONE (sea level)
- Fault Modes: TWO (HPC Degradation, Fan Degradation)

*_FD004
- 248 Trajectories
- Conditions: SIX
- Fault Modes: TWO (HPC Degradation, Fan Degradation)

The 4 data sets each for training, test and RUL classification were combined into one data set in this thesis (suffix *_all) to get the largest possible data set for each epoch. In this laboratory test, the epoch time is to be measured. In order to determine runtime differences between the two models, a larger dataset is advantageous. The merged data set then comprises 708 trajectories with the conditions ONE/SIX and fault modes ONE/TWO. Merging the different conditions and fault modes results in an unbalanced data set where certain conditions and fault modes are potentially overrepresented. It can be assumed that this reduces the quality (accuracy) achieved by the models. However, since the present laboratory test is a relative comparison of two models, this disadvantage was accepted as a compromise for more data per epoch. But this compromise is only accepted under the condition that the achieved model quality still reaches a value of over 90% and thus remains comparable to the Azure reference project.

To simplify the later evaluation, an additional feature was introduced in the training data, which defines the RUL class of the current cycle (sample). For this purpose, the number of cycles remaining until the end of the time series and thus the failure of the turbine was calculated for each cycle. This value was then assigned using the binary classification RUL <= 30 cycles or RUL > 30 cycles. The newly introduced label has the value 1 if the cycle is to be assigned to the class RUL <= 30 cycles and 0 otherwise.

Subsequently, all 21 sensor values, as well as the 3 values of the operational settings and the cycle number were normalised to the value range [0, 1]. The cycle number was included in data because it not only has an ordering function in the sequence of the data, but also provides an indication of the operating time of the turbine.

The time series of data prepared in this way were then converted into feature sequences with a fixed length of 50 cycles per sequence. Time series shorter than the selected window were discarded. For example, a time series with 150 cycles $z = z^1, z^2, \dots, z^{150}$ thus results in the following 101 feature sequences:

- $z^1, \dots, z^{50}$
- $z^2, \dots, z^{51}$
- …
- $z^{101}, \dots, z^{150}$

In general, dividing a series of *n* cycles into sequences of window length *k* results in *x* as the number of sequences defined as follows:

$$x = \begin{cases} 0, & n < k \\ n - k + 1, & else \end{cases}$$

In the present laboratory setup, this results in 125.909 sequences. 10% of those sequences are randomly chosen for validation. A validation rate between 10% and 20% is a common order of magnitude and has been chosen in other projects using the same NASA data set (Liu, et al., 2022). The feature sequences represent the live processing of a data stream with a sliding window of size 50. In each cycle, the new data arriving is processed together with data from the 49 previous cycles.

6.3.3 LSTM Models

In the following, the implementation of the LSTM models for the experimental study in the form of a laboratory experiment will be explained. In the study design in section 4.4 it was explained that it is recommended to use similar studies and their results as a comparison in order to avoid errors (Duckett, 2021). This recommendation was also taken into account when implementing the LSTM models in this thesis. Therefore, a suitable implementation of a standard LSTM is selected first. The implementation of the SlicedLSTM will then be done by adapting and extending this code base. Such an implementation is preferable to an own development by the author, if only to avoid errors and reduce effort. The implementation of both models on the same code base also supports the comparability of the two models.

The PoC described in section 6.2.3 of this thesis was implemented using the open source Deep Learning library Keras (Chollet, et al., 2022) and a TensorFlow backend (Google Brain, 2022). By using these libraries, the implementation effort for the PoC was greatly reduced. Keras offers highlevel

APIs with which a neural network with LSTM cells can be created and parameterised with a few commands. During construction and dimensioning, Keras offers intensive assistance and optimisation to the user. The Keras library is widely used in many projects.

However, Keras is not considered suitable for the implementation of the experimental study in the form of a laboratory test. The main argument for this is that the APIs of Keras do not allow adaptation of the inner model architecture. The Keras library is open source and therefore all source code is available. However, the implementations of an LSTM cell in Keras is very extensive and includes a lot of code. Many code parts seem to have nothing to do with the actual function of an LSTM and are probably due to the already mentioned ease of use and compatibility with other frameworks. Converting the code of the LSTM implementation in Keras to a SlicedLSTM would, in the author's opinion, involve too much effort and risk. Errors resulting from adjustments to the model architecture would probably be difficult to solve. In addition, it cannot be determined what influence certain code parts assigned to optimisation or framework compatibility would have after manipulation on the comparative measurements to be carried out here.

For the reasons mentioned above, the open-source program library PyTorch was chosen for the implementation of the Standard LSTM and the SlicedLSTM. PyTorch, like Keras, uses Python as programming language and is also geared towards machine learning. PyTorch was developed by Facebook's artificial intelligence research team and is released under the Facebook Open Source Privacy Policy (PyTorch, 2020). However, PyTorch is much more rudimentary compared to Keras and allows the implementation of custom machine learning models.

The search for an open, well-documented implementation of a standard LSTM with PyTorch, preferably used in a comprehensible project context, led to the project *"Building a LSTM by hand on PyTorch"*. The article associated with the project describes the implementation of an LSTM in PyTorch step by step and also deals with variants such as the PeepholeLSTM (Esposito, 2020a). Due to the extensive documentation of the implementation and the freely available sources on GitHub (Esposito, 2020b), this project was chosen as the basis for the implementations in this thesis. The code of the implemented LSTM is clear to understand and each step is comprehensible. It should be mentioned again that the study described here performs a relative comparison between two models. The results are only valid in relation and under the defined test conditions and cannot be generalised. From the author's point of view, the selection of the mentioned implementation is justifiable on the basis of the listed reasons. The SlicedLSTM will be developed by extending the selected PyTorch code of the standard LSTM. It can therefore be assumed that possible disadvantages or errors in the included implementation apply to both models and do not affect the relative comparison.

In the following, the code-side procedure of the implementation of the SlicedLSTM will be explained. Appendix A.4 contains the associated code extracts of the SlicedLSTM. The implementation of the SlicedLSTM follows the theoretical specification from section 6.2.2. For implementation reasons, the order of operations of the inner layers in the code of the SlicedLSTM deviates somewhat from its theoretical considerations. In particular, the intermediate storage of partial results and their structuring for further processing is adapted to the use of tensors in PyTorch. One significant difference in the implementation is that the four weight matrixes for forget gate, input gate, tanh gate and output gate of each slice are stored in one matrix. In the theoretical consideration in section 6.2.2., the weight matrices of the individual slices were shown for each gate. Even though the order of the processing steps in the code is slightly different, it corresponds exactly to the theoretical specification from section 6.2.2.

Based on the example 3 in section 6.2.2 with input and hidden size of 3 and a division into split 1 ($\mathrm{x}_1 \quad \mathrm{x}_2 \quad \mathrm{h}_1 \quad \mathrm{h}_2$) and split 2 ($\mathrm{x}_3 \quad \mathrm{h}_3$), the following processing sequence results from an implementation point of view (note: "|" is not part of the formula and only used as visual separator):

$$\begin{aligned}(x_1 \quad x_2 \quad h_1 \quad h_2) \cdot (\boldsymbol{W_I^1} \quad | \quad \boldsymbol{W_F^1} \quad | \quad \boldsymbol{W_G^1} \quad | \quad \boldsymbol{W_O^1}) \\ = ((\mathrm{s_{I1}}, \mathrm{s_{I2}}) \quad (\mathrm{s_{F1}}, \mathrm{s_{F2}}) \quad (\mathrm{s_{G1}}, \mathrm{s_{G2}}) \quad (\mathrm{s_{O1}}, \mathrm{s_{O2}}))\end{aligned}$$

$$(\mathrm{x_3} \quad \mathrm{h_3}) \cdot (\mathrm{W_I^2} \quad | \quad \mathrm{W_F^2} \quad | \quad \mathrm{W_G^2} \quad | \quad \mathrm{W_O^2}) = (\mathrm{s_{I3}} \quad \mathrm{s_{F3}} \quad \mathrm{s_{G3}} \quad \mathrm{s_{O3}})$$

The gate slices are then merged by component-wise concatenating the individual tuples, indicated by the symbol $\oplus$:

$$\begin{aligned}((\mathrm{s_{I1}}, \mathrm{s_{I2}}) \quad (\mathrm{s_{F1}}, \mathrm{s_{F2}}) \quad (\mathrm{s_{G1}}, \mathrm{s_{G2}}) \quad (\mathrm{s_{O1}}, \mathrm{s_{O2}})) \oplus (\mathrm{s_{I3}} \quad \mathrm{s_{F3}} \quad \mathrm{s_{G3}} \quad \mathrm{s_{O3}}) \\ \Longrightarrow ((\mathrm{s_{I1}}, \mathrm{s_{I2}}, \mathrm{s_{I3}}) \quad (\mathrm{s_{F1}}, \mathrm{s_{F2}}, \mathrm{s_{F3}}) \quad (\mathrm{s_{G1}}, \mathrm{s_{G2}}, \mathrm{s_{G3}}) \quad (\mathrm{s_{O1}}, \mathrm{s_{O2}}, \mathrm{s_{O3}}))\end{aligned}$$

Subsequently, the ConnectionLayer generates the respective output, shown here for the forget gate:

$$(\mathrm{s_{F1}}, \mathrm{s_{F2}}, \mathrm{s_{F3}}) \cdot \begin{pmatrix} u_{\mathrm{F11}} & u_{\mathrm{F12}} & u_{\mathrm{F13}} \\ u_{\mathrm{F21}} & u_{\mathrm{F22}} & u_{\mathrm{F23}} \\ u_{\mathrm{F31}} & u_{\mathrm{F32}} & u_{\mathrm{F33}} \end{pmatrix} = (f_1 \quad f_2 \quad f_3)$$

The following are for further explanation of the demonstrated implementation in the gates and refer to the first slice of a SlicedLSTM with input size 4 and hidden size 3:

$$\mathbf{x^t} = (x_1 \quad x_2 \quad x_3 \quad x_4)^t$$

$$\text{and } \mathbf{h^{t-1}} = (h_1 \quad h_2 \quad h_3)^{t-1}$$

As mentioned, the input data is stored in a tensor in such a way that a weight matrix with all weights of all gates per slice is created. The storage is done within the slice separately for input data and hidden data. The dimension of the tensor of the input weights therefore is $(l_i) \times (4 \cdot l_h)$ with l_i as size of the input vector and l_h as size of the hidden vector (i,n $\subset \mathbb{N}$) The tensor of the hidden weights therefore is $(l_h) \times (4 \cdot l_h)$. The vector matrix product of input data and weight matrix in slice 1 for forget gate (f), input gate (i), tanh gate (g) and output gate (o) looks as follows:

$$(\mathrm{x}_1 \quad \mathrm{x}_2 \quad \mathrm{x}_3 \quad \mathrm{x}_4) \cdot \begin{pmatrix} w_{\mathrm{f}11} & w_{\mathrm{f}12} & w_{\mathrm{f}13} & | & w_{\mathrm{i}11} & w_{\mathrm{i}12} & w_{\mathrm{i}13} & | & w_{\mathrm{g}11} & \cdots & w_{\mathrm{g}13} & | & w_{\mathrm{o}11} & \cdots & w_{\mathrm{o}13} \\ w_{\mathrm{f}21} & w_{\mathrm{f}22} & w_{\mathrm{f}23} & | & w_{\mathrm{i}21} & w_{\mathrm{i}22} & w_{\mathrm{i}23} & | & \vdots & \ddots & \vdots & | & \vdots & \ddots & \vdots \\ w_{\mathrm{f}31} & w_{\mathrm{f}32} & w_{\mathrm{f}33} & | & w_{\mathrm{i}31} & w_{\mathrm{i}32} & w_{\mathrm{i}33} & | & \vdots & \ddots & \vdots & | & \vdots & \ddots & \vdots \\ w_{\mathrm{f}41} & w_{\mathrm{f}42} & w_{\mathrm{f}43} & | & w_{\mathrm{i}41} & w_{\mathrm{i}42} & w_{\mathrm{i}43} & | & w_{\mathrm{g}41} & \cdots & w_{\mathrm{g}43} & | & w_{\mathrm{o}41} & \cdots & w_{\mathrm{o}43} \end{pmatrix}$$

$$= (\mathrm{y}_1 \quad \mathrm{y}_2 \quad \mathrm{y}_3 \quad | \quad \mathrm{y}_4 \quad \mathrm{y}_5 \quad \mathrm{y}_6 \quad | \quad \mathrm{y}_7 \quad \mathrm{y}_8 \quad \mathrm{y}_9 \quad | \quad \mathrm{y}_{10} \quad \mathrm{y}_{11} \quad \mathrm{y}_{12}) = \boldsymbol{y_1}$$

(Note: According to the syntax defined in this document for the meaning of normal font and bold font, y_1 is a scalar value and $\boldsymbol{y_1}$ is a vector.)

The weight matrix of the hidden data in slice 1 for forget gate (f), input gate (i), tanh gate (g) and output gate (o) looks as follows:

$$(\mathrm{h}_1 \quad \mathrm{h}_2 \quad \mathrm{h}_3) \cdot \begin{pmatrix} w'_{\mathrm{f}11} & \cdots & w'_{\mathrm{f}13} & | & w'_{\mathrm{i}11} & \cdots & w'_{\mathrm{i}13} & | & w'_{\mathrm{g}11} & \cdots & w'_{\mathrm{g}13} & | & w'_{\mathrm{o}11} & \cdots & w'_{\mathrm{o}13} \\ \vdots & \ddots & \vdots & | & \vdots & \ddots & \vdots & | & \vdots & \ddots & \vdots & | & \vdots & \ddots & \vdots \\ w'_{\mathrm{f}31} & \cdots & w'_{\mathrm{f}33} & | & w'_{\mathrm{i}31} & \cdots & w'_{\mathrm{i}33} & | & w'_{\mathrm{g}31} & \cdots & w'_{\mathrm{g}33} & | & w'_{\mathrm{o}31} & \cdots & w'_{\mathrm{o}33} \end{pmatrix}$$

$$= (\mathrm{y}'_1 \quad \mathrm{y}'_2 \quad \mathrm{y}'_3 \quad | \quad \mathrm{y}'_4 \quad \mathrm{y}'_5 \quad \mathrm{y}'_6 \quad | \quad \mathrm{y}'_7 \quad \mathrm{y}'_8 \quad \mathrm{y}'_9 \quad | \quad \mathrm{y}'_{10} \quad \mathrm{y}'_{11} \quad \mathrm{y}'_{12}) = \mathbf{y'_1}$$

The next step is to connect the gate vectors of the input layer and the hidden layer with the bias vector. In the SlicedLSTM code, the resulting gate vector is stored in the variable named $\mathbf{gates}_{\boldsymbol{n}}$.

$$\mathbf{y_1} + \mathbf{y'_1} + \mathbf{b_1} = \mathbf{gates_1}$$

The next step is to apply the activation function:

$$\mathbf{gates_1} = (\mathrm{g}_1 \quad \mathrm{g}_2 \quad \mathrm{g}_3 \quad | \quad \mathrm{g}_4 \quad \mathrm{g}_5 \quad \mathrm{g}_6 \quad | \quad \mathrm{g}_7 \quad \mathrm{g}_8 \quad \mathrm{g}_9 \quad | \quad \mathrm{g}_{10} \quad \mathrm{g}_{11} \quad \mathrm{g}_{12})$$

- Forget gate of the first slice: $\sigma(g_1 \quad g_2 \quad g_3) = \mathbf{f_1}$
- Input gate of the first slice: $\sigma(g_4 \quad g_5 \quad g_6) = \mathbf{i_1}$
- Tanh gate of the first slice: $tanh(g_7 \quad g_8 \quad g_9) = \mathbf{g_1}$
- Outputgate of the first slice: $\sigma(g_{10} \quad g_{11} \quad g_{12}) = \mathbf{o_1}$

All further slices are done in the same way.

All concatenated gate slices are then merged in the ConnectionLayer. In the code of the SlicedLSTM, the gate slices are designated as **total_f**, **total_i**, **total_g** and **total_o**. In the explanation so far, only one slice was considered, which has input data size of 4 units and hidden data size of 3 units. To illustrate the ConnectionLayer, two more slices are given. Slice 2 contains 3 input data units and 2 hidden data units. Slice 3 contains 2 input data units and 1 hidden data unit. The ConnectionLayer for these three slices is then structured in the SlicedLSTM code as shown below.

For a better understanding, the assignment of the vectors $(\mathbf{f_1} \quad \mathbf{f_2} \quad \mathbf{f_3})$ to the individual values $(\mathrm{f}_1 \quad \mathrm{f}_2 \quad \mathrm{f}_3 \quad \mathrm{f}_4 \quad \mathrm{f}_5 \quad \mathrm{f}_6)$ is highlighted in colour in the following at the forget gate (**total_f**). The values of slice 1 $(\mathbf{f_1})$ are highlighted in green colour, values belonging to slice 2 $(\mathbf{f_2})$ in turquoise and the values of slice 3 $(\mathbf{f_3})$ in grey. The resulting vector $\mathbf{f_t}$ is denoted by **f_t** in the code of the SlicedLSTM. The biases have not been specified for reasons of better readability, but they do exist.

$$\textbf{total\_f} = (\mathbf{f_1} \quad \mathbf{f_2} \quad \mathbf{f_3}) = (\mathrm{f}_1 \quad \mathrm{f}_2 \quad \mathrm{f}_3 \quad \mathrm{f}_4 \quad \mathrm{f}_5 \quad \mathrm{f}_6)$$

$$(\mathrm{f}_1 \quad \mathrm{f}_2 \quad \mathrm{f}_3 \quad \mathrm{f}_4 \quad \mathrm{f}_5 \quad \mathrm{f}_6) \cdot \begin{pmatrix} u_{\mathrm{f}11} & \cdots & u_{\mathrm{f}16} \\ \vdots & \ddots & \vdots \\ u_{\mathrm{f}61} & \cdots & u_{\mathrm{f}66} \end{pmatrix} = \mathbf{f_t}$$

The other gates are then carried out analogously:

$$\textbf{total\_i} = (\mathbf{i_1} \quad \mathbf{i_2} \quad \mathbf{i_3}) = (\mathrm{i}_1 \quad \mathrm{i}_2 \quad \mathrm{i}_3 \quad \mathrm{i}_4 \quad \mathrm{i}_5 \quad \mathrm{i}_6)$$

$$(i_1 \quad i_2 \quad i_3 \quad i_4 \quad i_5 \quad i_6) \cdot \begin{pmatrix} u_{\mathrm{i}11} & \cdots & u_{\mathrm{i}16} \\ \vdots & \ddots & \vdots \\ u_{\mathrm{i}61} & \cdots & u_{\mathrm{i}66} \end{pmatrix} = \boldsymbol{i}_\mathbf{t}$$

$$\textbf{total\_g} = (\mathbf{g_1} \quad \mathbf{g_2} \quad \mathbf{g_3}) = (\mathrm{g}_1 \quad \mathrm{g}_2 \quad \mathrm{g}_3 \quad \mathrm{g}_4 \quad \mathrm{g}_5 \quad \mathrm{g}_6)$$

$$(g_1 \quad g_2 \quad g_3 \quad g_4 \quad g_5 \quad g_6) \cdot \begin{pmatrix} u_{\mathrm{g}11} & \cdots & u_{\mathrm{g}16} \\ \vdots & \ddots & \vdots \\ u_{\mathrm{g}61} & \cdots & u_{\mathrm{g}66} \end{pmatrix} = \boldsymbol{g}_\mathbf{t}$$

$$\textbf{total\_o} = (\mathbf{o_1} \quad \mathbf{o_2} \quad \mathbf{o_3}) = (\mathrm{o}_1 \quad \mathrm{o}_2 \quad \mathrm{o}_3 \quad \mathrm{o}_4 \quad \mathrm{o}_5 \quad \mathrm{o}_6)$$

$$(o_1 \quad o_2 \quad o_3 \quad o_4 \quad o_5 \quad o_6) \cdot \begin{pmatrix} u_{\mathrm{o}11} & \cdots & u_{\mathrm{o}16} \\ \vdots & \ddots & \vdots \\ u_{\mathrm{o}61} & \cdots & u_{\mathrm{o}66} \end{pmatrix} = \boldsymbol{o}_\mathbf{t}$$

The resulting vectors $\mathbf{f_t}, \boldsymbol{i_t}, \boldsymbol{g_t}, \boldsymbol{o_t}$ form the result of the respective gates of the SlicedLSTM for a processing step. These are now processed by component-wise multiplication into the new cell state $\boldsymbol{c_t}$ and the new hidden state $\boldsymbol{h_t}$ just as in the standard LSTM:

$$\boldsymbol{c_t} = \boldsymbol{f_t} \odot \boldsymbol{c_{t-1}} + \boldsymbol{i_t} \odot \boldsymbol{g_t}$$

$$\mathbf{h_t} = \mathbf{o_t} \odot \tanh(\mathbf{c_t})$$

6.3.4 Implementation

The aim of the laboratory test is to compare the newly developed SlicedLSTM with the standard LSTM. The values to be compared are the runtime required to process an epoch and the achieved model quality (accuracy). The measured values determined in this way are to provide quantitative proof that the targeted runtime improvement of the SlicedLSTM compared to the standard LSTM is present and significant. This thesis was originally focused only on the runtime behaviour of the models. During the development of the PoC and the preparation of the laboratory test, the author decided to include the entire training time in the measurements. This includes the processing time at runtime (forward pass) as well as the actual training time (backward pass). In the definition of the quantitative study in section 4.4 of this thesis, this fact was already taken into account in the study design and the reasons for this were explained.

In the following, the conduct of the laboratory tests is described first. Standard LSTM and SlicedLSTM differ in the internal architecture of the cell. The desired optimisation should therefore be independent of the selected model configuration. Therefore, to compare standard LSTM and SlicedLSTM cells multiple model configurations were tested. The first test results show that the two hyperparameters dropout rate and learning rate significantly influence the results. Therefore, different dropout rates and learning rates were experimented with and the best ones were selected for the laboratory test. In this context a model A is considered better than a model B if A has better accuracy or lower runtime or both. The results are analysed using different methods (aggregate functions) to aggregate the measurements taken in the multiple runs for each model configuration. All definitions were made empirically, as is the state of the art in the field of machine learning (Jeyakumar, et al., 2020). Subsequently, the test runs were carried out with the different model configurations and hyperparameters. A certain number of repetitions of the epochs was taken into account in order to allow training and compensate for perturbations and the effects of the stochastic nature of neural networks (Detorakis, et al., 2019). This section ends with the graphical representation and textual explanations of the most important measurement results and its evaluation. The graphics of all test runs carried out can be found in Appendix A.3.

In the search for a suitable model configuration for the laboratory tests for both models, different layer configurations are to be compared and the best selected. The goal was to achieve an accuracy of at least 90%. In machine learning, model configurations are typically found empirically through trial and error (Jeyakumar, et al., 2020). A model configuration is described by the definition of its layers. The layers are of a certain type. In this laboratory test, the layer types used were Dense, LSTM, SlicedLSTM and Dropout. For a Dense layer and for the standard LSTM, the implementation requires the specification of input size and output size (for LSTM hidden size = output size). The SlicedLSTM also requires the input size and hidden size of the individual slices to be specified. Dropout layers must be defined for framework implementation reasons. Their input size and output size is the output size of the previous layer. Dropout layers are only necessary for setting parts of the input values to 0 during training the network in order to mitigate overfitting effects (Jeyakumar, et al., 2020). Therefore, dropout layers are defined in the following layer configurations only by specifying the dropout rate as a percentage. For example, a dropout rate of 20 means that 20% of the input values are randomly set to 0.

The parameters used in the laboratory test to code the layer configuration are as listed in Table 8. The symbol l_i stands for the input size and l_h for the hidden size.

Parameter	**Syntax**	**Example**
Dropout	[Identifier]+[Proportion]	drop20 means 20% dropout rate
Standard LSTM	([l_i]-[l_h])	(25-10) for a layer with input size 25 and hidden size 10
SlicedLSTM	([l_i Slice 1]-[l_h Slice1]_ [l_i Slice 2]-[l_h Slice2] _ … _ [l_i Slice k]-[l_h Slice k])	(5-2_4-1_3-2) defines a SlicedLSTM with Slice 1: l_i =5, l_h = 2, Slice 2: l_i =4, l_h = 1 and Slice 3: l_i =3, l_h = 2. Input size and hidden size of the entire layer are thus the sum of the sizes of the individual slices, here input size =12 and hidden size = 5

Table 8 - Parameters used in the laboratory test to code the layer configuration

The layer configuration of the entire network is then done using the syntax *(coding layer 1)_(coding layer 2), ... (coding the output layer)*. Due to the way neural networks work, a layer *i+1* must always have the output size of layer *i* as its input size. The first layer of the network must have as input size the number of features that enter the model. The output size of the output layer is defined by the desired result class. In the present laboratory test, the result of the model should be a binary classification into the two RUL classes RUL <= 30 cycles and RUL > 30 cycles. Therefore, a dense layer with output size 1 (value range (0, 1)) was always used as the output layer.

The following examples will show the layer configuration syntax used in the laboratory test. Since the output dense layer is always derived from the second last layer, this is omitted in the following syntax definition.

Example 1: Layer configuration (10-3)_drop10_(3-2), this results in the following model:

- Layer 1: Standard LSTM with input size 10 and hidden size 3.
- Layer 2: Dropout Layer with 10% Dropout

- Layer 3: Standard LSTM with input size 3 and hidden size 2
- Output layer: Dense layer with input size 2 (=hidden size layer 3) and output size 1 (always 1)

Example 2: Layer configuration (8-3_5-1)_(4-2)_drop15, this results in the following model:

- Layer 1: SlicedLSTM with Slice1: input size 8 and hidden size 3 and Slice2: input size 5 and hidden size 1.
- Layer 2: Standard LSTM with input size 4 and hidden size 2
- Layer 3: Dropout layer with 15% dropout
- Output layer: Dense layer with input size 2 (=hidden size layer 2) and output size 1

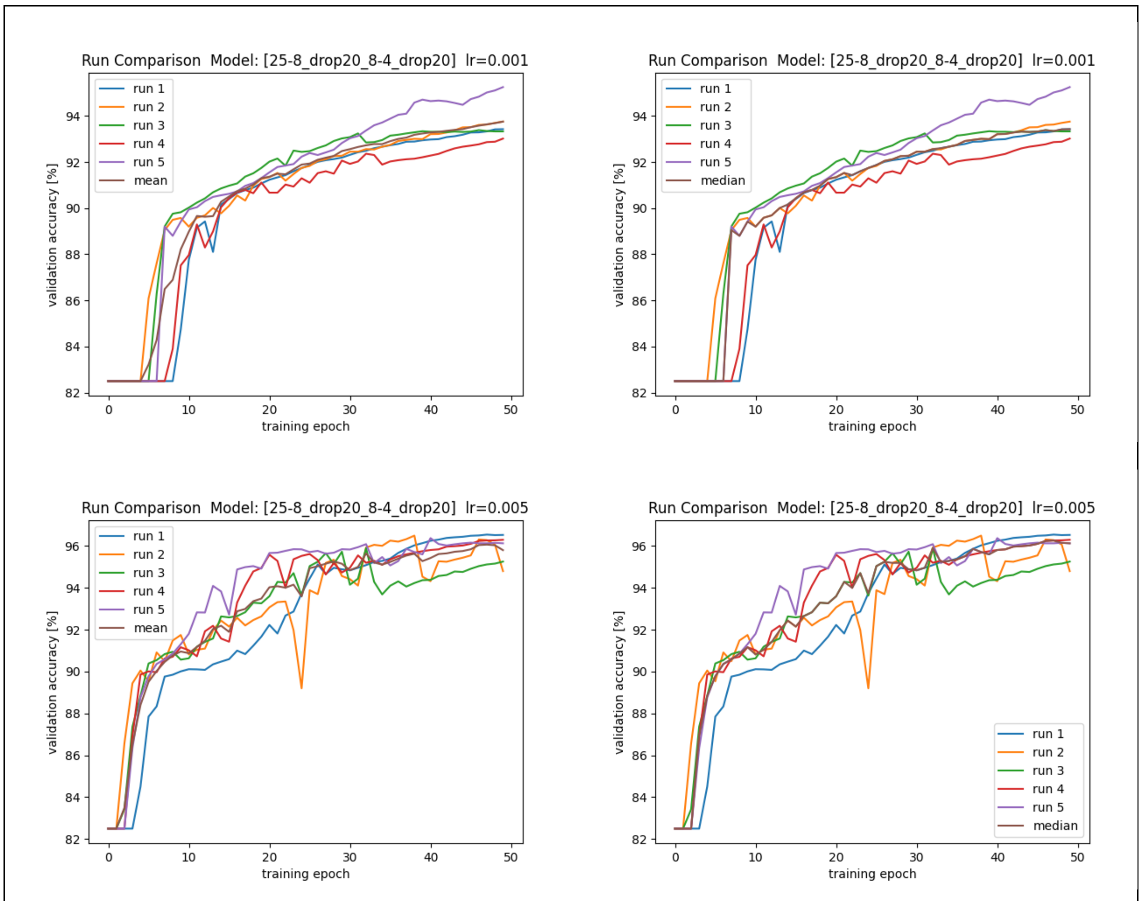
Theoretically, one could also define a SlicedLSTM with only one split. However, the coding of the SlicedLSTM with only one split would not be unambiguous in the chosen syntax, since a standard LSTM is used for a specification such as (4-2). For the comparison test of the two models carried out here, a test with a SlicedLSTM with only one slice is not considered necessary. For simplicity, this potential ambiguity in the syntax was accepted for the laboratory test. In Example 2, a SlicedLSTM and a standard LSTM were combined in one neural network definition. This is possible, but is only intended here to illustrate the chosen syntax. In the laboratory tests carried out, however, such a mixture was not used because of the goal of model comparison. In the laboratory tests described below, an output layer is always used whose input size is derived from the hidden size of the second last layer and which has an output size of 1. As already mentioned, the output layer is not explicitly named in the configurations for reasons of simplicity, but it is always present.

For the empirical search for suitable model configurations, approx. 20 different model configurations were tested for the standard LSTM as a reference model in initial trials. From these, the following four best model configurations (hereinafter referred to as the reference model) were selected for more extensive test runs:

- Reference model 25-8_drop20_8-4_drop20
- Reference model 25-6_drop20_6-3_drop20
- Reference model 25-5_drop20_5-2_drop20
- Reference model 25-5_drop10_5-2_drop10

Tests were then conducted for each reference model over 50 epochs, each with four different learning rates 0.001, 0.005, 0.01 and 0.02. Each test run was repeated five times to minimise interferences and outliers. The results of the five test runs were then determined once using the aggregate function average and once using the median. Due to the four different learning rates and the two aggregate functions, there are eight results per reference model, which are shown in the following plots.

Figure 32 shows the development of the validation accuracy over 50 epochs of all test runs for reference model 25-8_drop20_8-4_drop20. Each plot has the reference model configuration and the learning rate used as heading. Two plots are shown next to each other for each of the five tested learning rates. The respective left plot shows the evaluation based on the aggregate functions mean, the respective right plot based on median. Compared to all four tested configurations, reference model 25-8_drop20_8-4_drop20 developed a high level of accuracy after 30-40 epochs. The best results were achieved with a learning rate of 0.01 (see also section A.3.1). For the model comparison carried out in section 6.3.5, the learning rate of 0.01 is used for this model configuration.



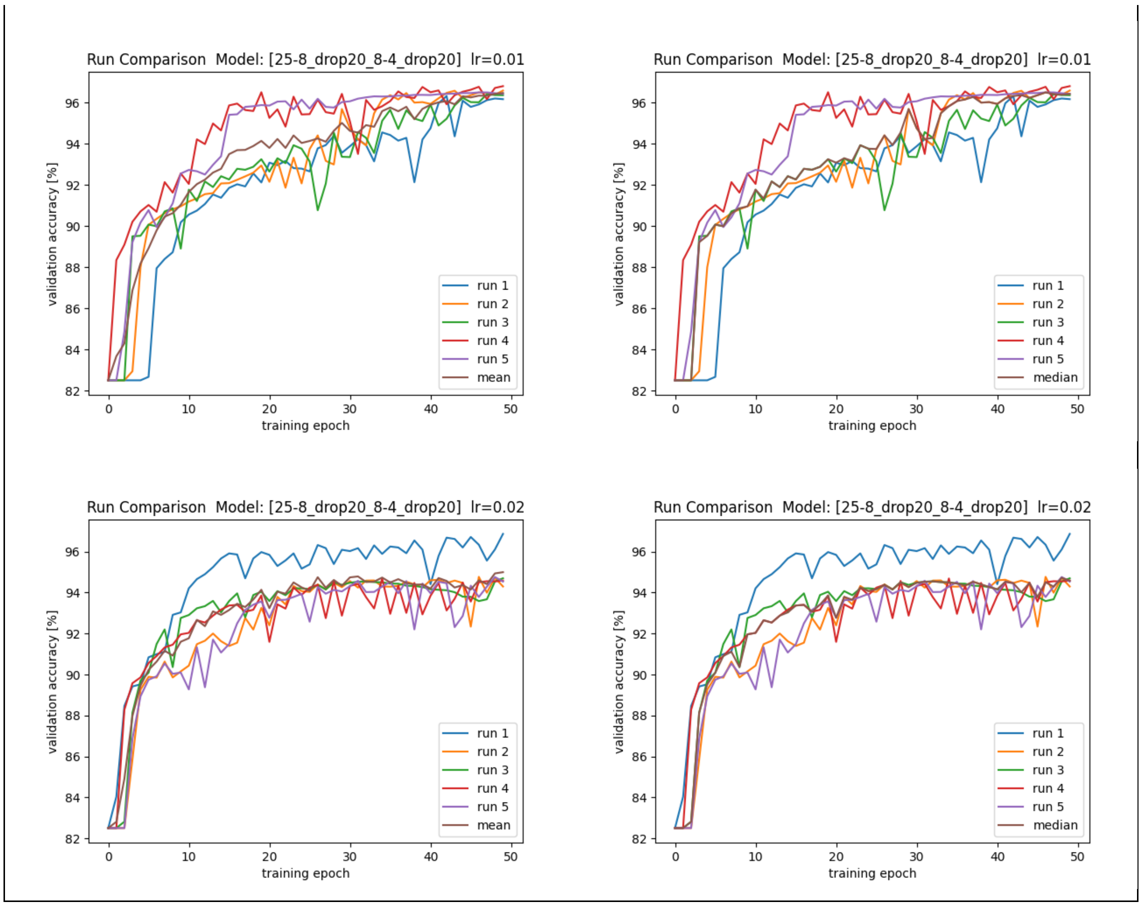


Figure 32 – Accuracy of reference model configuration 25-8_drop20_8-4_drop20

Figure 33 shows a boxplot of the ordinal distribution of the epoch run times of all epochs of all runs of all learning rates for reference model 25-8_drop20_8-4_drop20. The learning rate influences the accuracy, but not the runtime behaviour of the model. Therefore, all tested learning rates are summarised in the boxplot. The Y-axis shows the epoch runtime in seconds and the X-axis one box representation each for the forward times per epoch and the backward times per epoch. For presentation reasons, the total time is not shown. The representation by boxplots was chosen to give a clear impression of the range in which the epoch times of all test variations of a model layout lie and how they are distributed over this range. The orange line in the box shows the median of the epoch times as the value at which the smallest 50 % of the measured epoch times are less than or equal to. The box itself describes the range in which the middle 50 % of the epoch times lie. The lower end of the box, also called the lower quantile, is accordingly the value at which the smallest 25 % of the measured epoch times are smaller or equal. The upper end of the box (upper quantile) is the value at which the smallest 75 % of the measured epoch times are smaller or equal. The length of the box is called the interquartile range (IQR) and shows the dispersion of data in the mean range of values. The end of each of the two antennas, also called the whiskers, define the value closest to 1.5 times the IQR from the lower and upper quantiles. The epoch times shown as circles above and below the antennas are classified as outliers (Hunter, et al., 2022).

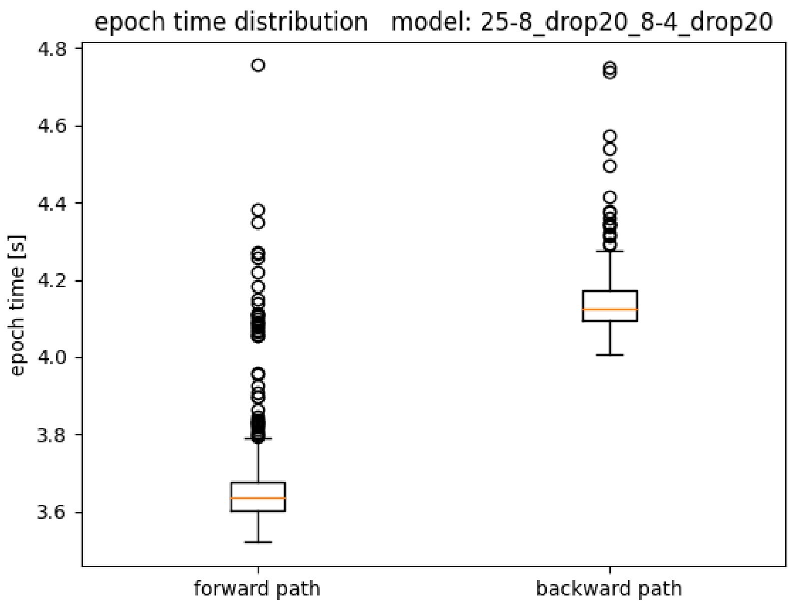


Figure 33 – Epoch time distribution of reference model configuration 25-8_drop20_8-4_drop20

As a result, the reference model 25-8_drop20_8-4_drop20 showed a competitive accuracy in the measurement comparison of the different model configurations but long epoch times compared to the other model configurations measured in this laboratory test. All measured values shown here graphically are listed in concrete form in the evaluation section 6.3.5 in Table 9 to Table 13.

For the following reference models only the test results with the best learning rate in each case are shown below. The diagrams used to choose the best learning rate for each of the model configurations are shown in Appendix A.3.1. The box plots of all test results with all the different learning rates are shown in Appendix A.3.2.

Figure 34 and Figure 35 show the results of the test runs with the best learning rate of 0.005 for reference model 25-6_drop20_6-3_drop20. The accuracy of this reference model develops better than those of the first reference model 25-8_drop20_8-4_drop20. The epoch time of the forward pass is on the same level as for model 25-8_drop20_8-4_drop20, but the epoch time of the backward pass was significantly lower in 25-6_drop20_6-3_drop20. Again, all measured values shown here graphically are listed in concrete form in the evaluation section 6.3.5 in Table 9 to Table 13.

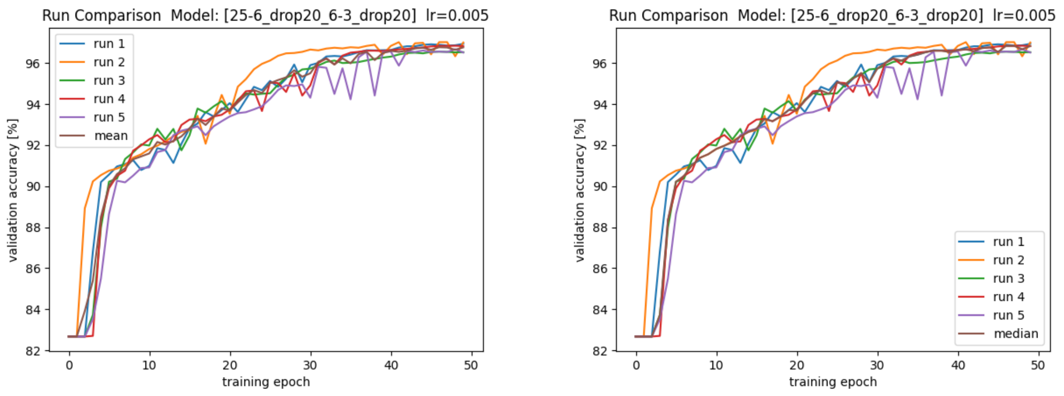


Figure 34 – Accuracy of reference model configuration 25-6_drop20_6-3_drop20

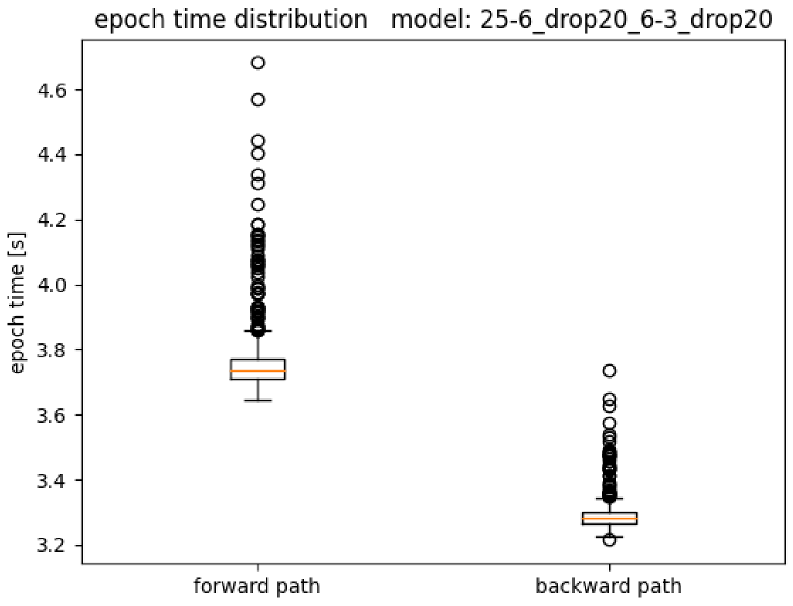


Figure 35 - Epoch time distribution of reference model configuration 25-6_drop20_6-3_drop20

Figure 36 and Figure 37 show the results of the test runs with the best learning rate of 0.01 for reference model 25-5_drop20_5-2_drop20. This model configuration showed better epoch times than the two models tested before but significantly less accuracy than model configuration 25-6_drop20_6-3_drop20. All measured values shown here graphically again listed in concrete form in the evaluation section 6.3.5 in Table 9 to Table 13. The diagrams for selecting the best learning rate are shown in Appendix A.3.1. The diagrams of all test results and with all learning rates can be found in Appendix A.3.2.

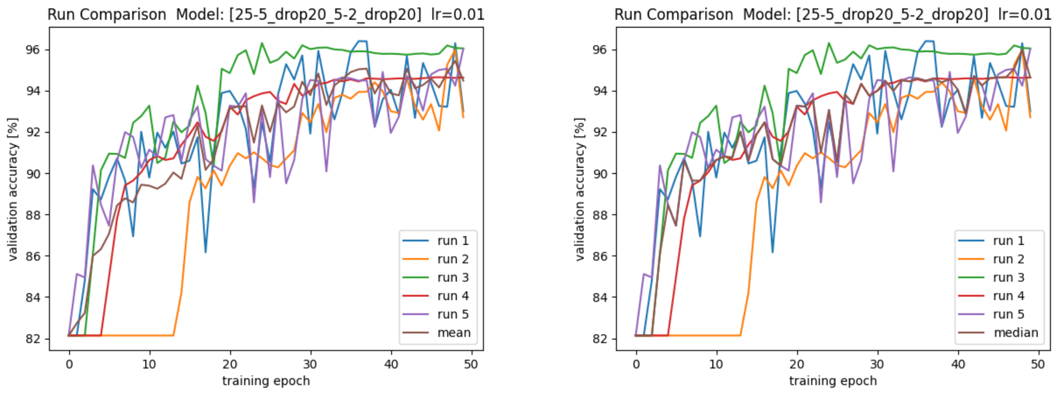


Figure 36 – Accuracy of reference model configuration 25-5_drop20_5-2_drop20

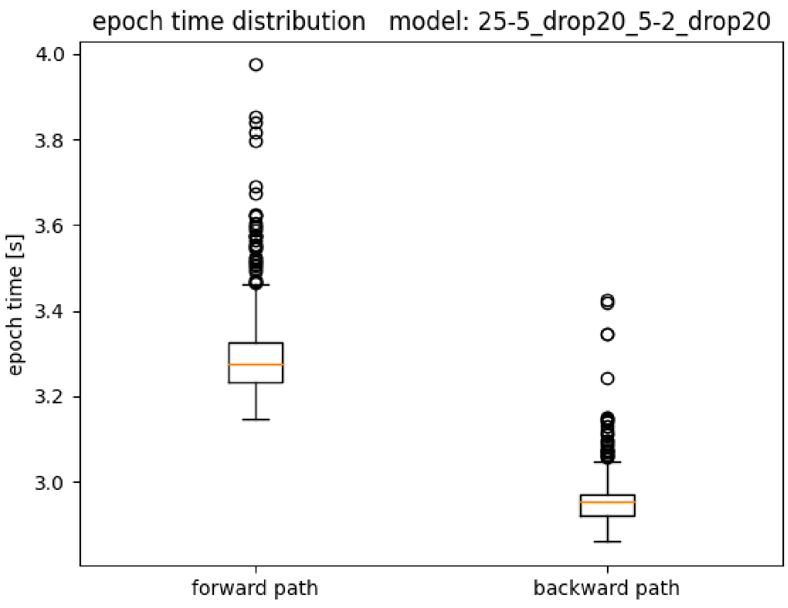


Figure 37 - Epoch time distribution of reference model configuration 25-5_drop20_5-2_drop20

Figure 38 and Figure 39 show the results of the test runs with the best learning rate of 0.01 for reference model 25-5_drop10_5-2_drop10. This reference model showed an early and fast development of accuracy to a maximum of around 96% after 50 epochs. The achieved accuracy corresponds to those of the best reference model 25-6_drop20_6-3_drop20 so far. Figure 38 shows very strong drops in accuracy for run 5 in the range of epochs 20 to 30. Such drops can also be observed in almost all runs of all models, but they are much less pronounced than in run 5 of the model configuration 25-5_drop10_5-2_drop10. These drops are attributed to the stochastic component of the optimiser used (Adam). Depending on the aggregation function used, these drops have a greater or lesser effect on the result. In addition to accuracy, reference model 25-5_drop10_5-2_drop10 was significantly faster than the previous fastest reference model 25-5_drop20_5-2_drop20. In the course of the empirical model finding carried out here, the reference model 25-5_drop10_5-2_drop10 seems to be the best configuration. Reference model 25-5_drop10_5-2_drop10 was therefore chosen as a strong competitor to the SlicedLSTM. All measured values shown here graphically again listed in concrete form in the evaluation section 6.3.5 in Table 9 to Table 13. The diagrams for selecting the best learning rate are shown in Appendix A.3.1. The diagrams of all test results and with all learning rates can be found in Appendix A.3.2.

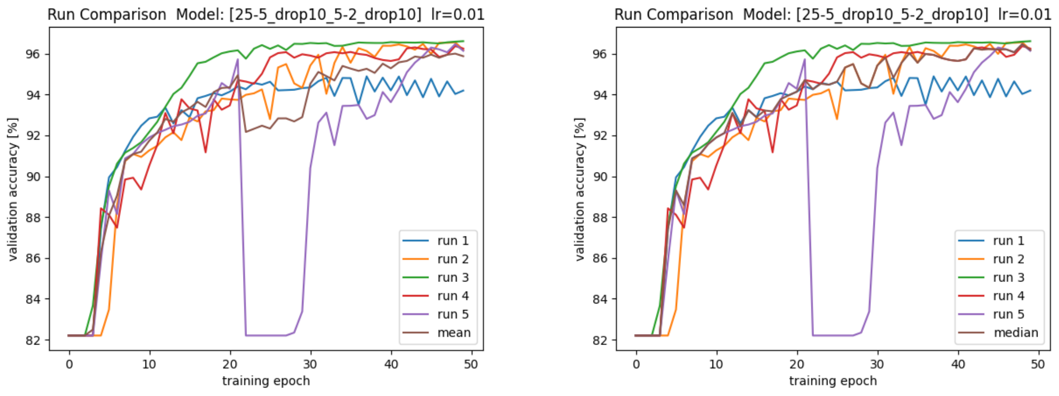


Figure 38 – Accuracy of reference model configuration 25-5_drop10_5-2_drop10

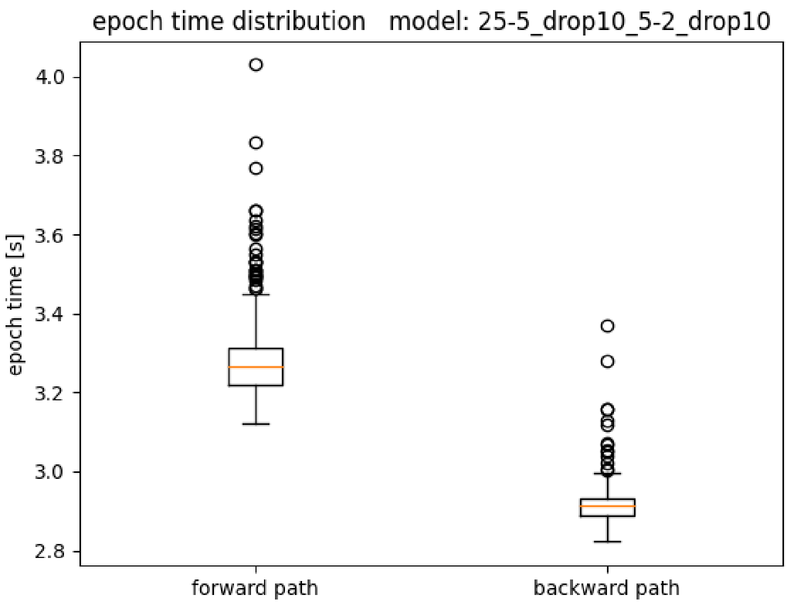


Figure 39 - Epoch time distribution of reference model configuration 25-5_drop10_5-2_drop10

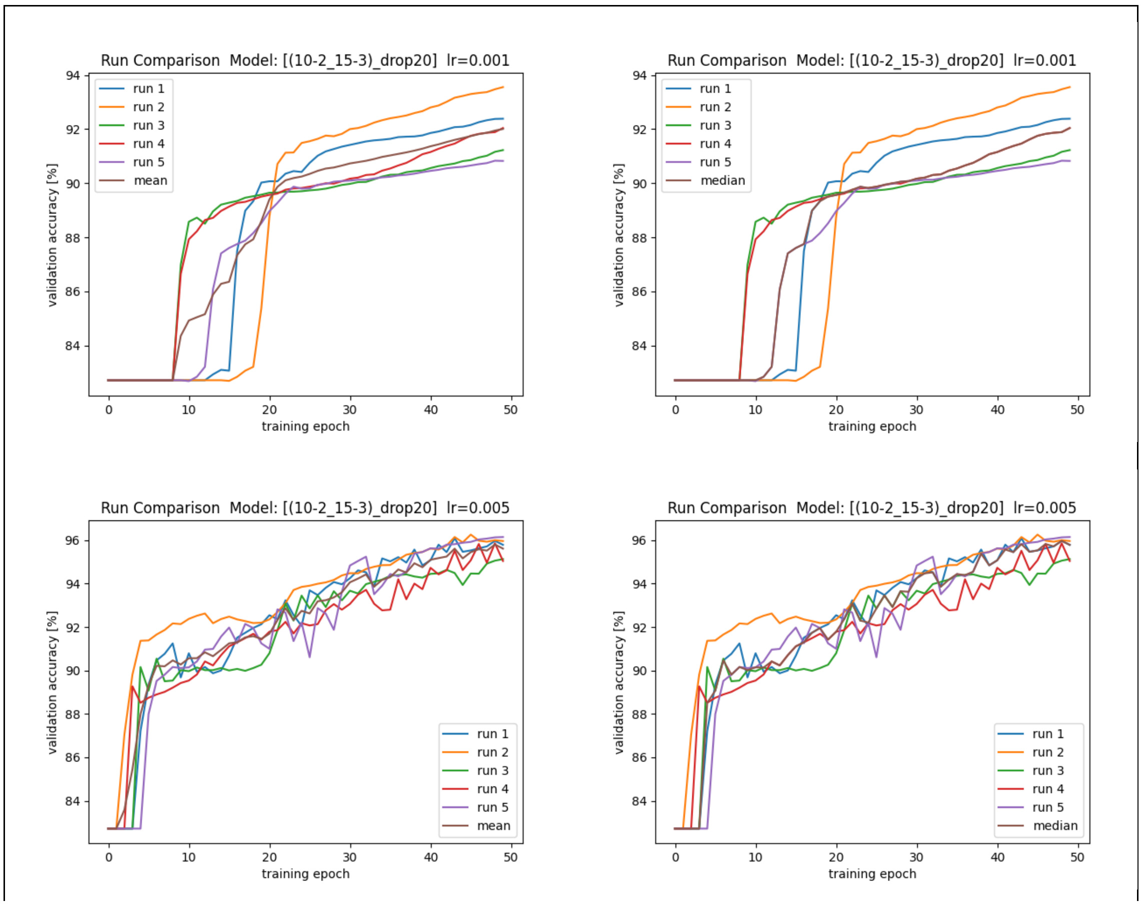
Due to the internal structure of the SlicedLSTM with SplitLayer and ConnectionLayer, one could assume that a neural network with a SlicedLSTM layer is similarly deep as a network with two standard LSTM layers. An empirical search for suitable model configurations was also conducted for the SlicedLSTM. Due to the splits, the SlicedLSTM results in a significantly higher number of possibilities for a model configuration. Therefore, the number of trials for the SlicedLSTM was increased in contrast to that of the standard LSTM. Approximately 40 different model configurations were tested in upstream experiments. Here, tests were first carried out with one SlicedLSTM layer and then the tests were repeated with a network with two SlicedLSTM layers. The network with two SlicedLSTM layers did not achieve significantly better accuracy than the network with one SlicedLSTM layer. Based on these initial findings, 8 model configurations with one SlicedLSTM layer and one model configuration with two SlicedLSTM layers (hereinafter referred to as sliced models) were tested in more depth.

- Sliced model (12-4_13-4)_drop20
- Sliced model (12-8_13-8)_drop20_(8-4_8-4)_drop20
- Sliced model (8-3_9-3_8-3)_drop20
- Sliced model (10-2_15-3)_drop20
- Sliced model (10-1_15-3)_drop20
- Sliced model (10-1_15-2)_drop15
- Sliced model (10-1_15-2)_drop10
- Sliced model (10-1_15-1)_drop10
- Sliced model (11-1_14-1)

The first test with the SlicedLSTM was carried out with the sliced model configuration (12-4_13-4)_drop20. This model configuration pursues a split of the input and hidden vectors in half as far as possible. However, the results achieved were not high enough to make a further comparison with the reference model. The sliced model configuration (12-4_13-4)_drop20 was therefore not investigated further.

The previous tests had already shown that more slices and more layers in SlicedLSTM do not lead to an improvement in accuracy, but do worsen the runtime. To verify this again, two more model configurations were tested. The first defines three layers with the configuration (12-8_13-8)_drop20_(8-4_8-4)_drop20. The results showed, as expected, no significant improvement in accuracy with longer runtime. The second model with the configuration (8-3_9-3_8-3)_drop20 uses only one SlicedLSTM layer, but splits the input into three slices. Even with this configuration, the accuracy was not significantly better with a longer runtime. The model configurations sliced model (12-8_13-8)_drop20_(8-4_8-4)_drop20 and (8-3_9-3_8-3)_drop20 are therefore not competitive and are not considered for further investigations.

After the first model configuration (12-4_13-4)_drop20 with a half division of the slices did not produce competitive results, the configuration (10-2_15-3)_drop20 was tested, dividing the slices unevenly. The tests were again performed with the four learning rates 0.001, 0.005, 0.01 and 0.02 and again five repetitive runs per learning rate. The results of these tests are shown in Figure 40. Compared to the first sliced model (12-4_13-4)_drop20 with an almost half split of the slices, sliced model (10-2_15-3)_drop20 shows significantly better accuracy. According to Figure 41 the measured epoch times of model configuration (10-2_15-3)_drop20 also showed competitive values compared to the reference model. The learning rate 0.01 performed best, as shown in details in Appendix A.3.1. For the model comparison carried out in section 6.3.5, the learning rate of 0.01 is used for sliced model configuration (10-2_15-3)_drop20. All measured values shown here graphically are listed in concrete form in the evaluation section 6.3.5 in Table 9 to Table 13.



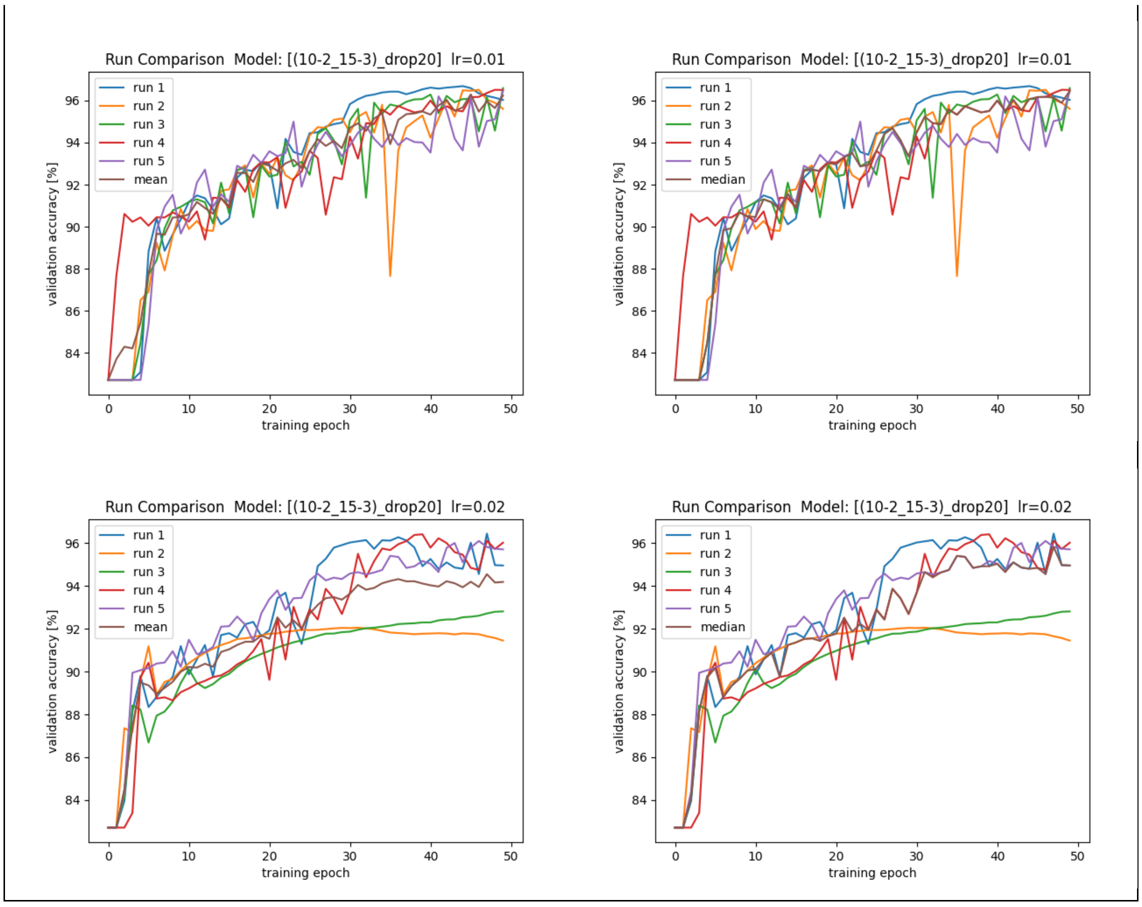


Figure 40 - Accuracy of sliced model configuration (10-2_15-3)_drop20

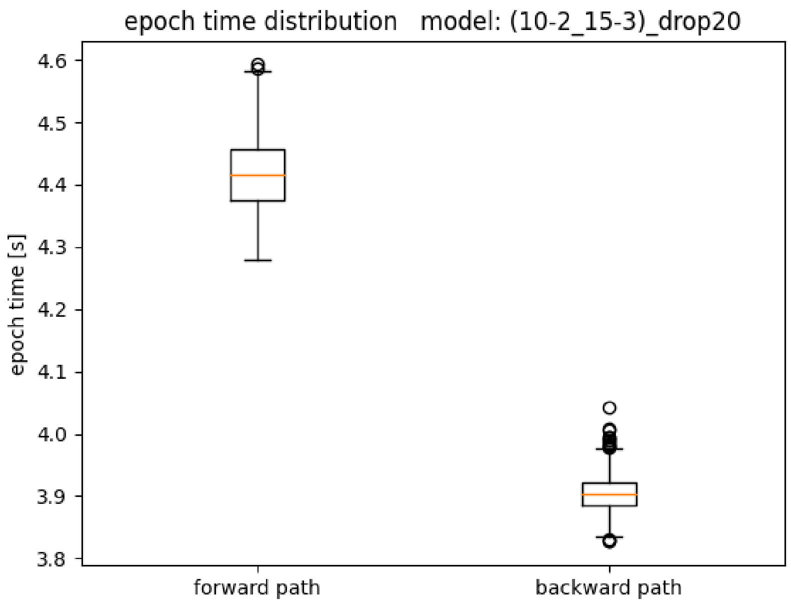


Figure 41 - Epoch time distribution of sliced model configuration (10-2_15-3)_drop20

For the sliced model configuration tested in the following, only the accuracy plots with the best learning rate in each case are shown below. The diagrams used to choose the best learning rate for each of the model configurations are shown in Appendix A.3.3. The diagrams of all test results and with all learning rates are shown in Appendix A.3.4. All measured values are listed in concrete form in the evaluation section 6.3.5 in Table 9 to Table 13.

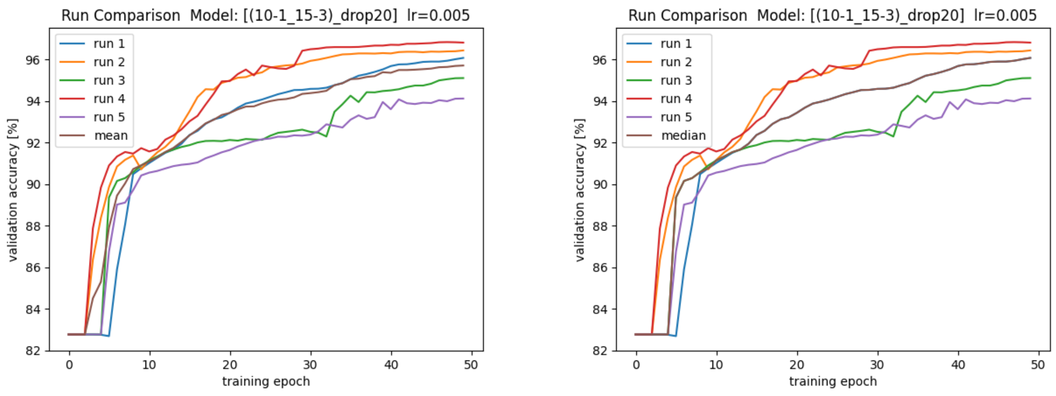


Figure 42 - Accuracy of sliced model configuration (10-1_15-3)_drop20

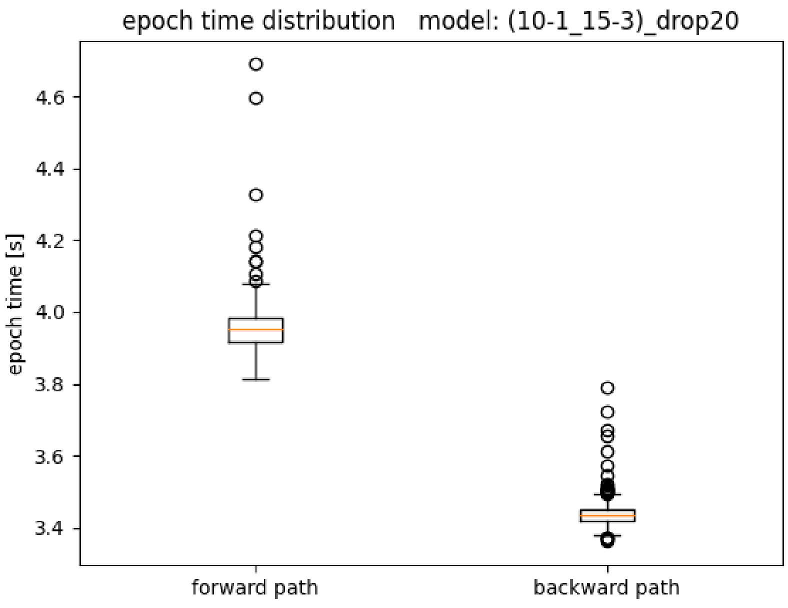


Figure 43 - Epoch time distribution of sliced model configuration (10-1_15-3)_drop20

Motivated by the results with uneven distribution of the slices, an attempt was made with sliced model (10-1_15-3)_drop20 to dimension the model smaller in order to save runtime. For this purpose, the hidden size was reduced by 1. Figure 42 shows the measured accuracy of sliced model (10-1_15-3)_drop20 with the best learning rate of 0.005. The model achieved an accuracy comparable to the previously tested sliced model (10-2_15-3)_drop20 with a smaller size of the network layers and therefore faster epoch times (Figure 43). For comparison with the reference model, sliced model (10-1_15-3)_drop20 is thus better suited than sliced model (10-2_15-3)_drop20.

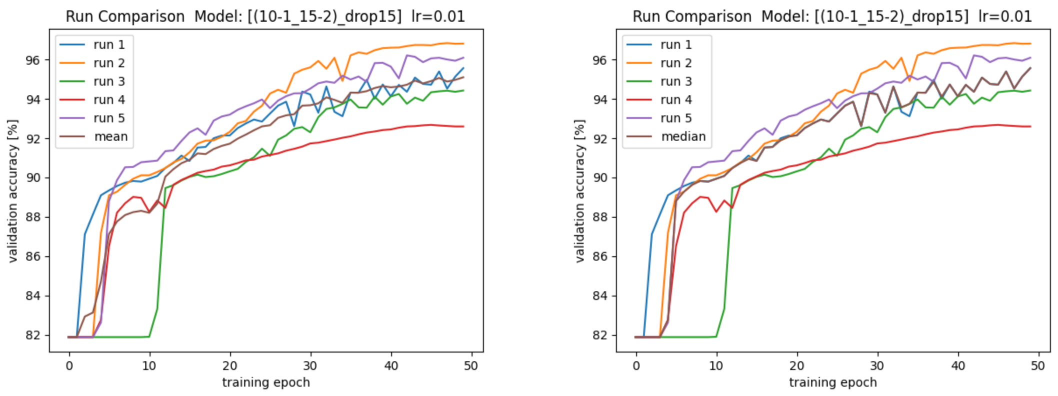


Figure 44 - Accuracy of sliced model configuration (10-1_15-2)_drop15

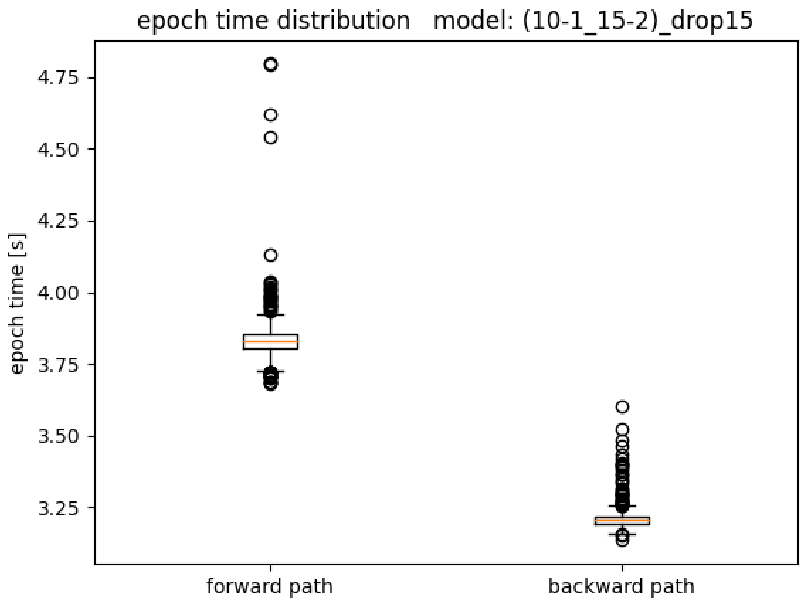


Figure 45 - Epoch time distribution of sliced model configuration (10-1_15-2)_drop15

In the next test step with sliced model (10-1_15-2)_drop15 the size of the hidden layers was reduced by 1 and the dropout rate was reduced from 20% to 15%, due to the lower size of the hidden layers. This model configuration showed faster epoch times but significantly worse accuracy was achieved than with the previous model configuration (10-1_15-3)_drop20. Figure 44 shows the accuracy and Figure 45 the measured epoch times for the model configuration (10-1_15-2)_drop15 with the best learning rate 0.01.

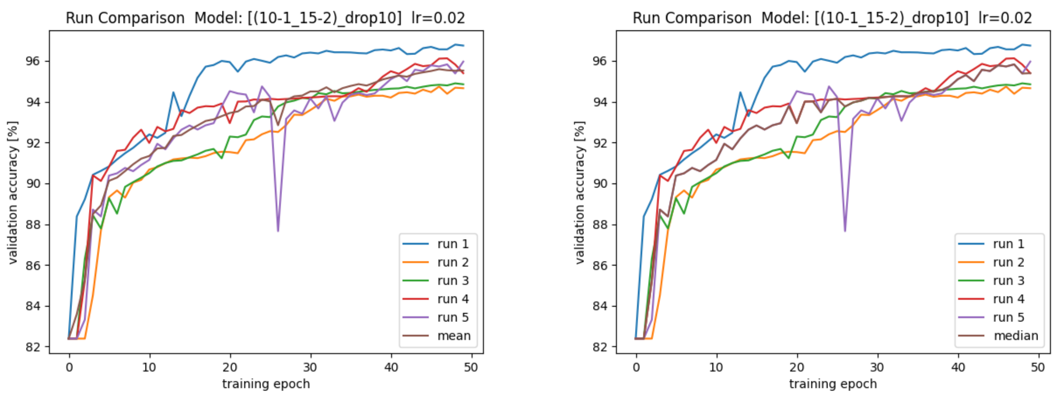


Figure 46 - Accuracy of sliced model configuration (10-1_15-2)_drop10

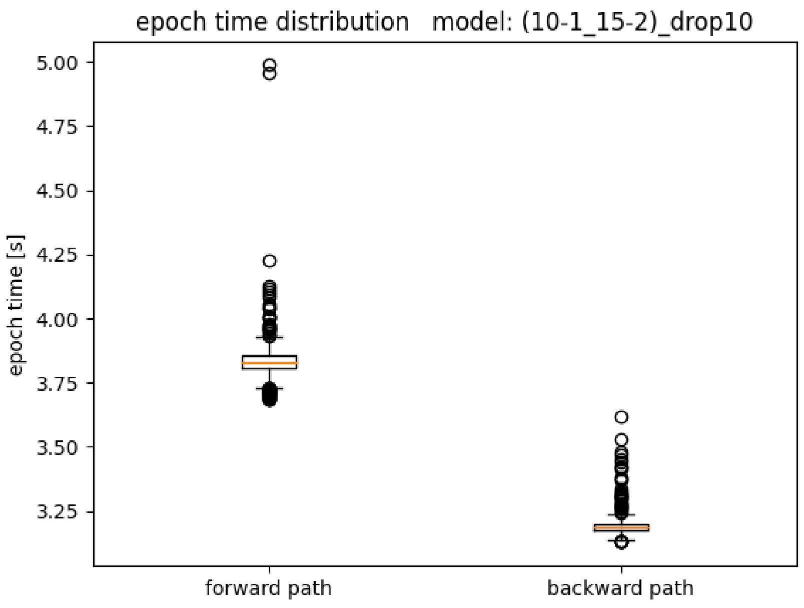


Figure 47 - Epoch time distribution of sliced model configuration (10-1_15-2)_drop10

With the model configuration (10-1_15-2)_drop10, only the dropout rate was further reduced from 15% to 10% compared to sliced model (10-1_15-2)_drop15. As a result, sliced model (10-1_15-2)_drop10 achieved better accuracy comparable to that of sliced models (10-2_15-3)_drop20 and (10-1_15-3)_drop20 at lower epoch times. Figure 46 shows the measured accuracy, Figure 47 the epoch times for sliced model (10-1_15-2)_drop10 with the best learning rate 0.02.

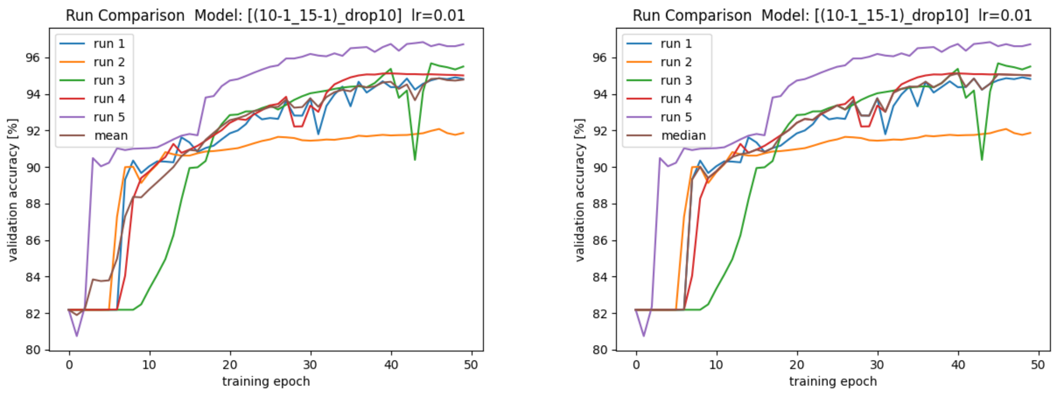


Figure 48 - Accuracy of sliced model configuration (10-1_15-1)_drop10

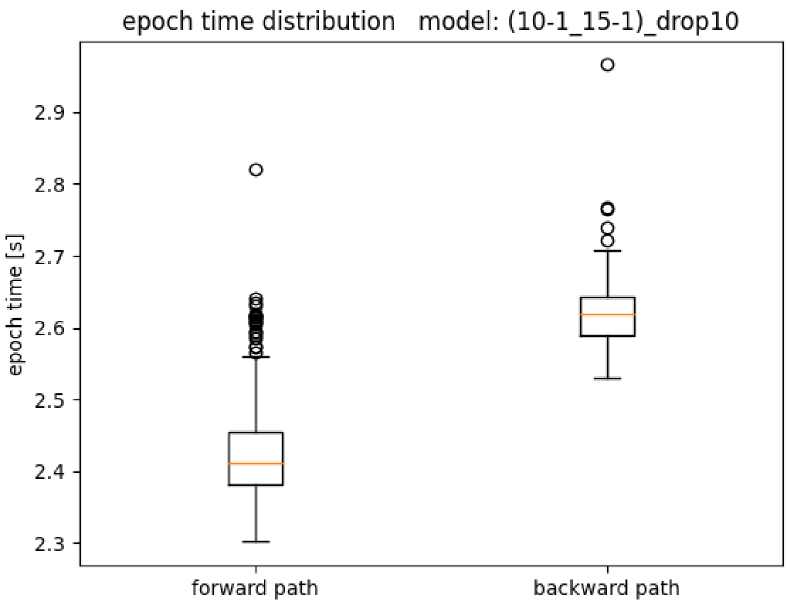


Figure 49 - Epoch time distribution of sliced model configuration (10-1_15-1)_drop10

After the reduction of the dropout rate with reduced hidden size restored competitive accuracy, the hidden size was reduced again for sliced model (10-1_15-1)_drop10. This model provided a accuracy comparable to the previous model configuration (10-1_15-2)_drop10. However, the epoch times were again lower. Figure 48 and Figure 49 show the results for sliced model (10-1_15-1)_drop10 with the best performing learning rate 0.01.

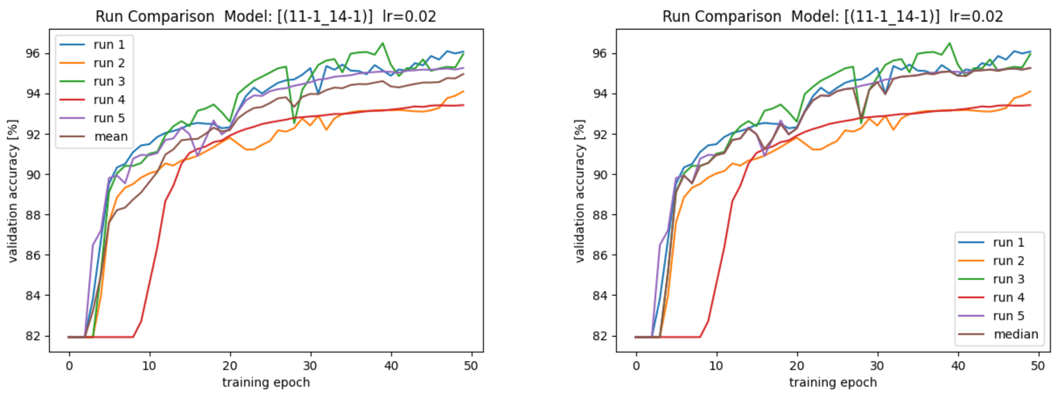


Figure 50 - Accuracy of sliced model configuration (11-1_14-1)

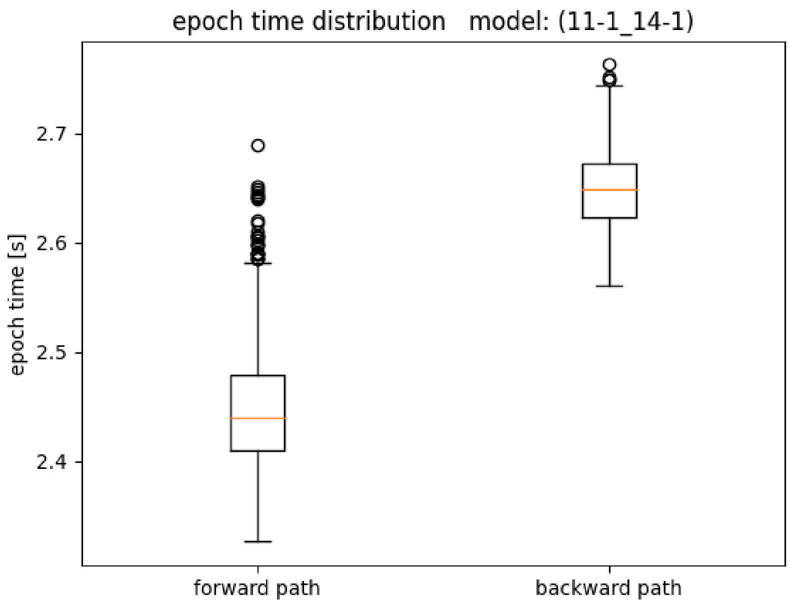


Figure 51 - Epoch time distribution of sliced model configuration (11-1_14-1)

After the hidden size could be minimised, sliced model (11-1_14-1)_drop10 was used to change the distribution of the slices again. The results are comparable with those of the previous sliced model (10-1_15-1)_drop10. In the later comparative tests, the additional aggregate function peak accuracy value was introduced in addition to average and median. In this type of evaluation, however, sliced model (11-1_14-1) delivers significantly worse values than sliced model (10-1_15-1)_drop10. Figure 50 and Figure 51 show the results for sliced model (11-1_14-1) with the best performing learning rate 0.02.

As a result, the model configurations (10-1_15-1)_drop10 and (11-1_14-1) appear to be best suited for comparison with the reference model. Both configurations achieve a comparable accuracy as the reference model (Standard LSTM). The extent to which the runtime of the two SlicedLSTM models can outperform that of the standard LSTM is determined in the following section 6.3.5.

6.3.5 Evaluation

For the evaluation of the measurement results for model comparison, the runs carried out in section 6.3.4 are combined graphically. Three different aggregate functions were used to determine the results for the comparison of the models. These are the average and the median already used in section 6.3.4 and additionally a self defined metric named Peak Validation Accuracy (PVA). Normally, when training neural networks, the value used as the result is not the maximum, but the value that was reached or exceeded with a certain number. This allows the sensitivity of measurement to be regulated, outliers are filtered out and overfitting is avoided. For all measurements in this laboratory test, the value of this number, designated *k-best*, was set to five (10% of the 50 epochs). This number was considered plausible by the author in order to take the best values into account and filter out outliers sufficiently at the same time.

Arithmetic mean and median are proven statistical methods, also in machine learning (Melnikov, et al., 2016). For each of the 50 epochs performed in the experimental study, the mean of the result values of the five replicate runs per learning rate is calculated. The result for the aggregation function mean then is the k-best out of the measured mean values. The median is calculated in the same way. Particular in the figures Figure 44, Figure 48 and Figure 50, the accuracy of the individual runs fluctuates quite strongly in some cases. This shows that the results of the aggregate functions mean and median are noticeably negatively influenced by poorly performing runs. This disregards the motto often followed in the field of machine learning that only the best model counts (Melnikov, et al., 2016). Motivated by this motto, the self defined metric named PVA was defined. In accordance with the approach used to measure mean and median, the PVA is the k-best peak value of the 50 epochs. However, this does not have to be the accuracy at the end of the 50 epochs performed. While mean and median represent an aggregate of the values of all five runs per epoch, PVA always considers only the best value. In this sense, PVA is not an aggregate function.

The following plots show all the comparison models with their required runtime and the achieved accuracy. Figure 52 illustrates the comparative data of the models calculated with the aggregate function mean for the entire epoch time. The corresponding measured values are listed in Table 9. The Standard LSTM no. 9 achieves the highest accuracy with 96.71%. The SlicedLSTM no. 1 is the fastest model with 5.374 seconds for the entire epoch. The fastest SlicedLSTM no. 1 is 26.92% faster and with 94.73% accuracy only 2.05% less accurate than the most accurate Standard LSTM no. 9. Comparing the fastest SlicedLSTM no. 1 (5.374s) with the fastest Standard LSTM, which is the no. 7 with 6.570s, the fastest SlicedLSTM is still 18.20% faster than the fastest Standard LSTM. The accuracy of the fastest SlicedLSTM no. 1 is only 1.18% lower than that of the fastest Standard LSTM Nr. 7 (94.73% to 95.86%). This measurement shows that the SlicedLSTM clearly outperforms the standard LSTM when the entire epoch is considered.

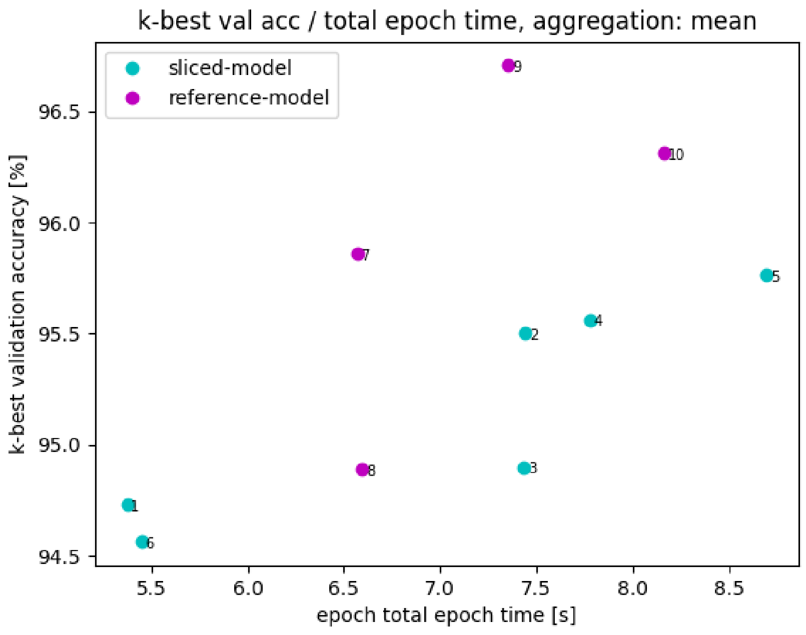


Figure 52 - Model comparison total epoch time with aggregate function mean

Number	Type	Configuration	LR	Accuracy	Time
1	SlicedLSTM	(10-1_15-1)_drop10	0.01	94.73%	5.374s
2	SlicedLSTM	(10-1_15-2)_drop10	0.02	95.50%	7.444s
3	SlicedLSTM	(10-1_15-2)_drop15	0.01	94.90%	7.436s
4	SlicedLSTM	(10-1_15-3)_drop20	0.005	95.56%	7.777s
5	SlicedLSTM	(10-2_15-3)_drop20	0.01	95.76%	8.697s
6	SlicedLSTM	(11-1_14-1)	0.02	94.56%	5.450s
7	Standard LSTM	25-5_drop10_5-2_drop10	0.01	95.86%	6.570s
8	Standard LSTM	25-5_drop20_5-2_drop20	0.01	94.89%	6.595s
9	Standard LSTM	25-6_drop20_6-3_drop20	0.005	96.71%	7.354s
10	Standard LSTM	25-8_drop20_8-4_drop20	0.01	96.31%	8.164s

Table 9 - Model comparison total epoch time with aggregate function mean

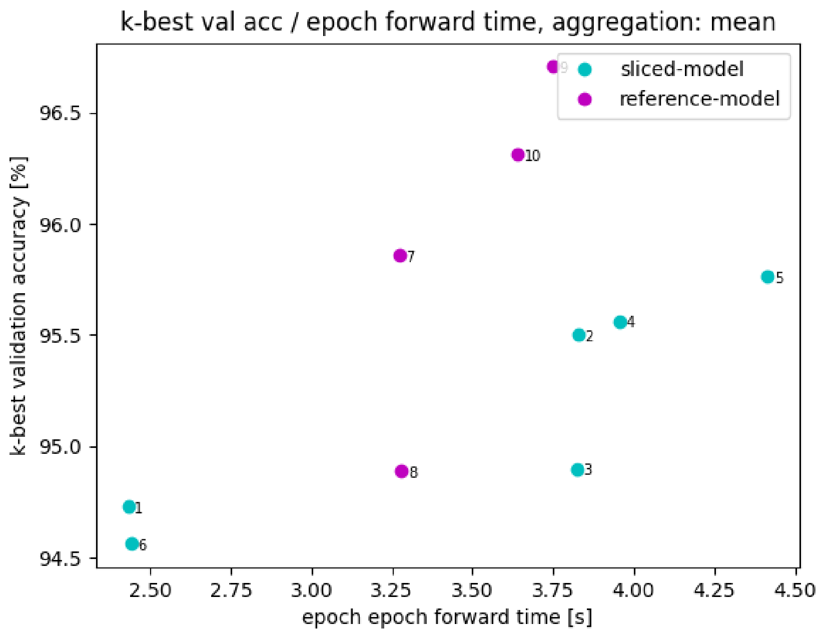


Figure 53 - Model comparison forward time with aggregate function mean

Number	Type	Configuration	LR	Accuracy	Time
1	SlicedLSTM	(10-1_15-1)_drop10	0.01	94.73%	2.431s
2	SlicedLSTM	(10-1_15-2)_drop10	0.02	95.50%	3.828s
3	SlicedLSTM	(10-1_15-2)_drop15	0.01	94.90%	3.825s
4	SlicedLSTM	(10-1_15-3)_drop20	0.005	95.56%	3.955s
5	SlicedLSTM	(10-2_15-3)_drop20	0.01	95.76%	4.415s
6	SlicedLSTM	(11-1_14-1)	0.02	94.56%	2.442s
7	Standard LSTM	25-5_drop10_5-2_drop10	0.01	95.86%	3.273s
8	Standard LSTM	25-5_drop20_5-2_drop20	0.01	94.89%	3.280s
9	Standard LSTM	25-6_drop20_6-3_drop20	0.005	96.71%	3.747s
10	Standard LSTM	25-8_drop20_8-4_drop20	0.01	96.31%	3.639s

Table 10 - Model comparison forward time with aggregate function mean

Figure 53 and Table 10 show the same test series as shown in Figure 52 and Table 9 but here only the times of the forward path. The forward path is the processing time at runtime, which is the focus of this thesis. Aggregation function is still mean. Again, the SlicedLSTM no. 1 is the fastest model and Standard LSTM no. 9 the most accurate. SlicedLSTM no. 1 is 35.12% faster on the forward path and still only 2.05% less accurate than the most accurate Standard LSTM Nr 9. Standard LSTM no. 7 is the fastest reference model. When comparing the fastest SlicedLSTM no. 1 with the fastest Standard LSTM no. 7, SlicedLSTM no. 1 is still 25.73% faster than the Standard LSMT no. 7 although the accuracy is only 1.18% lower (2.431s compared to 3.273s and 94.73% accuracy compared to 95.86%). As with mean, the SlicedLSTM again clearly outperforms the standard LSTM, especially in terms of runtime (forward pass).

Figure 54 and Table 11 again show the same test series but here only the times of the backward path. The SlicedLSTM no. 1 is here also clearly the fastest model during training, Standard LSTM no. 9 still the most accurate. In the backward pass the SlicedLSTM no. 1 is 20.34% faster with only 2.05% lower accuracy than the most accurate standard LSTM no. 9. Compared with the fastest Standard LSTM no. 7, the SlicedLSTM no. 1 is 10.32% faster at only 1.18% lower accuracy (2.616s compared to 2.917s and 94.73% accuracy compared to 95.86%). Even though the training time (backward pass) is not the focus of this thesis, the results show that the SlicedLSTM is able to outperform the standard LSTM even while learning.

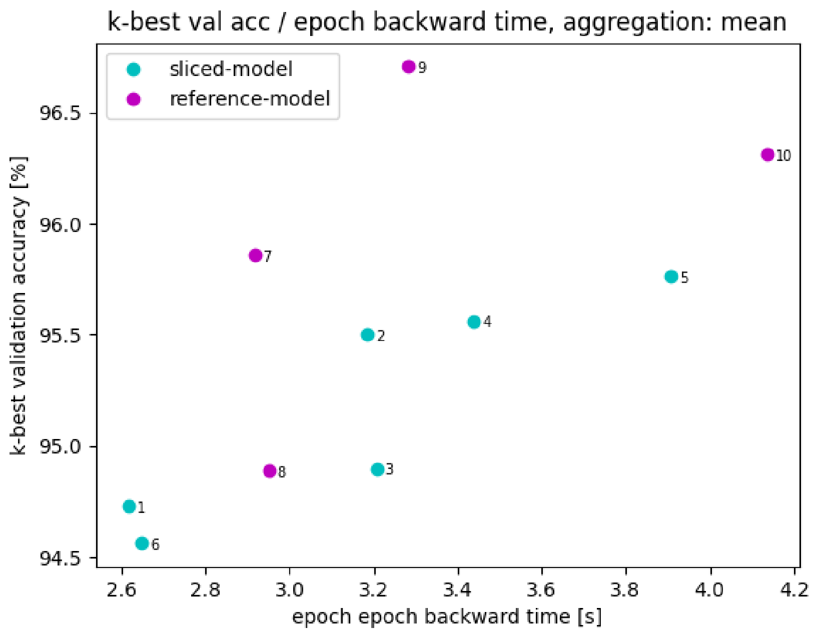


Figure 54 - Model comparison backward time with aggregate function mean

Number	Type	Configuration	LR	Accuracy	Time
1	SlicedLSTM	(10-1_15-1)_drop10	0.01	94.73%	2.616s
2	SlicedLSTM	(10-1_15-2)_drop10	0.02	95.50%	3.186s
3	SlicedLSTM	(10-1_15-2)_drop15	0.01	94.90%	3.207s
4	SlicedLSTM	(10-1_15-3)_drop20	0.005	95.56%	3.438s
5	SlicedLSTM	(10-2_15-3)_drop20	0.01	95.76%	3.906s
6	SlicedLSTM	(11-1_14-1)	0.02	94.56%	2.649s
7	Standard LSTM	25-5_drop10_5-2_drop10	0.01	95.86%	2.917s
8	Standard LSTM	25-5_drop20_5-2_drop20	0.01	94.89%	2.951s
9	Standard LSTM	25-6_drop20_6-3_drop20	0.005	96.71%	3.284s
10	Standard LSTM	25-8_drop20_8-4_drop20	0.01	96.31%	4.138s

Table 11 - Model comparison backward time with aggregate function mean

The following figures and tables show the corresponding results of the comparisons with the aggregate function median. The times for forward, backward and total epoch time are illustrated in Figure 55, Figure 56 and Figure 57 and summarised in Table 12.

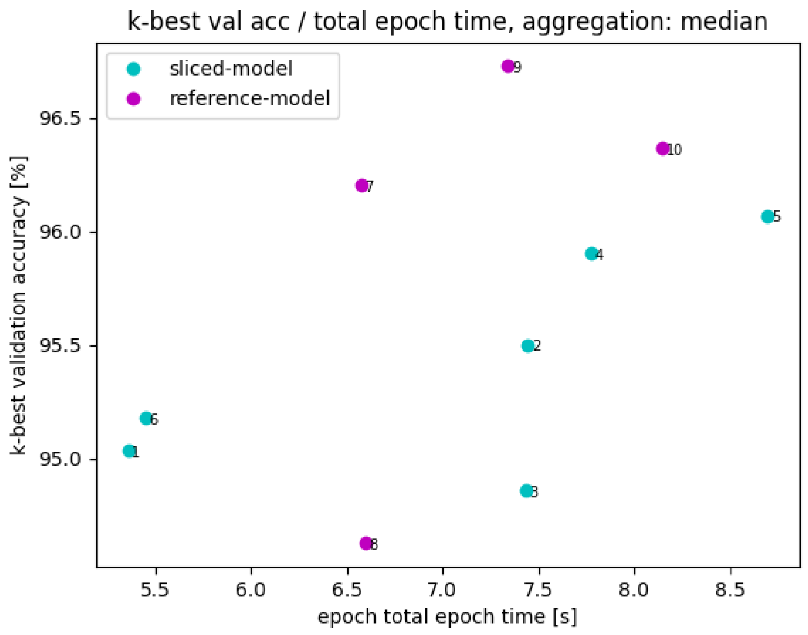


Figure 55 - Model comparison total epoch time with aggregate function median

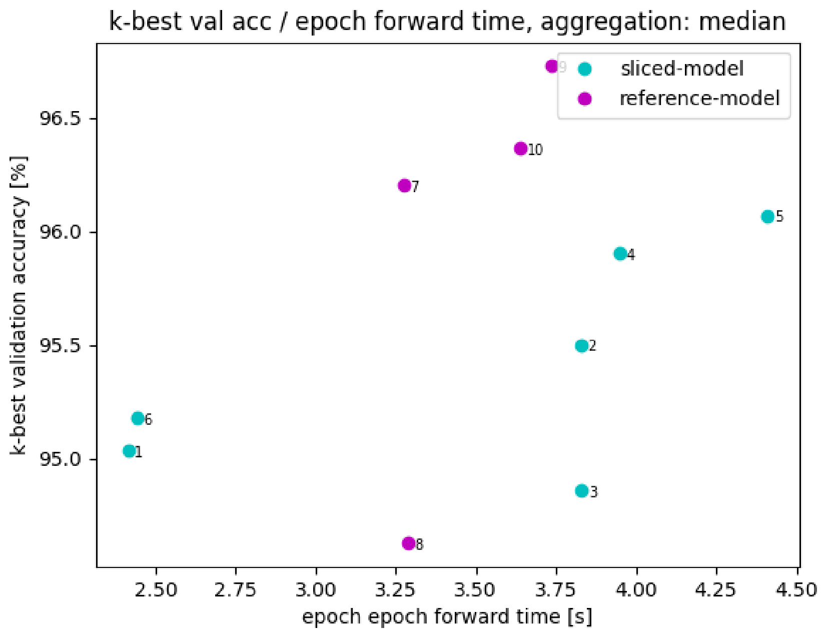


Figure 56 - Model comparison forward time with aggregate function median

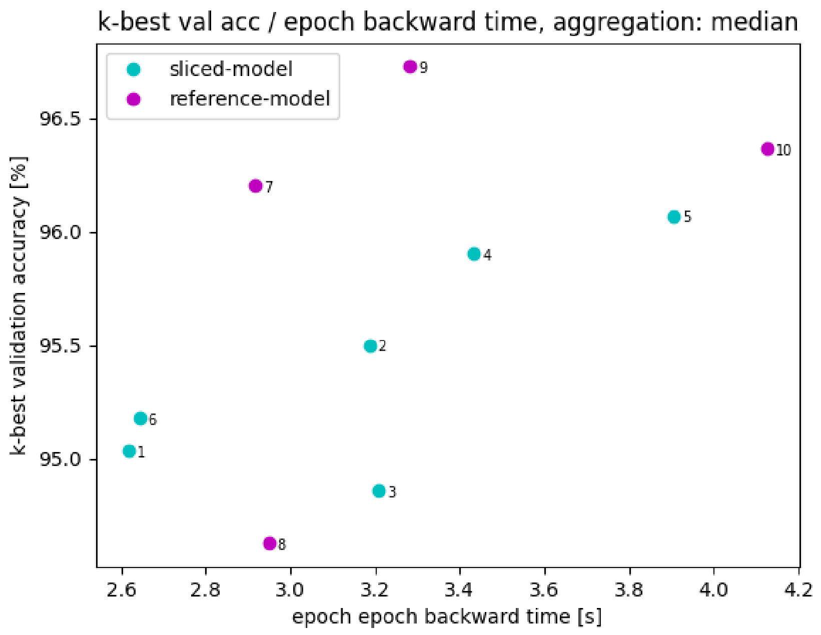


Figure 57 - Model comparison backward time with aggregate function median

Number	Type	LR	Accuracy	Total Epoch	Forward	Backward
1	SlicedLSTM	0.01	94.73%	5.357s	2.414s	2.616s
2	SlicedLSTM	0.02	95.50%	7.446s	3.829s	3.186s
3	SlicedLSTM	0.01	94.90%	7.436s	3.830s	3.209s
4	SlicedLSTM	0.005	95.56%	7.775s	3.949s	3.434s
5	SlicedLSTM	0.01	95.76%	8.699s	4.411s	3.905s
6	SlicedLSTM	0.02	94.56%	5.448s	2.442s	2.644s
7	Standard LSTM	0.01	95.86%	6.573s	3.275s	2.918s
8	Standard LSTM	0.01	94.89%	6.599s	3.287s	2.949s
9	Standard LSTM	0.005	96.71%	7.342s	3.737s	3.282s
10	Standard LSTM	0.01	96.31%	8.149s	3.639s	4.128s

Table 12 - Model comparison forward, backward and total epoch time with aggregate function median

Also when using median as the aggregation function, there is a clear improvement in runtime with the SlicedLSTM compared to the standard LSTM as the reference model. SlicedLSTM no. 1 again is always the fastest. For the total epoch time (see Figure 55 and Table 12), it is 18.5% faster (5.357s) than the fastest standard LSTM no. 7, which needs 6.573s for the same calculation. The accuracy achieved by the sliced LSTM no. 1 with 94.73% again is only 1.18% lower than that of the fastest reference model no. 7 with 95.86%. Comparing the fastest SlicedLSTM with the most accurate Standard LSTM no. 9, it is even 27.04% faster with only 2.05% loss of accuracy (5.357s compared to 7.342s and 94.73% accuracy compared to 96.71%).

In the forward pass and backward pass, SlicedLSTM no. 1 is also the fastest SlicedLSTM whereas Standard LSTM no. 7 is the fastest and Standard LSTM no. 9 the most accurate reference model. In the forward pass (see Figure 56 and Table 12), the SlicedLSTM no. 1 needs 2.414s and is thus 26.29% faster than the fastest reference model no. 7 with 3.275s and 35.40% faster than the most accurate reference model no. 9 with 3.737s. The SlicedLSTM no. 1 completes training in the backward pass (see Figure 57 and Table 12) with 2.616s which is 10.35% faster than the fastest standard LSTM no. 7 with 2.918s and 20.29% faster than the most accurate reference model no. 9 with 3.282s. The performance gain was thus greatest in the forward pass. The improvement in runtime targeted with the SlicedLSTM could thus be demonstrated. The accuracies are always the same for total epoch time, forward pass and backward pass.

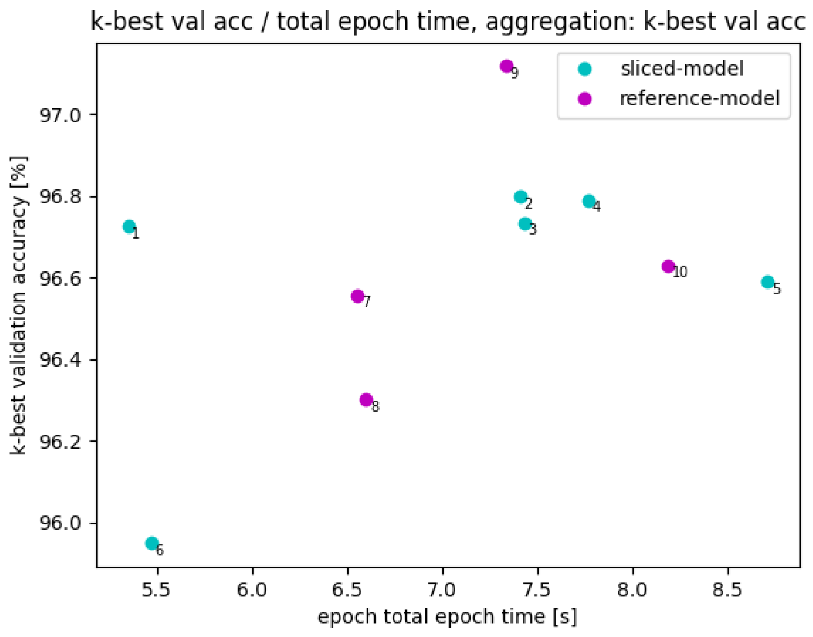


Figure 58 - Model comparison total epoch time with aggregate function PVA

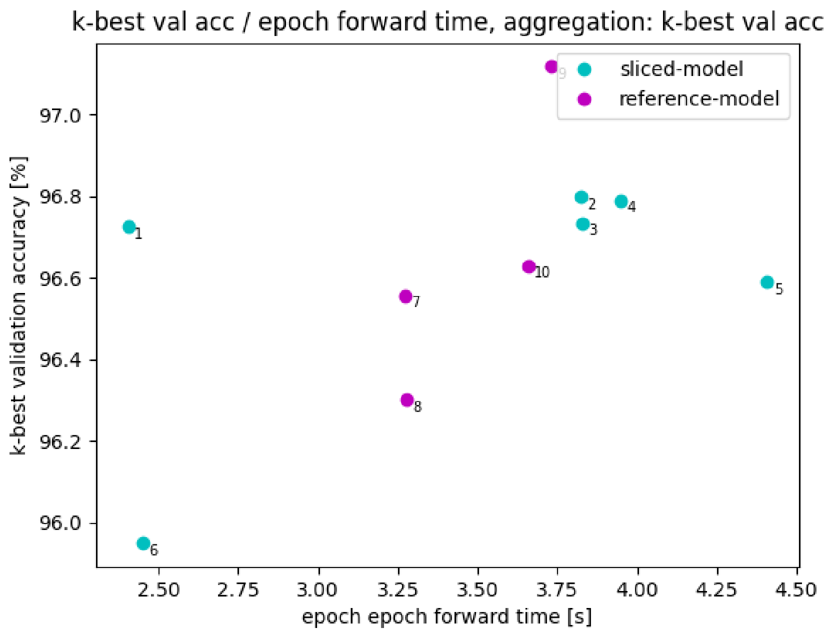


Figure 59 - Model comparison forward time with aggregate function PVA

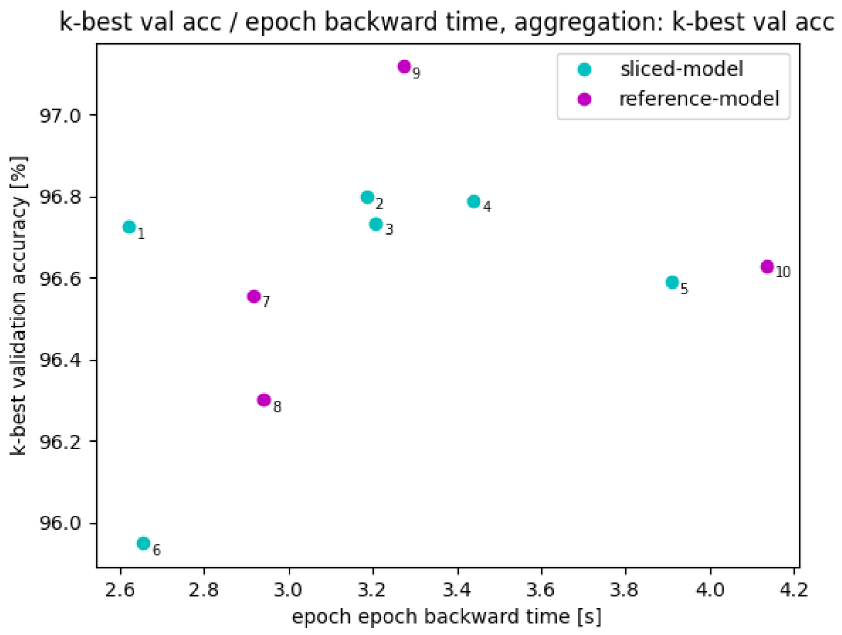


Figure 60 - Model comparison backward time with aggregate function PVA

Number	Type	LR	Accuracy	Total Epoch	Forward	Backward
1	SlicedLSTM	0.01	94.73%	5.349s	2.404s	2.619s
2	SlicedLSTM	0.02	95.50%	7.409s	3.822s	3.185s
3	SlicedLSTM	0.01	94.90%	7.430s	3.828s	3.207s
4	SlicedLSTM	0.005	95.56%	7.767s	3.951s	3.439s
5	SlicedLSTM	0.01	95.76%	8.712s	4.409s	3.908s
6	SlicedLSTM	0.02	94.56%	5.469s	2.449s	2.655s
7	Standard LSTM	0.01	95.86%	6.555s	3.271s	2.915s
8	Standard LSTM	0.01	94.89%	6.601s	3.278s	2.940s
9	Standard LSTM	0.005	96.71%	7.334s	3.732s	3.273s
10	Standard LSTM	0.01	96.31%	8.185s	3.658s	4.137s

Table 13 - Model comparison forward, backward and total epoch time with aggregate function PVA

The evaluation with PVA as the evaluation function again clearly shows that the SlicedLSTM outperforms the standard LSTM, especially in terms of runtime (forward pass). Considering all measurements with total epoch time, forward pass and the backward pass, the sliced LSTM no. 1 is always the fastest sliced LSTM, while the standard LSTM no. 7 is the fastest and the standard LSTM no. 9 the most accurate reference model.

At total epoch time (see Figure 58 and Table 13) SlicedLSTM no. 1 is 18.40% faster than the fastest Standard LSTM no. 7 (5.349s compared to 6.555s). Comparing the fastest SlicedLSTM no. 1 with the most accurate Standard LSTM no. 9, the measured improvement in runtime is 27.07%.

As already established with the aggregate functions mean and median, the SlicedLSTM achieves the greatest runtime improvement in the forward pass (see Figure 59 and Table 13). This is a very pleasing result, since the focus of this thesis is on the forward pass, i.e. on the processing of the model at runtime. Looking at the measured results of the forward pass, the fastest SlicedLSTM no. 1 is 26.51% faster than the fastest Standard LSTM no. 7 and 35.58% faster than the most accurate Standard LSTM no. 9 (2.404s compared to 3.271s and 3.732s). With the backward pass, the measured runtime improvements are somewhat smaller than with the forward pass, but still significant (see Figure 60 and Table 13). In the backward pass, the fastest SlicedLSTM no. 1 is 10.15% faster than the fastest Standard LSTM no. 7 and 19.98% faster than the most accurate Standard LSTM no. 9 (2.619s compared to 2.915s and 3.273s).

The loss in accuracy for all measurements is again only 1.18% when comparing the fastest SlicedLSTM no. 1 with the fastest Standard LSTM no. 7 (94.73% compared to 95.86%). Compared to the most accurate Standard LSTM no. 9, the fastest SlicedLSTM no. 1 is only 2.05% less accurate (94.73% compared to 96.71%). In all measurements performed in this experimental study, the accuracy achieved by the fastest SlicedLSTM no. 1 was always close to 95%. This is far more than the value of 90%, which was defined as the minimum accuracy in the context of this thesis.

In all measurements with different aggregate functions, but especially when using PVA, the SlicedLSTM outperforms the standard LSTM as a reference model in runtime with comparable accuracy. The best performance values for all aggregate functions were achieved by SlicedLSTM model no. 1. The fastest reference model was the Standard LSTM no. 7 where the Standard LSTM no.9 was the most accurate of the models compared. Table 14 summarises the measurement results presented in this section for the best models no. 1, 7 and 9. Table 14 lists the measured accuracies and the epoch times determined using the different aggregate functions.

Aggregate	**Model***	**Type**	**Accuracy**	**Total Epoch**	**Forward**	**Backward**
Mean	1	SlicedLSTM	94.73%	5.374s	2.431s	2.616s
	7	Standard LSTM	95.86%	6.570s	3.273s	2.917s
	9	Standard LSTM	96.71%	7.354s	3.747s	3.284s
Median	1	SlicedLSTM	94.73%	5.357s	2.414s	2.616s
	7	Standard LSTM	95.86%	6.573s	3.275s	2.918s
	9	Standard LSTM	96.71%	7.342s	3.737s	3.282s
PVA	1	SlicedLSTM	94.73%	5.349s	2.404s	2.619s
	7	Standard LSTM	95.86%	6.555s	3.271s	2.915s
	9	Standard LSTM	96.71%	7.334s	3.732s	3.273s

(* Model 1 = fastest SlicedLSTM. Model 7 = fastest reference model. Model 9 = most accurate reference model)

Table 14 - Measurement results for the best models no. 1, 7 and 9

Table 15 shows the calculated percentage improvements of the SlicedLSTM no. 1 compared to the two reference models Standard LSTM no. 7 and no. 9 for the total epoch time, the forward pass and the backward pass.

Aggregate function	**SlicedLSTM no. 1 compared to:**	**Loss of accuracy**	**Level of improvement in time [%]**		
			Total Epoch	Forward	Backward
Mean	Standard LSTM no. 7	1.18%	18.20%	25.73%	10.32%
	Standard LSTM no. 9	2.05%	26.92%	35.12%	20.34%
Median	Standard LSTM no. 7	1.18%	18.50%	26.29%	10.35%
	Standard LSTM no. 9	2.05%	27.04%	35.40%	20.29%
PVA	Standard LSTM no. 7	1.18%	18.40%	26.51%	10.15%
	Standard LSTM no. 9	2.05%	27.07%	35.58%	19.98%
Average*	Standard LSTM no. 7	1.18%	18.37%	26.17%	10.27%
	Standard LSTM no. 9	2.05%	27.01%	35.37%	20.21%

* Average as simple mean over all three aggregate functions per model

Table 15 – Level of improvement achieved by SlicedLSTM no. 1

It is worth mentioning that SlicedLSTM no 1 was the fastest SlicedLSTM model, but closely followed by SlicedLSTM model no 6. This was to be expected, as these two models have the smallest internal size and yet offer comparable accuracy. These two model configurations already delivered the best results when selecting the model configurations in section 6.3.3.

During the measurements, it was noticed that the accuracy achieved by the SlicedLSTM converge more slowly during training compared to the reference model. This effect is attributed to the fact that the inner SplitLayer and ConnectionLayer of the SlicedLSTM require more training effort at the beginning of the training than the standard LSTM. After about 20 epochs, however, this effect is almost completely balanced and the accuracy of the two model approaches develops similarly well. In the laboratory test, 50 epochs were performed and measured for each of the test runs. If the laboratory test were extended to significantly more than 50 epochs, even better PVA values will be expected with the SlicedLSTM. However, the accuracy values of always more than 94.5% achieved in the laboratory test are considered sufficient for the comparison test and the objective of this thesis.

6.4 Summary

As expected in the theoretical definition of the SlicedLSTM in section 6.2.2, the deeper inner structure of a SlicedLSTM cell allows a certain task to be solved with a less complex neural network structure than is the case when using standard LSTM cells. This confirms hypothesis 2 set out in the research design in section 5.5: *if a neural cell has a deeper inner structure, this leads to less complex networks*. The lower complexity of the SlicedLSTM then results in reductions in the required runtime, which confirms hypothesis 1 set out in the research design in section 5.5: *if the input data of a neural cell is divided into several simpler sub-layers, the neural cell can process data faster overall despite the resulting overhead and with comparable quality of results*. The reduction of the runtime was first proven in the PoC in section 6.2.3 and then confirmed in the laboratory test and under the described conditions and data. According to research question RQ4 (see section 1.2), the aim of this thesis, to find a novel model type that significantly improves runtime behaviour with comparable result quality, was achieved with the SlicedLSTM and proven in the quantitative Study in this thesis. Even in the backward pass, the fastest SlicedLSTM was always more than 10% faster than all reference models. The greatest improvements in runtime were achieved in the forward pass, i.e. the processing time at runtime. The average improvement in processing time (forward pass) of the fastest SlicedLSTM no 1 over the three aggregation functions was 26.17% compared to the fastest reference model Standard LSTM no. 7 and 35.37% compared to the most accurate reference model Standard LSTM no. 9 while the loss in accuracy was only between 1.18% and 2.05%.

7 Conclusion & Future Work

The use of neural networks for Predictive Maintenance systems in IIoT environments is becoming increasingly widespread. The reasons for this were comprehensively elaborated and presented in the basic chapters of this thesis. It was also explained and confirmed by means of a qualitative study that almost all currently used model architectures of neural networks focus exclusively on achieving the highest possible result accuracy. In contrast, the runtime behaviour and resource requirements of a network do not play a significant role. In the basic chapters of this thesis, it was shown that in the area of Predictive Maintenance applications there are definitely use cases in which the performance of neural networks is a decisive factor for their usability. This was also confirmed by the statements of the experts interviewed during the qualitative study. Especially in IIoT environments and Predictive Maintenance systems, it was proven that such a need exists. Since there is currently no architecture for neural networks that is explicitly geared towards performance, the corresponding technologies are lacking for these application areas. This inhibits the use of neural networks in the area of Predictive Maintenance and IIoT or leads to the application of optimisation approaches that introduce more complexity into the network and thus complicate its trainability and applicability. In the context of this thesis, the question was raised whether it is possible to achieve better runtimes with similar result quality using a new architecture of neural network cells. At the same time, however, the complexity of the neural network should not be further increased by the new architecture. Additional mechanisms or pre-processing steps purely for performance optimisation, whether executed manually or automatically, should be avoided.

From the basic chapters of this thesis, and in particular from the related works listed, it is evident that reducing the complexity of a neural network represents the greatest lever for performance improvements. Improved parallelisability also leads to an improvement in performance, but the effect of reduced complexity may be estimated to be much higher. This fact has already been pointed out in the chapter on related work. It was also shown in the chapter on related work that existing optimisation approaches either refer to the optimisation of the network structure or accelerate the processing in the actual network by means of data-side pre-processing steps. A neural cell whose architecture is designed for use in time-critical Predictive Maintenance applications and IIoT environments did not exist until now. Furthermore, this was also shown in the chapter on related work and in the qualitative study.

Based on the approaches of performance optimisation by splitting data in an upstream, usually manually performed, processing step, the Sliced Long Short-Term Memory (SlicedLSTM) emerged as a new model architecture of a neural cell in the context of this thesis. The SlicedLSTM is based on the architecture of the Standard Long Short-Term Memory Neural Network (Standard LSTM) with its advantages in the application field. However, due to a new inner architecture of the SlicedLSTM, the focus is on the processing time and the resource requirements of the cell and not on its maximum result quality. The basic principle here is to already map a segmentation of the input data and the ability for parallelisation in the internal architecture. A similarly oriented procedure could not be found in all the research carried out in the context of this work. The SlicedLSTM cell itself is capable of independently splitting the incoming data, processing it in parallel layers and aggregating it again in a final result layer.

Within the framework of a quantitative study, the improvement of the runtime behaviour of the SlicedLSTM was verified. In this study, the SlicedLSTM was able to clearly outperform the standard LSTM under the given framework conditions. The general framework conditions include in particular the use case of Predictive Maintenance and the data used in the study. However, a general validity cannot be derived from the results of this study. The improvements achieved by the SlicedLSTM developed in this thesis should be further tested in future research activities. This should be done under further use cases and with other data. From the basic idea of the SlicedLSTM, it may also be possible to derive other cell architectures that further improve the effect of the SlicedLSTM as part of future research.

The standard LSTM is an established and widely used model architecture. Various forms of optimisation (e.g. Peephole or GRU) improve its capabilities. Highly developed frameworks offer ready-made LSTM implementations whose programming and framework compatibility are well developed. The application case and the data are always a determining factor for the performance of a particular model type. The newly developed SlicedLSTM approach outperforms the Standard LSTM as a reference model in the framework of this thesis. All measurements and interpretations of the results are only made under the framework conditions of this thesis. The results achieved in this thesis were not examined for their general validity. However, the results of this thesis provide strong evidence that the SlicedLSTM is a competitive model architecture that can achieve significant runtime improvements.

For all measurements carried out in the context of this thesis the measured improvements in runtime result solely from the reduction in complexity. The SplitLayer of the SlicedLSTM consists of the parallel arrangement of several sublayers in which the split input data are processed independently of each other. The parallel and independent arrangement of the SplitLayer is an ideal prerequisite for parallel processing. Additional performance improvements can be expected from this. Programming parallel processing functions for the SplitLayer of the SlicedLSTM would exceed the scope of this thesis in terms of time and programming effort. However, the SlicedLSTM model developed within the scope of this thesis offers the starting point for making use of this further potential of the SlicedLSTM.

In section 3.4 of this thesis, approaches are described in which the input data are manually split in an upstream step. The split data leads to a simplification of the subnetworks and thus to shorter processing times overall. In this, the manual division of data is done by expert knowledge. The experts know the data and the meaning precisely and take this into account when splitting. Data that have strong dependencies on each other with regard to the result to be determined by the neural network are ideally placed in the same data sub-package. This form of partitioning should result in as little information as possible being lost from the relationships between data, which improves the quality of the results achieved (Ma, et al., 2018). (Jin, et al., 2022). The splitting of the input and hidden data in the SlicedLSTM was done arbitrarily for the purpose of this thesis. The data used are anonymised and do not allow any conclusions to be drawn about dependencies within data. If the distribution of data in the SlicedLSTM were to be carried out on the basis of deep expert knowledge, it can be assumed that the quality (accuracy) of the SlicedLSTM would increase further compared to the standard LSTM. In turn, the complexity of the SlicedLSTM model is expected to be further reduced while still achieving comparable accuracy. It can be assumed that this will lead to a further significant improvement in the runtimes of the SlicedLSTM.

The SlicedLSTM was developed in such a way that the splitting of the slices can be specified in detail. A user-defined division of the data is already possible with the current implementation, which was carried out as part of the experimental study of this thesis. It is also conceivable to extend the splitting mechanism of the SlicedLSTM so that instead of an expert, an intelligent mechanism automatically finds an optimal splitting of the data. For example, an advantageous slice partitioning and appropriate model configurations could possibly be found automatically by applying Automated Machine Learning (AutoML) processes (AutoML, 2022). The SlicedLSTM will in any case be able to benefit from such future research in these topics.

8 References

Aggarwal, Raghav. 2019. Bi-LSTM. [Online] 4. July 2019. [Quote from: 6. January 2022.] https://medium.com/@raghavaggarwal0089/bi-lstm-bc3d68da8bd0.

Aha, David, et al. 2017. UCI Machine Learning Repository. *UCI Machine Learning Repository.* [Online] UC Irvine, 2017. https://archive.ics.uci.edu/ml/index.php.

Ahmed, M. und Soo, L. 2008. *Supervisory Control and Data Acquisition System (SCADA) Based Customized Remote Terminal Unit (RTU) for Distribution Automation System.* s.l. : 2nd IEEE International Conference on Power and Energy (PECon 08), December 1-3, 2008, Johor Baharu, Malaysia, 2008.

Ait-Alla, Abderrahim, et al. 2015. Real-time fault detection for advanced maintenance of sustainable technical systems. *Procedia CIRP.* 2015, 41.

Åkerman, Magnus. 2018. *Implementing Shop Floor IT for Industry 4.0 - THESIS FOR THE DEGREE OF DOCTOR OF PHILOSOPHY.* s.l. : ResearchGate https://www.researchgate.net/publication/326224890, 2018.

Akwensi, Perpetual Hope, et al. 2021. *Performance evaluation of artificial neural networks for natural terrain classification.* s.l. : Springer, 2021.

Alippi, Cesare, et al. 2014. A Self-Building and Cluster-Based Cognitive Fault Diagnosis System for Sensor Networks. *IEEE Transactions on neural networks and learning systems.* 2014, Bd. 6, 25.

Al-Yateem, Nabeel. 2012. *The effect of interview recording on quality of data obtained: A methodological reflection.* s.l. : University of Sharjah, https://www.researchgate.net/publication/230621515, 2012.

Asanovic, Krste, et al. 2006. *The Landscape of Parallel Computing Research: A View from Berkeley.* s.l. : Electrical Engineering and Computer Sciences University of California at Berkeley, 2006. Technical Report No. UCB/EECS-2006-183.

Atteslander, Peter, et al. 2010. *Methoden der empirischen Sozialforschung - 13. Auflage.* s.l. : Erich Schmidt Verlag, 2010.

AutoML. 2022. AutoML. *AutoML.* [Online] AutoML, 2022. [Zitat vom: 2022. Juli 14.] https://www.automl.org/.

AzureML Team, for Microsoft. 2018. Azure-Samples - MachineLearningSamples-DeepLearningforPredictiveMaintenance. *Azure-Samples - MachineLearningSamples-DeepLearningforPredictiveMaintenance.* [Online] Microsoft, Public archive on Github under Microsoft Open Source Code of Conduct, 2018. [Zitat vom: 26. 05 2022.] https://github.com/Azure-Samples/MachineLearningSamples-DeepLearningforPredictiveMaintenance/.

—. 2015. Predictive Maintenance Template. *Predictive Maintenance Template.* [Online] Microsoft, 2015. [Zitat vom: 26. 05 2022.] https://gallery.azure.ai/Collection/Predictive-Maintenance-Template-3.

Bahdanau, Dzmitry, Cho, KyungHyun und Bengio, Yoshua. 2015. *Neural Machine Translation by Jointly Learning to Align and Translate.* s.l. : arXiv:1409.0473v7 [, 2015.

Bauer, Dennis, Stock, Daniel und Bauernhansl, Thomas. 2017. Movement towards service-orientation and app-orientation in manufacturing IT. *10th CIRP Conference on Intelligent Computation in Manufacturing Engineering - CIRP ICME '16.* 2017.

Bengio, Yoshua, Boulanger-Lewandowski, Nicolas und Pascanu, Razvan. 2012. *Advances in Optimizing Recurrent Networks.* s.l. : arXiv.org > cs > arXiv:1212.0901v2, 2012.

Beyer, Kevin, et al. 1999. When Is "Nearest Neighbor" Meaningful? 1999.

Beyer, Mark, et al. 2011. 'Big Data' Is Only the Beginning of Extreme Information Management. [Online] 07. April 2011. [Zitat vom: 05. Oktober 2017.] https://www.gartner.com/doc/1622715/big-data-beginning-extreme-information.

Bischl, Bernd, et al. 2021. *OpenML Benchmarking Suites.* s.l. : 35th Conference on Neural Information Processing Systems (NeurIPS 2021) Track on Datasets and Benchmarks, 2021.

Bogner, Alexander, Littig, Beate und Menz , Wolfgang. 2014. *Interviews mit Experten.* Wiesbaden : VS Verlag für Sozialwissenschaften, 2014.

Bouaziz, Mohamed, et al. 2017. *Parallel Long Short-Term Memory for Multi-stream Classification.* s.l. : arXiv:1702.03402v1 [cs.LG] 11 Feb 2017, 2017.

Bu, N. und Fukuda, O. 2017. *Classification Accuracy.* s.l. : Copyright © 2018 Elsevier B.V. or its licensors or contributors, 2017.

Byeon, Wonmin, et al. 2015. *Scene Labeling with LSTM Recurrent Neural Networks.* s.l. : IEEE Xplore, 2015.

Canziani, Alfredo und Culurciello, Eugenio, Paszke, Adam. 2016. *An Analysis of Deep Neural Network Models for Practical Applications.* s.l. : Weldon School of Biomedical Engineering Purdue University, Faculty of Mathematics, Informatics and Mechanics University of Warsaw, arXiv:1605.07678v4, 2016.

Cardona, Albert, et al. 2010. *An Integrated Micro- and Macroarchitectural Analysis of the Drosophila Brain by Computer-Assisted Serial Section Electron Microscopy.* s.l. : PLoS Biology | www.plosbiology.org, 2010.

Chatfield, Ken, et al. 2014. *Return of the Devil in the Details: Delving Deep into Convolutional Nets.* s.l. : Visual Geometry Group, Department of Engineering Science, University of Oxford, arXiv:1405.3531v4, 2014.

Chen, C.L Philip und Zhang, Chun-Yang. 2014. Data-intensive applications, challenges, techniques. *Information Sciences.* 2014, 275.

Chiu, Jason P.C. und Nichols, Eric. 2016. *Named Entity Recognition with Bidirectional LSTM-CNNs.* s.l. : University of British Columbia, arXiv:1511.08308v5 [cs.CL] 19 Jul 2016, 2016.

Cho, Kyunghyun, et al. 2014a. *Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation.* s.l. : arXiv:1406.1078v3 [cs.CL] 3 Sep 2014, 2014a.

Cho, Kyunghyun, van Merrienboer, Bart und Bahdanau, Dzmitry. 2014b. *On the Properties of Neural Machine Translation: Encoder–Decoder.* s.l. : arXiv:1409.1259v2 [cs.CL] 7 Oct 2014, 2014b.

Chollet, François und Community, keras. 2022. Keras - Simple. Flexible. Powerful. *Keras - Simple. Flexible. Powerful.* [Online] 25. 05 2022. [Zitat vom: 25. 05 2022.] https://keras.io/.

Chung, Junyoung, et al. 2014. *Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling.* s.l. : arXiv:1412.3555v1 [, 2014.

Ciresan, Dan, et al. 2012a. *Deep Neural Networks Segment Neuronal Membranes in Electron Microscopy Images.* s.l. : Supervised Deep / Recurrent Nets SNF grant, Project Code, 2012a.

Corbett, Felix. 2022. *How Industry 4.0 is Changing the.* s.l. : TTI Europe, 2022.

Creswell, John W. 2014. *Research Design. Qualitative, quantitative, and mixed methods approaches.* London : SAGE Publications, 2014.

De Vaus, David. 2013. *Surveys in Social Research* . London, New York : Routledge, 2013.

Deppermann, Arnulf. 2013. Interview als Text vs. Interview als Interaktion. *Forum: Qualitative Sozialforschung - Social Research.* 14, 2013, Bd. 3, 13.

Detorakis, Georgios, et al. 2019. *Inherent Weight Normalization in Stochastic Neural Networks.* s.l. : arXiv:1910.12316v1 [cs.LG] 27 Oct 2019, 2019.

Devcoons. 2016. Literature Review of Deep Machine Learning for feature extraction. *Literature Review of Deep Machine Learning for feature extraction.* [Online] Devcoons, 28. 06 2016. [Zitat vom: 16. 05 2021.] http://www.devcoons.com/literature-review-of-deep-machine-learning-for-feature-extraction/.

Ding, Congzhang, et al. 2021. *Continuous Human Activity Recognition through Parallelism LSTM with Multi-Frequency Spectrograms.* s.l. : MDPI, Basel, Switzerland 2021, 2021.

Dodgson, Joan E. 2020. *Quality in Research: Asking the Right Question.* s.l. : Journal of Human Lactation 2020, Vol. 36(1) 105–108, 2020.

Duckett, Laura J. 2021. *Quantitative Research Excellence: Study Design and Reliable and Valid Measurement of Variables.* s.l. : Journal of Human Lactation 2021, Vol. 37(3) 456–463, 2021.

Egmason. 2010. Egmason. [Online] 2010. https://commons.wikimedia.org/wiki/File:Fourdrinier.svg.

eoda. 2014. *Predictive Maintenance (mit R). Potenziale und Möglichkeiten der freien Statistiksprache R für neue Geschäftsmodelle im Industrie 4.0 Zeitalter.* Kassel : eoda gmbH, 2014.

Ertan, Halil. 2021. CNN-LSTM-Based Models for Multiple Parallel Input and Multi-Step Forecast. [Online] Towards Data Science, 2021. [Zitat vom: 21. 04 2022.] https://towardsdatascience.com/cnn-lstm-based-models-for-multiple-parallel-input-and-multi-step-forecast-6fe2172f7668.

Esposito, Piero. 2020a. Building a LSTM by hand on PyTorch. *Building a LSTM by hand on PyTorch.* [Online] Towards Data Science, 25. 05 2020a. [Zitat vom: 06. 06 2022.] https://towardsdatascience.com/building-a-lstm-by-hand-on-pytorch-59c02a4ec091.

—. 2020b. piEsposito/pytorch-lstm-by-hand (GitHub Repository). *piEsposito/pytorch-lstm-by-hand (GitHub Repository).* [Online] 25. 05 2020b. [Zitat vom: 06. 06 2022.] https://github.com/piEsposito/pytorch-lstm-by-hand.

Fernandez, Santiago, Graves, Alex und Schmidhuber, Jürgen. 2007. *Sequence Labelling in Structured Domains with Hierarchical Recurrent Neural Networks.* s.l. : https://www.researchgate.net/publication/220815625, 2007.

Finkbeiner, Patrick. 2016. Qualitative Research: Semi-structured Expert Interview. *Social Media for Knowledge Sharing in Automotive Repair.* s.l. : Springer International Publishing, 2016.

Flick, Uwe, Kardoff, Ernst von und Steinke, Ines. 2004. *A Companion to Qualitative Research.* s.l. : SAGE, 2004.

Fraccaro, Marco, et al. 2016. *Sequential Neural Models with Stochastic Layers.* s.l. : arXiv:1605.07571v2 [stat.ML] 13 Nov 2016, 2016.

Frederiksson, Gustav und Larsson, Hanna. 2012. *An analysis of maintenance strategies and development of a model for strategy formulation - A case study.* Göteborg, Sweden : CHALMERS UNIVERSITY OF TECHNOLOGY, 2012.

Fredriksson, Gustav und Larsson, Hanna. 2012. An analysis of maintenance strategies and development of a model for strategy formulation – A case study. *Master of Science Thesis.* Schweden : Chalmers University of Technology, 2012.

Freitag, Michael, et al. 2015. A Concept for the Dynamic Adjustment of Maintenance Intervals by Analysing Hereogenoeous Data. *Applied Mechanics and Materials.* 794, 2015.

Gao, Jing, et al. 2008. Classifying Data Streams with Skewed Class Distributions and Concept Drifts. *IEEE Internet Computing.* 6, 2008, 12.

Gartner. 2022. What's New in the 2022 Gartner Hype Cycle for Emerging Technologies. *What's New in the 2022 Gartner Hype Cycle for Emerging Technologies.* [Online] Garnter Inc., 10. 08 2022. [Zitat vom: 21. 11 2022.] https://www.gartner.com/en/articles/what-s-new-in-the-2022-gartner-hype-cycle-for-emerging-technologies.

Gensler, André, et al. 2016. *Deep Learning for Solar Power Forecasting - An Approach Using Autoencoder and LSTM Neural Networks.* s.l. : IEEE International Conference on Systems, Man, and Cybernetics • SMC 2016 | October 9-12, 2016 • Budapest, Hungary, 2016.

Gers, Felix A. und Schmidhuber, Jürgen. 2000. *Reccurent Nets that Time and Count.* s.l. : IDSIA, 2000.

Gers, Felix A., Schmidhuber, Jürgen und Cumins, Fred. 1999. *Learning to Forget: Continual Prediction with LSTM.* s.l. : IDSIA-01-99, 1999.

Glende, Sebastian, et al. 2011. Erfolgreiche AAL_Lösungen durch Nutzerintegration. [Online] Januar 2011. http://docplayer.org/4508719-Erfolgreiche-aal-loesungen-durch-nutzerintegration.html.

Göb, Rainer. 2014. Discussion of ‘‘Reliability Meets Big Data: Opportunities and Challenges’’. *Quality Engineering.* 2014, 26.

Google Brain, Team. 2022. TensorFlow - An end-to-end open source machine learning platform. *TensorFlow - An end-to-end open source machine learning platform.* [Online] 25. 05 2022. [Zitat vom: 23. 05 2022.] www.tensorflow.org.

Graves, Alex. 2011. *Practical Variational Inference for Neural Networks.* s.l. : Department of Computer Science University of Toronto, Canada, 2011.

Graves, Alex und Schmidhuber, Jürgen. 2005. *Framewise Phoneme Classification with Bidirectional LSTM Networks.* s.l. : IDSIA, 2005.

—. 2009. *Offline Handwriting Recognition with Multidimensional Recurrent Neural Networks.* s.l. : In Advances in Neural Information Processing Systems 21, pp. 545–552. MIT Press, Cambridge, MA,, 2009.

Graves, Alex, Fernandez, Santiago und Schmidhuber, Jürgen. 2013a. *Multi-Dimensional Recurrent Neural Networks.* Switzerland : IDSIA, 2013a.

Graves, Alex, Mohamed, Abdel-rahman und Hinton, Geoffrey. 2013b. *Speech Recognition with Deep Recurrent Neural Networks.* s.l. : Department of Computer Science, University of Toronto, arXiv:1303.5778v1 [cs.NE] 22 Mar 2013, 2013b.

Greff, Klaus, et al. 2017. *LSTM: A Search Space Odyssey.* s.l. : arXiv:1503.04069v2 [cs.NE] 4 Oct 2017, 2017.

Häder, Michaela. 2015. *Empirische Sozialforschung. Eine Einführung.* Wiesbaden : Springer Fachmedien, 2015.

Hafsa, Nur-E. 2019. *Mixed Methods Research: An Overview for Beginner Researchers.* s.l. : Faculty of Education, Department of Language and Literacy Education, University of Malaya, 2019.

Hagenberg. 2017. Software Competence Center Hagenberg. [Online] 2017. [Zitat vom: 04. Oktober 2017.] https://www.scch.at/en/das-news/scch-presented-predictive-analytics.

Hammerton, James. 2003. *Named Entity Recognition with Long Short-Term Memory.* s.l. : Alfa-Informatica, University of Groningen Groningen, The Netherlands, 2003.

He, Zhen, et al. 2017. *Wider and Deeper, Cheaper and Faster: Tensorized LSTMs for Sequence Learning.* s.l. : arXiv:1711.01577v3 [stat.ML] 13 Dec 2017, 2017.

Heng, Aiwina, et al. 2009. Rotating machinery prognostics: State of the art, challenges and opportunities. *Machanical Systems and Signal Processing.* 2009, 23.

Hochreiter, Sepp und Schmidhuber, Jürgen. 1997. *LONG SHORT-TERM MEMORY.* s.l. : Neural Computation 9(8):1735{1780, 1997, 1997.

Hu, Xiyuan, Peng, Silong und Hwang, Wen-Liang. 2014. Learning adaptive interpolation kernels for fast single-image super resolution. *Signal, Image and Video Processing.* 6, 2014, 8.

Huang, Po-Yao, et al. 2016. *Attention-based Multimodal Neural Machine Translation.* s.l. : Association for Computational Linguistics, Proceedings of the First Conference on Machine Translation, Volume 2: Shared Task Papers, pages 639–645, 2016.

Hunter, John, et al. 2022. Documentation matplotlib.pyplot.boxplot. *Documentation matplotlib.pyplot.boxplot.* [Online] he Matplotlib development team, 2022. [Zitat vom: 02. 07 2022.] https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.boxplot.html.

Ibrahim, Mostafa S., et al. 2016. *A Hierarchical Deep Temporal Model for Group Activity Recognition.* s.l. : School of Computing Science, Simon Fraser University, Burnaby, Canada, 2016.

IEC 61511-1. 2016. *IEC 61511-1 INTERNATIONAL STANDARD Edition 2.0 2016-02.* 2016. ISBN 978-2-8322-3216-3.

ISO 55000. 2022. ISO 55000. *ISO 55000.* [Online] ICON Services GmbH, 17. 11 2022. [Zitat vom: 17. 11 2022.] http://www.iso55000.de/de/asset-management/iso-55000.

Izadi, Maliheh, Akbari, Kiana und Heydarnoori, Abbas. 2021. *Predicting the Objective and Priority of Issue Reports in Software Repositories.* s.l. : arXiv:2012.10951v3 [cs.SE] 28 Sep 2021, 2021.

Jaeger, Herbert, et al. 2007. *Optimization and Applications of Echo State Networks with Leaky Integrator Neurons.* s.l. : International University Bremen, Planet intelligent systems GmbH, 2007.

Jeng, J.-C. 2010. Adaptive process monitoring using efficient recursive PCA and moving window PCA algorithms. *Journal of the Taiwan Institute of Chemical Engineers.* 41, 2010, Bd. 4.

Jeyakumar, Jeya Vikranth, et al. 2020. *How Can I Explain This to You? An Empirical Study of Deep Neural Network Explanation Methods.* s.l. : Advances in Neural Information Processing Systems 33 (NeurIPS 2020), 2020.

Jia, Xiaodong, et al. 2018. *Review of PHM Data Competitions from 2008 to 2017: Methodologies and Analytics.* s.l. : NSF I/UCR Center for Intelligent Maintenance Systems, Department of

Mechanical Engineering, University of Cincinnati, PO Box 210072, Cincinnati, Ohio 45221-0072, US, 2018.

Jin, Ning, et al. 2022. Highly accurate energy consumption forecasting model based on parallel LSTM neural networks. *Advanced Engineering Informatics.* 2022.

Johnson, B und Christensen, L. 2008. *Educational research: Quantitative, qualitative, and mixed approaches.* s.l. : Thousand Oaks, CA: Sage Publications, 2008.

Johnson, R. Burke, Onwuegbuzie, Anthony J. und Turner, Lisa A. 2007. *Toward a Definition of Mixed Methods Research* . s.l. : Sage Publications, 2007.

Jozefowicz, Rafal, Zaremba, Wojciech und Sutskever, Ilya. 2015. *An Empirical Exploration of Recurrent Network Architectures.* s.l. : Proceedings of the 32 nd International Conference on Machine, 2015.

Kalchbrenner, Nal und Blunsom, Phil. 2013. *Recurrent Continuous Translation Models.* s.l. : Department of Computer Science University of Oxford, 2013.

Kalchbrenner, Nal, Danihelka, Ivo und Graves, Alex. 2015. *Grid Long Short-Term Memory.* s.l. : Google DeepMind London, arXiv:1507.01526v3 [cs.NE] 7 Jan 2016, 2015.

Karanam, Shashmi. 2021. Curse of Dimensionality — A “Curse” to Machine Learning. *Curse of Dimensionality — A “Curse” to Machine Learning.* [Online] Towards Data Science, 11. 08 2021. [Zitat vom: 17. 11 2022.] https://towardsdatascience.com/curse-of-dimensionality-a-curse-to-machine-learning-c122ee33bfeb.

Kassner, Karsten und Wassermann, Petra. 2002. Nicht überall, wo Methode draufsteht, ist auch Methode drin. Zur Problematik der Fundierung von ExpertInneninterviews. [Buchverf.] Alexander Bogner, Beate Littig und Wolfgang Menz. *Das Experteninterview.* Wiesbaden : VS Verlag für Sozialwissenschaften, 2002.

Kothamasu, Ranganath, Huang, Samuel H und VerDuin, William H. 2006. System health monitoring and prognostics — a review of current paradigms and practices. *The International Journal of Advanced Manufacturing Technology.* 28, 2006, Bd. 9.

Koutník, Jan, et al. 2014. *A Clockwork RNN.* s.l. : IDSIA, USI&SUPSI, Manno-Lugano, CH-6928, Switzerland, arXiv:1402.3511v1 [cs.NE] 14 Feb 2014, 2014.

Krauß, Jonathan, et al. 2019. *Maschinelles Lernen in der Produktion, Anwendungsgebiete und frei verfügbare Datensätze.* s.l. : Fraunhofer IPT und WZL RWTH Aachen, https://doi.org/10.30844/I40M_19-4_S39-42, 2019.

Krawczyk, Bartosz und Wozniak, Michal. 2015. Data stream classification and big data analytics. *Neurocomputing.* 2015, 150.

Krizhevsky, Alex, Sutskever, Ilya und Hinton, Geoffrey E. 2012. *ImageNet Classification with Deep Convolutional Neural Networks.* Switzerland : NIPS, 2012.

Krueger, David, et al. 2017. *Zoneout: Regularizing RNNs by Randomly Preserving Hidden Activations.* s.l. : arXiv:1606.01305v4 [cs.NE] 22 Sep 2017, 2017.

Kuß, Alfred, Wildner, Raimund und Kreis, Henning. 2012. *Marktforschung. Grundlagen der Datenerhebung und Datenanalyse. 4. Auflage.* Wiesbaden : Gabler Verlag , 2012.

Lee, Jay, et al. 2014. Prognostics and health management design for rotary machinery systems—Reviews, methodology and applications. *Mechanical Systems and Signal Processing.* 2014, 42.

Lee, L. N., et al. 2015. Risk Perceptions for Wearable Devices. *Cornell University Library.* [Online] 2015. http://arxiv.org/pdf/1504.05694.pdf.

Lee, Sangkeum, et al. 2021. *Smart Metering System Capable of Anomaly Detection by Bi-directional LSTM Autoencoder.* s.l. : arXiv.org > cs > arXiv:2112.03275, 2021.

Lee, Yoonho, et al. 2020. *Neural Complexity Measures.* s.l. : 34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada, 2020.

Liang, Tengyuan, et al. 2019. *Fisher-Rao Metric, Geometry, and Complexity of Neural Networks.* s.l. : arXiv:1711.01530v2 [cs.LG] 23 Feb 2019, 2019.

Liang, Victor C., et al. 2016. *Mercury: Metro Density Prediction with Recurrent Neural Network on Streaming CDR Data.* s.l. : ICDE 2016 Conference 978-1-5090-2020-1/16, 2016.

Lichtman, M. 2006. *Qualitative research in education: A user's guide.* s.l. : Thousand Oaks, CA: Sage Publications, 2006.

Lietz, Peter. 2010. Research into questionnaire design. *International Journal of Market Research.* 52, 2010, 2.

Liu, Lu, Song, Xiao und Zhou, Zhetao. 2022. *Aircraft engine remaining useful life estimation via a double attention-based data-driven architecture.* s.l. : 0951-8320/© 2022 Elsevier Ltd. All rights reserved., https://doi.org/10.1016/j.ress.2022.108330, 2022.

Liu, Ting, et al. 2014. *A modular hierarchical approach to 3D electron microscopy image segmentation.* s.l. : Elsevier, 2014.

Lu, Yuzhen und M. Salem, Fathi. 2017. *Simplified Gating in Long Short-term Memory (LSTM) Recurrent Neural Networks.* s.l. : arXiv.org > cs > arXiv:1701.03441, 2017.

Ma, Chih-Yao, et al. 2018. *TS-LSTM and Temporal-Inception: Exploiting Spatiotemporal Dynamics for Activity Recognition.* s.l. : arXiv:1703.10667v1 [cs.CV] 30 Mar 2018, 2018.

MacLennan, Jamie. 2012. 5 Myths about Predictive Analytics. [Online] 1. Mai 2012. [Zitat vom: 05. Oktober 2017.] https://tdwi.org/articles/2012/05/01/5-predictive-analytics-myths.aspx.

Märtens, Marcus und Izzo, Dario. 2019. *Neural Network Architecture Search with Differentiable Cartesian Genetic Programming for Regression.* s.l. : GECCO ’19 Companion, July 13–17, 2019, Prague, Czech Republic, https://doi.org/10.1145/3319619.3322003, 2019.

Marz, Nathan und Warren, James. 2015. *Big Data. Principles and best practices of scalable real-time data systems.* New York : Manning Publications Co., 2015.

MathWorks, Inc. 2021. *Deep Learning for Signal Processing with MATLAB.* s.l. : MathWorks, Inc., 2021.

Melnikov, Vitalik und Hüllermeier, Eyke. 2016. *Learning to Aggregate using Uninorms.* s.l. : Department of Computer Science Paderborn University, Germany, 2016.

Mendrik, Adriënne M., et al. 2015. *MRBrainS Challenge: Online Evaluation Framework for Brain Image Segmentation in 3T MRI Scans.* s.l. : Hindawi Publishing Corporation Computational Intelligence and Neuroscience, 2015.

Mikolov, Tomas, et al. 2015. *Learning Longer Memory in Recurrent Neural Networks.* s.l. : arXiv:1412.7753v2 [cs.NE], 2015.

Mocanu, Elena, et al. 2016. *Deep learning for estimating building energy consumption.* s.l. : Department of Electrical Engineering, Eindhoven University of Technology, The Netherlands, 2016.

Mohammadi, Mehdi, et al. 2018. *Deep Learning for IoT Big Data and Streaming Analytics: A Survey.* s.l. : arXiv:1712.04301v2 [cs.NI], 2018.

Montgomery, Douglas C. 2014. Big Data and the Quality Profession. *Quality and Reliability engeineering International.* 4, 2014, Bd. 30.

NASA. 2008b. Prognostics Center of Excellence Data Repository. *Prognostics Center of Excellence Data Repository.* [Online] 2008b. [Zitat vom: 26. 05 2020.] http://ti.arc.nasa.gov/projects/data_prognostics.

—. 2008a. *RCM GUIDE RELIABILITY- CENTERED MAINTENANCE GUIDE For Facilities and Collateral Equipment.* 2008a.

Neil, Daniel, Pfeiffer, Michael und Liu, Shih-Chii. 2016. *Phased LSTM: Accelerating Recurrent Network Training for Long or Event-based Sequences.* s.l. : arXiv:1610.09513v1 [cs.LG] 29 Oct 2016, 2016.

Newington, Lisa und Metcalfe, Alison. 2014. *Factors influencing recruitment to research: qualitative study of the experiences and perceptions of research teams.* s.l. : BioMed Central Ltd, BMC Medical Research Methodology 2014, http://www.biomedcentral.com/1471-2288/14/10, 2014.

Ngulube, Patrick, Romm, Norma. 2015. Mixed methods research. *South African Journal of Economics and Management Sciences.* 18, 2015, 1.

Nunan, Daniel und Domenico, Maria. 2013. Market research & the ethics of big data. *International Jounal of Market Research.* 4, 2013, 55.

Orenstein, Gary, et al. 2015. *Building Real-Time Data Pipelines - Unifying Applications and Analytics with In-Memory Architectures.* Sebastopol, CA : O’Reilly Media, Inc., 2015. 978-1-491-93547-7.

Pan, Yingnan und Yang, Guang-Hong. 2021. *Event-Driven Fault Detection for Discrete-Time Interval Type-2 Fuzzy Systems.* s.l. : IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS: SYSTEMS, VOL. 51, NO. 8, AUGUST 2021 4959, 2021.

Peng, Ying, Dong, Ming und Zuo, Ming Jian. 2010. Current status of machine prognostics in condition-based maintenance: a review. *The International Journal of Advanced Manufacturing Technology.* 1-4, 2010, 50.

Pranckevicius, Tomas und Marcinkevičius, Virginijus. 2017. *Comparison of Naive Bayes, Random Forest, Decision Tree, Support Vector Machines, and Logistic Regression Classifiers for Text Reviews Classification.* s.l. : Vilnius University, Institute of Mathematics and Informatics, 2017.

Praveena1809. 2020. Praveena1809/Predictive-Maintenance-Using-LSTM. *Praveena1809/Predictive-Maintenance-Using-LSTM.* [Online] github, 11. 7 2020. [Zitat vom: 12. 08 2022.] https://github.com/Praveena1809/Predictive-Maintenance-Using-LSTM.

Pusala, Murali, et al. 2016. Massive Data Analysis: Tasks, Tools, Applications and Challenges. *Big Data Analytics.* s.l. : Springer Verlag, 2016.

PyTorch. 2020. PyTorch. *PyTorch.* [Online] 2020. [Zitat vom: 06. 06 2022.] https://pytorch.org/.

Rippel, Daniel, Lütjen, Michael und Freitag, Michael. 2017. SIMULATION OF MAINTENANCE ACTIVIES FOR MICRO-MANUFACTURING SYSTEMS BY USE OF PREDICTIVE QUALITY CONTROL CHARTS. 2017.

Roweis, Sam T. und Saul, Lawrence K. 2000. Nonlinear Dimensionality Reduction by Locally Linear Embedding. *Science.* 290, 2000.

SAP.com. 2022. What is ERP? *What is ERP?* [Online] SAP Deutschland SE & Co. KG, 17. 11 2022. [Zitat vom: 17. 11 2022.] https://www.sap.com/insights/what-is-erp.html.

Saxena, A. und Goebel, K. 2008. *Damage Propagation Modeling for Aircraft Engine Run-to-Failure Simulation.* s.l. : U.S. National Aeronautics and Space Administration (NASA), 2008.

Schmidt, Bernard und Wang, Lihui. 2015. Predictive Maintenance: Literature Review and Future Trends. 2015.

Schuster, Mike und Paliwal, Kuldip K. 1997. *Bidirectional Recurrent Neural Networks.* s.l. : Article in IEEE Transactions on Signal Processing · December 1997, 1997.

Shaker, Ammar. 2016. Novel Methods for Mining and Learning from Data Streams. *Dissertation.* Paderborn : Universität Paderborn, 2016.

Shaker, Ammar und Hüllermeier, Eyke. 2013. Recovery Analysis for Adaptive Learning from Non-stationary Data Streams. *Proceedings of the 8th International Conference on Computer Recognition Systems CORES 2013.* 2013.

Sikorska, J.Z, Hodkiewicz, M. und Ma, L. 2011. Prognostic modelling options for remaining useful life estimation by industry. *Mechanical Systems and Signal Processing.* 2011, 25.

Silverman, David. 2016. *Qualitative Research.* London : Sage Publication, 2016.

Singh, Maulshree, et al. 2021. *Digital Twin: Origin to Future.* s.l. : MDPI Appl. Syst. Innov. 2021, 4, 36. https://doi.org/10.3390/asi4020036, 2021.

Sokolova, Marina und Lapalme, Guy. 2009. *A systematic analysis of performance measures for classification tasks.* s.l. : 2009 Elsevier Ltd. All rights reserved, 2009.

Song, Xuan, Kanasugi, Hiroshi und Shibasaki, Ryosuke. 2016. *DeepTransport: Prediction and Simulation of Human Mobility and Transportation Mode at a Citywide Level.* s.l. : Center for Spatial Information Science, The University of Tokyo, Japan, 2016.

Srivastava, Rupesh Kumar, Greff, Klaus und Schmidhuber, Jürgen. 2015. *Highway Networks.* s.l. : arXiv:1505.00387v2 [cs.LG] 3 Nov 2015, 2015.

Stollenga, Marijn F., et al. 2015. *Parallel Multi-Dimensional LSTM, With Application to Fast Biomedical Volumetric Image Segmentation.* s.l. : Advances in Neural Information Processing Systems 28 (NIPS 2015), 2015.

Straub, A. 2012. Maintenance and Repair. *International Encyclopdia of Housing and Home.* 2012.

Sutskever, Ilya und Hinton, Geoffrey. 2009. *Temporal Kernel Recurrent Neural Networks* . s.l. : Neural Networks, 23(2):239–243, 2009.

Sutskever, Ilya, Vinyals, Oriol und Le, Quoc V. 2014. *Sequence to Sequence Learning with Neural Networks.* s.l. : arXiv:1409.3215v3 [cs.CL] 14 Dec 2014, 2014.

Tenenbaum, Joshua B., de Silva, Vin und Langford, John C. 2000. A Global Geometric Framework for Nonlinear Dimensionality Reduction. *Science.* 290, 2000, 5500.

Tran, Van Tung, Leng, Chan Hian, et al. 2022. *IoT and machine learning enable predictive maintenance for manufacturing systems: a use-case of laser welding machine implementation.* s.l. : Singapore Institute of Manufacturing Technology (SIMTech), A*STAR, Singapore at 12th Conference on Learning Factories, CLF2022, 2022.

Uffelmann, Joachim R., Wienzek, Peter und Jahn, Myriam. 2021. *IO-Link - Band 1: Anwendung: Schlüsseltechnologie für Industrie 4.0.* s.l. : Vulkan-Verlag GmbH; 3. Edition (4. Januar 2021), 2021. ISBN-10 : 3835674390 ISBN-13 : 978-3835674394.

van der Westhuizen, Jos und Lasenby, Joan. 2018. *The unreasonable effectiveness of the forget gate.* s.l. : arXiv:1804.04849v3 [cs.NE] 13 Sep 2018, 2018.

VDP, Association of German Paper Factories e.V. 2022. Paper make. *Paper make.* [Online] 16. 11 2022. [Zitat vom: 16. 11 2022.] https://www.papierindustrie.de/fileadmin/0002-PAPIERINDUSTRIE/07_Dateien/7_Publikationen/Papiermachen.pdf.

VDW, Association of the Corrugated Board Industry e.V. 2022. Paper machine. *Paper machine.* [Online] 16. 11 2022. [Zitat vom: 16. 11 2022.] https://www.wellpappe-wissen.de/wissen/papier/papierherstellung/papiermaschine.html.

Verleysen, Michel und Francois, Domien. 2005. The Curse of Dimensionality in Data Mining and Time Series Prediction. [Buchverf.] Joan Cabastany, Alberto Prieto und Francisco Sandoval. *Computational Intelligence and Bioinspired Systems* . Berlin : Springer Verlag, 2005.

Wang, Bo, et al. 2019. *Parallel LSTM-Based Regional Integrated Energy System Multienergy Source-Load Information Interactive Energy Prediction.* s.l. : Hindawi Complexity Volume 2019, Article ID 7414318, 13 pages https://doi.org/10.1155/2019/7414318, 2019.

Wang, Gang, Nixon, Mark und Boudreaux, Mike. 2019. *Toward Cloud-Assisted Industrial IoT Platform for Large-Scale Continuous Condition Monitoring.* s.l. : Proceedings of the IEEE PP(99):1-13, 2019.

Wang, Li, et al. 2015. *LINKS: Learning-based multi-source IntegratioN frameworK for Segmentation of infant brain images.* s.l. : Elsevier, 2015.

Wick, Ryan R., Judd, Louise M. und Holt, Kathryn E. 2019. *Performance of neural network basecalling tools for Oxford Nanopore sequencing.* s.l. : Open Access, https://genomebiology.biomedcentral.com/articles/10.1186/s13059-019-1727-y, 2019.

Wu, Yujie, et al. 2018. *Spatio-Temporal Backpropagation for Training High-Performance Spiking Neural Networks.* s.l. : Open Access, https://www.frontiersin.org/articles/10.3389/fnins.2018.00331/full, 2018.

Xavier Unversitiy Library . 2016. QUALITATIVE VERSUS QUANTITATIVE RESEARCH. *QUALITATIVE VERSUS QUANTITATIVE RESEARCH.* [Online] Unversitiy Library Xavier, 2016. http://www.xavier.edu/library/students/documents/qualitative_quantitative.pdf.

Xie, Xiaofeng, et al. 2017. *IoT Data Analytics Using Deep Learning.* s.l. : Key Laboratory for Embedded and Networking Computing of Hunan Province, Hunan University, 2017.

Yao, Kaisheng, et al. 2015. *Depth-Gated LSTM.* s.l. : arXiv:1508.03790v4 [cs.NE] 25 Aug 2015, 2015.

Yiu, Tony. 2019. The Curse of Dimensionality - Why High Dimensional Data Can Be So Troublesome. *The Curse of Dimensionality - Why High Dimensional Data Can Be So Troublesome.*

[Online] Towards Data Science, 20. 07 2019. [Zitat vom: 17. 11 2022.] https://towardsdatascience.com/the-curse-of-dimensionality-50dc6e49aa1e.

Zhang, Dalin, et al. 2017b. *EEG-based Intention Recognition from Spatio-Temporal Representations via Cascade and Parallel Convolutional Recurrent Neural Networks.* s.l. : arXiv:1708.06578v1 [cs.HC] 22 Aug 2017, 2017b.

Zhang, Liangwei. 2016c. Big Data Analytics for Fault Detection and its Application in Maintenance. *Doctoral Thesis.* s.l. : Luleå University of Technology, 2016c.

Zhang, Liangwei und Karim, Ramin. 2014. Big Data Mining in eMaintenance: An Overview. 2014.

Zhang, Liangwei, et al. 2017a. Sliding Window-Based Fault Detection From High-Dimensional Data Streams. *IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS.* 2017a, Bd. 47, 2.

—. 2017. Sliding Window-Based Fault Detection From High-Dimensional Data Streams. *IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS.* 2017, Bd. 47, 2.

Zhang, Liangwei, Lin, Janet und Karim, Ramin. 2016b. Adaptive Kernel Density-based Anomaly Detection for Nonlinear Systems. [Buchverf.] Liangwei Zhang. *Big Data Analytics for Fault Detection and its Application in Maintenance.* Sweden : s.n., 2016b.

Zhang, Liangwei, Lin, Jing und Karim, Ramin. 2016a. An angle-based subspace anomaly detection approach to high-dimensional data: With an application to industrial fault detection. *ReliabilityEngineeringandSystemSafety.* 142, 2016a.

A Appendix

A.1 Interview Guide

These guidelines are intended to serve as a framework and support for conducting interviews. The questions listed here help the interviewer to head the interview and to order the relevant questions in the meeting thematically. The interview is conducted openly. The interviewee should be able to talk freely. Not all questions listed here need to be asked and other subjects may be broached spontaneously by the interviewee.

A.1.1 Introduction

As part of my scientific work I am currently performing a study in the form of qualitative and explorative interviews of experts. The study deals with the subject area of predictive analytics in industrial applications (IIoT). The focus is on methods and algorithms for predicting certain results under the conditions associated with data streams and real-critical applications.

The aim of the study is to identify those methods and algorithms relevant to the application field and to determine the need for optimisation and development of new methods and algorithms.

This is an open interview. The questions asked are merely a framework. The main purpose is to learn your personal opinion as an expert. The themes discussed in the interview are determined by you and you can always skip questions and switch between topics.

You formally gave your consent to this interview. I would like to reiterate that all the information from this interview will be processed anonymously. You may refuse to answer individual questions and terminate the interview at any time as you wish. You may also revoke your consent to evaluation of your data. However, since data will be anonymised, at a later stage it cannot be deleted anymore (as it will be impossible to be identified).

I would like to make an audio recording of our conversation. The recording is only used for evaluation of this interview. No names will be used in the report written. Your information will be kept confidential. The primary data will be kept on a CD and stored with data of this study for 10 years. It will subsequently be destroyed.

This interview will take about 45 minutes to one hour.

By way of an introduction. I will pose a couple of specific questions. This will be followed by open questions, which are only intended to set up a framework for the interview.

A.1.1.1 Question: subject area

What are your main subjects of concern in the area of predictive analytics, stream processing and IIoT?

Follow-up question: To what extent do you deal professionally with the subject and with what kind of projects have you been involved recently? You do not have to name any specific projects, only the general activities and methods used are of interest.

A.1.1.2 Question: algorithms

In your opinion, which methods and algorithms are optimal for data streams and real-time processing? Which of them are used in predictive maintenance (PdM) applications and why do you think of these specifically?

Follow-up question: Which of these algorithms are you familiar with from practical applications?

A.1.1.3 Question: Predictive Maintenance

- What are the most important goals of predictive maintenance and why? *[If examples are requested: RUL, maximum lifetime, early point ...]*
- How relevant, in your opinion, is real-time processing in PdM, particularly today and in the future and why?

A.1.2 Predictive Systems

A.1.2.1 Question: Approaches for PdM

What basic approach do you regard as important for a predictive system tasked with predictive maintenance in IIoT environments, data streams and real-time requirements and why? *[If a definition of "basic" is requested: "Approaches such as regression techniques, machine learning techniques ...]*

A.1.2.2 Question: Predictive Methods

From your experience, are you aware of any predictive methods that work with mathematical formulas, or rather with classifications, or with other techniques?

A.1.2.3 Question: Learning

In the area of machine learning, training methods exist in which data is classified by an expert. The model is therefore trained on the basis of expert knowledge. In contrast, there are also approaches where no expert knowledge is required. In these cases, patterns are automatically recognised in data by algorithms without the use of knowledge. One often talks about unsupervised learning or supervised learning.

Which of the two approaches to training models is more appropriate in your opinion and why?

Follow-up question: What are the advantages and disadvantages of the two approaches and why?

A.1.2.4 Question: Experts

For supervised learning: in your opinion, how available are experienced experts in the area of PdM to perform supervised learning?

Follow-up question: How comprehensive, in your opinion, is the knowledge of experts on average in the area of PdM? Is their knowledge sufficient to generate competitive models, or should, in your opinion, self-learning processes complement expert knowledge?

A.1.2.5 Question: Supervised learning

Supervised learning is performed either by data classification or regression analyses.

- Which of the two methods do you think is more important for PdM in IIoT environments or, in other words, is better to which applications in your opinion and why?
- Are there any other, relevant processes in your opinion?
- In your opinion, what are the most important classification processes for PdM in IIoT environments and why? *[If examples are requested: nearest neighbours classification, support vector machines (SVM), naive Bayesian classification, classification trees, neural networks, discriminant analysis. ...)]*?
- Which of these processes do you consider already suitable for processing data streams in real-time and why?
- What in your opinion is necessary to make the processes you stated ready for processing data streams in real-time?

A.1.2.6 Question: Machine Learning (ML)

What approaches in machine learning do you consider being the most important and how do you justify your opinion? *[If examples are requested: neural networks, support vector machines (SVM). Density-based approaches ...]*

A.1.2.7 Question: Unsupervised Learning

Which methods in the area of unsupervised machine learning are the most relevant in your opinion, why and for which applications?

A.1.2.8 Question: Adaptive Models

Concept drifts exist for example in behavioural research, where behavioural patterns change gradually or abruptly depending on developments and influences.

- How relevant, in your opinion, are concept drifts in IIoT environments and, in particular, in PdM applications and why?
- Apart from concept drift, how adaptive do PdM models need to be in IIoT environments in general and why?
- Do you see any other reasons why adaptive models are required?

A.1.2.9 Question: Data Complexity

In IIoT environments, data streams with high dimensionality are created due to the existence of many heterogeneous data sources and continuous sensor data. This rapidly increasing dimensionality places high demands on data processing and on algorithms, especially when real-time behaviour is desired.

- How relevant are the problem of high dimensionality in PdM systems and IIoT and what do you base your opinion on?
- What processes do they know and/or recommend and why?
- Follow-up question: What do you think about approaches to reduce data dimensionality via a preceding step and why?

A.1.2.10 Question: Non-Linearity

Non-linear data streams also place high demands on methods and algorithms.

- Do industrial processes, in your opinion, generate a relevant number of non-linear data why are you this opinion?
- Do methods and algorithms have to deal with non-linearity in your opinion and why are you this opinion?
- What processes do you know and/or recommend and why?

A.1.2.11 Question: Data Quality with respect to Historical Data

In order to process data streams within distributed and scalable systems in real-time (stream processing), the individual calculations for each data set (sample) must be as simple and as inexpensive, in terms of resources, as possible (CPU. MEM. I/O). Stream processing is a discipline in which data streams are continuously processed. Approaches such as onepass (reading, processing and forgetting data only once) or time series mean that historical data can only be conditionally included. This is different for classic batch-oriented big data systems since the entire data history is included here for each calculation, which leads to absolute truth, but also to arbitrary latency when used correctly. Missing or compromised history reduces the quality of predictions. In addition, certain knowledge, such as the re-learning of already forgotten patterns (recurrent concept), cannot be achieved.

- What is your experience concerning the importance of historical data and on what do you base your opinion?
- What, in your opinion, is more important in PdM systems: prediction quality or real-time behaviour, and why?
- To what extent do you see opportunities for improvement through the use of modern IT technologies and in which areas?

A.1.2.12 Question: Validation

The evaluation of the prediction quality of methods and algorithms in a specific use case is of great importance.

- What basic approaches to testing the prediction quality do you recommend and why? *[If examples are requested; false-positive rate, positive-false rate. ...]*
- Which specific approaches do you recommend and why?
- Do you know any approaches that are specially designed for data streams and real-time behaviour are which of their characteristics do see as most important?
- In your opinion, do you think that prediction quality should be evaluated on the basis of real data or on specially produced synthetic data, and why?

A.1.2.13 Question: Methods and Algorithms in general

To sum up:

- Which algorithms are, in your opinion, the most important and for what type of applications?
- How suitable do you think the current methods and algorithms are for analysis of data streams in real-time and why?
- What optimisation potential do you see here and why?
- In your opinion, how can the possibilities provided by modern IT systems for optimisation of existing methods and algorithms contribute and why?
- Which methods and algorithms do you think are specifically applicable here and which approaches may lead to optimisation in your opinion?

A.1.2.14 Question: Other Subjects

Is there anything associated with this subject that we haven’t discussed which you consider relevant?

A.1.3 Wrapping up and table

Finally. I would like to ask you to answer some specific questions using a table. *[Table will be handed out to the interviewee]*

What are the most relevant analytics techniques?		
Regression techniques		
Machine learning techniques		
Others / more?		

What are the most relevant regression techniques?		
Linear regression model		
Discrete choice models		
Logistic regression		
Multinomial logistic regression		
Probit regression		
Time series models		
Survival or duration analysis		
Classification and regression trees (CART)		
Multivariate adaptive regression splines		
Others / more?		

What are the most relevant Machine learning techniques?		
Neural networks		
	supervised learning	
	unsupervised learning	
	reinforcement learning	
Multilayer perceptron (MLP)		
Radial basis functions (RBF)		
Support vector machines (SVM)		
Naïve Bayes		
k-nearest neighbours (KNN)		

Geospatial predictive modelling		
Others / more?		

I would like to thank you very much for the interview, your time and for the many, important answers and opinions. You have been a great help to me and my work.

If you are interested in the subsequent course of my work and its results. I'd be happy to provide you with information. You can contact me at any time regarding this project.

Contact information:

Thomas Rieger. Mphil/PhD Student

Faculty of Science and Environment. DTC: Computing and Mathematics. Enrolment N.: 10536298

E-Mail: thomas.rieger@plymouth.ac.uk

Phone: +49 160 3616461

A.2 Evaluation of Interviews

The following details the steps taken to evaluate the expert interviews. The selected procedure is described in section 5.4.

First evaluation step was the search for similar thematic units. In the run-up to the interviews, a text document was compiled for this purpose, which contains the thematic blocks and a box for handwritten entries per block. The thematic units are listed in Table 5.

Thematic unit	**Description**
Experts Main Areas of Interest	Expert's main subjects of concern in the area of Predictive Analytics. Data Streams and IIoT; professional experience.
Targets of Predictive Maintenance Applications	Find out the most important goals of predictive maintenance.
Methods and Algorithms used by the Experts	Expert's opinion which methods and algorithms are widely used and/or optimal for PdM and real-time processing.
Tools used by the Experts	What tools and software libraries are known and/or used by the expert?
The Relevance of Real-time Demands and Stream Processing	How relevant is real-time and stream processing in PdM today and in the future?
Assessment of Complexity Factors	How important are the known complexities for PdM applications and are these already considered in practice?
The Relevance of Historical Data	Value of historical data and its use and impact on the forecast quality
Learning Methods and Adaptive Models	Relevance of continuous learning and adaptive models in practice.

Table 16 - Thematic units of the expert interviews

At the end of the text document, sufficient space was provided for possible further topics. During the interview, most important statements were entered manually into the fields. Each interviewed expert was assigned an individual colour in order to distinguish statements (see Table 6). It was possible to make audio recordings of all interviews. Using the audio recording, handwritten entries were subsequently verified and supplemented. This was respectively done promptly after each interview.

Interview	Colour
Interview 1	Black
Interview 2	Red
Interview 3	Green
Interview 4	Blue
Interview 5	Olive
Interview 6	Purple
Interview 7	Magenta

Table 17 - Colour coding of interviews

After all interviews were performed, important passages were paraphrased or quoted verbatim within the defined thematic blocks. The respective author was still colour coded. For time reasons a complete transcription was waived.

After this step had been completed for all interviews, the text passages were reorganized and reduced to their essential components through the abstraction of common motifs. Based on this, superordinate statements or findings on the respective thematic blocks were collected in note form. These superordinate statements formed the outline of the evaluation section 5.5.

A.2.1 Thematic blocks with quotations

A.2.1.1 Experts Main Areas of Interest

Mainly engaged in the field of data mining and manual preparation of data using various methods. ... The goal in the current projects is to identify optimization potential based on data. Optimizations can be design-related, operational, or maintenance-oriented. These findings are sold to customers. Online systems pending, currently not yet in use, but the topic will come.

…

Active in the field of maintenance of offshore wind turbines (WTG); ... methods are usually based on image processing and analysis of the image data; ... increasingly also Augmented Reality (AR) and thus real-time and streaming requirements for processing image data; ... performs very intensive analytics and predictions for failures and maintenance tasks ... failures are very expensive as offshore wind turbines are very difficult to reach and are accessible only in good weather.

Further topics are safety assistance systems in robotics. Also here only offline analytics because still in testing phase … for later use online operation in real-time is necessary.

…

Active in port facilities and offshore wind turbines; ...a great deal of sensor data available ... extensive PdM solutions in use, as failures are very expensive ... data mostly delivered á block and analysed manually in batch processing

…

Performs various industrial projects as a consultant. ... the focus is on analysis of existing data and processes and advice on possible improvements. ... maintenance strategies and the design of PdM systems is everyday business ... previously mostly offline analyses with batch-oriented processes ... real-time and data streams will become standard in the near future ... currently, systems and communication infrastructure within industry are not advanced enough to be able to deliver online data ... but this will change soon

…

Works mainly as an expert consultant for industrial projects... anomaly detection is currently the most frequent requirement of the customers ...for streaming analytics, the environments of the customers are usually not yet sufficiently developed ...currently data are still predominantly supplied manual and in packages, therefore mainly batch processing is used ... Industry 4.0 will fundamentally change this ...concepts for online and stream processing are already being worked on internally, especially in the area of ML

…

Active in Industry 4.0 projects ... main objectives of current applications are predictions and anomaly detection ... predictions are made for reliability and probability of failures and flow into the customer's maintenance strategies ... predictions are also made for other areas, for example an application for predicting water consumption with automatic pump control was developed ... currently, data are mostly delivered in packages, for example, in 1-year samples, therefore offline evaluation ...however real-time is the future topic which is currently being worked on internally ... mostly the customers are not yet advanced enough and still require basic knowledge about what they can learn from their data ...therefore currently mostly condition monitoring solutions as a first step ... real-time anomaly detection will be the next step ... PdM with data streams will definitely be the standard of the future.

A.2.1.2 Methods and Algorithms used by the Experts

Methods of sensor fusion and data fusion are often used, since data preparation is part of the activity … in image processing methods from the field of ANNs are mainly used, also because potential errors are mostly unknown;

Camera systems are installed, and then one begins to examine what might be relevant within the image data. This is done together with the experts (customers) ... only when the relevant features have been identified can one start to consider which models fit best;

Applied processes are the Wiener Process for determination of degression in the case of known physical wear and for simple systems such as wind turbines ... but can only be used when material behaviour is known or for simple processes such as rotating machine parts ... ANNs are preferred for complex problems;

The knowledge of the experts usually has to be worked out first ... in particular the question of which parameters are actually relevant is of importance ... ANNs are preferred for this purpose;

Classifications are rarely applied because data are not classifiable due to a variety of factors that are often not deterministic ... this is especially true in the offshore environment;

…

For security assistance systems usually ANNs used, since only the relevant features have to be ascertained and no valid training data is available ... in the practical application usually combinations of algorithms are used ... which ones they are, depends on the application case ... for each system a special application case is created with its own strategies and objectives ... Algorithms have to be selected for each application case ... selection is usually done iteratively and according to t“e "trial and error” approach;

A combination of ANN and reduction techniques (in an upstream step) is often used ... especially for image and video data, the reduction of data is decisive .... often semantic reduction, for example, certain layers, colours or shapes are filtered out before the actual pattern recognition takes place;

Conclusion: if behaviour is known, then usually the Wiener Process is used, otherwise ANNs;

…

In the area of port facilities, mostly short-term forecasts using statistical methods such as ARMA models (ARIMA. ARMAX) ... otherwise data mining. ML and CEP procedures are used ... Procedures usually two-step, first attempt is made to roughly identify relevant features and relationships using statistical methods, then mathematical models are used when behaviour/degression is known precisely and is deterministic ... if not. ANNs are applied;

From experience, application of supervised ANN provides the best results … this requires that the results of an ANN is constantly discussed with experts and the ANN is optimised using their evaluation ... SVM would theoretically be suitable for many applications (pattern recognition), but application is too complicated;

Which method is used depends on the application and varies widely ... no general statements are possible, as to which methods can be used for which requirements ... only general suitability of the methods (for example linear degressive or non-linear degressive) is taken into account upon selection

…

Using the ThingWorx tool for analytics and PdM ... Algorithms are not directly visible in the tool ... in everyday project work, the used algorithms are rarely followed, mostly the automatic selection and the recommendations of the tool are followed ... the tool analyses all data and specifies which areas should be classified;

ThingWorx automatically displays 5 algorithms that are best and worst suited to the task, based on the provided and classified data … based on this suggestion, one could now decide which algorithms are applied ... but usually the ThingWorx proposal is followed ... Neural Net (ANN) is most often suggested as a suitable solution, next to it is Decision Tree. Gradient Boost. Linear/Logistic Regression. Random Forest ... a table outlining the suitability of the above procedures for specific applications is given on the help pages of ThingWorx (https://support.ptc.com/help/thingworx_hc/thingworx_analytics_8), in the "ThingWorx Analytics Functionality / Prediction Model Generation" section ... the statements of ThingWorx are not valid for all applications however;

Classification is mostly binary in the first step and then multiclass in the second step ... classes are mostly events, which are defined and specified by ThingWorx ... learning is fully automated by means of iterative classification ... evaluation of the results with experts and then adaptation of the classifications for subsequent continuous improvements ... corresponds to a supervised approach.

…

Use mainly of PCA and the various modifications thereof ... often also clustering and sliding window to reduce data to the relevant features ... in current projects it is usually about customers taking the first step in the direction of data analytics and PdM ... therefore only simple steps can be implemented at first ... the customer first needs to learn what is possible and what goals are important for him ... in addition, the infrastructure is usually not sufficiently advanced to allow data to be transmitted online in streams … therefore, data first needs to be analysed and discussed with the customer ... this requires simple standard procedures, even if it is not possible to achieve 100% valid results;

Pre-processing for complexity reduction is very important (clustering, sliding window) ... GMM and decision trees for estimating features in statistical models ... ANN have not been used really, because of missing data, but are certainly relevant in the future;

…

At the beginning a project, mostly binary classifications in order to become familiar with data and to take the first steps ... Naive Bayes classifiers and decision trees are often used here ... decision trees are very good, because of understandable representation for discussion with experts ... decision trees especially when causal connections between failures and causative component can be represented ... always iterative process with the customers experts, consisting of intensive pre-processing of data, evaluation of the results by the experts and renewed processing with the new findings ... sometimes the customer also provides the classification.

For anomaly detection. Gaussian Mixture Models (GMM) are mostly used ... also used for pre-processing of data to separate relevant data from non-relevant ... this currently serves primarily to detect relevant signals and eliminate noise, less to reduce the complexity ... in real time applications, the reduction of complexity is important ... but currently, real projects only offline;

Application of linear methods when conclusions on the cause are desired ... this only works with simple systems ... usually non-linear methods are used for more complex applications ... ANNs are used in complex applications, for example auto-encoders are used.

In general, the choice of methods depends strongly on the application and should be selected individually ... ANNs will become the most common method for complex online systems in the future...

A.2.1.3 Tools used by the Experts

No tools in use ... a special application is implemented for each project ... projects very different and usually very special application areas, such as offshore wind turbines ... algorithms are not implemented themselves, use of open-source libraries;

…

No tool in use, only libraries for the known algorithms

…

RapidMiner. WeKa and various libraries for algorithms, as well as aforge vb.net for image processing

…

ThingWorx Analyse as a tool for Predictive Analytics and PdM applications ... but only deals with the algorithms, data is stored in an upstream IoT platform and transferred in a pre-processed and reduced form to ThingWorx ... as an IoT platform for data management. Cloudera Data Hub is used ... here full pre-processing of data takes place ... data streams are processed and stored here and then passed on in prepared batches to ThingWorx … since one works only in ThingWorx, one does not come into contact with data streams ... so far with real-time one never came into contact … this is performed by the Cloudera Data Hub … data processing in data hub and response times of ThingWorx have been sufficient in all projects so far;

Within the company, it is generally believed that ThingWorx cuts out the data scientist ... therefore only IT professionals are involved in projects;

Partly cloud-based platforms such as Microsoft Azure, currently still very few, but strongly increasing demand for cloud solutions;

…

Own software solution / analytics product, which uses open source libraries ... is the core product of the company, so all projects are implemented with this tool ... own solution uses a specially developed data format, which is part of the company's IP ... in projects, all heterogeneous data sources and formats are converted into the internal format … in this conversion step, data is classified with the involvement of the customer's experts;

…

Scikit-learn (machine learning library for Python) is used ... scikit already has numerous algorithms for classification, regression and clustering ... mathematical libraries like NumPy and SciPy can be integrated directly ... for those working in science, this is the ideal tool since it is highly flexible and open source ... since being open source, the code can be viewed and analysed in case of problems ... errors can be identified by compiling the entire source code;

A.2.1.4 Targets of Predictive Maintenance Applications

RUL and TTF are the most frequent objectives of PdM applications ... further objectives depend on the specific application ... costs of failures are certainly the most important, as unplanned failures cause the highest costs ... often though, indirect costs are also relevant ... for example, for rapid mass production of small parts, a fast forecast is important, since a lot of costly rejects can be produced very quickly ... avoid follow-on costs caused by heavy and therefore expensive accessibility … example: offshore wind turbines, in this case cheap parts, such as the carbon brushes of the generator, are exchanged at every entry regardless of their wear;

Achieve pre-set numbers, reliable production, avoid interruptions in production processes with just in time and just in sequence;

Profit optimization ... in case of wind turbines, maintenance planned in the low yield seasons … planning subject to weather conditions and weather periods providing high-yield;

…

In case of offshore wind turbines the most important goal is to prevent breakdowns ... system standstills and unplanned offshore operations are very expensive ... in case of unfavourable weather conditions it may be that, after an unscheduled failure, the system cannot reached and services for days ... optimal planning of maintenance work is required due to difficult accessibility and many other external influences;

…

Objectives vary greatly and depend on the application ... the most common objectives are RUL and TTF to avoid unplanned failures while maximizing possible useful lifespan ... the original goal is usually to reduce life cycle costs;

…

At the beginning, there is always the reduction of production costs ... unplanned failures and reactive maintenance strategies generate the highest costs (production losses, required maintenance personnel, storage of spare parts e.g.) ... therefore PdM always aims at predicting RUL and TTF ... the predictions of the PdM system should also be made available to higher-level planning systems (EAM. ERP);

A further goal is analysis of when and how often a system can be operated at maximum load or even over it ... this is one of the analyses that our company offers its customers ... through such peak load analyses, companies are able to meet production peaks without requiring additional systems ... maximization of system utilization is an important cost factor in the industry ... nevertheless, it must also be possible to fully meet production peaks ... the knowledge of how long you can operate a system at its limits and how long any interim recuperation phases are, is determined through physical knowledge and historical data and model calculations/simulations;

…

Prediction of failures (TTF) and the remaining useful lifetime (RUL) are most important ... in a specific project, the aim was to save on sensors and to replace many sensors with a few multifunctional sensors with integrated diagnostic functions

…

Avoiding unplanned failures ... prediction of RUL and TTF ... causality analyses in real time; the aim is to use patterns to determine which components or which state of a system caused the error;

In spite of very mature procedures and ever-increasing knowledge in predictive analytics, one must realize that not everything is predictable ... even if ideal data would be available in an unlimited quantity, there are events in industrial systems that cannot be explained by data ... the topics Predictive Analytics. PdM. IIoT, and I4.0 have been very hyped, which is why there are often large expectations which are cannot always be met in practice;

A.2.1.5 The Relevance of Real-time Demands and Stream Processing

Very important for systems for the production of micro components, as large quantities are produced in a very short time ... this area is mainly dominated by CBM and CEP. PdM even less;

With offshore wind turbines, data volume is currently not quite as high ... but this will be changed by more and more sensor technology ... precisely because failures in the offshore area are so expensive, the sensor technology is greatly expanded here ... data streams will also be available at wind turbines ... for the rotating parts of a wind turbine, the predictions are not time-critical; here simple degression analyses are sufficient ... but, for carbon fibre components, very time-critical predictions (fractures) could be necessary with availability of appropriate sensors;

…

Whether real-time is required depends on the objectives and the physical properties of the system... but also the existing IT infrastructure is important ... mobile service teams need information in real-time

…

Real-time is still rarely a special requirement ... so far, the prerequisites have not been met in most projects ... in the future, most systems will have to deal with data streams and meet real-time requirements ... Esper has been used so far, especially for event stream processing (ESP) ... otherwise OPC UA was used;

…

Response times are given by the platform Thing Worx ... streaming techniques are already included ... as an IoT platform for data storage. Cloudera Data Hub is used ... the entire pre-processing of data takes place here ... Data streams are processed and stored here and then passed on as prepared batches to ThingWorx. … because you work only in ThingWorx, one does not come into contact with data streams ... real-time has never been an explicit requirement ... Data processing in data hub and response times of ThingWorx have so far been sufficient in all projects;

Applications for face detection have already developed ... here real-time is very important ... will be achieved mostly by data reduction ... the algorithms themselves were not precisely real-time capable ... here was missing a procedure, or the knowledge how to process the algorithms more quickly;

…

So far no real-time was not an explicit requirement ... response times have been sufficient with batch approach so far ... in the future, this will certainly be more relevant ... in feedback loops and adaptive online systems real-time is essential;

OPC UA is currently used as a communication protocol and so far fulfils the time requirements ... so far stream processing has no application field, but projects in planning alongside or with OPC UPA

…

Currently, our own product is already being developed with real-time support ... real-time processing is to be implemented in manageable application scenarios with OPC UA ... for complex applications and data streams, a streaming platform (Apache Spark Streaming, including ML functions) is being integrated;

Previous real-time requirements had the goal of providing result data very quickly in SCADA systems ... otherwise, the requirements are currently mostly in minutes or hours ... the desire for visualization of data and forecasts is increasingly expressed by customers ... reactive web front ends also need correspondingly fast systems to provide data;

A.2.1.6 Assessment of Complexity Factors

High dimensionality

In PdM applications, image processing is used and there is a high dimensionality in data (resolution, depths, planes, contrasts) ... a preliminary step to reduction is common here ... but leads to problems in the models ... superordinate relationships are often lost ... overfitting and underfitting problems ... multi-sensors generate enormous amounts of data, so dimensionality will become relevant in systems without image processing;

With available raw data, high dimensionality is created quickly ... current methods are partly useless for this ... reduction techniques can help here but can’t solve the problem of lacking real-time support of the algorithms ... in reduction, the question "what are the relevant data" is an endless research topic;

Classification for reduction to the relevant data is required ... mostly multi-level ... in the first step binary, then multiclass;

Reduction by clustering and window technologies ... in ThingWorx no topic, is done by upstream Cloudera Data Hub;

In real projects dimensionality was a problem, the combination of high data availability and too few meaningful data ... reduction techniques must recognize the relevant data ... then dimensionality in current systems is no problem ... in industrial systems, a lot of sensors are installed and lots of data are generated, but often the knowledge of which data are relevant for analytics applications is missing;

Concept Drifts - non-stationary or evolving data

Rarely occurs in production plants … in the field of work safety, spatial monitoring for robots - human interaction ... movement of people can change rapidly ... movement forecasts drift ... currently it's not attempted to solve methods for recognizing concept drifts, but via probability calculations

Concept drifts are also present in adaptive systems. ... own project with autonomous conveyor vehicles ... so far, only autonomous vehicles with defined behaviour patterns were travelling in a closed environment ... so far no adaptive systems were necessary ... now also humans can move in the same environment ... the control systems of the autonomous Vehicles must now become adaptive and can react to unforeseeable events;

This is a very difficult topic ... with PdM for production plants, this rarely occurs, so this is hardly taken into account in practice and is often interpreted as an anomaly;

Known from energy installations here, sometimes basic behaviour changes, for example by energy trading ... with PdM for industrial system is not so dynamic ... changes in industrial system are known ... therefore not usually considered in the modelling;

Concept-drifts are rather rare in PdM ... usually these are in fact errors in data interpretation which have been erroneously interpreted as drift ... they are hardly considered in real applications;

Is treated implicitly by Thing Worx

Concept drifts are little known or not at all in real projects ... seldom is a permanent validation of the models on drifts ... therefore they would not even be detected ... drifts arise in production plants mostly by changing the hardware or calibration of sensors ... the models have to be adjusted thereafter ... special operating modes, such as maintenance operations, look like concept drifts ... special operating modes are usually suppressed in the PdM system;

Concept drifts are rare in industrial systems ... data analytics is usually carried out on stationary data

Non-linear data

In offshore wind turbines data are often missing due to non-transmission ... this makes the forecast very difficult and leads to non-linear situations and misinterpretations ... systematic treatment of non-linearity currently not available ... in the case of forecasts based on image data, missing data is not acceptable ;

Non-linear data generate higher modelling effort ... relationships are determined step-by-step through simulations and validated again and again ... most systems assume linearity and, for example, ignore non-linear areas ... in image processing, this is often modelled over probabilities and events ... It's a critical topic also for PdM applications ... but difficult to handle, always has to be considered for the respective application ... this has so far been little handled in previously known systems ... if not handled, this can lead to errors in the prediction;

Non-linearity is rarely considered in PdM applications ... often non-linear relationships are erroneously interpreted as errors in data interpretation

Non-linearity is treated implicitly by Thing Worx, how exactly is not known ... actually, all factors of complexity are treated implicitly by ThingWorx ... so far no own measures have been carried out;

Simple systems contain mostly linear relationships ... in complex systems, relationships are mostly non-linear ... maintenance slots also produce such effects, in which other values are measured or not all sensors are active ... special operating modes are excluded in the modelling or specially modelled;

For the modelling of non-linear regressions methods like MLP. ANN. SVM are used ... is however a very complex topic ... in particular the correct parametrisation is very difficult ... errors in the parametrisation quickly lead to incorrect results ... causes for non-linearity are often different operating modes ... less frequent changes in the hardware or calibrations ... non-linear physical properties are usually ignored in PdM applications and the resulting errors are accepted

A.2.1.7 The Relevance of Historical Data

Is strongly dependent on the application ... with online systems, historical data are usually processed incrementally ... loss of quality must be accepted ... with offshore wind turbines, the full history of data is currently being processed as no real time is required;

…

Very important for training the models ... rarely available in practical projects, since the systems already exist, but so far data from the process control systems were not saved ... if historical data were present, then these were often wrong or incomplete ... if systems were rebuilt in the past, then this must be known, otherwise this leads to incorrect interpretation ... in practice it often happens that such evens in the historical data are not known as such ... historical data are therefore to be considered very critical at the beginning of a project;

…

Availability of valid historical data is extremely important in projects ... it serves as a basis for extracting the expert knowledge for supervised learning ... in adaptive systems, forecasting models must be used, which can also handle the historical data ... dealing with historical data in online systems is usually incremental

…

Data is stored in the IoT platform (Cloudera Data Hub) ... database technology is selectable (MS SQL. H2. Hive. PostgreSQL. No SQL) ... historical data are incrementally transferred to ThingWorx as part of the current data ... the overall historical data can be queried by ThingWorx in the IoT platform, but is only made with manual analysis (manually create, adapt or extend the models);

Historical knowledge about the system is very important and usually not available or only incomplete ... must be critical validated at the start of the project ... events in historical data are always discussed with the experts ... in modelling, an attempt is made to explain the relationship between historical data and the physical model;

…

Historical data are very important in order to gain knowledge about the system ... in most cases few or incomplete historical data are available ... analytics applications always includes all historical data ... no incremental procedure ... current applications are all operated in batch mode ... response times are currently no problem ... in projects, data sets are not too large ... data volumes will increase in the future and thus also the amount of historical data ... then certainly other approaches will be required ... incremental treatment of historical data is certainly a way;

…

Historical data is very important ... often only incomplete and not valid ... based on historical data, the feasibility of the desired application is often checked in projects at the beginning ... feasibility studies often only use historical data ... in current projects, historical data are streamed from a database using a software to simulate the effect of data streams;

A.2.1.8 Learning Methods and Adaptive Models

In solutions based on image processing, data patterns of errors are unknown ... camera systems are installed and then one begins to examine what could be relevant in the image data ... this is done together with the experts (customers) ... models can only be generated if the relevant features are identified ... the process starts initially unsupervised and is then enriched with expert knowledge ... non-adaptive models are only suitable for initial data familiarisation with a stable model ... otherwise the models have to be adaptive ... either manually or online ... in current projects not online, therefore also no problems with computing capacities;

…

In the case of security assistance systems, the training data is very important ... to have enough training data, image databases are usually connected to supply additional (synthetic) training data ... these are, for example, people from different perspectives ... such external training sources are not available in industrial systems;

In PdM applications, adaptive models are important but not as critical as in image processing, since industrial systems do not change their behaviour so abruptly;

…

Adaptive learning is very important ... supervised learning takes place in many loops for optimization ... models must always adapt themselves automatically ... in the operation of a system, the conditions rarely change ... here it is more about constantly improving the model ... adaptive models come with heavy processing requirements, and always cause load spikes in the system ... in real-time systems, special measures are required, such as parallel processing and scalable infrastructures;

…

Adaptive behaviour is important for permanent training of models ... at the beginning of the project, only about 50 % of the errors are detected (false positive rate) in addition, misinterpretations are added (false negative rate) ... ThingWorx learns permanently and adapts the models automatically ... ThingWorx decides how adaptive models are and how often they are adapted ... presumably this happens with every new sample ... the learning process is based on classifications of the results, which have to be done by the user in ThingWorx ... learning is based on ANNs;

…

Adaptive learning is essential to develop valid models ... expert knowledge is always required ... training is a constant process ... mostly concerning classifications ... because currently only batch systems are in use, the updating of the models is triggered manually ... in online systems this must be done automatically ... for real-time requirements, the underlying system must be able to adapt the models in time ... this is a difficult problem where real-time IT infrastructures and processing techniques are required;

…

Right now, only batch systems are actually used ... model adaptation is performed manually ... it's a constant process, as new insights are always being gained from the system results ... the system is constantly being improved ... online systems with real-time requirements are currently under internal development ... to overcome overloading the system with continuous model adaptations, incremental learning was selected as an approach ... in addition, learning processes are processed in micro-batches using parallel processing;

The adaptive models also learn concept drifts ... depending on the application, this is either desirable or leads to errors ... adaptive learning is therefore a complex task and must be adapted to the particular application;

In real projects, the amount of really relevant data is often too small ... unbalanced data, missing data from errors and worst case scenarios make it difficult to develop adaptive learning processes ... with missing historical data it will also be very difficult to develop adaptive models;

A.2.1.9 Validation of Prediction Quality

Mostly on the basis of synthetically generated data, since the necessary error scenarios are rarely present in real data … tests are often done by simulations ... real tests on the live system rarely possible (example offshore wind turbines) ... synthetic data are easy to generate and can all known error scenarios cover…

…

Within image processing by using visual control (false positive rate) ... this is only possible in this area, as a tester without special knowledge can, for example, recognize whether a person is in a picture or not ... in industrial systems, the matching of results always takes place with experts (was this error recognized correctly, was the measure taken correct. ...) ... field tests are also carried out, although rarely, thanks to the availability of the equipment and possible damage caused by provoked errors ... validation is mostly done in simulations ... ideally validation is performed with real data, but they rarely have the necessary characteristics;

…

Validation with current and historical data, if available ... usually error scenarios have to be inserted into data ... 80/20-rule for AN–s - 80% of data are used for training. 20% for validation of the results ... since data is often unbalanced, this approach often does not work … validation always requires expert knowledge ... no automatic procedures for validation or benchmarking are known;

…

In practice, validation is often not based on data, but on monitoring the results, in specific terms, the failure rates are in operation ... if the failure rate drops, the model seems to be getting better … the failure rate increases, manual search is performed for modelling errors ... ultimately it's always about trying out and gaining experience;

…

Results are evaluated by experts, new findings are incorporated into the model ... mathematical or statistical procedures for validation are not known ... presumably, they do not exist, as results are always very much dependent on the application and no valid assessment is possible ... indeed studies have already been developed on this subject and approaches for validation procedures have also been developed, but these are not generally valid

…

Validation is always by expert knowledge ... manual assessment and evaluation of results by experts ... used methods include cross-validations and classifications ... no methods for automatic validation are known ... it's probably also not possible due to strong application-dependence;

A.2.2 Superordinate Statements

In this section the superordinate statements or findings on the respective thematic blocks were presented in note form. These superordinate statements formed the outline of the evaluation section 5.5.

A.2.2.1 Experts Main Areas of Interest

Field of work are usually concentrated on the implementation of basic steps.

Feature extraction, i.e. the determination of the relevant features, is an important task area.

Currently still frequent anomaly detection and rarely predictions.

In addition to prediction of failures, data analyses are also performed to identify optimization potentials in the industrial plants.

Currently mostly offline systems with batch processing in practical use.

Online systems with real-time requirements are not yet widespread, the industry is not yet as far as one would assume due to the topics IIoT and I4.0.

In the future, online and real-time systems will be the most important applications.

Autonomous systems and image processing require real-time processing.

Many people are already working on solutions for online systems with streaming analytics, real-time and ML.

I4.0 applications are the innovation drivers for such technologies.

The top 5 mostly mentioned terms in this area were: depends on data (8), online systems (4), batch-approach (3), fault detection (3), real-time approaches (3)

A.2.2.2 Methods and Algorithms used by the Experts

Data pre-processing and reduction techniques to reduce to the relevant features are always advisable.

ANNs can help identify relevant features when no expert knowledge is available.

Loops, consisting of the steps of analysing data, evaluating results by experts, and optimizing models.

No general rules for selecting the methods, deep expert knowledge and trial-and-error, always depending on the application.

Mostly advanced step to reduction or filtering, often statistical methods or SW upstream to an ANN.

If physical behaviour is known and low complexity, then linear regressions like Wiener process.

In deterministic systems and existing expert knowledge classifications or PCA.

For complex systems and many unknowns, primary ANNs

In case of bad training data, use ANNs

Expert knowledge facilitates the modelling of the layers of ANN's and improves their results.

SVM too difficult to parametrize in real applications.

In real systems usually combinations of several algorithms apply, strongly depends on the application case, no general statement about combinations possible.

Prognosis horizon has an influence on the selection of the method.

Tools are very powerful and adaptive, but the internal functionality is opaque.

The top 5 mostly mentioned terms in this area were: ANN (14). Application case (11). Classification (6). Feature and feature extraction (5) and decision trees (4)

A.2.2.3 Tools used by the Experts

Often no data science tool is used.

Development of systems takes place individually for every application

Open-source libraries for algorithms and mathematical functions.

Open-source data science tools in the preparation phase of projects to analyse data and find suitable methods.

If a commercial data science tool is used, it is completely entrusted to it.

Cloud-based platforms are the future environment.

In addition to the tool used, data formats and protocols used are also relevant (OPC UA).

In research and science primarily open-source libraries and tools are used.

A.2.2.4 Targets of Predictive Maintenance Applications

Ultimately, the goal is always to reduce the Lifecycle Costs (LCC) and Total Cost of Ownership (TCO).

Unplanned failures cause the highest costs, so the prediction of the errors and the calculation of the remaining useful lifetime is the focus.

The goals of a PdM system always depend on the application.

Costs that cause an unscheduled failure always depend on the application case.

Other objectives are to optimise the maintenance strategy in order to reduce costs and ensure reliable production, maximize plant utilization / yield optimization, minimising the reject rate and other qualitative criteria.

The predictions from the PdM system should be transferred to the higher-level planning systems (EAM. ERP) and used there for overall planning.

A.2.2.5 The Relevance of Real-time Demands and Stream Processing

At the moment still relatively rare, as the necessary communication infrastructure and the development of the sensor systems in industrial environments is only just beginning.

At the moment, often concentrating on real-time-capable CMB and CEP systems. PdM systems will then build on it.

Whether real-time is required depends on the application case and the physical properties of the system.

For simple systems and with linear regression, the processing speed is sufficient to date.

PdM systems are often are only at the beginning, concentrate on first simple steps using batch processing.

Industry has yet to gain experience until complex online systems with stream processing in real time become reality.

For autonomous systems (I4.0) in IIoT environments, the PdM system must provide fast response times.

Future mobile service teams that interact with systems (M2M Communication. OPCUA) need real-time information.

Online systems and real-time processing will be the dominant form in the future.

However, there will always be applications where offline and batch processing is sufficient.

Technologically, streaming platforms and real-time OPC UA servers will be relevant.

Data science tools often integrate an IoT platform (IoT hub) to realize stream processing and provide real-time results.

Reduction techniques are very important at real-time because the actual algorithms are rarely real-time capable and not suitable for distributed and parallel processing.

Methods and knowledge are missing in order to make algorithms real-time.

In the case of adaptive online systems, the permanent learning process must be simple and therefore very efficient. Real-time techniques would certainly help.

A.2.2.6 Assessment of Complexity Factors

High dimensionality

Whether dimensionality is a relevant complexity depends on the application.

In current applications it is usually not yet clear which data is relevant at all.

Data volume, however, increases very strongly in industrial environments (IIoT. I4.0), so the dimensionality of data will increase steadily.

Reduction techniques are very important in order to liberate the relevant data.

Reduction of the relevant data can also lead to problems, e.g. the meaning of data can be lost.

In complex systems, reduction is a complex and error-prone task.

With reduction techniques, good response times are still being achieved, but this will change with increasing data volume.

Reduction techniques reduce the complexity for the following algorithms, but the lack of real-time capability of the algorithms themselves is not solved.

In the future, the entire PdM system must be scalable and must be able to react elastically to high loads.

Concept Drifts - non-stationary or evolving data

Until now, concept drifts are rarely considered in PdM systems.

In industrial plants, relevant drifts do not actually occur.

Methods for the automatic detection of drifts are not known.

If changes to the hardware or recalibrations of the system are carried out, the model is also adapted.

Effects of external factors (e.g. humidity) are treated as measured values and not as concept drifts.

In the case of highly adaptive systems, drifts are relevant and are currently usually dealt with by probability calculations and CEP.

When concept drifts occur, these are usually interpreted as anomaly.

Adaptive models learn drifts automatically. It must be decided whether this is desired or have to be prevented.

Non-linear data

The problem of non-linear data is often ignored in practical applications.

For non-linear data, the effort involved in creating the model increases.

Non-linear physical properties are often simply ignored or linearly modelled.

This causes errors in the results, the cause of which is often not interpreted correctly.

Often, non-linear data is simply filtered out to apply linear models.

Special operating modes and maintenance operations can cause non-linear data and are usually not handled, or handled via separate sub-models.

For nonlinear models, methods from the area of neural networks (ANN. MLP) and ML (SVM) are used.

Modelling is very complex; imprecise parameterization can quickly lead to wrong results.

How precisely high-level data science tools deal with non-linearity is not known.

A.2.2.7 The Relevance of Historical Data

The historical data, if available and valid, comprises the entire knowledge of the industrial plant.

Historical data are very important for predictions and should always be included.

On the basis of the historical data, the existing knowledge can be worked out in dialogue with experts.

With batch systems with any latency, all historical raw data can be processed easily.

In the case of online systems and real-time requirements, historical data are usually processed incrementally in order to keep the computational effort to a minimum.

In the case of incremental treatment of historical data, important details or connections can be lost.

The resulting loss of quality in the results must still be accepted at the moment.

Ideally, historical data should be processed in raw form.

In future with highly optimised and very powerful PdM systems, it would be desirable to process historical data also in its raw form.

Adaptive models should be trained with historical raw data.

High-level data science tools often use IoT platforms (IoT Hub) to store all raw data.

A.2.2.8 Learning Methods and Adaptive Models

Adaptive learning is essential to develop a valid model.

The main goal is to improve the model permanently.

The learning of new conditions (e.g. concept drifts) is less important.

Often, a lot of data is available, but few are errors. This is problematic for the learning process.

In batch systems, models are adapted manually.

In online systems, models must automatically and permanently adapt themselves.

A permanent learning and adaptation process is complex in implementation.

Requires high computing power.

Automatic learning and adaptation processes must be scalable and elastic.

Required scalability is highly dependent on the application case.

Manual adaptation usually takes place by classification with expert knowledge.

Automatic adaptation requires methods from the ML area, but also expert knowledge.

A.2.2.9 Validation of Prediction Quality

Validation of the results of a PdM system is very important.

Expert knowledge is always required for validation.

From the evaluation of the results further expert knowledge for the adaptation of the models can be gained.

Assessment depends on the application case.

Evaluation mostly by simulations.

Real data usually contains too few errors, especially critical errors.

Usually use of synthetic data, or enrichment of real data with artificial errors.

If data contain sufficient errors. 80/20 rule (80% of data for training the model. 20% for validation).

Validation mostly through visual control and through discussions with the experts.

No universal procedure known, no general mathematical or statistical procedures.

No way for benchmarks known.

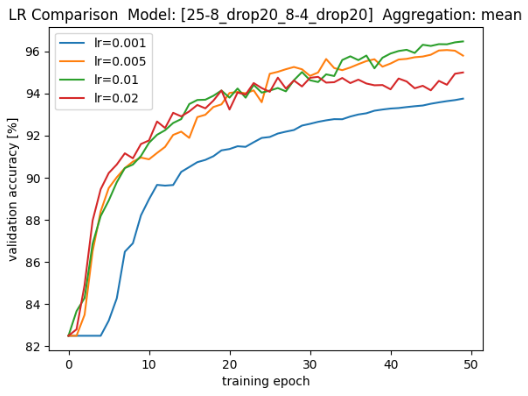
The evaluation criteria are methods such as type I error and type II error or indirect values such as, for example, failure rates.

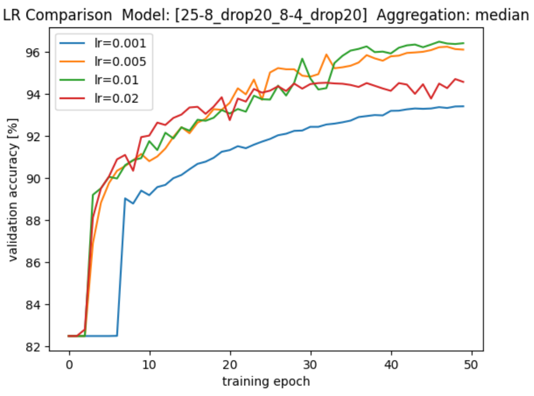
Evaluation by cross-validation and classifications of results.

A.3 Laboratory Test Plots

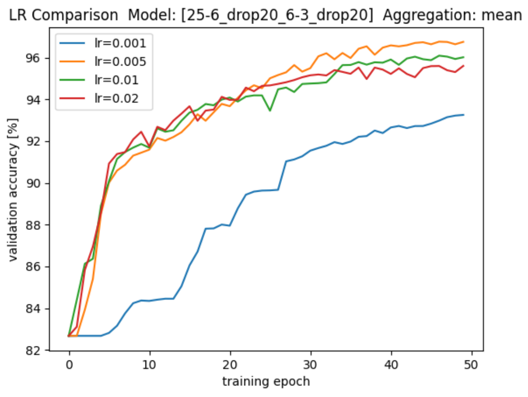
A.3.1 Comparison of the Reference Model Learning Rates

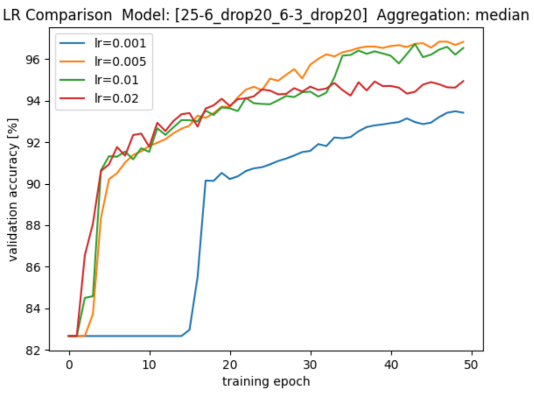
Comparison of the different learning rates for reference model configuration 25-8_drop20_8-4_drop20. Learning rate 0.01 was chosen as the best one.



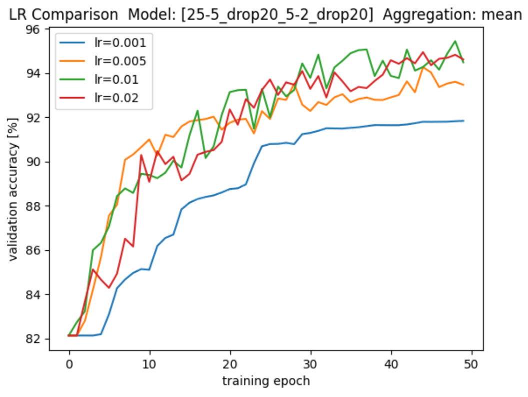


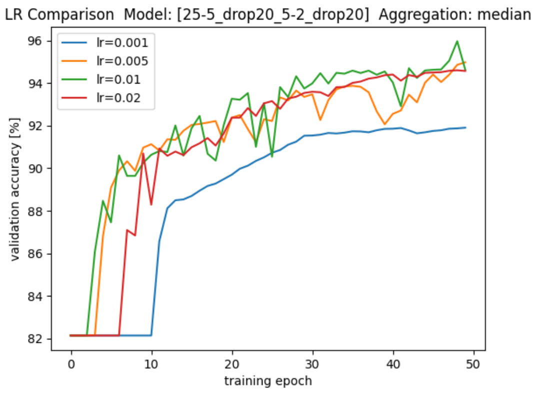
Comparison of the different learning rates for reference model configuration 25-6_drop20_6-3_drop20. Learning rate 0.005 was chosen as the best one.



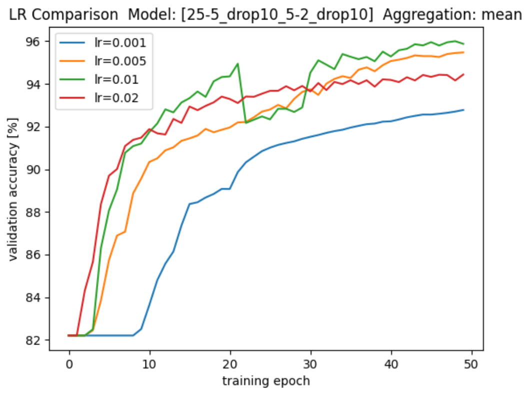


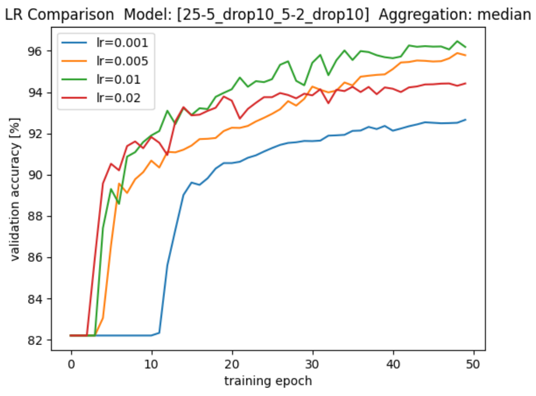
Comparison of the different learning rates for reference model configuration 25-5_drop20_5-2_drop20. Learning rate 0.01 was chosen as the best one.





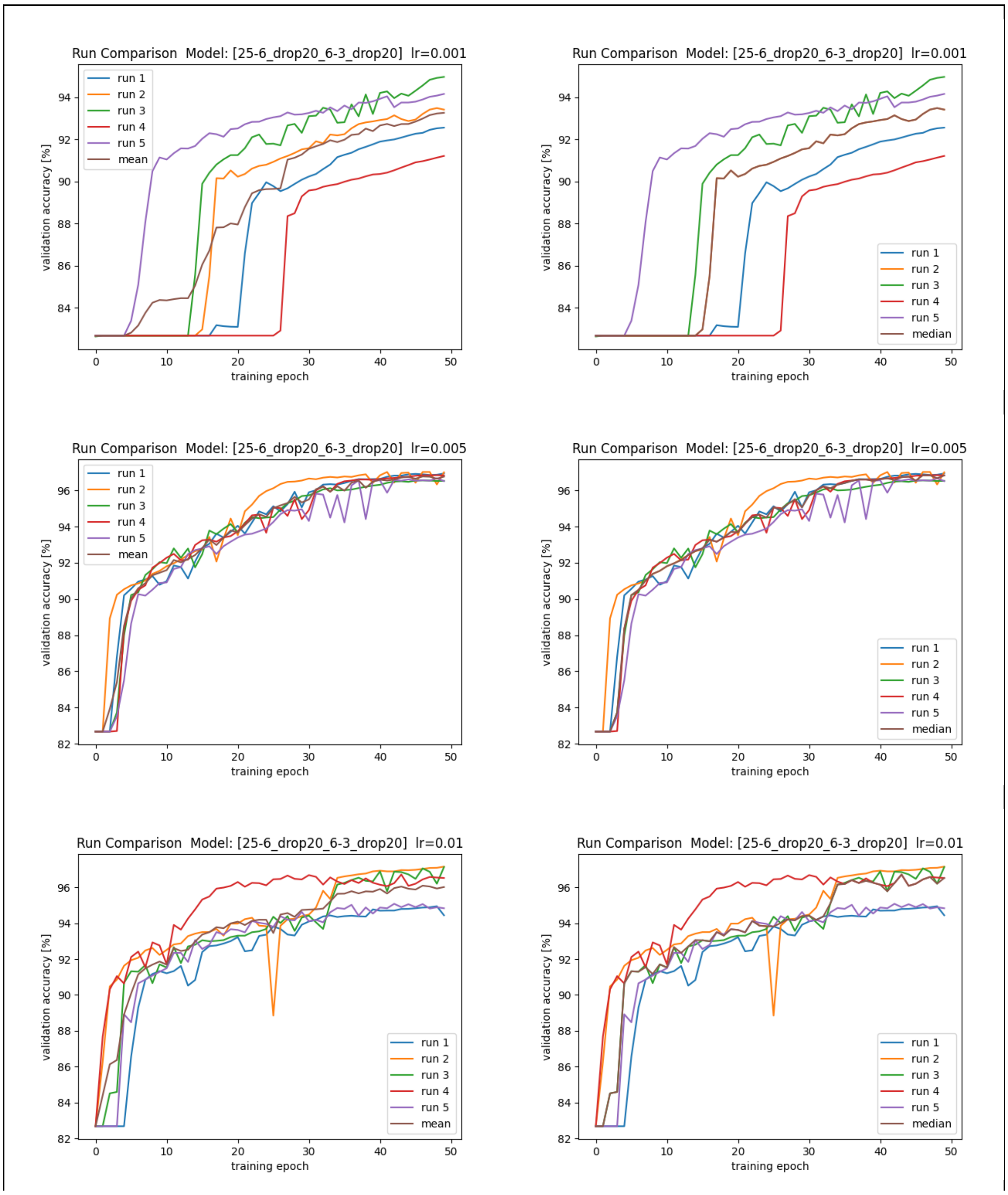
Comparison of the different learning rates for reference model configuration 25-5_drop10_5-2_drop10. Learning rate 0.01 was chosen as the best one.

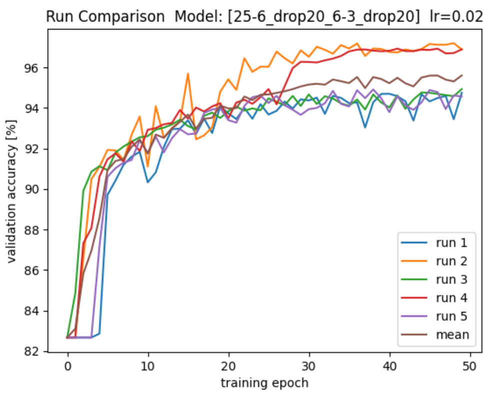


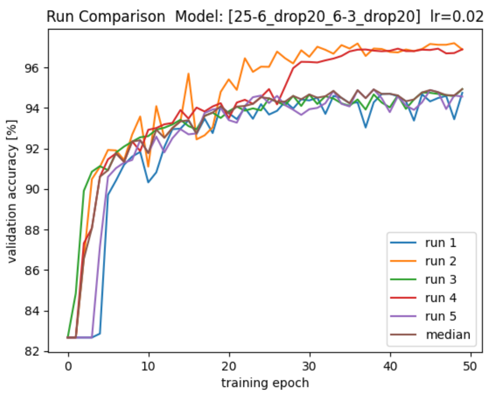


A.3.2 Comparison of the Reference Model Runs

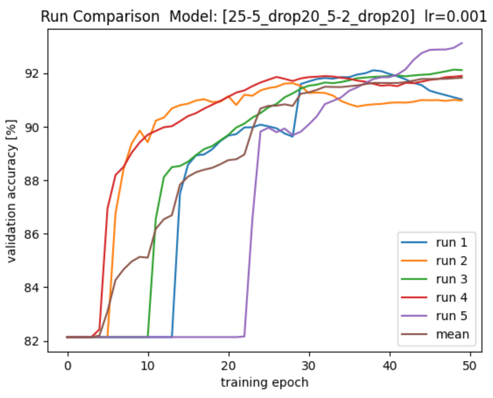
Results for reference model configuration 25-6_drop20_6-3_drop20 with all different learning rates. Learning rate 0.005 was chosen as the best one.

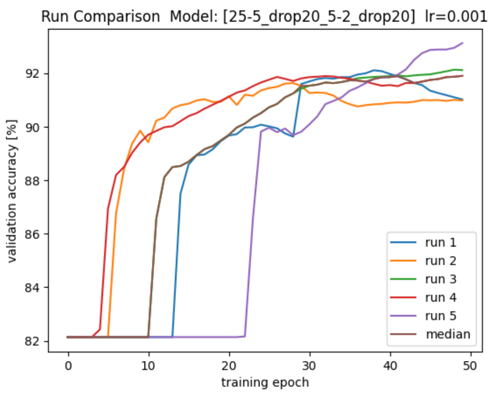


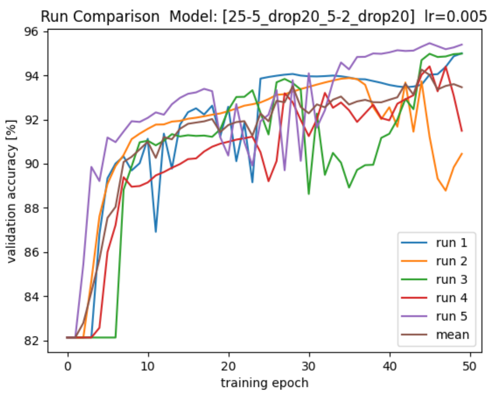


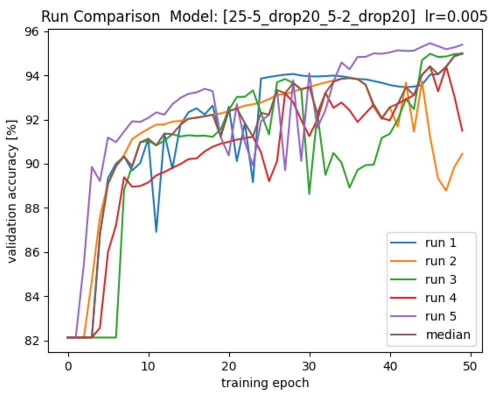


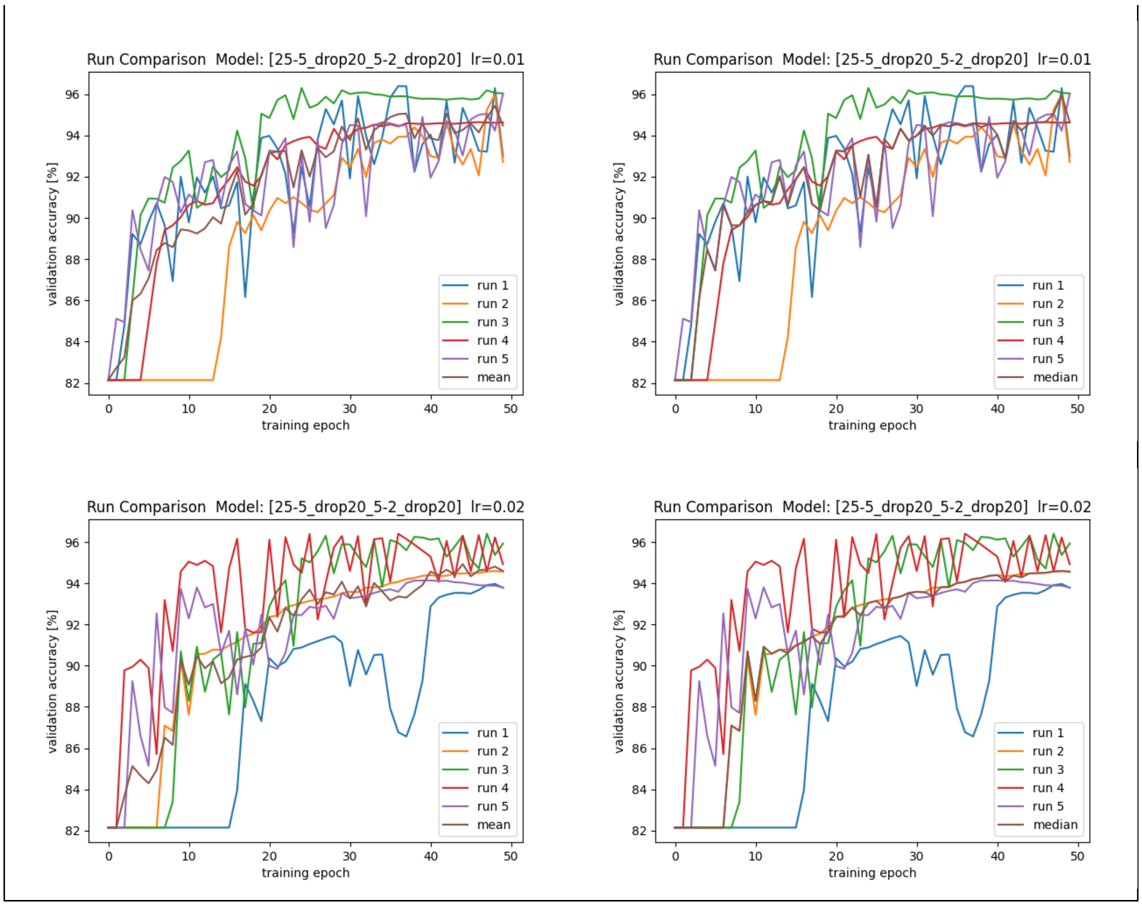
Results for reference model configuration 25-5_drop20_5-2_drop20 with all different learning rates. Learning rate 0.01 was chosen as the best one.



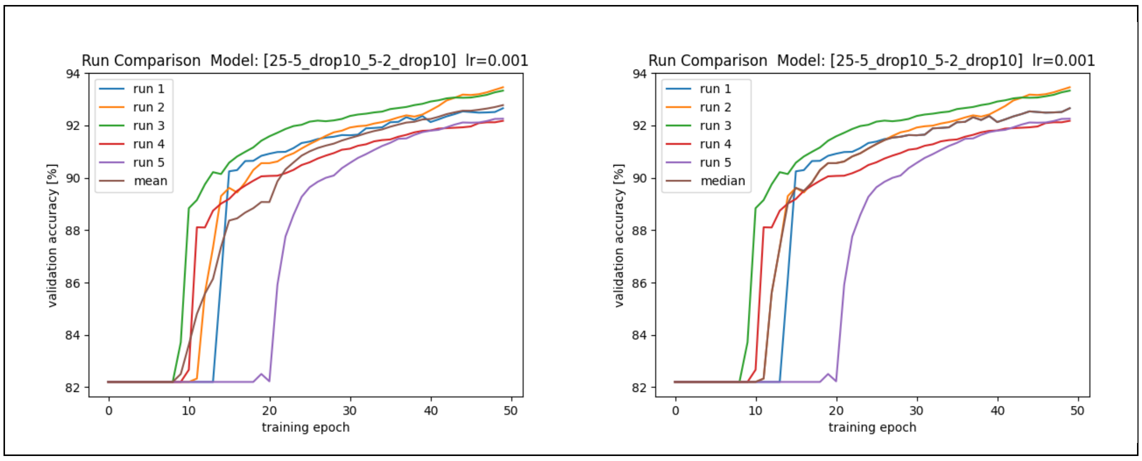


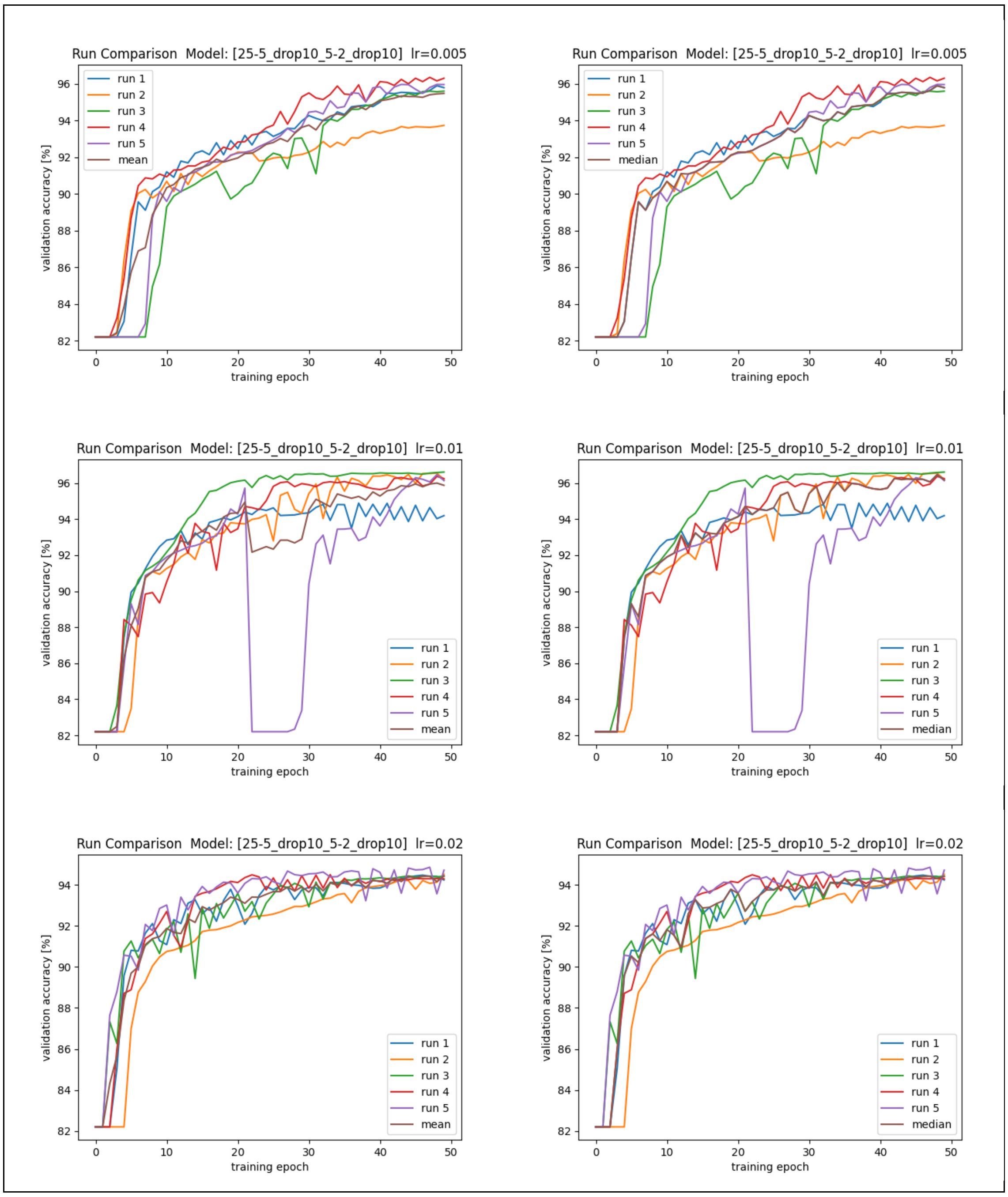






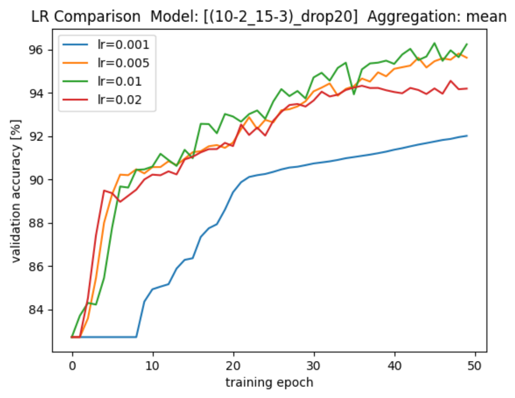
Results for reference model configuration 25-5_drop10_5-2_drop10 with all different learning rates. Learning rate 0.01 was chosen as the best one.

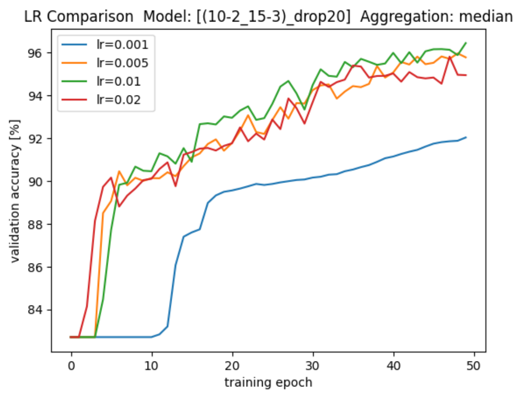



Run Comparison Model: [25-5_drop10_5-2_drop10] lr=0.005
Run Comparison Model: [25-5_drop10_5-2_drop10] lr=0.005
Run Comparison Model: [25-5_drop10_5-2_drop10] lr=0.01
Run Comparison Model: [25-5_drop10_5-2_drop10] lr=0.01
Run Comparison Model: [25-5_drop10_5-2_drop10] lr=0.02
Run Comparison Model: [25-5_drop10_5-2_drop10] lr=0.02
validation accuracy [%]
training epoch
run 1
run 2
run 3
run 4
run 5
mean
median

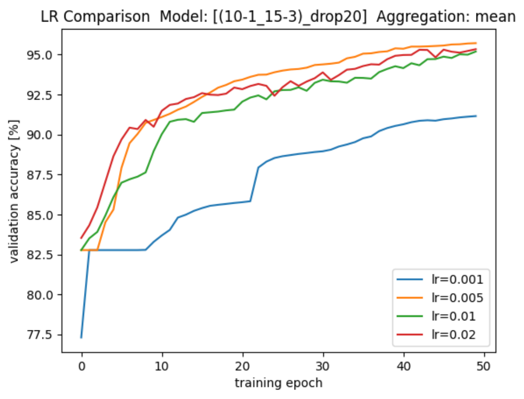
A.3.3 Comparison of the SlicedLSTM Model Learning Rates

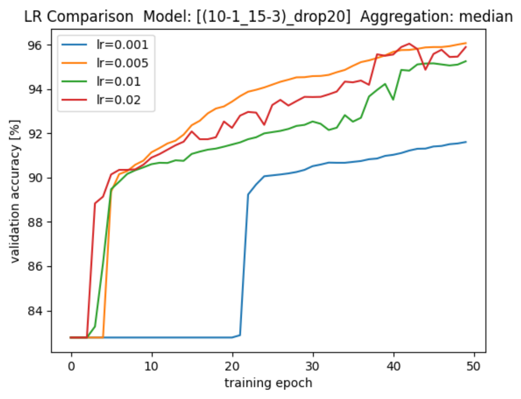
Comparison of the different learning rates for SlicedLSTM model configuration (10-2_15-3)_drop20. Learning rate 0.01 was chosen as the best one.



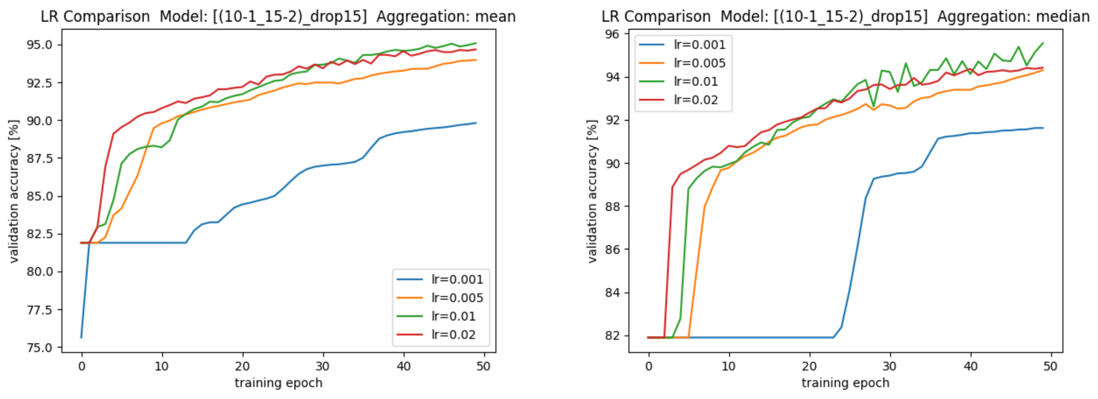


Comparison of the different learning rates for SlicedLSTM model configuration (10-1_15-3)_drop20. Learning rate 0.005 was chosen as the best one.

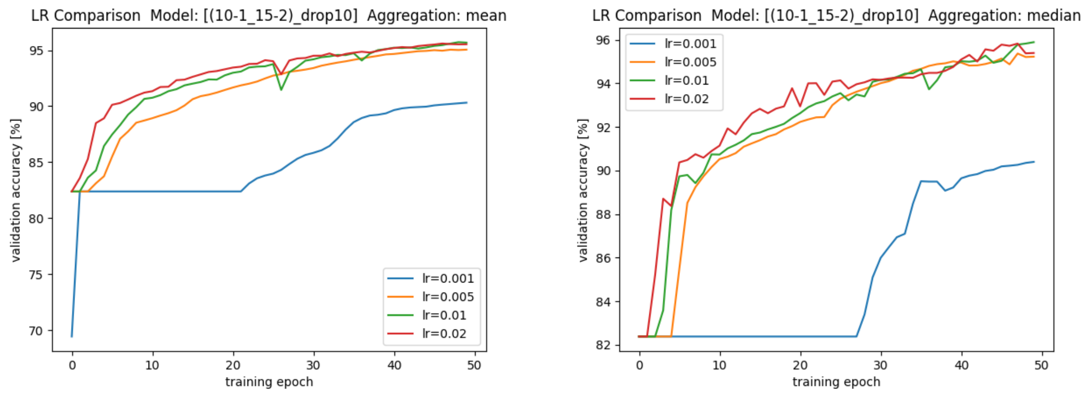




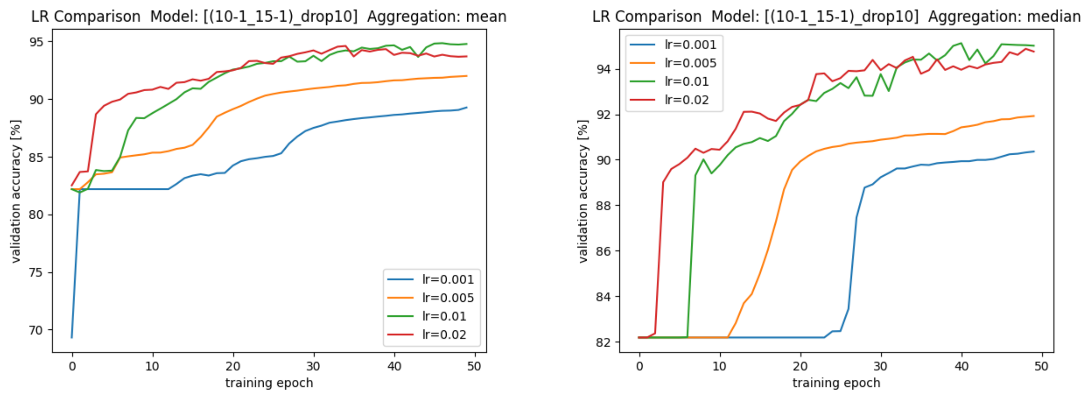
Comparison of the different learning rates for SlicedLSTM model configuration (10-1_15-2)_drop15. Learning rate 0.01 was chosen as the best one.



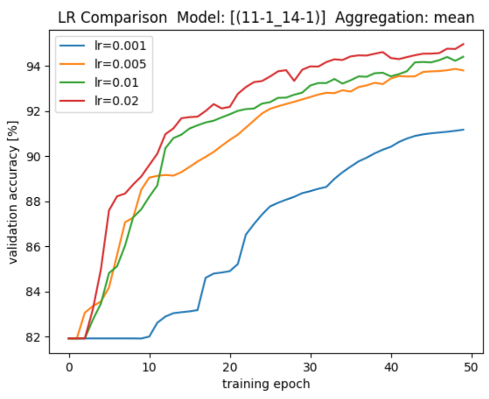
Comparison of the different learning rates for SlicedLSTM model configuration (10-1_15-2)_drop10. Learning rate 0.02 was chosen as the best one.

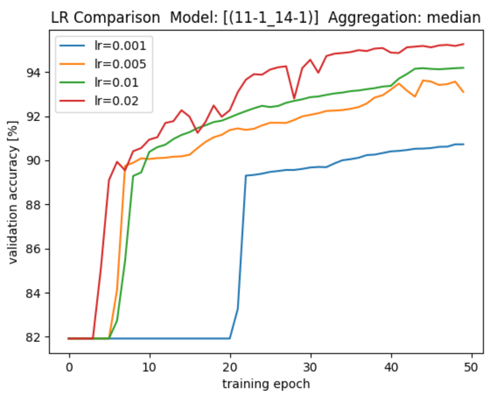


Comparison of the different learning rates for SlicedLSTM model configuration (10-1_15-1)_drop10. Learning rate 0.01 was chosen as the best one.



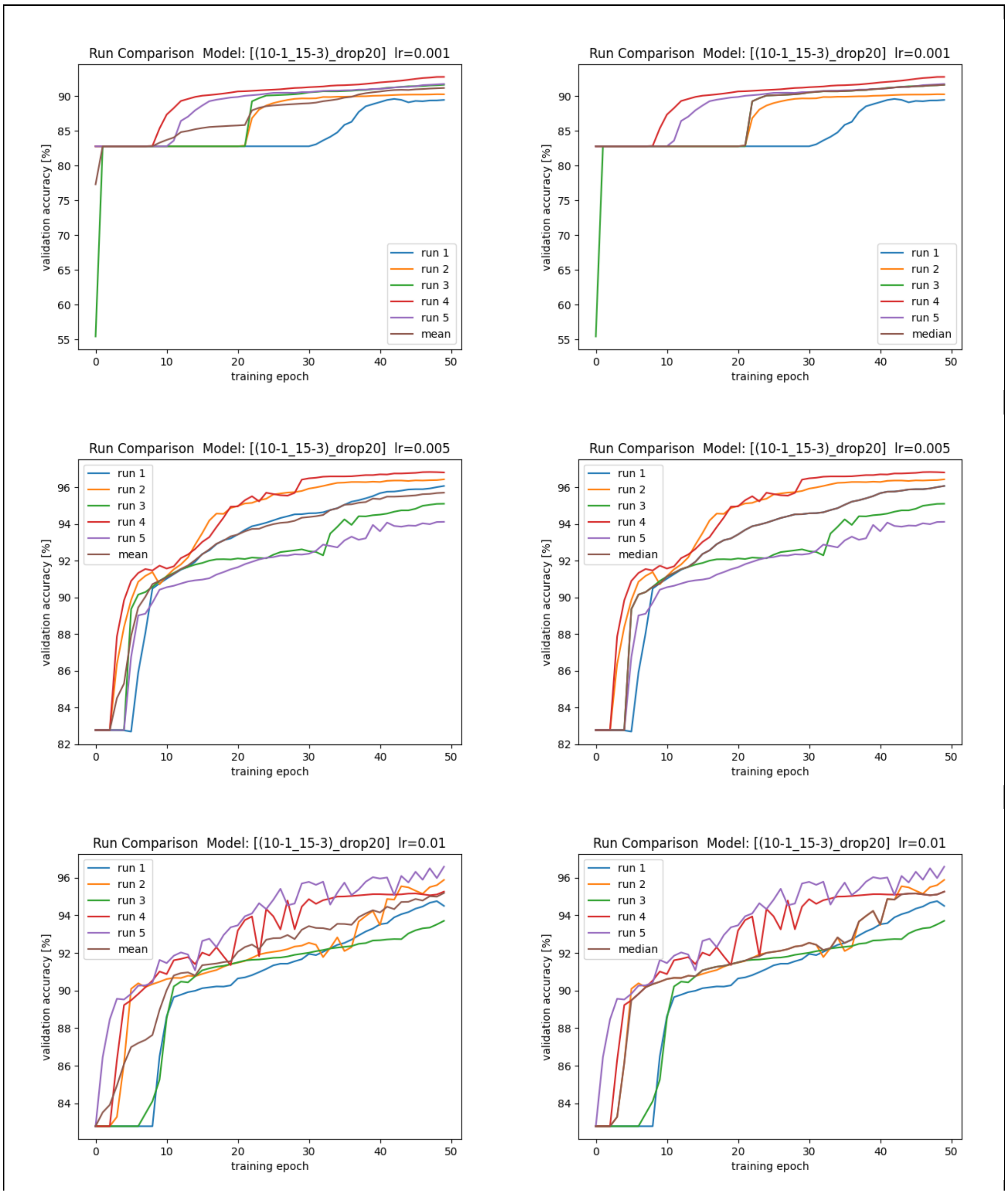
Comparison of the different learning rates for SlicedLSTM model configuration (11-1_14-1). Learning rate 0.02 was chosen as the best one.

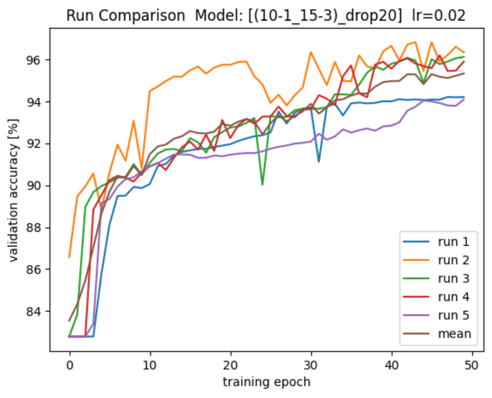


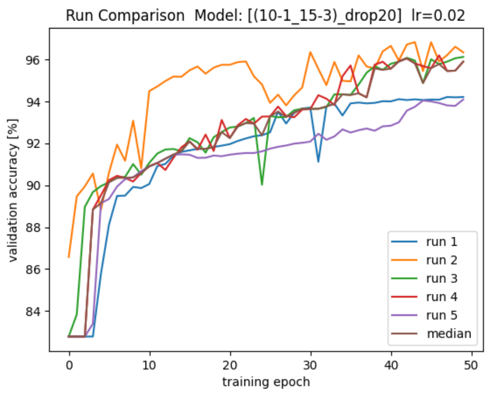


A.3.4 Comparison of the SlicedLSTM Models Runs

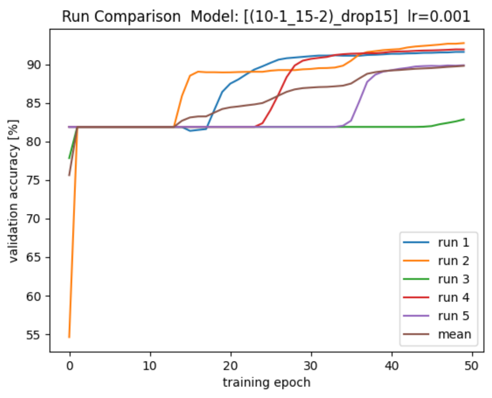
Results for SlicedLSTM model configuration (10-1_15-3)_drop20 with all different learning rates. Learning rate 0.005 was chosen as the best one.

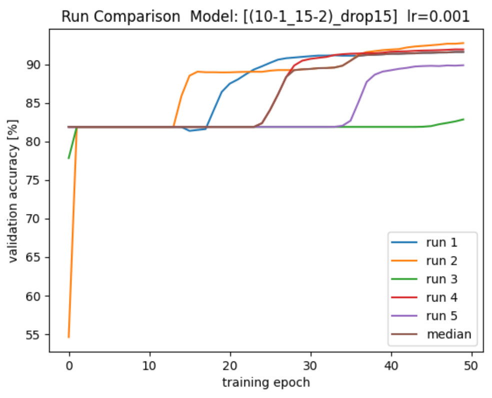


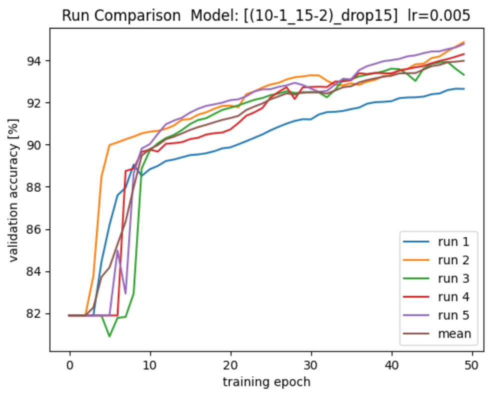


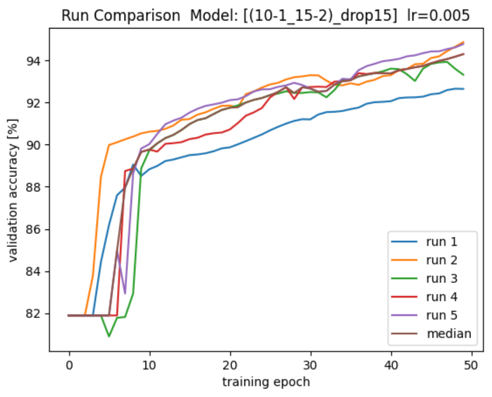


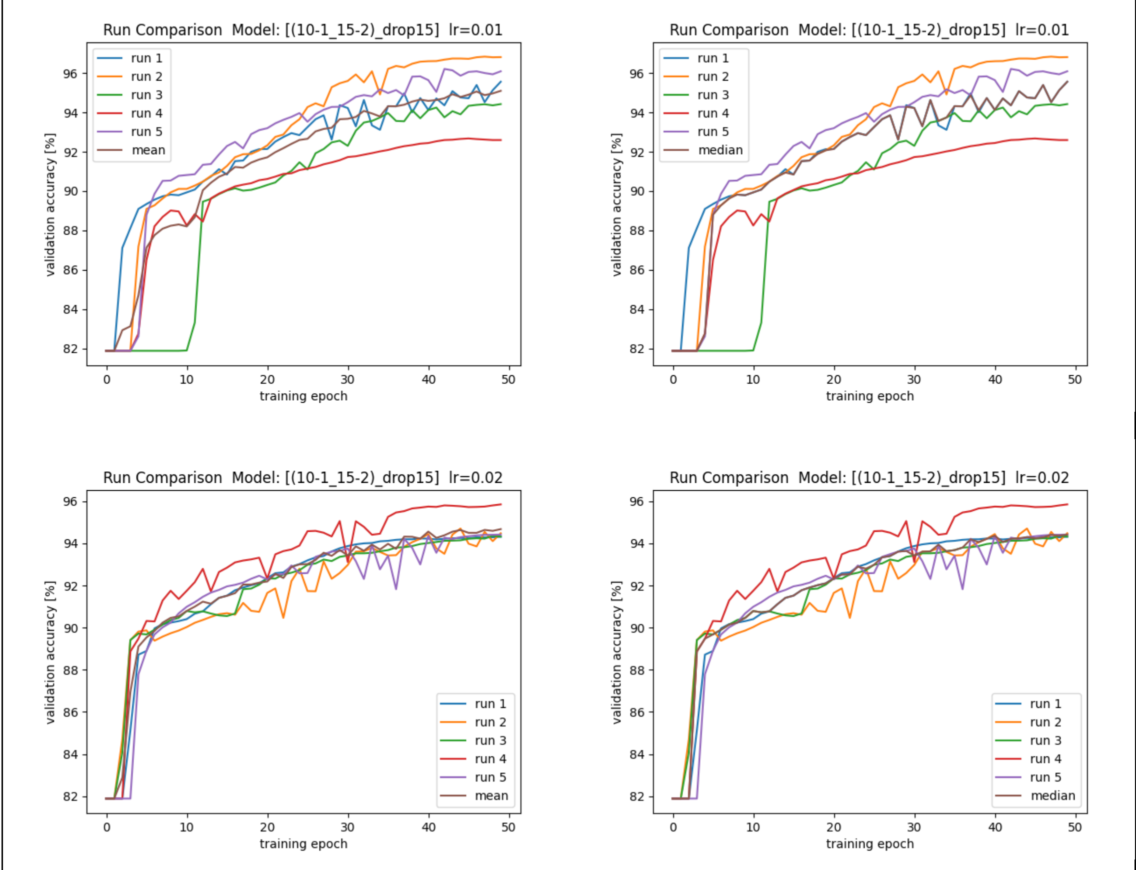
Results for SlicedLSTM model configuration (10-1_15-2)_drop15 with all different learning rates. Learning rate 0.01 was chosen as the best one.



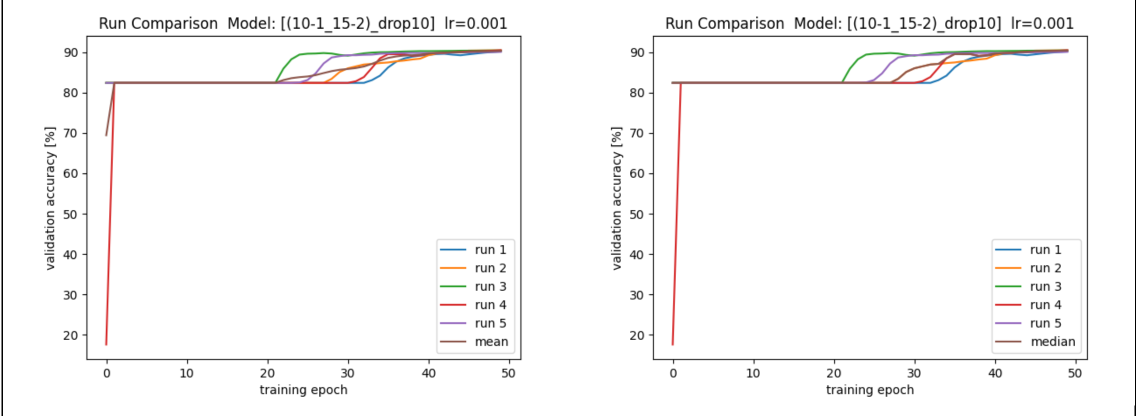


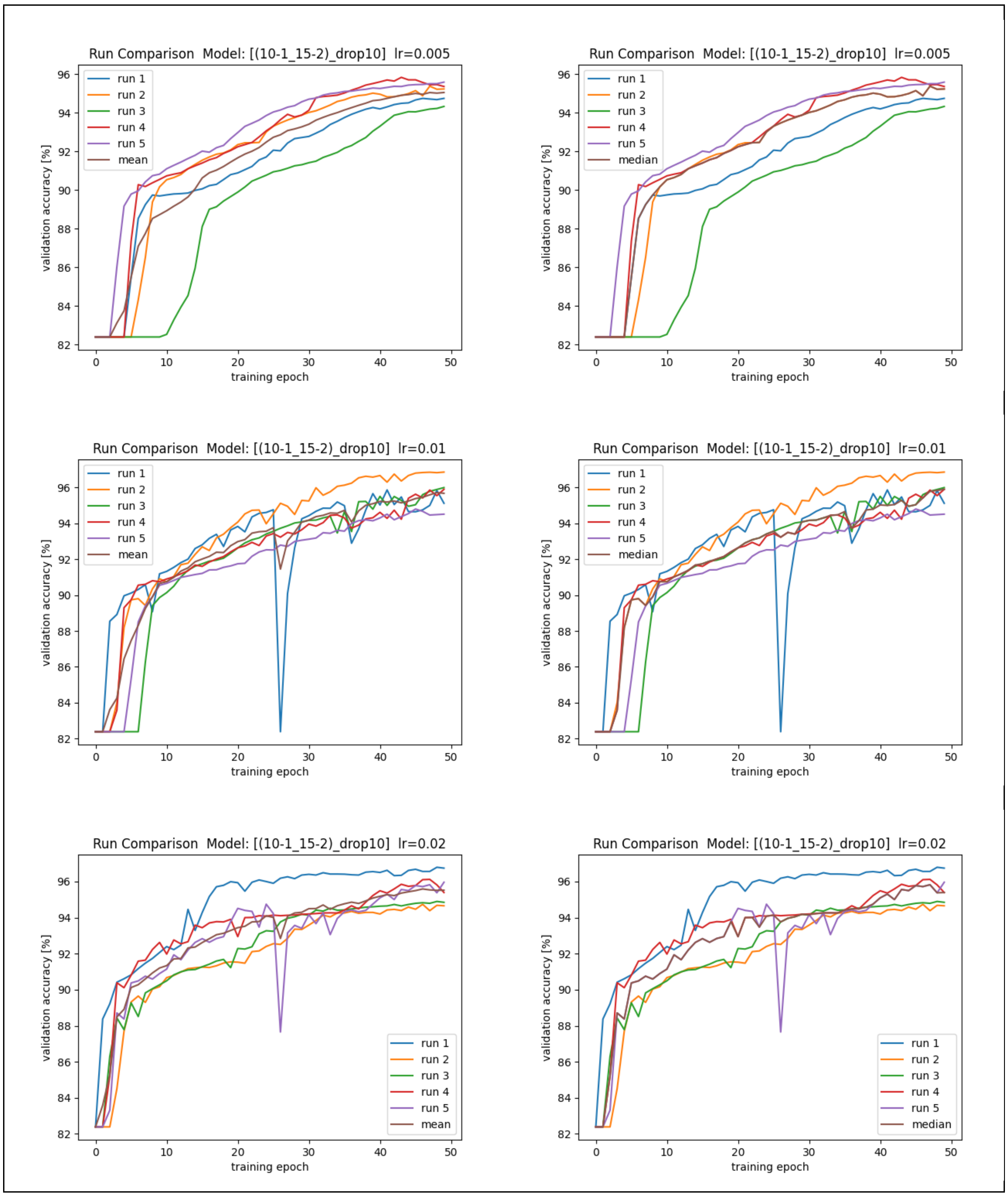




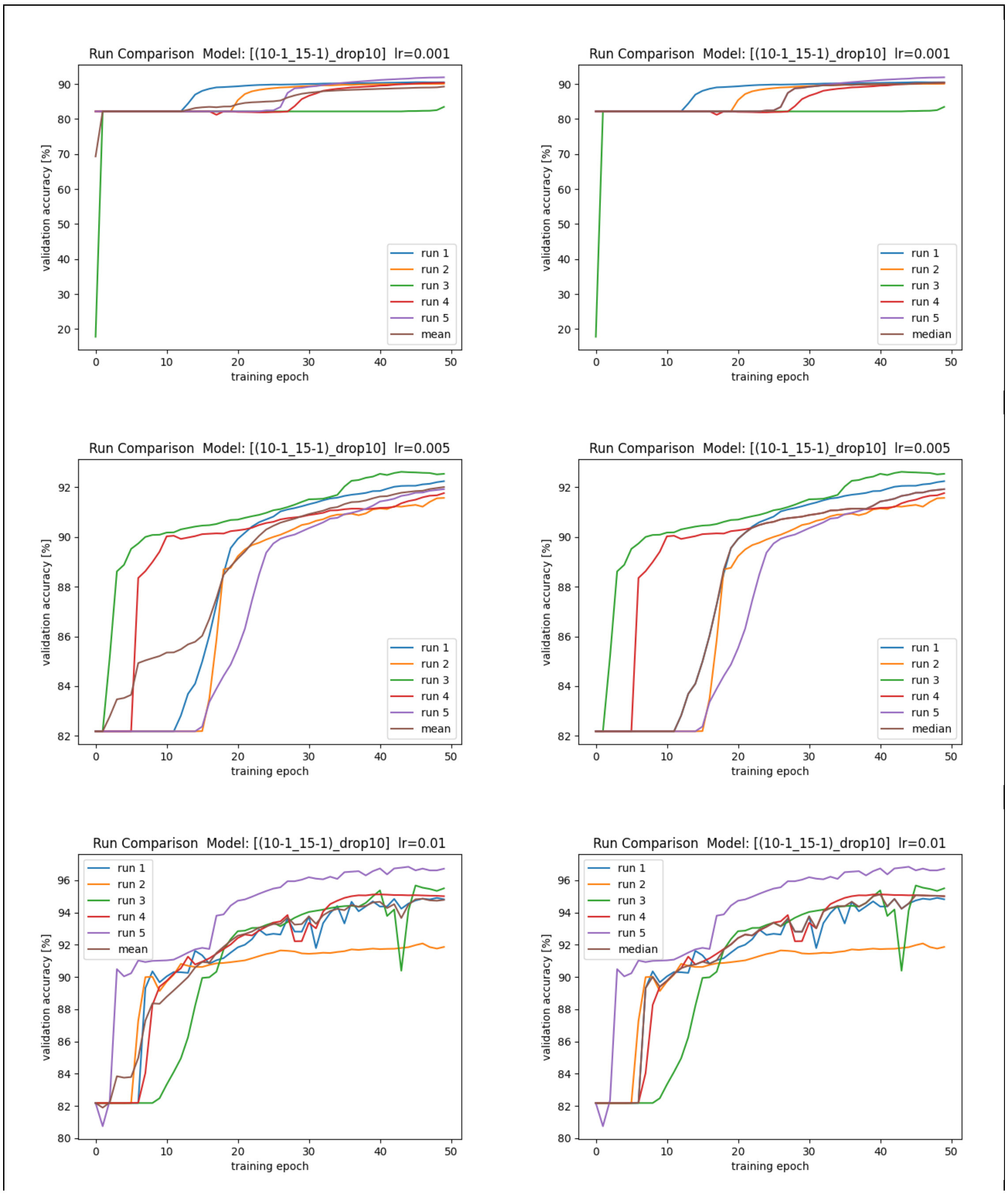


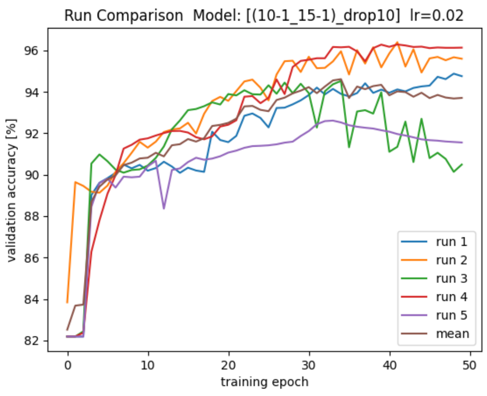
Results for SlicedLSTM model configuration (10-1_15-2)_drop10 with all different learning rates. Learning rate 0.02 was chosen as the best one.

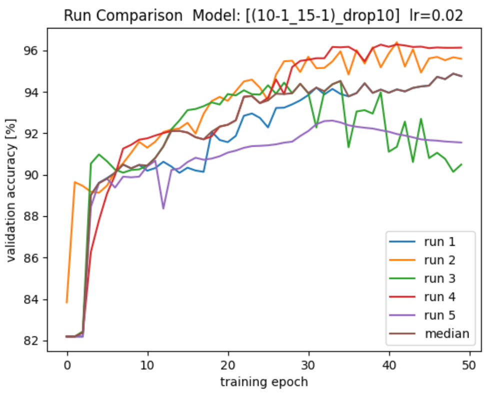




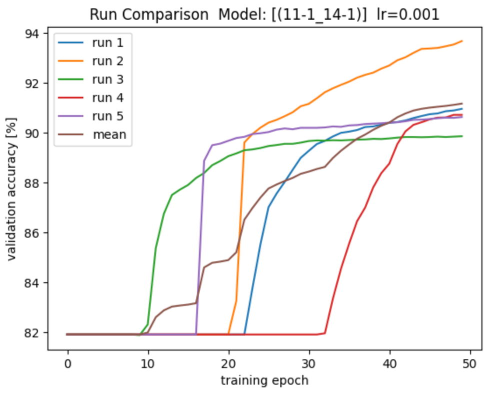
Results for SlicedLSTM model configuration (10-1_15-1)_drop10 with all different learning rates. Learning rate 0.01 was chosen as the best one.

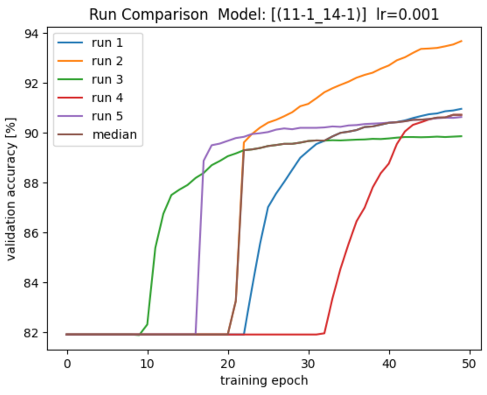

Run Comparison Model: [(10-1_15-1)_drop10] lr=0.001
Run Comparison Model: [(10-1_15-1)_drop10] lr=0.001
Run Comparison Model: [(10-1_15-1)_drop10] lr=0.005
Run Comparison Model: [(10-1_15-1)_drop10] lr=0.005
Run Comparison Model: [(10-1_15-1)_drop10] lr=0.01
Run Comparison Model: [(10-1_15-1)_drop10] lr=0.01
validation accuracy [%]
training epoch
run 1
run 2
run 3
run 4
run 5
mean
median

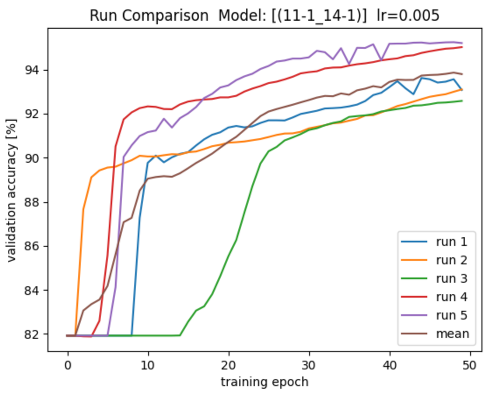


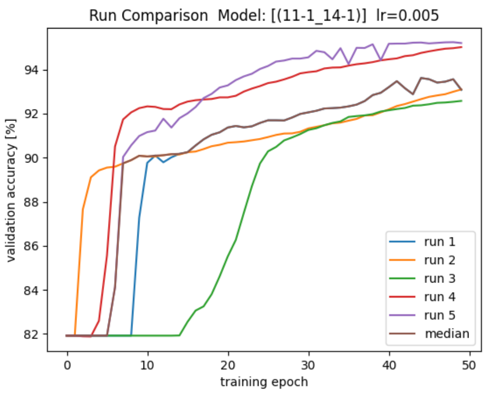


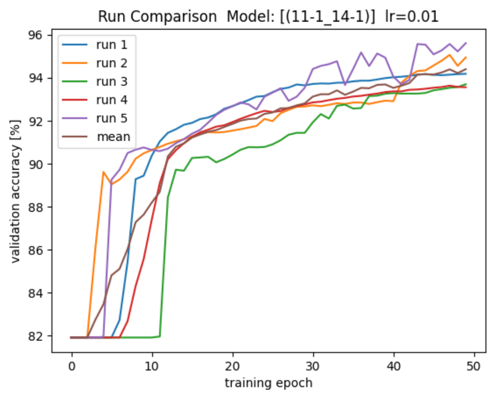
Results for SlicedLSTM model configuration (11-1_14-1) with all different learning rates. Learning rate 0.02 was chosen as the best one.

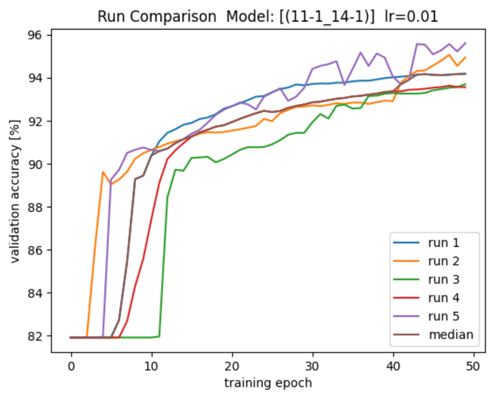


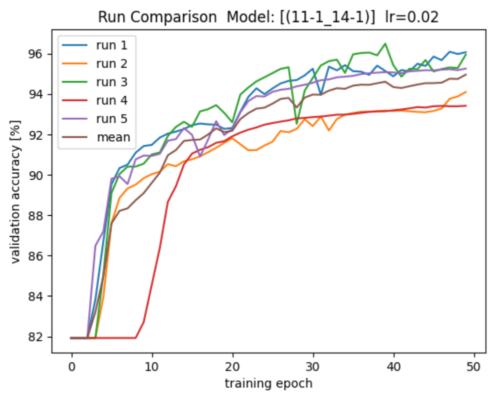


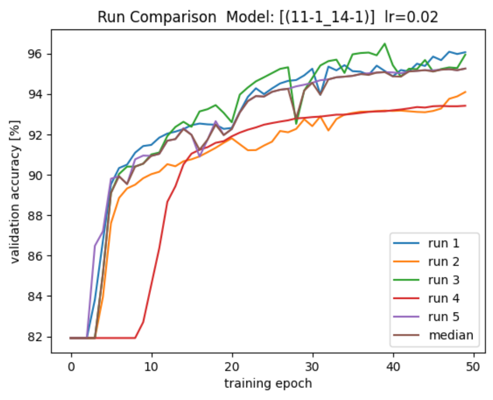





Run Comparison Model: [(11-1_14-1)] lr=0.01
run 1
run 2
run 3
run 4
run 5
mean
validation accuracy [%]
training epoch
82
84
86
88
90
92
94
96
0
10
20
30
40
50


Run Comparison Model: [(11-1_14-1)] lr=0.01
run 1
run 2
run 3
run 4
run 5
median
validation accuracy [%]
training epoch
82
84
86
88
90
92
94
96
0
10
20
30
40
50


Run Comparison Model: [(11-1_14-1)] lr=0.02
run 1
run 2
run 3
run 4
run 5
mean
validation accuracy [%]
training epoch
82
84
86
88
90
92
94
96
0
10
20
30
40
50


Run Comparison Model: [(11-1_14-1)] lr=0.02
run 1
run 2
run 3
run 4
run 5
median
validation accuracy [%]
training epoch
82
84
86
88
90
92
94
96
0
10
20
30
40
50

A.4 Important extracts of the Program Code

A.4.1 SlicedLSTM

The following source code shows the implementation of the SlicedLSTM in python.

```
# Slice LSTM implementation
# inspired by example from https://towardsdatascience.com/building-a-lstm-by-hand-on-pytorch-59c02a4ec091
# The LSTM-Cell uses the optimization of calculating the weights of all four LSTM gates in one big matrix
# (for input and hidden state units respective). Since gates are sliced, this results in a list of matrices.

Import math
import torch
import torch.nn as nn


class SlicedLSTM(nn.Module):
    """ Experimental new version of an LSTM layer, in which data slices of the input are processed by 'slices'
        of the typical LSTM gates (input gate i/g, forward gate f, output gate o) and afterwards connected by a dense
         ConnectorLayer  """

    def __init__(self, lstm_slices: list[tuple]):
        super().__init__()
        # list with tuples: (input units for this slice, hidden units for this slice) for each slice of the layer
        self.slices = lstm_slices
        # amount of slices in this layer
        self.num_slices = len(lstm_slices)
        # hidden size of the entire layer (sum of length of slices)
        self.hidden_size = sum(x[1] for x in self.slices)
        # list of matrices of weights for input unit slices. Concatenation of weight matrix of all four gates
        # (i, f, g, o). One matrix contains weights for one slice of  i, f, g and o.
        self.Ws = nn.ParameterList([nn.Parameter(torch.Tensor(input_size, hidden_size * 4)) for input_size, hidden_size in lstm_slices])
        # list of matrices of weights for hidden unit slices. Concatenation of weight matrix of all four gates
        # (i, f, g, o). One matrix contains weights for one slice of  i, f, g and o.
        self.Us = nn.ParameterList([nn.Parameter(torch.Tensor(hidden_size, hidden_size * 4)) for _, hidden_size in lstm_slices])
        # list of bias vectors for each slice. One vector contains concatenated biases for all four gates i, f, g and o.
        self.biases = nn.ParameterList([nn.Parameter(torch.Tensor(hidden_size * 4)) for _, hidden_size in lstm_slices])
        # weight matrix for the dense connector layer connecting the gate slices. Contains concatenated weights for all
        # four gates i, f, g and o.
        self.connector_Ws = nn.Parameter(torch.Tensor(self.hidden_size, self.hidden_size * 4))
        # bias vector for the dense connector layer connecting the gate slices. Contains concatenated biases for all
        # four gates i, f, g and o.
        self.connector_biases = nn.Parameter(torch.Tensor(self.hidden_size * 4))
        self.init_weights()

    def init_weights(self):
        stdv = 1.0 / math.sqrt(self.hidden_size)
        for weight in self.parameters():
            weight.data.uniform_(-stdv, stdv)

    def forward(self, x.
                init_states=None):

        # assumes x is of shape (batch, sequence, feature) !!!!
        bs, seq_sz, feat_sz = x.size()

        # some input sanity checks
        # print(f"input shape: {x.size()}")
        # print(f"batch size: {batch_size}, sequence_length: {sequence_length}, feature size: {feature_size}")
        total_input_size = sum(x[0] for x in self.slices)
        assert(feat_sz == total_input_size)

        hidden_seq = []
        # non-zero initialization also possible
        if init_states is None:
            init_states = (torch.zeros(bs, self.hidden_size).to(x.device).
                           torch.zeros(bs, self.hidden_size).to(x.device))
        h_t, c_t = init_states

        gate_shape = (x.shape[0], self.hidden_size)

        # iterate over time steps in sequence
        for t in range(seq_sz):
            in_start = 0
            hid_start = 0

            # aggregate lists to store resulting gate tensors for each slice
            slice_is: list[torch.Tensor] = [None] * self.num_slices
            slice_fs: list[torch.Tensor] = [None] * self.num_slices
            slice_gs: list[torch.Tensor] = [None] * self.num_slices
            slice_os: list[torch.Tensor] = [None] * self.num_slices

            # iterate over slices for each timestep
            for index. (input_size, hidden_size) in enumerate(self.slices):
                in_end = in_start + input_size          # end index exclusive in slicing
                hid_end = hid_start + hidden_size       # end index exclusive in slicing

                cur_x_t = x[:, t, in_start:in_end]      # input units for current slice (for entire batch)
                cur_h_t = h_t[:, hid_start:hid_end]     # hidden units for current slice (batch independent)

                # batch the computations for each slice into one single matrix multiplication
                gates = cur_x_t @ self.Ws[index] + cur_h_t @ self.Us[index] + self.biases[index]

                # apply activation functions for each gate slice
                i_t = torch.sigmoid(gates[:. :hidden_size])                     # input gate slice
                f_t = torch.sigmoid(gates[:, hidden_size: hidden_size * 2])     # forget gate slice
                g_t = torch.tanh(gates[:, hidden_size * 2: hidden_size * 3])  # 'gate gate' slice
                o_t = torch.sigmoid(gates[:, hidden_size * 3:])  # output gate slice

                slice_is[index] = i_t
                slice_fs[index] = f_t
                slice_gs[index] = g_t
                slice_os[index] = o_t

                # start index (inclusive) of next slice is previous (exclusive) end index
                in_start = in_end
                hid_start = hid_end

            total_i = torch.cat(slice_is, dim=1)
            total_f = torch.cat(slice_fs, dim=1)
            total_g = torch.cat(slice_gs, dim=1)
            total_o = torch.cat(slice_os, dim=1)

            # apply dense connector-layer
            i_t = torch.sigmoid(total_i @ self.connector_Ws[:. :self.hidden_size]
                                + self.connector_biases[:self.hidden_size])
            f_t = torch.sigmoid(total_f @ self.connector_Ws[:, self.hidden_size:self.hidden_size * 2]
                                + self.connector_biases[self.hidden_size:self.hidden_size * 2])
            g_t = torch.tanh(total_g @ self.connector_Ws[:, self.hidden_size * 2:self.hidden_size * 3]
                             + self.connector_biases[self.hidden_size * 2:self.hidden_size * 3])
            o_t = torch.sigmoid(total_o @ self.connector_Ws[:, self.hidden_size * 3:]
                                + self.connector_biases[self.hidden_size * 3:])

            c_t = f_t * c_t + i_t * g_t
            h_t = o_t * torch.tanh(c_t)

            hidden_seq.append(h_t.unsqueeze(0))

        hidden_seq = torch.cat(hidden_seq, dim=0)
        # reshape from shape (sequence, batch, feature) to (batch, sequence, feature)
        hidden_seq = hidden_seq.transpose(0. 1).contiguous()

        return hidden_seq. (h_t, c_t)
```

A.4.2 Excerpt experimental Study

The following code excerpt shows the data processing pipeline for the experimental study. The implementation was based on the code of and inspired by the two available GitHub projects Predictive-Maintenance-Using-LSTM by by the author Praveena1809 and pytorch-lstm-by-hand by the author Pi Esposito (Praveena1809, 2020) (Esposito, 2020a).

```
    ...
    # STEP 1: READING DATA
    # read training data
    train_df = pd.read_csv(current_train_filepath, sep=" ", header=None)
    # train_df.drop(train_df.columns[[26. 27]], axis=1, inplace=True)
    train_df.columns = ['id'. 'cycle'. 'setting1'. 'setting2'. 'setting3'. 's1'. 's2'. 's3'.
                        's4'. 's5'. 's6'. 's7'. 's8'. 's9'. 's10'. 's11'. 's12'. 's13'. 's14'.
                        's15'. 's16'. 's17'. 's18'. 's19'. 's20'. 's21']

    # read test data
    test_df = pd.read_csv(current_test_filepath, sep=" ", header=None)
    # test_df.drop(test_df.columns[[26. 27]], axis=1, inplace=True)
    test_df.columns = train_df.columns

    # read ground truth data
    truth_df = pd.read_csv(current_groundtruth_filepath, sep=" ", header=None)
    # truth_df.drop(truth_df.columns[[1]], axis=1, inplace=True)


    # STEP 2: DATA LABELING
    # Data Labeling - generate column RUL
    rul = pd.DataFrame(train_df.groupby('id')['cycle'].max()).reset_index()
    rul.columns = ['id'. 'max']
    train_df = train_df.merge(rul, on=['id'], how='left')
    train_df['RUL'] = train_df['max'] - train_df['cycle']
    train_df.drop('max', axis=1, inplace=True)

    # generate label columns for training data
    w1 = 30
    w0 = 15

    # Label1 == 1 indicates a failure will occur within the next 30 cycles (otherwise: 0).
    train_df['label1'] = np.where(train_df['RUL'] <= w1. 1. 0 )

    # label2 is multiclass, value 1 is identical to label1.
    # value 2 indicates failure within 15 cycles
    train_df['label2'] = train_df['label1']
    train_df.loc[train_df['RUL'] <= w0. 'label2'] = 2


    # STEP 3: DATA NORMALIZATION (values within [0.0.1.0])
    # MinMax normalization - train data
    train_df['cycle_norm'] = train_df['cycle']
```

```
cols_normalize = train_df.columns.difference(['id'. 'cycle'. 'RUL'. 'label1'. 'label2'])
min_max_scaler = MinMaxScaler()
norm_train_df = pd.DataFrame(min_max_scaler.fit_transform(train_df[cols_normalize]).
                             columns=cols_normalize.
                             index=train_df.index)
join_df = train_df[train_df.columns.difference(cols_normalize)].join(norm_train_df)
train_df = join_df.reindex(columns=train_df.columns)

# MinMax normalization - test data
test_df['cycle_norm'] = test_df['cycle']
norm_test_df = pd.DataFrame(min_max_scaler.transform(test_df[cols_normalize]).
                            columns=cols_normalize.
                            index=test_df.index)
test_join_df = test_df[test_df.columns.difference(cols_normalize)].join(norm_test_df)
test_df = test_join_df.reindex(columns=test_df.columns)
test_df = test_df.reset_index(drop=True)


# STEP 4: GENERATE CLASS LABELS
# generate column max for test data
rul = pd.DataFrame(test_df.groupby('id')['cycle'].max()).reset_index()
rul.columns = ['id'. 'max']
truth_df.columns = ['more']
truth_df['id'] = truth_df.index + 1
truth_df['max'] = rul['max'] + truth_df['more']
truth_df.drop('more', axis=1, inplace=True)

# generate RUL for test data
test_df = test_df.merge(truth_df, on=['id'], how='left')
test_df['RUL'] = test_df['max'] - test_df['cycle']
test_df.drop('max', axis=1, inplace=True)

# generate label columns w0 and w1 for test data
test_df['label1'] = np.where(test_df['RUL'] <= w1. 1. 0)
test_df['label2'] = test_df['label1']
test_df.loc[test_df['RUL'] <= w0. 'label2'] = 2


# FILE 2: MODEL BUILDING AND TRAINING
# pick a large window size of 50 cycles
sequence_length = 50

# pick the feature columns
sequence_cols = ['setting1'. 'setting2'. 'setting3'. 'cycle_norm']
key_cols = ['id'. 'cycle']
label_cols = ['label1'. 'label2'. 'RUL']

input_features = test_df.columns.values.tolist()
sensor_cols = [x for x in input_features if x not in set(key_cols)]
sensor_cols = [x for x in sensor_cols if x not in set(label_cols)]
sensor_cols = [x for x in sensor_cols if x not in set(sequence_cols)]

# The time is sequenced along
# This may be a silly way to get these column names, but it's relatively clear
sequence_cols.extend(sensor_cols)

# generator for the sequences
seq_gen = (list(gen_sequence(train_df[train_df['id']==id], sequence_length, sequence_cols))
           for id in train_df['id'].unique())

# generate sequences and convert to numpy array
seq_array = np.concatenate(list(seq_gen)).astype(np.float32)

# generate labels and convert to numpy array
label_gen = [gen_labels(train_df[train_df['id']==id], sequence_length. ['label1'])
             for id in train_df['id'].unique()]
label_array = np.concatenate(label_gen).astype(np.float32)
...
```