DEGREE PROJECT IN TECHNOLOGY,
SECOND CYCLE, 30 CREDITS
STOCKHOLM, SWEDEN 2021

Master's Thesis

Development and evaluation of an LSTM-VAE based anomaly detection pipeline for battery time-series measurements

Nabakumar Singh Khongbantabam

KTH ROYAL INSTITUTE OF TECHNOLOGY
ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

Author

Nabakumar Singh Khongbantabam <nskh@kth.se>
EIT ICT Innovation, Autonomous Systems
KTH Royal Institute of Technology

Place for Project

Fortum Power and Heat Oy
Keilalahdentie 2-4
Espoo 02150, Finland

Examiner

Prof. Amir H. Payberah <payberah@kth.se>
Assistant Professor
Division of Software and Computer Systems
KTH Royal Institute of Technology
Kistagången 16
Stockholm 10044 Sweden

Supervisor

Prof. Seif Haridi <haridi@kth.se>
Professor
Division of Software and Computer Systems
KTH Royal Institute of Technology
Kistagången 16
Stockholm 10044 Sweden

Abstract

This thesis presents the development and evaluation of an anomaly detection pipeline to detect possible anomalies during the operation of a fleet of batteries and assist fleet owners in better maintenance. Maintaining their lead-acid batteries well can reduce operational costs by preventing downtime and increasing the service life of the batteries.

The anomaly detection pipeline employs battery sensors that connect to the battery fleet, one sensor per battery, to remotely collect real-time time-series measurements of their operating characteristics, such as voltage, current, and temperature. The time-series anomaly detector was developed using a Variational Autoencoder (VAE) architecture based on Long Short-Term Memory (LSTM) units (LSTM-VAE architecture).

The evaluation of the pipeline indicates that LSTM-VAE architecture was able to reproduce finer details of the input measurement. However, it falls short in reproducing the low-frequency components of the input. The conclusion drawn is that LSTM-VAE architecture in its base form is unable to capture the much longer patterns in the measurement time series. As a result, a substantial change in the design and the development of the LSTM-VAE is necessary to accommodate the nature of the battery time-series measurements. The thesis provides a detailed analysis of the phenomena and identifies future work with several proposed changes to the architecture, design, and training methods.

Keywords

Master's thesis, Degree project, Forklifts, Battery sensors, Data pipeline, Predictive maintenance, Anomaly detection, Battery failure prediction, Time-series, Long short-term memory, Variational autoencoder.

Abstract

Denna avhandling presenterar utvecklingen och utvärderingen av en pipeline för upptäckt av anomalier för att upptäcka eventuella anomalier under driften av en flotta av batterier och hjälpa flottans ägare med bättre underhåll. Att underhålla sina blybatterier väl kan minska driftskostnaderna genom att förhindra stillestånd och öka batteriernas livslängd.

Anomalidetekteringspipelinen använder batterisensorer som ansluter till batteriflottan, en sensor per batteri, för att på distans samla in realtidsmätningar i tidsserier av deras driftsegenskaper, såsom spänning, ström och temperatur. Tidsserieanomalidetektorn utvecklades med en VAE-arkitektur baserad på LSTM-enheter (LSTM-VAE-arkitektur).

Utvärderingen av pipelinen indikerar att LSTM-VAE-arkitekturen kunde reproducera finare detaljer i ingångsmätningen. Det misslyckas dock med att återskapa lågfrekventa komponenter i ingången. Slutsatsen som dras är att LSTM-VAE-arkitekturen i sin basform inte kan fånga de mycket längre mönstren i mättidsserien. Som ett resultat är en väsentlig förändring i designen och utvecklingen av LSTM-VAE nödvändig för att tillgodose typen av batteritidsseriemätningar. Avhandlingen ger en detaljerad analys av fenomenen och identifierar framtida arbete med flera föreslagna förändringar av arkitektur, design och träningsmetoder.

Nyckelord

Examensarbete, Examensarbete, Gaffeltruckar, Batterisensorer, Datapipeline, Prediktivt underhåll, Avvikelsedetektering, Förutsägelse av batterifel, Tidsserier, Långt korttidsminne, Varierande autoencoder.

Acknowledgements

I would like to acknowledge and thank the following important people and organizations, without whose support and contribution, this degree project would not be successful.

Firstly, I would like to thank Fortum Oyj for providing me with this golden opportunity to conduct the degree project as part of eFleetly product development and customer pilot. I would like to also thank the entire eFleetly development team, whose support was crucial in addressing many technical challenges faced during the implementation of this project.

Secondly, I would like to thank (a) Outi Kettunen, Director, Digital Services and (b) Arttu Kautonen, Solutions Product Manager, both Mishubishi Logisnext Europe, and Logisnext Finland Oy, for permitting me to use the data acquired from the operation of one of their largest customers. For confidentiality reasons, no actual battery measurement data used in the project is published as part of this thesis.

Thirdly, I would like to thank Pavel Marek, CEO, and Battery Intelligence Oy team, whose excellent battery sensors and infrastructure were used to collect the battery operational data used in this project.

Fourthly, I would like to thank my examiner and supervisors, (a) Prof. Amir H. Payberah, Assistant Professor, (b) Prof. Seif Haridi, both Division of Software and Computer Systems, KTH, and (c) Prof. Quan Zhou, Electrical Engineering and Automation, Aalto University for staying with me all along during the execution of this degree project.

And finally, I would like to thank my family for providing me with the vital support and motivation to perform and complete the project despite the tough challenges posed by the COVID-19 pandemic during 2020/2021.

Acronyms

AGM Absorbent Glass Mat
VRLA Valve Regulated Lead–Acid
SLA Sealed Lead–Acid
DoD Depth of Discharge
CSV Comma Separated Values
MSE Mean Squared Error
ML Machine Learning
VAE Variational Autoencoder
LSTM Long Short-Term Memory
LSTM-VAE Long Short-Term Memory Variational Autoencoder
GAN Generative Adversarial Network
ELBO Evidence Lower Bound
KL Kullback–Leibler
RNN Recurrent Neural Network
GRU Gated Recurrent Unit
CNN Convolutional Neural Network
Seq2Seq Sequence to Sequence
MNIST Modified National Institute of Standards and Technology database
SGD Stochastic Gradient Decent
OOD Out Of Distribution
SVM Support Vector Machine
ROC Receiver Operating Characteristic

Contents

Chapter 1

Introduction

1.1 Background

Forklifts are essential equipment used in warehouse indoor operations which are often operated electrically using batteries. Different types of batteries exist when it comes to powering electric forklifts in the industry. However, a big portion of the industry currently relies on lead-acid batteries for their operation, while other types, such as lithium-ion batteries, are growing [6].

Lead-acid batteries, unlike other maintenance-free batteries, such as lithium-ion batteries, require significant maintenance to keep their service life intact. They are not very well tolerant of usage abuses, such as overheating or deep discharges.

As a result, the maintenance cost of lead-acid batteries can be a significant part of the customer operation. For example, a fleet of 1000 batteries can easily run in millions of euros in just the cost of batteries. Hence, ensuring that the batteries are well maintained maximizes their service life and becomes a good cost-saving measure for the business.

One key aspect of maintaining lead-acid batteries is identifying maintenance issues early to avoid bigger failures which could lead to disruption in customer operation and shorter battery lives. Predictive maintenance refers to such identification of potential issues before they actually occur.

This project explores and researches the extent to which deep learning can be used to help identify battery issues before they occur. The theoretical basis utilized in this

exploration and research is known as 'Anomaly Detection'.

Anomaly Detection, also known as Outlier Detection, is a machine learning method where a set of events are analyzed to identify events that are abnormal when compared to the rest of the events. It assumes the system operates in a normal circumstance most of the time and further assumes there are rare events that represent defective or outlier events. Anomaly Detection, therefore, identifies those outlier events by their abnormal deviation from the norm.

The data used in this project are collected from a pilot deployment of a digital battery maintenance service known as eFleetly[1] at a warehouse of a customer belonging to the company Logisnext Finland Oy. eFleetly is a battery maintenance service developed by Fortum[2], in collaboration with Logisnext Finland[3], TietoEvry[4] and Bamomas[5].

1.2 Problem

The battery data collected through the pilot consist of multi-variate time-series readings of voltage, current, temperature, and water level from each battery, sampled at a fixed time period. When the batteries operate normally, the said variables change in a consistent manner, depending on the usage of the forklifts. However, if a failure occurred or is impending, the performance of a battery is expected to change in some subtle ways and should be reflected in those variables.

In this multi-variate time-series data, an anomaly detector has the potential to identify the aforementioned subtle changes in operating patterns. The anomaly detector in the project is implemented using a deep learning-based model described in Chapter 4.

1.3 Purpose

The purpose of the degree project was to develop and evaluate an anomaly detection pipeline utilizing the time-series measurements from a customer's battery fleet. The anomaly detection pipeline was implemented using LSTM-VAE neural network.

[1] https://efleetly.com
[2] https://fortum.com
[3] https://rocla.com
[4] https://tietoevry.com
[5] https://bamomas.com

The purpose of this thesis is to present the development and analysis of the said anomaly detection pipeline so that someone knowledgeable in the area can implement a similar pipeline and further the research in predictive maintenance of battery fleets.

Hence, the thesis documents the development process of the battery monitoring and data collection pipeline, analysis of time-series data as a viable input for anomaly detector, the design and development of a deep-learning-based anomaly detector based on LSTM-VAE architecture, and the analysis of the results from the network.

1.4 Objectives

The objectives of the degree project are as follows:

1. Initiate a pilot at the customer site to collect battery data.
2. Setup battery sensors and data collection network at a customer site, and transfer the raw data (voltage, current, temperature, and water level) to eFleetly cloud.
3. Collect baseline data for a period of three months.
4. Deploy eFleetly battery maintenance tools (charging room monitors and maintenance user interface)
5. Collect post-deployment data.
6. Design and develop a deep learning anomaly detector.
7. Analyse the performance of the anomaly detector from the study of training performance, input reproduction and reproduction error.

1.5 Benefits, Ethics and Sustainability

The forklift fleet owners or rental service providers benefit the most from this project. By extending the lives of batteries, they reduce the cost of operation, thereby increasing their competitiveness and profits.

In the industry, forklift fleet owners are often served by maintenance service providers. Hence, forklift maintenance service providers can also benefit from this project by

improving their service competitiveness.

By improving the lives of operating batteries, this project reduces the number of batteries used over a period of time. Thereby, reducing the environmental impact caused by the customer's operations.

1.6 Methodology

The project follows a mix of quantitative experimental methods and qualitative methods to answer the research questions specified in the section 1.2.

Since the battery measurement data can be monitored using sensors and the historical maintenance activities can be observed from the ground records, the qualitative method is selected as a method to detect possible anomaly phenomena. Analysing the reconstruction errors are in particular used as a method to qualitatively analyse the performance of the anomaly detection pipeline.

One of the key assumptions is that, under normal operation, all batteries generate the battery measurement data from an underlying set of functional battery models. An imminent failure of a battery would, therefore, be an exception to such a set of functional models and would generate a different pattern of measurement data. An anomaly detector is used to identify these abnormal patterns and flag them.

Anomaly detectors identify possible near-future failures. However, they do not identify the type of failures they are. As a result, their usefulness for a forklifts operator is not apparent at a first glance. Hence, to identify and establish the usefulness of the results, a qualitative analysis is performed by interviewing the customer employees.

Additionally, the project was planned as a key feature of a predictive maintenance product. As a result, it was executed as a real customer pilot to deploy battery sensors and collect the data from the real use of the batteries. The engineering method was applied to execute this project.

Chapter 3 describes the methodologies in full detail.

1.7 Stakeholders

Stakeholders of the project comprise battery fleet owners and rental operators, businesses providing battery maintenance services, and businesses that develop solutions for battery maintenance.

The aforementioned businesses which own or operate rental fleets are in general interested in how anomaly detection could be incorporated into their overall maintenance services and operation. Businesses which develop solutions for battery maintenance or their technical team would be in general interested in the implementation of the anomaly detection utilized in this project.

1.8 Delimitations

The training and evaluation data in this project were collected over only a single year and may not represent the full distribution of the battery measurements. Ideally, the anomaly detection system should be trained with more operational data.

The pilot with the customer consisted of three different phases, which may affect the patterns involved in the data collected during different phases. The first phase was a period of their old operational routines. The second phase was a period where eFleetly tools were deployed, which affect how the batteries are used. The second phase data may reflect a mixture of the first and the third stages. The third stage is when the eFleetly tools have been fully deployed and the usage patterns of the batteries have, more or less, improved.

This thesis does not distinguish between the three stages of the pilot and their corresponding data collection. It assumes that all three phases represent the normal behavior of customer's operation.

Furthermore, the said data were collected during the global coronavirus pandemic year (2020). This may restrict the customer's operating patterns to only a certain subset of its normal operating patterns.

Anomaly detection and matching the anomalies to their corresponding real-life maintenance events is a bit of a subjective matter, considering it is based on human perception of time and association, in particular the author's perception in this matter. Hence, the interpretation of the results should take this shortfall into account.

1.9 Outline

Chapter 2 ”Background” describes the technical functioning of lead-acid batteries, how their lifetime are affected by various factors, what are various failure modes, how anomaly detection can be utilized to predict the failures and highlights of various past researches performed around this area, including existing time-series anomaly detection models and methods.

Chapter 3 ”Methods” describes the method used to acquire the battery data. The chapter describes the format of anomaly detection suitable for this purpose. It further describes the method of analysis of the results.

Chapter 4 ”Work” describes the data pipeline development and the architecture design of the Long Short-Term Memory Variational Autoencoder (LSTM-VAE) anomaly detector, mathematical description of the model, its development, training, and evaluation.

Chapter 5 ”Results” analyses the functioning of the pipeline and functioning of anomaly detection from reproduction errors.

Finally, Chapter 6 ”Conclusions” draws some key conclusions from the results of this project and identifies future work.

1.10 Examples

The table 1.10 illustrates the battery measurement data acquired during the pilot. The data are organized into different sampling locations inside the warehouse. However, for the purpose of this project, they are not distinguished.

Table 1.10.1: Sample battery measurements.

Battery ID	Time	Voltage	Current	Temp.	Water level
73	1601499600	27.12	0.0	29.87	1
53	1601499600	88.35	-0.1	27.00	1
85	1601499600	24.18	-55.1	27.37	0
26	1601499600	77.72	-86.4	30.12	1
110	1601499600	25.45	-0.9	23.12	0

Data acquisition environment for the batteries

Operating environment

Warehouse, indoor

Equipment: Forklifts (small, medium and large)

Sensors

Number of battery sensors: 100

Number of charging rooms: 3

Number of batteries per forklift: about 2

Connectivity: Dedicated 4G routers and switches

Chapter 2

Background

This chapter presents a detailed description of the background of predictive maintenance for batteries. It further presents related work in the development of time-series anomaly detection systems. It also discusses what is found useful from the prior research and what is less useful, and explains how they are utilized as supporting research for this project.

2.1 Lead-acid battery

Gaston Planté invented the first practical lead-acid battery in 1860 [14]. Lead-acid batteries, despite being a 150 years old technology, it is still commonly used in many industries because of their low cost and high power-to-weight ratio. The forklifts industry is one such industry. Although slowly transitioning to more modern alternatives, such as lithium-ion batteries, the industry still uses lead-acid batteries in a large portion.

2.1.1 Design and construction

A lead-acid battery is composed of multiple independent battery cells connected either in series, parallel or mixed configuration. Each cell provides roughly 2V across its terminals, while the energy capacity or draw current depends on the cell's design. For practical applications, multiple cells are combined in order to achieve the desired voltage or draw current requirements. For example, a 12V battery would often consist of 6 cells connected in series.

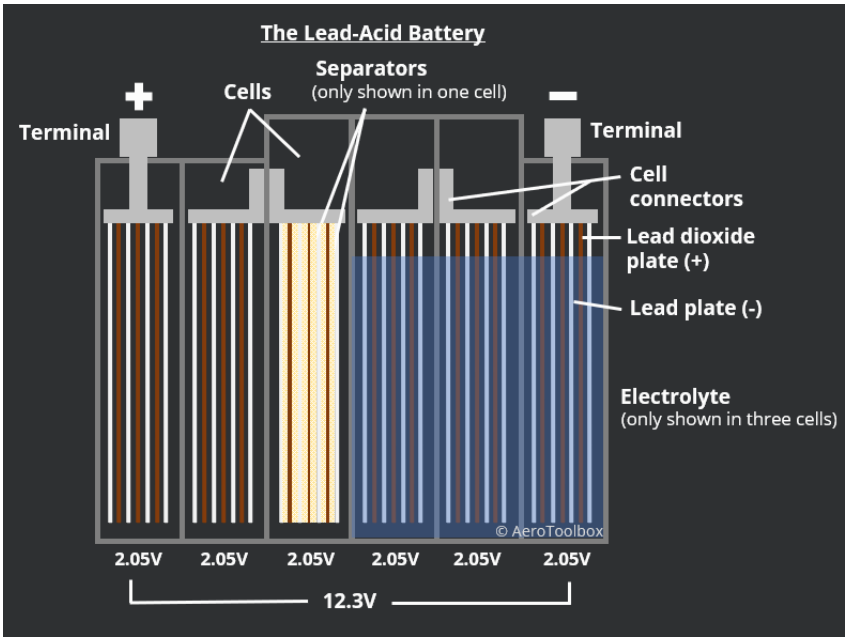


Figure 2.1.1: Lead-acid battery construction

Source: AeroToolbox.com

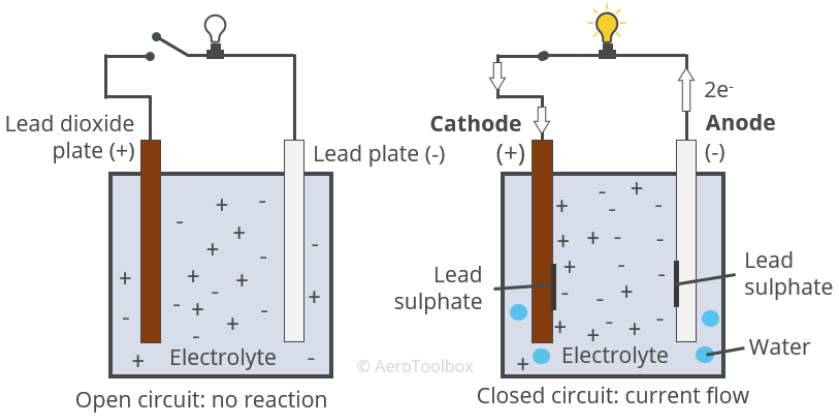
Figure 2.1.1 shows the inside of a lead-acid battery. Each cell is physically isolated and contains positive plates, negative plates, separators, and electrolytes. During a fully discharged state or construction stage (before first charging), the positive plates are made of lead dioxide (PbO_2), but often mixed with materials known as expanders. The negative plates are made of lead sheets. The separators, made of inert materials, are held between the positive plates and negative plates to provide structural support and electrical isolation. The electrolyte comprises diluted sulphuric acid (H_2SO_4) and floods the inside of the cell chamber. Both the positive plates and negative plates are therefore immersed in the electrolyte.

Lead-acid battery goes through a cycle of life when it goes from fully charged state to about 80% discharged state and back to fully charged state. The service life of a battery is often specified in terms of the number of such charge-discharge cycles when operated under an ideal operating condition. For example, manufacturers often specify a typical forklift battery to constitute a lifetime of about 1500 cycles, when operated at a room temperature of 25 deg C.

2.1.2 Charge-discharge cycle

The positive plates, the negative plates, and the electrolyte undergo a significant chemical change during each charge-discharge cycle. When ideally fully discharged, both the positive plates and negative plates become lead sulfate ($PbSO_4$) while the electrolyte is heavily diluted sulphuric acid (H_2SO_4). In practice, batteries are not allowed to discharge to such a state because heavy $PbSO_4$ on the electrodes impede electrical conductivity, which often results in failure to convert back all $PbSO_4$ to PbO_2 when fully charged (hence losing battery capacity).

The energy in the battery is stored during charging by splitting the water molecule (H_2O) into hydrogen ions ($2H^+$) and oxygen ions (O^{2-}) [27]. The hydrogen ions ($2H^+$), also known as hydrated protons, become part of the electrolyte (sulphuric acid, H_2SO_4), and the oxygen ions (O^{2-}) become part of the lead dioxide on the positive plates. The stored energy is released during discharge when those ions combine back into water molecules [27].



The r

Figure 2.1.2: Reaction inside a lead-acid battery during charging and discharging

Source: AeroToolbox.com

During discharge, the chemical reaction produces $PbSO_4$ (on both plates) water. The reaction on the negative plate follows:

$$Pb(s) + HSO_4^-(aq) \rightarrow PbSO_4(s) + H^+(aq) + 2e^-$$

and on the positive plate:

$$PbO_2(s) + HSO_4^-(aq) + 3H^+(aq) + 2e^- \rightarrow PbSO_4(s) + 2H_2O(l)$$

The reaction above produces net electrical energy with a potential difference of 2.05V [27].

The reaction is reversed during charging, which produces back the lead dioxide (PbO_2) on the positive plate, lead (Pb) on the negative plate, and sulphuric acid (H_2SO_4) in the electrolyte. These reactions are illustrated in the figure 2.1.2.

2.2 Battery failure modes

Lead-acid batteries are sensitive to maintenance and operational conditions. Poor maintenance and abuses during use can lead to a shorter life. Listed below are some of the major degradation that occurs in a battery.

2.2.1 Capacity degradation

Capacity degradation is the most common and obvious failure mode that batteries encounter during their daily operation. The main cause of capacity degradation in a lead-acid battery is due to reduction in the performance of the positive plates. Repeated cycling of the battery introduces crystallization of the active materials on the positive plates, which introduces morphological changes on the surface of the plates, ultimately reducing the mechanical integrity of the plates over time. This reduced mechanical integrity affects the performance of the active material (PbO_2) on the plates causing softening or shedding [5].

Furthermore, a process known as sulfation reduces the electrical performance of the plates, further affecting the capacity of the lead-acid battery. During discharge of a battery, the active material lead oxide (PbO_2) converts to discharge material lead sulfate ($PbSO_4$). Ideally, this process is fully reversed during the charging of the battery. However, not all discharge material necessarily convert back to PbO_2, leaving sulfate deposits on the plates which are no longer reversible. The deposits reduce the electrical conductivity of the plates, resulting in reduced electrical performance and reduced capacity [5].

Sulphuric acid is either consumed during the discharge process and produced during the charging process. This may cause a density difference of sulphuric acid in the battery. As a result, higher density sulphuric acid can settle on the bottom, while lighter

density sulphuric acid stays on top. Such a difference in density can cause the plates to degrade differently along the height of the battery [5].

2.2.2 Degradation of negative plates

Negative plates have 'expander' materials added to them to improve their performance and life [3]. The expanders mainly consist of a mixture of barium sulfate, lignosulfonate, and carbon black added to the paste used to create the negative plates. The presence of expanders provides structural stability and prevents largely localized crystallization of lead on the surface of the negative plate during charging and discharging processes. Without the expanders, the leads crystallize with reduced surface area, which produces reduced electrically active surface, and consequently reduces electrical conductivity [5].

The expanders degrade over time as the plates are subject to strong electrolytes during charging and discharging cycles. This effect is accelerated at higher temperatures. At high temperatures, the expanders degrade much faster. For instance, at or above 60 deg C, almost all expanders degrade. As a result, batteries operating at such extreme temperatures would lose cycle life significantly [23].

2.2.3 Degradation of separators

The separators between the positive plates and negative plates can also degrade over time and usage. The separators provide electrical insulation between positive and negative plates and provide structural support for holding the plates together. Degradation to the separators can occur due to mechanical stress caused by movements of the plates, as well as from the lead crystals growing on the plates. High temperatures can also create stress that can break down the separator materials [5].

2.3 Battery types

Depending on the plate separator design and the electrolyte used, the following types of lead-acid batteries currently exist:

2.3.1 Flooded lead-acid battery

Traditional lead-acid batteries have diluted sulfuric acid as electrolytes. This entails needing to keep all the battery cells ventilated and upright. The liquid electrolyte also requires constant refilling with water to maintain the electrolyte level. The biggest advantage of flooded batteries is that they last very long when compared to sealed batteries.

2.3.2 AGM lead-acid battery

Absorbent Glass Mat (AGM) batteries use glass fiber matt as a separator between the electrodes. The electrolyte is just soaked in the matt rather than flooding the cell's inside. Since there is no free-flowing electrolyte. AGM lead-acid batteries are also sealed, hence they are also Valve Regulated Lead–Acid (VRLA) batteries. In addition to low maintenance advantage, AGM batteries have extremely high surge current. Hence, they are often used as automobile starter batteries.

2.3.3 Gel lead-acid battery

Instead of liquid electrolytes, gel batteries have electrolytes in the form of a gel. This greatly reduces maintenance needs, and the batteries can freely orient in any direction during operation. Gel batteries are VRLA batteries since they are easily sealed. Gel batteries have similar low-maintenance advantages to AGM batteries, however, they have lower surge current owning to slower electro-chemical process.

2.3.4 VRLA battery

Both gel battery and AGM battery are VRLA design, allowing the cells to be fully sealed – only openings are through regulated pressure valves. As a result, the VRLA battery requires minimal maintenance and can operate in any orientation. They are also known as Sealed Lead–Acid (SLA) batteries. However, because of sealed operation, VRLA batteries have shorter life compared to traditional flooded lead-acid batteries.

2.4 Battery abuses and degradation

As explained in previous sections, the components inside a lead-acid battery can take degradation from various usage conditions. The following list describes various factors leading to accelerated degradation and leads to poor performance of the battery.

2.4.1 Overheating

Overheating the battery during operation can be quite detrimental to the health of the battery. While batteries degrade over their lifetime normally, overheating can cause to accelerate the degradation, especially the degradation of the plates and the expanders. For example, high-temperature operation increases gassing in the chemical process, which carries away active materials off the plates [1]. High temperature also increases the reaction rate, causing faster re-crystallization [1], which leads to the formation of larger crystals. Larger crystallization results in reduced surface area of the plates for electrical conductivity, thereby reducing the battery capacity and its usable life.

2.4.2 Deep-discharge

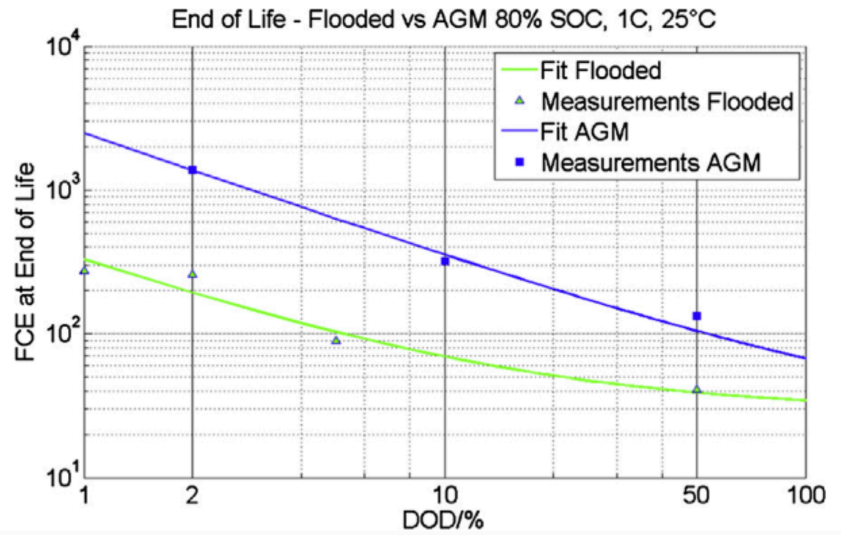


Figure 2.4.1: Impact of dept-of-discharge on the battery life-time.

Source: J. Badeda, J. Kabzinski, D. Schulte, H. Budde-Meiwes, J. Kowal, D.U. Sauer, in: Advanced Battery Power - Kraftwerk Batterie, Aachen,Germany, February 2013.

As described in the subsection2.1.2, when a lead-acid battery is discharged, both the plates start depositing lead sulfate ($PbSO_4$). However, if discharged too much, the $PbSO_4$ can clog up the plates, reducing electrical conductivity. As a result, when the

battery is charged back, some $PbSO_4$ will fail to convert back into active materials. This process is known as sulfation and reduces the available life of the battery before its capacity reduces to an unusable level.

Depth-of-discharge (DoD) is defined as the % of capacity discharged from a full charge. 100% Depth of Discharge (DoD) would mean the battery is discharged fully. Figure 2.4.1 graphically shows the lifetime cycles reduction when DoD is increased. At very high DoD, the cycles life can be an order of magnitude lower [1].

To avoid such loss in battery life, it is often advised to discharge the battery, not more than 80% DoD.

2.4.3 Low electrolyte level

For flooded lead-acid batteries, maintaining the electrolyte level is also crucial. The electrolyte is a key part of cell chemistry. If the level is not high enough, it would create non-uniform reactions on the plates, causing them to degrade non-uniformly. The reduced surface area available for the reaction also reduces the electrical performance and causes reduced capacity and power.

2.5 Related work: Deep neural networks

This degree project utilizes some key deep neural networks in the development of the anomaly detection system. This section introduces those neural networks in sufficient detail which are then utilized in the composition of various anomaly detection architectures in the next section. The following subsections introduce the deep neural network architectures known as Autoencoders, Variational Autoencoders (VAE), and Long Short-Term Memory (LSTM) cells.

2.5.1 Autoencoder

Autoencoders are part of a machine learning class known as ”unsupervised learning”. Unsupervised learning utilizes only the input data to train and learn the model that represents the training data. The learned model is subsequently used to infer if a given new data is part of the learned model. If not, it is typically interpreted as an anomaly.

More precisely, an autoencoder neural network is trained to encode the measurement data into smaller-dimension, meaningful, latent variable values. The figure 2.5.1 shows a general architecture of an autoencoder. The network essentially learns to map higher-dimensional space into a smaller dimensional space.

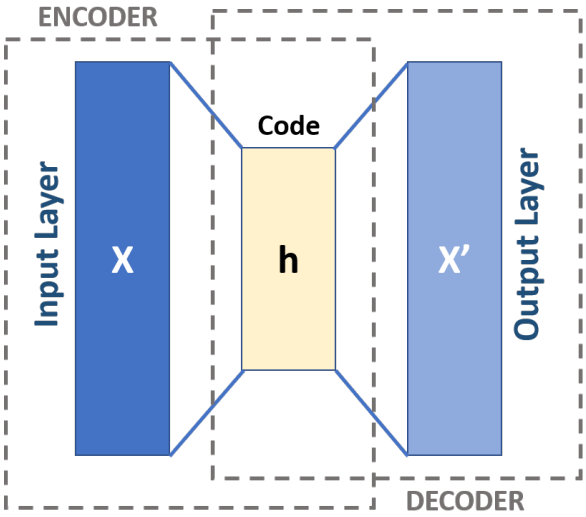


Figure 2.5.1: An illustration of Autoencoder architecture [19]

Mathematically, autoencoders implement the following transformation:

$$\begin{aligned} h &= E(X) \\ X' &= D(h) \end{aligned} \tag{2.1}$$

Where,
X is the training data,
h is (hidden) latent variable - usually smaller in dimension than X,
$E()$ is the encoding function,
$D()$ is the decoding function,
X' is the reconstructed data

Autoencoders are trained to minimize the reconstruction error between X' and the original data X:

$$\underset{D,E}{min}||X' - X||$$

Sakurada and Yairi (2014) demonstrate that autoencoders are able to detect subtle anomalies in which linear PCA fails [26]. The paper further demonstrates that denoising autoencoders can increase accuracy. This conclusion likely comes from the notion that denoising autoencoders would prevent encoding uncorrelated data, which anomalies often represent.

Deep denoising autoencoders require clean training data without outliers and noise. However, that may not be practical in real-world problems. Zhao and Paffenroth (2017) propose a noble extension that improves the quality of anomaly detection by a denoising autoencoder without access to any clean training data [29]. The author names the network ”Robust Deep Autoencoder” (RDA).

2.5.2 Variational Autoencoder (VAE)

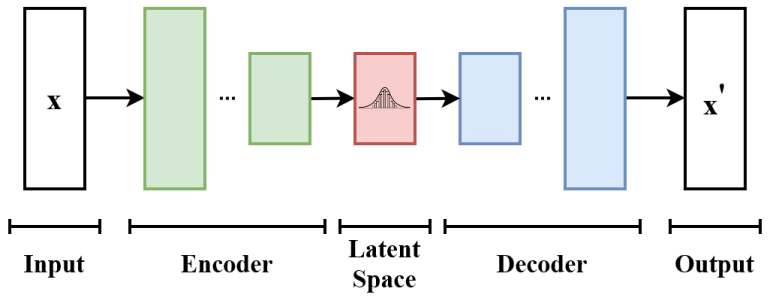


Figure 2.5.2: A simple illustration of Variational Autoencoder architecture [7]. The encoder output is a latent distribution represented by a normal distribution specified by a mean and a variance pair.

VAE is illustrated in the figure 2.5.2. Similar to a regular autoencoder, VAE also assumes there is a hidden and significantly lower dimension latent variable whose distribution maps closely to the distribution of the input space. However, it does so by learning to estimate the joint probability distribution, $p_\theta(x, z)$, of the input x and latent variable z [11]. The input variable x denotes all the input data instances, the hidden latent variable z denotes all the latent variable instances in the latent space, and θ denotes the learned parameters of the VAE. Both input space and latent space are treated as probability distributions and assumed to be continuous, which is how a VAE differs from a regular autoencoder.

Often the underlying distribution of the input data, x, is rather complex. Therefore, like in a regular autoencoder, the encoder in a VAE also learns to map this complex

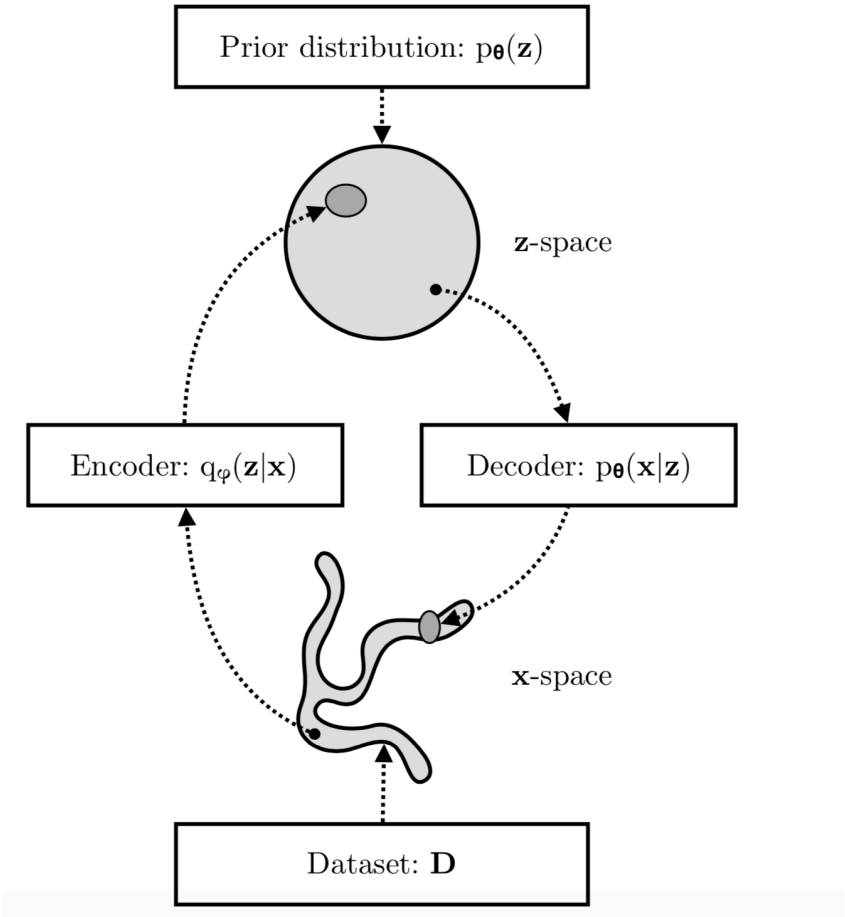


Figure 2.5.3: Variational Autoencoder process, [11]

distribution to a simpler and smaller-dimensional latent space. However, the encoder in a VAE evaluates the conditional probability distribution of z given x, i.e. $p_\theta(z|x)$, instead. This is in contrast to a regular autoencoder, where it evaluates to a discrete value of z itself - see equation 2.1. As a result, the resulting encoded output is actually an estimated probability distribution of the latent value, rather than the latent value itself. This estimate is denoted by $q_\theta(z|x)$ below:

$$q_\phi(z|x) \approx p_\theta(z|x) \tag{2.2}$$

where ϕ is the learned parameters of the encoder, also known as *variational parameters*.

In VAE implementation, and through its learning process, this conditional distribution $q_\phi(z|x)$ is often forced into a more manageable and simpler Gaussian distribution $\mathcal{N}(\mu, diag(\Sigma))$, where μ denotes the mean and Σ denotes the covariance of the Gaussian distribution. Hence, the encoder evaluates:

$$\begin{aligned} (\mu, log\ (diag\ (\Sigma))) &= Encoder_\phi(x) \\ q_\phi(z|x) &= \mathcal{N}(z; \mu, diag(\Sigma)) \end{aligned} \tag{2.3}$$

Notice that, in order to keep the outputs of the encoder network within functional range and to simplify loss computation, the encoder in practice evaluates $(\mu, log\ (diag\ (\Sigma)))$ pair instead of $(\mu, diag\ (\Sigma))$ pair.

The decoding, on the other hand, estimates the opposite conditional probability distribution, $p_\theta(x|z)$, from the learned joint probability distribution $p_\theta(x, z)$. In practice, however, the decoder is usually a generative process where the decoder samples from an encoded latent probability distribution as $z \sim q_\phi(z|x)$ and generates reconstructed x' samples representing the expected value of $p_\theta(x, z)$ distribution (i.e., the mean of the distribution). Unlike in the encoded representation, there is no additional use of the variance of $p_\theta(x, z)$ and, hence it is not generated by the

decoder.

$$\begin{aligned} z &\sim q_\phi(z|x) \\ &\sim \mathcal{N}(z; \mu, diag(\Sigma)) \\ x' &= \mathbb{E}(p_\theta(x|z)) \\ &= Decoder_\theta(z) \end{aligned} \tag{2.4}$$

By decoding a large number of sampled z values, producing that many samples of x', the distribution $p_\theta(x|z)$ can be approximated.

The VAE process described previously is illustrated graphically in figure 2.5.3. For a further detailed reading on VAE, see Kingma and Wellings (2014 [12] and 2019 [11]).

VAE training and Evidence Lower Bound (ELBO) loss function

Notice from the above that the objective of a VAE is to learn the true posterior. Hence from Bayes's theorem, we can express:

$$p_\theta(z|x) = \int_z \frac{p_\theta(x|z)p_\theta(z)}{p_\theta(x)} dz \tag{2.5}$$

However, the above integral is intractable because $p_\theta(x)$ is intractable. For this reason, a VAE instead attempts to approximate the posterior $p_\theta(x|z)$ by learning an estimate $q_\phi(x|z)$. For computational convenience, $q_\phi(x|z)$ is approximated by a normal distribution.

Hence, the training objective or the optimization objective of a VAE autoencoder is a bit more involved compared to a non-stochastic autoencoder. We want to learn both ϕ and θ parameters of the encoder and decoder, respectively, such that the following two goals are achieved:

1. The encoder estimates $q_\phi(z|x)$ as close as possible to the true latent space posterior $p_\theta(z|x)$ (see equation 2.2).
2. The decoder closely reproduces the input x with the least errors.

Consequently, the backpropagation of a VAE utilizes a loss function that captures the error specified in the previously listed two optimization goals [12]. The first goal can be achieved by increasing the Kullback–Leibler (KL) divergence of the two distributions, $q(z|x)$ and $p(z|x)$. The KL divergence of two distributions, denoted by $D_{KL}[q(z|x)||p(z|x)]$ measures the similarity of the distributions $q(z|x)$ and $p(z|x)$, and is given by the following identity [2]:

$$D_{KL}[q_\phi(z|x)||p_\theta(z|x)] = \int_{-\infty}^{\infty} q_\phi(z|x)\ log\left(\frac{q_\phi(z|x)}{p_\theta(z|x)}\right) dz \tag{2.6}$$

However, the above integral is also intractable. Instead, we notice that the log-likelihood of a single sample x^i evaluates as follows [12]:

$$log(p_\theta(x^i)) = D_{KL}[q_\phi(z|x^i)||p_\theta(z|x^i)] + \mathcal{L}(\phi, \theta|x^i) \tag{2.7}$$

Notice that the first term is the D_{KL} term of equation 2.6 itself. Since the log-likelihood $log(p_\theta(x^i))$ is constant for a given x^i, and that KL Divergence by definition is always $>= 0$, it implies that to increase $D_{KL}[q_\phi(z|x^i)||p_\theta(z|x^i)]$, we can also simply decrease $\mathcal{L}(\phi, \theta|x^i)$. We also see that the term $\mathcal{L}(\phi, \theta|x^i)$ is the lower bound in the value of $log(p_\theta(x^i))$ (the evidence of data), and is consequently known as "Evidence Lower Bound" or ELBO [12]. The ELBO term can be expanded to:

$$\begin{aligned} \mathcal{L}(\phi, \theta; x^i) &= \mathbb{E}_{q_\phi(z|x)}[-log(q_\phi(z|x)) + log(p_\theta(x, z))] \\ &= -D_{KL}[q_\phi(z|x^i)||p_\theta(z)] + \mathbb{E}_{q_\phi(z|x^i)}[log\ p_\theta(x^i|z)] \end{aligned} \tag{2.8}$$

From equation 2.3, we know $q_\phi(z|x^i)$ is a normal distribution with mean μ_z and variance vector diag(Σ_z), denoted further as consisting of elements ${\sigma^2}_j$, and prior $p(z)$ is defined as unit normal distribution $p(z) = N(0, I)$. Hence, the KL Divergence term in the

equation above simplifies further as [12]:

$$\mathcal{L}(\phi, \theta; x^i) = -\frac{1}{2}\sum_{l=1}^{J}\left[log\ \sigma_j^2 + 1 - \sigma_j - \mu_j^2\right] + \frac{1}{L}\sum_{l=1}^{L} log\ p_\theta(x^i|z^l) \tag{2.9}$$

Where, J is the number of dimensions in z, and L is the number of z samples drawn from $q(z|x^i)$ to approximate the decoder distribution $p_\theta(x^i|z)$ to compute the expectation using the monte-carlo method.

The equation above is the key to implementing the loss function in this project. Chapter 4 presents the implementation.

2.5.3 Long Short-Term Memory (LSTM) cell

The VAE model described earlier can be designed to be temporal aware by utilizing a Recurrent Neural Network (RNN) architecture for the encoding and decoding neural networks. A few notable RNN architectures used in VAEs are well known by now, namely LSTM, Gated Recurrent Unit (GRU), and their variants.

Earlier attempts at using RNN for a long sequence of temporal data faced challenges due to a phenomenon known as vanishing gradients during backpropagation. When the time sequence is relatively long, the recurring backpropagation through time gradually diminishes the contributing error gradient exponentially, until there is not much left for the network to learn at the end of the recurring sequence (Sepp Hochreiter, 1991). This issue made it impractical to use RNN in deep neural networks involving long time-series data.

However, with the advent of LSTM (Sepp Hochreiter, Jürgen Schmidhuber, 1997 [9]), this changed completely. LSTM architecture finally enabled very long sequence of recurring backpropagation in the network, and still be able to learn from error gradient much further down in time sequence. Similar to traditional RNN, LSTM also consists of feedback connections from and to itself (recurrent feedback). However, in contrast to RNN network, LSTM uses multiple internal gates to control the level of feedback during its recurrent steps. The gates have parameters which are part of the learned parameters. Figure 2.5.4 shows the internal architecture of an LSTM cell, and the three internal gates known as input gate, output gate and forget gate (seen as σ

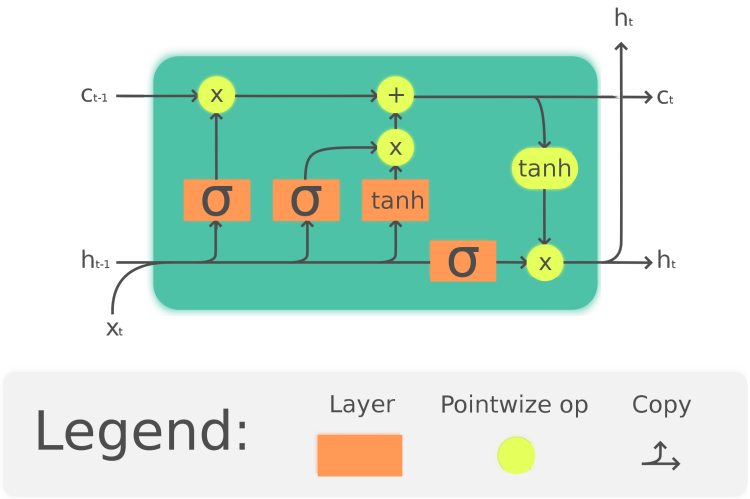


Figure 2.5.4: Long Short-Term Memory Unit [13] (figure is courtesy of Guillaume Chevalier, 2018)

blocks in the figure). Through these gates, the LSTM cells learn to control the level of feedback and can learn to hold memories from previous states. LSTM architecture was further improved by the introduction of GRU, mainly in the network stability and performance, owing to its fewer learned parameters. GRU was introduced in 2014 by Kyunghyun Cho et al [4].

There are other variations to LSTM. For example, Morales-Forero1 and S. Bassetto, 2019 [20], proposes an architecture variation that enables semi-supervised learning of an LSTM based anomaly detection. LSTM is not the only suitable architecture for anomaly detection in temporal data. For example, Ribeiro et al, 2018 [25], proposes an anomaly detection architecture for video surveillance purposes using an autoencoder based on Convolutional Neural Network (CNN).

2.6 Related work: Anomaly detection

This degree project developed a time-series anomaly detection system utilizing previously described VAE and LSTM architectures as the core components for detecting anomalies in battery operation. To describe such a system, this section first presents some approaches to anomaly detection using deep neural networks and then describes the architecture of interest that combines VAE and LSTM.

2.6.1 Prediction-based anomaly detection

Prediction-based anomaly detection works by predicting the anomaly based on past measurements. This method requires supervised training, hence the anomalies need to be labeled in the training data. A simple example is a feedforward neural network trained to predict an anomaly at time t based on the past measurements from time t to $t - w$, where w is the detection window of the input time-series (see figure 2.6.1).

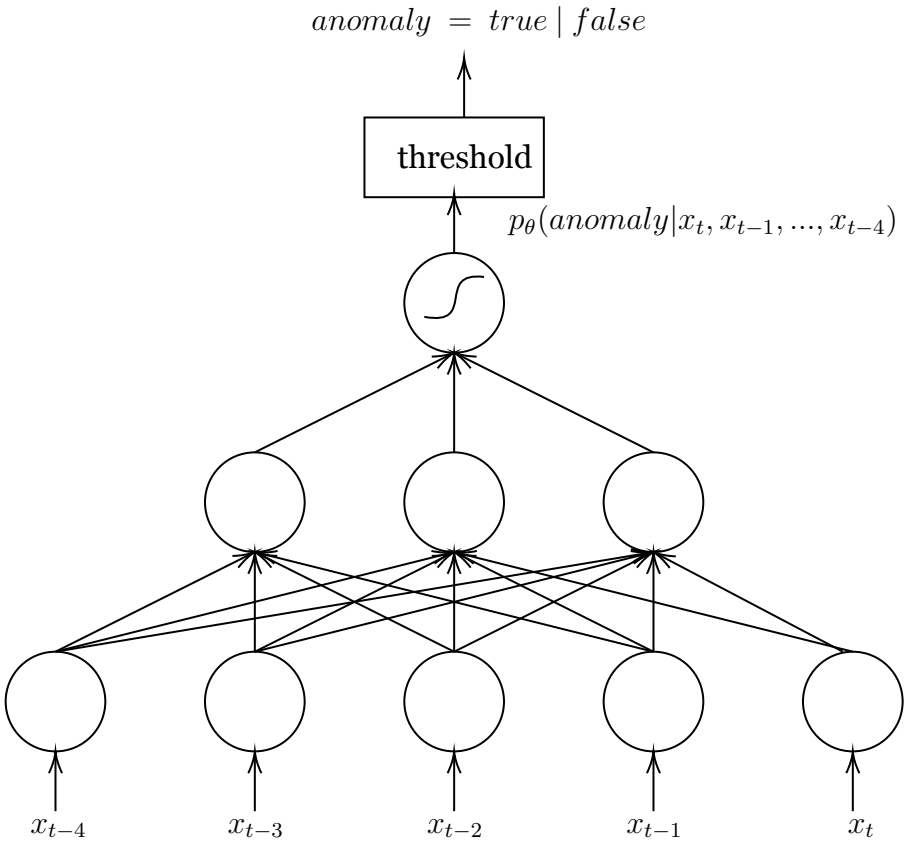


Figure 2.6.1: An example feedforward binary classifier neural network for anomaly detection. The network is trained using known anomalous data and predicts the probability of a given new input being an anomaly during inference.

Unfortunately, in many practical applications, the anomalies are rare, expensive to reproduce (since they are actual failures), and are not readily labeled. As a result, unsupervised training, or semi-supervised training offers a better choice. The following subsections explore various deep learning networks that can learn using unsupervised training methods.

2.6.2 Reconstruction-based anomaly detection

Reconstruction-based anomaly detection relies on encoding the input measurement into a smaller-dimension latent variable, which attempts to reconstruct back into the original measurement. The process of reducing input to a smaller dimension variable is known as dimensionality reduction. However, the architecture also includes a decoding counter-part, which reconstructs the lower-dimension variable to its original higher-dimensional measurement values. The network assumes the training would have captured all non-anomalous data in the training set into their corresponding latent variable representations. The latent values, in turn, could be reconstructed back to the real measurement data (or close to them). The architecture just described is, in fact, an autoencoder, familiar from the previous section. This is illustrated in figure 2.6.2. There exists a variety of deep neural network autoencoders which have been adapted for anomaly detection purposes.

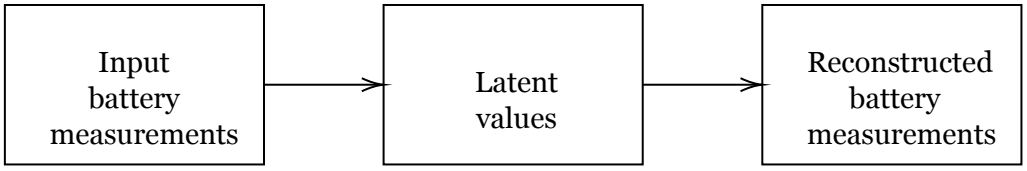


Figure 2.6.2: Reconstruction-based autoencoder.

Given the function of an autoencoder, a normal measurement should correctly encode to an existing latent variable value, which would then be able to correctly reconstruct back to the original measurement. An anomaly, on the other hand, would encode to a latent variable which would fail to reconstruct to a close match of the input. This failure, the anomaly, can then be detected as an unusually large deviation of the reconstructed value from its original measurement.

2.6.3 Stochastic anomaly detection

The type of anomaly detectors discussed in the previous section utilizes architectures, such as autoencoders, that maps the input data features directly to their lateral representation in a non-probabilistic / discrete approach. The class of anomaly detectors that utilizes stochastic autoencoders are colloquially known as stochastic anomaly detectors. They learn the probabilistic mapping of the input data to the latent space representation. One example would be an anomaly detector that utilizes VAE as the main component. Unlike a regular autoencoder, we recall from section 2.5.2 that

VAE maps the underlying probability distribution of the input data to the distribution of the latent variable hence is quite a natural component for such type of anomaly detector.

2.6.4 Time-series anomaly detection

We finally arrive at the architecture of interest for this degree project. The battery data is a collection of time-series measurements of a battery's operational parameters, such as voltage, current, and temperature. All of them change over time, where the present values are dependent on previous measurements. As a result, the anomaly detector needs to take into account the time-series nature of the measurement data.

Coming back to the time-series autoencoder, it is, therefore, natural to consider LSTM [17] [16] [24] [10], GRU [8], or a similar unit as an architecture of choice for a time-series anomaly detector.

Malhotra et al., 2015 [17], proposed a prediction-based anomaly detector that uses stacked LSTM units (names it LSTM Anomaly Detector or LSTM-AD). The network is trained on non-anomalous data to enable predicting the time-series data. Based on the deviation error of the predicted output and the actual data, the detector then identifies an anomaly. This approach is similar to the non-time-series approach described in section 2.6.1.

Maleki et al., 2021 [16], improves upon LSTM based autoencoder by incorporating filtering of samples prior to feeding into the network for training. It assumes that the input samples are drawn from a normally distributed population. Any sample that is outside a certain threshold of deviation from the mean of this population is considered an anomalous sample (considered drawn from different distribution) and, therefore, is excluded from the training. This improved the performance of the network despite the training dataset containing anomaly data [16].

2.6.5 LSTM and VAE based anomaly detection

In the previous cases, we have so far separately covered VAE based anomaly detector (used for non-temporal data) or LSTM based anomaly detector (used for time-series data). In this subsection, we now look into anomaly detectors that are designed using both LSTM and VAE. As noted earlier, VAE based autoencoder

architecture generalizes the encoding model better, avoiding the inherent overfitting issue associated with a regular autoencoder. Such implementation is also known as Sequence-to-Sequence (Seq2Seq) LSTM-VAE autoencoder.

In recent times, researchers have developed and studied LSTM-VAE architecture in several different applications. For example, Park et al. 2018 [24] utilize it in a robot-assisted feeding system. Hsu et al. 2017 [10] also utilize LSTM in a VAE architecture for the purpose of speech recognition. Hsu et al. 2017 [10] implementation is the closest architecture that this project would be utilizing.

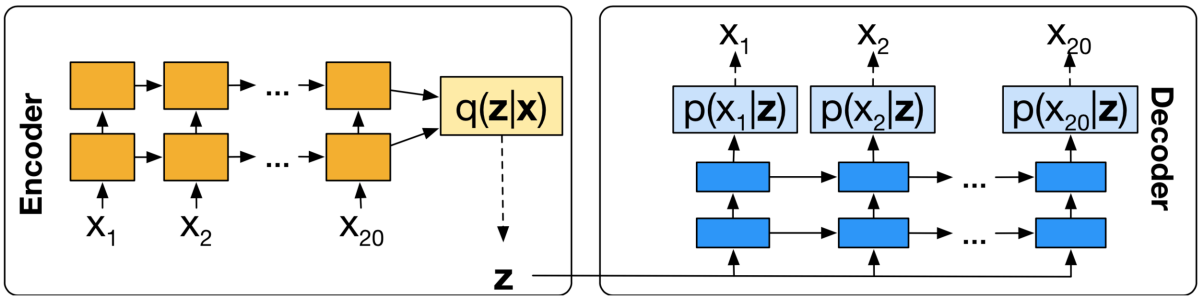


Figure 2.6.3: Variational Autoencoder (VAE) using Long Short-Term Memory (LSTM) units

Source: Hsu et al. 2017 [10]

We will have a closer look at it in the chapter 3, but briefly, it consists of an LSTM encoder with a number of steps equaling the input time-series steps. The encoder processes the input sequence or the time-series data and produces the latent distribution $q_\phi(z|x)$. The decoder consists of another LSTM network with steps equalling the sequence or time-series length. It samples z from the encoded latent distribution and learns to reproduce x as $\mathbb{E}(p_\theta(x|z))$. Hence, it follows the steps described in the equation 2.4. The Sequence to Sequence (Seq2Seq) LSTM-VAE architecture is shown in figure 2.6.3.

2.6.6 Anomaly score

Reconstruction-based anomaly detection, such as the LSTM-VAE network described above, utilizes the reconstruction likelihood as a score to identify the probability of the inferred input being an anomaly. Reconstruction likelihood is a measure used to identify Out Of Distribution (OOD) inputs, which translates to Mean Squared Error (MSE) between the reconstructed measurement and the corresponding input measurements. In an application, a suitable threshold to the anomaly score is set to

classify the input as either normal, when the score is low, and anomaly, if the score is above the threshold.

2.6.7 Known issues

Recent research has shown that there are some issues when using generative deep neural networks, such as VAE, for anomaly detection purposes. An anomaly detection relies on identifying inputs that are within the distribution of the normal inputs. Inputs outside the distribution (OOD) are assumed to be an anomaly. For this reason, it is considered that generative models, such as VAE are good in mapping the distribution of the training data, such that normal input data that have not been seen before would also correctly map. However, this notion is challenged by Nalisnick et al. 2019 [21]. They argue if generative models truly know the input data that have not yet been known [in the training dataset]. According to them, not always. They illustrate this failure in their 2019 publication [21].

Xiao et al. 2020 [28] observe that the likelihood method as described in the previous section could fail and propose an alternative measure known as ”Likelihood Regret” and claims it to be a more efficient OOD detector for VAEs.

Chapter 3

Engineering Methodologies and Method

This chapter describes the methods followed in this project to reach the objectives described in section 1.4 and answer the research questions posed in chapter 1. The project follows a combination of an engineering method and a scientific method because of their suitability to the project's goals and implementation. The engineering method helps to streamline the heavy data acquisition process and software development needed in the project, while the scientific method aligns well with analyzing the results of the experiment and answering the research questions.

The engineering method follows the model proposed in *"The Electrical and Computer Engineering Design Handbook"*, a book by students of Tufts University as a collaborative effort. The method described in the book aligns well with the need of this project to develop from idea to solving the hypothesis posed in the beginning.

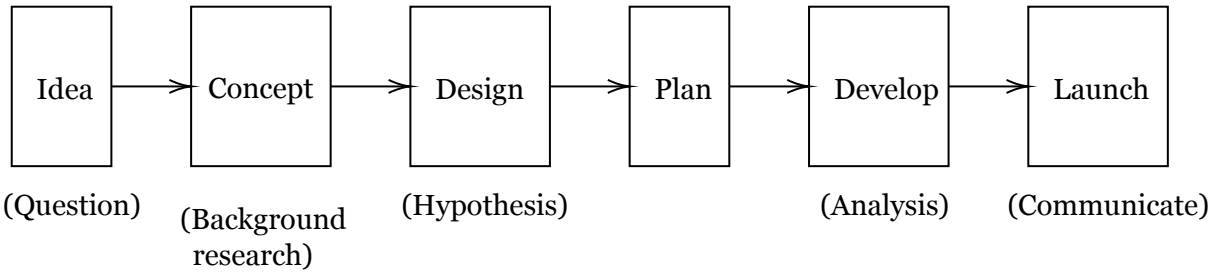


Figure 3.0.1: The engineering method stages followed in the project.

The method consists of six stages of engineering: idea formation, concept design,

planning, design, development, and launch. The stages are shown in figure 3.0.1. Alongside each stage is also shown their scientific method stage that would be utilized in combination.

The following sections describe the reasoning, methodology implementation, and resources involved in each of these stages.

3.1 Ideation

The idea for this degree project originated from the need for Fortum's eFleetly project. eFleetly is a battery fleet management platform focusing on predictive maintenance. As part of its development, it became necessary to enable the prediction of failures beyond what may be obvious from the traditional approaches.

As a result, the study of anomaly detection for batteries has been proposed as the topic of this degree project. Initial interviews and discussions with the eFleetly pilot customer, Logisnext Finland, and partners provided the groundwork on the requirements of such a system and validated the usefulness of the topic in the development of the eFleetly platform. Furthermore, the eFleetly team and partner provided permission and full support needed in the project.

During the evaluation of the idea of anomaly detection for batteries, it became interesting to question the potential failure modes that could be detected by applying state-of-the-art anomaly detection techniques on the customer's battery measurement data. Eventually, this curiosity sealed the research question of this project.

In order to develop the idea, an onsite co-development workshop with the potential users (of the system) from Logisnext Finland Oy and Hartwall Oy was conducted in the early stage of the project. The co-development workshop provided useful insights and validation of several ideas for eFleetly, including the one implemented in this project.

Chapter 1 details the formulation of this idea in terms of project goals and research questions that this project aims to resolve.

3.2 Concept development

Anomaly detection is not particularly a new topic in the literature. However, designing one for battery monitoring hasn't been particularly addressed widely, especially from an engineering and product perspective. Therefore, a key part of the concept development for this project comprised background studies and identification of prior research in the area. Chapter 2 provided the results of the background studies and introduced various technologies involved in this project.

The other aspect of the concept development was to design the basis of anomaly detection as a feature for the users and address the design questions: (1) how it will be used, (2) what are the expected interaction, and (3) what are the end benefits to the users. The concept is tested and validated with potential users and other stakeholders in the project before implementation.

Chapter 4, section 4.1 presents the result of this process and the final concept that was eventually developed.

3.3 Planning

Planning was a key project management activity performed early in the project to create a viable execution plan. In planning, I identified all the anticipated tasks involved in the execution of the project. For all tasks identified in the plan, I also estimated the length of the tasks, identified any dependencies between them, and estimated their likely finishing time. Planning often involved inputs from other experts involved in the eFleetly project.

Chapter 4, section 4.2 presents the project plan outcome of this degree project.

3.4 Design process

The system design process involved exploring the possible approaches to implement the battery anomaly detection system and designing the system components. It consisted of three stages in the project. The first stage was the background research conducted in chapter 2 to identify existing technologies and deep learning approaches in the area. The second stage consisted of designing the data pipeline and system

architecture, beginning from the sensors all the way to the user interface. Lastly, the third stage consisted of designing the anomaly detection system architecture, its training, and analysis of the results.

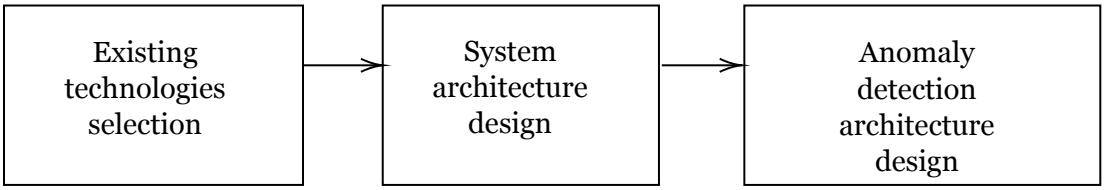


Figure 3.4.1: The design process stages that were conducted in the project.

The three states in the design process are illustrated in figure 3.4.1. Chapter 4, section 4.2 presents resulting designs developed in this project.

3.5 Development

The development process consisted of the bulk of work in the project, namely, the implementation of the data pipeline, the data collection, the implementation of the anomaly detector, training the network and analyzing the results.

The development process followed the agile software development methodology with iterative cycles. We used the *Kanban* method [1] to manage the project and tracked it using a *Trello* board. Trello[2] is a lightweight online project management tool. Trello was selected to manage the project because of its lightweight usability. The software was developed using three different datasets with progressively increasing complexity. The approach of using datasets progressively helped ensure development and troubleshooting progressed more systematically. Figure 3.5.1 shows the three stages corresponding to the three datasets used in the development. The final one is the real battery dataset collected from a real customer.

The LSTM-VAE deep neural network was implemented using python Keras framework [3] as the programming language and development framework, Jupiter Notebook [4] as the editing and development platform and Google Colab [5] as the execution environment. Keras framework was chosen for its rich community support in

[1]https://en.wikipedia.org/wiki/Kanban_(development)
[2]https://trello.com/
[3]https://keras.io/
[4]https://jupyter.org/
[5]https://colab.google.com/

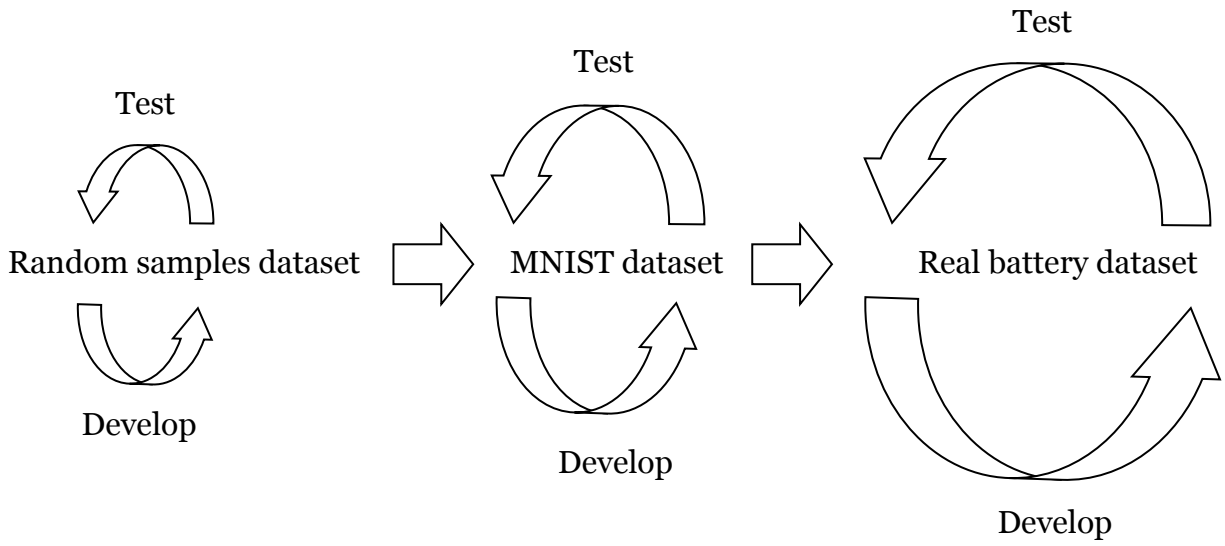


Figure 3.5.1: The iterative process and stages are used in the development of the project. Each stage used progressively more complex data to enable progressive development and debugging.

machine learning and deep learning development. Jupiter notebook enables rapid prototyping development. And, Google Colab provides an excellent GPU-enabled host for developing collaborative projects. It has especially been popular among the Machine Learning (ML) development community.

3.6 Launch and communication

The project does not have a *Launch* stage, per se, in the traditional sense of launching a software product. However, this thesis comprises its final documentation and communication of all the activities and results of the project. The hope is that the outcome of this project would be available to the community and the literature so that others can further the research and development in this domain.

Chapter 4

Anomaly Detection in Battery Time-Series Measurements

This chapter describes the project and all the steps performed and implemented to accomplish the result of this thesis. The methods described in the previous chapter are used as guidance for performing the tasks in the project.

4.1 The concept

The concept of anomaly detection for batteries arises from the need to optimize the maintenance of battery fleets. Customers are often blind to potential failures before it is too late, often causing potential loss of productivity, and sometimes, loss of equipment life.

Before this project, the pilot customer used their forklift batteries without any digital feedback or monitoring. As a result, many cases of abuse occurred to the batteries during daily operations. Therefore, identifying anomalies during operations could provide valuable management feedback. By acting on that feedback, management can potentially improve their productivity.

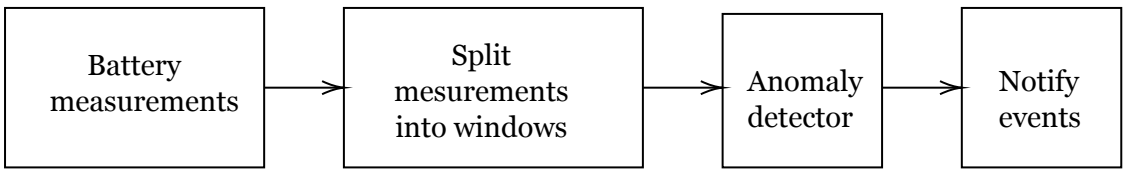


Figure 4.1.1: The high-level concept of battery anomaly detection.

The high-level concept consists of: (i) the battery measurement, (ii) dividing them into smaller window segments, (ii) testing each window for anomaly event, and (iii) notifying the events to the users. Measurements samples are 17 seconds apart. However, the size of the window is to be determined by experimentation. The last stage of notifying the users is outside the scope of this project. However, in finality, that is the primary purpose of this concept. Figure 4.1.1 illustrates this high-level concept. The section 4.3 elaborates and develops this concept into the system pipeline architecture.

4.2 The project plan

The project plan consisted of a series of tasks planned at the beginning of the project. Appendix B lists all the major tasks planned for the project. The major tasks were further broken down into minor tasks. However, they are not shown in the table since they were implemented in an agile development process just-in-time on a bi-weekly sprint basis.

4.3 Pipeline and system design

The system architecture consists of communication hardware, storage databases, and software processes configured in order to implement the full anomaly detection pipeline. Figure 4.3.1 illustrates the system architecture.

The customer site is considered to be organized into multiple separate locations. Each location hosts multiple batteries, and each battery has a battery sensor attached to it. WiFi is the primary communication medium for the battery sensors to connect to the nearby 4G routers. Some site locations can be relatively large, where a single 4G router may not be enough to cover the whole area. In such cases, it is typical to add one or more WiFi repeaters.

The measurement samples arrive at a central database where all raw data are collected for further processing at more convenient times. Usually, samples arrive at a much higher frequency than the actual processing. For example, in this project, the samples arrive in 5 minutes batches, while the anomaly detection process is performed every 24 hours. Before the anomaly detection process begins, the measurement data are pre-

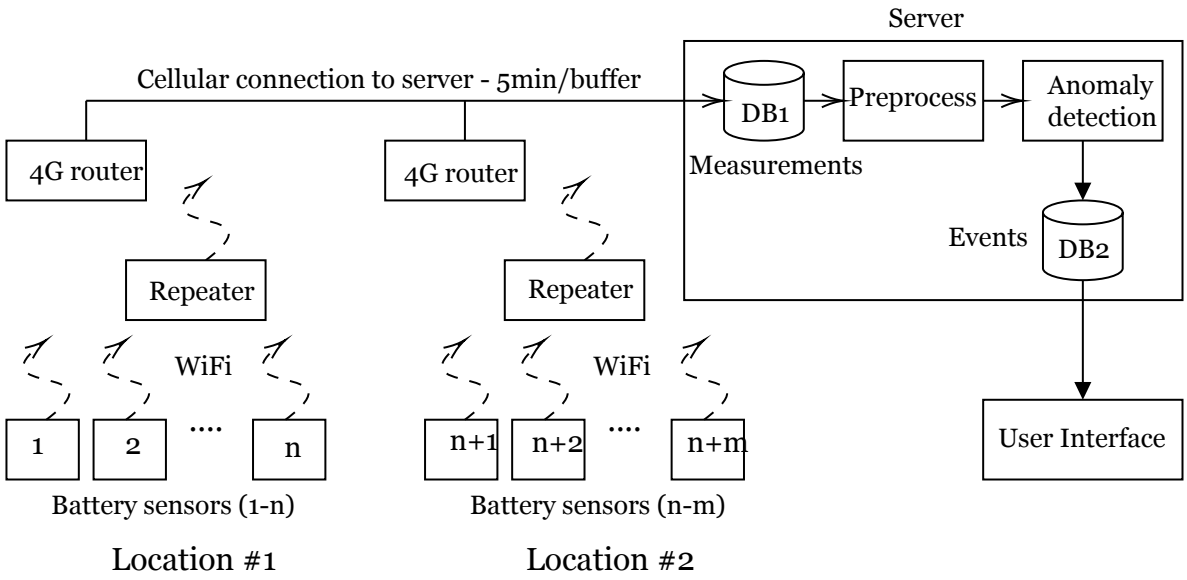


Figure 4.3.1: Data pipeline and system architecture.

processed into a more suitable form for the anomaly detector to work with. The output from the anomaly detection process is known as events, which are stored in a second database for processing by the user interface.

The next section walks through the various stages in the pipeline in greater detail, describing the entire process from data collection, from the sensors to the final pre-processing.

4.4 Data collection

The data for this project were collected from a warehouse belonging to the Finnish company Hartwall Oy. They are a drinks manufacturing company and this particular warehouse is used for storing and distributing their final products. The warehouse is located in the central Helsinki region, south of Finland.

The warehouse contains over 100 electric forklifts, all operated by lead-acid batteries. 100 lead-acid battery sensors were installed on 100 of those forklifts.

4.4.1 Battery sensor

The sensors installed were Bamomas lead-acid battery sensors. As shown in figure 4.4.1, the sensor has several probes connecting to the battery and is capable of measuring voltage, current, temperature, and water level in near real-time.

Figure 4.4.1: Bamomas lead-acid battery sensor device,

Source: Bamomas Battery Intelligence Oy

The sensor is powered by the battery itself and attached to the top of the battery securely. The voltage leads and current leads connect to the battery's supply lines leading to the forklift. The water level sensor is inserted into one of the cells, replacing its electrolyte filling cap. The temperature sensor is attached to the body of one cell, ensuring good thermal contact.

The sensors connect to nearby WiFi routers and repeaters, which in turn connect to the Internet via 4G cellular networks. For isolation reasons, the pilot data collection consisted of its own local area network. The local network for the sensors consisted of several 4G/WiFi routers and WiFi repeaters strategically placed around battery charging rooms (where there is most battery traffic). This enabled the sensors to connect to Bamomas cloud servers without interfering with the warehouse's network.

The warehouse contains three charging rooms equipped with arrays of lead-acid battery chargers. Forklifts would arrive at the charging rooms with nearly empty batteries and the drivers would swap them for fully charged ones. The drivers would then put the empty batteries on the chargers. These three charging rooms were identified as the ideal places to act as the main hub for syncing the battery sensors since almost all batteries would arrive there at some point. As a result, the WiFi network and the 4G routers were set up in and around these rooms. Some additional WiFi repeaters were also set up in high traffic areas, linking them to the main WiFi routers.

4.4.2 Sensors deployment and data collection periods

The sensors were deployed to the forklift batteries over a period of 2 months in June and July of 2020. The deployment process was slow due to the corona pandemic ongoing at the time. With no physical access to the warehouse site due to coronavirus pandemic restrictions, we had to remotely train a local technician to install and set up the sensors. As a result, the setup took more time than would normally take.

Following the setup and installation period, control data were collected for a period of about two months (July and Aug 2020), before some of the eFleetly maintenance features were activated. The reason was to use this data as control data for comparing the changes in batteries behavior once eFleetly maintenance features were activated.

Following the period of control data collection, there existed a period of about two months period (Sep and Oct 2020), during which eFleetly battery maintenance features were deployed. Unfortunately, that activity also took more time than anticipated due to various technical issues.

Lastly, the period following, Nov 2020 - June 2021, was the data collection with the eFleetly service in full operation. Figure 4.4.2 illustrates the timeline of data collection.

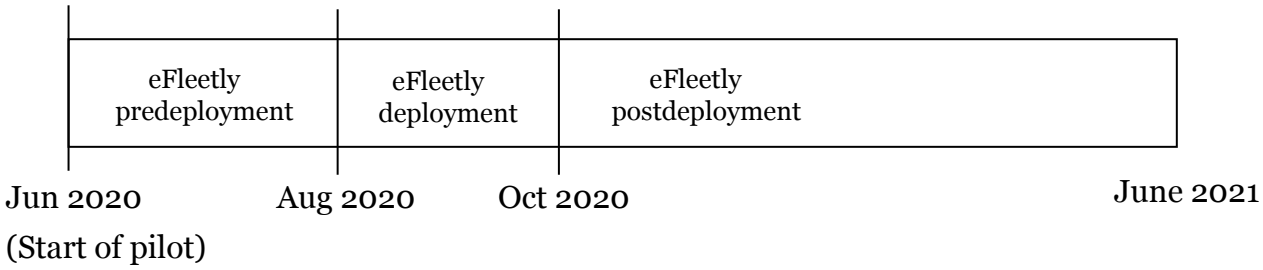


Figure 4.4.2: Collection timeline of battery measurement data during eFleetly pilot.

The periods above were mostly for the verification of other predictive maintenance features in eFleetly. This thesis does not distinguish the periods in terms of analyzing the anomaly detection behavior. However, the periods could be noted for identifying any changes in the frequency of anomaly incidences.

The sensors collect measurement data every 17 seconds. However, they are programmed to deliver the measurement data every 5 minutes in a batch. This is to reduce needless traffic between the sensors and the cloud server. Furthermore, if

some sensors are out of range from the WiFi stations, they would buffer the data until they are able to reconnect back to the server. Figure 4.4.3 shows a screenshot showing graphically some of the measurement data collected in the server.

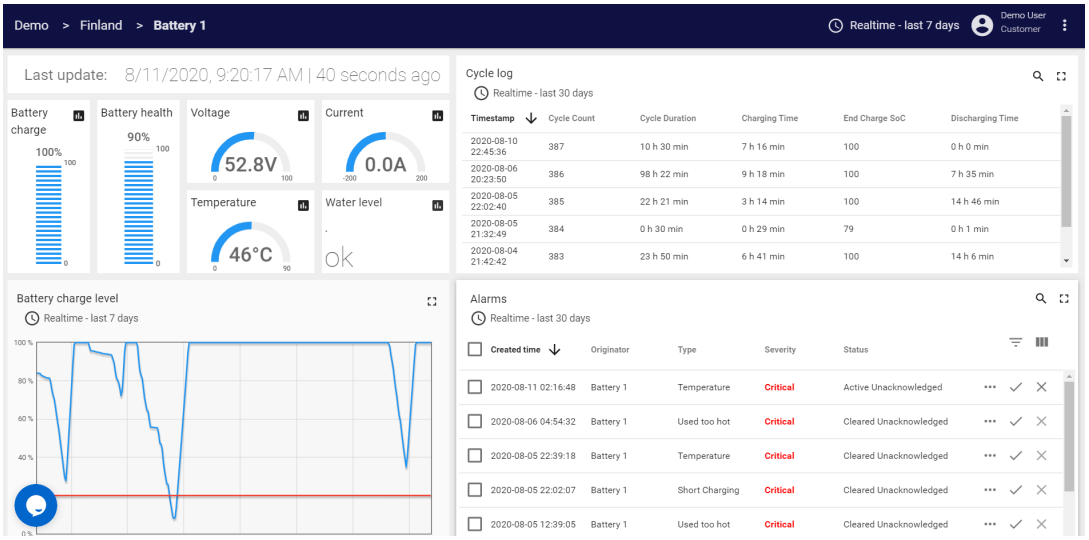


Figure 4.4.3: Measurement data collected by the sensors as seen from the server

Source: Bamomas Battery Intelligence Oy

4.5 Data pre-processing

4.5.1 Data extraction

eFleetly stores the data in a server in a MongoDB database. There are about 50G of data in the server, organized in MongoDB collections. It would not have been trivial to utilize the data directly. It became necessary to extract them into some simpler tabular form, such as flat Comma Separated Values (CSV) files, for use by the deep learning network. To avoid disruption in the production, the network is run separately (inside Google's Collab environment). Hence the challenge became to extract a large amount of data from the production server into CVS files and upload it to google drive for use by the Collab's Jupiter notebook, where the network is implemented.

The extraction processes could cause disruption to the running eFleetly service. In order to avoid the risk, the entire eFleetly database was copied to a local computer using backup and restore tools of MongoDB. The data was then extracted into tabular CSV files using MongoDB queries that extracted only the relevant data, namely battery_id, timestamp, voltage, current, and water level.

Given the size of the data, handling them in a single CSV file would be rather difficult. As a result, they were exported separately for each month, each roughly forming a CSV file of approximately 500MB size. They were then all compressed, copied to the Google drive, and uncompressed in the project directory.

4.5.2 Pre-processing

The eFleetly battery measurements arrive in timestamped measurement samples for each battery identified by a unique ID. Since sensors buffer the data and upload arbitrarily at opportune times, the data samples are not assumed to be sorted in any particular order. Figure 4.5.1 shows a plot of the voltage, current, and temperature variation of a battery on a typical working day at the warehouse.

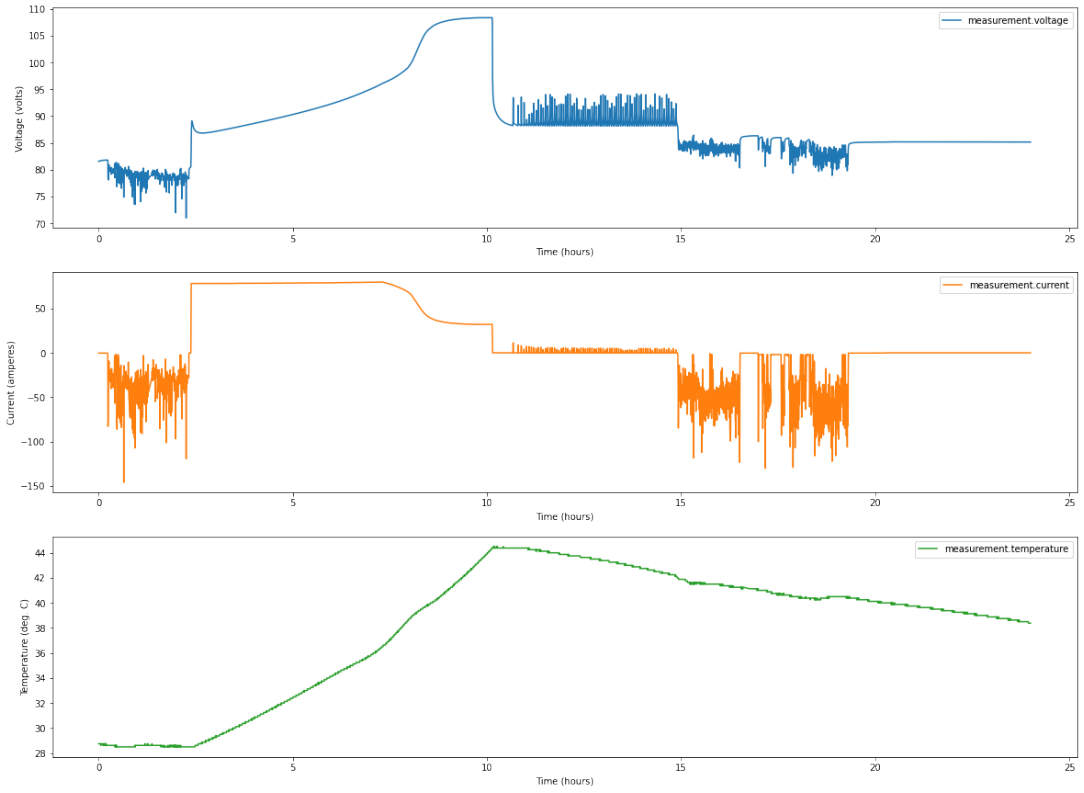


Figure 4.5.1: A plot of battery data on a typical working day)

The network training requires that the data samples are windowed in time series. However, all battery readings are mixed in the samples collection. Since every battery, the measurements are considered separate time-series. Hence, the network training needs to ensure that the samples from one battery are not mixed up with another.

In order to accomplish windowing correctly, the measurement samples are first separated into individual battery time series. The individual battery time series are

then sliced into windowed data samples. Any trailing measurement samples in each battery time-series data that could not form a complete window are discarded. Finally, The windowed data samples from all the batteries are combined into a single dataset. The windowed data samples in the dataset can be mixed up without affecting the training since they don't need to be in any particular order during the network training or validation.

However, the aforementioned processing is implemented more efficiently by simply sorting the measurement samples by battery ID and timestamp (in that order), deleting the odd trailing samples that are not modulus of the window size, and then finally slicing them into window-sized data samples in one go. The process accomplices a similar result without having to separate the individual time series.

4.6 LSTM-VAE network architecture

Typical of a VAE autoencoder (see section 1.1), the network architecture consists of two sections, an encoder, and a decoder. The encoder is implemented using an LSTM layer with a number of time steps matching the window size of the data sample. The LSTM layer's output pass through two separate feed-forward networks to output latent variable's mean μ_z vector and variance vector σ_z^2 (representing diag(Σ) of the covariance discussed in chapter 2).

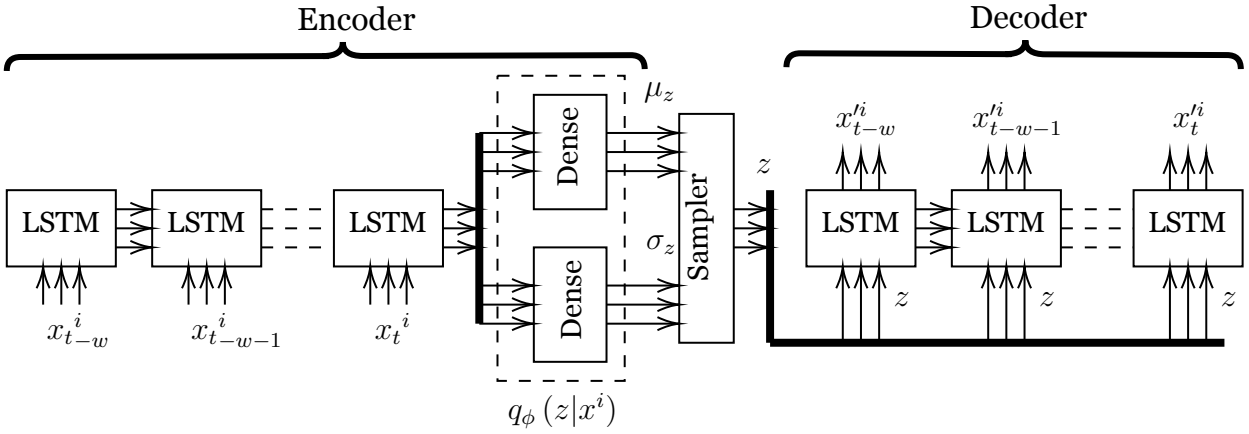


Figure 4.6.1: LSTM-VAE network architecture implemented for battery anomaly detection.

The latent variable z itself is sampled from the distribution with mean μ_z and variance σ_z^2. Once sampled, z is fed to the decoder network, which ideally should attempt to

reproduce the original input.

The decoder is another LSTM layer with a number of steps equal to the input window size. This is to produce an output that has the same size as the input data sample. The decoder LSTM network still takes input with full window size. For this purpose, the latent variable z is repeated a number of times equal to window size to form the input data sample for the decoder (see figure 4.6.1).

The final output of the network is simply the output of the decoder LSTM layer collected at each time step. Figure 4.6.1 illustrates the architecture.

4.6.1 Network loss

The deep neural network is trained following a loss computation described in chapter 2, section 2.5.2. It is a combination of reconstruction loss, given by cross-entropy loss computed with MSE of the output y and the input x, and latent variable distribution loss, given by the KL divergence value of the latent variable distribution.

$$\begin{aligned} Loss &= Reconstruction\ loss + Variational\ loss \\ &= Loss_{MSE} + Loss_{KL} \\ &= \frac{1}{N.W}\sum_{i=1}^{N}\sum_{w=1}^{W}(\hat{x}_{i,w} - x_{i,w})^2 - \frac{1}{2}\sum_{i=0}^{N}(1 + log\ \sigma_{z_i}^2 - \mu_{z_i}^2 - \sigma_{z_i}^2) \end{aligned} \tag{4.1}$$

Where,
N is the total number of training samples,
W is sample window size,
x is the input sensor reading,
$\hat{x}$ is the reproduced sensor reading,

KL Divergence loss, in particular for a VAE was described in the section 2.5.2 above.

4.6.2 Reparameterization trick

In a variational autoencoder, the encoder outputs a distribution (mean μ and variance σ^2 pair) of the latent variable. Subsequently, the decoder decodes a sample from this latent space as:

$$z \sim q_\phi(z|x)$$

This is okay in theory. However, in practice during implementation, random sampling makes it impossible to implement backpropagation of the network for its training. There exists a discontinuity in gradient at the random sampling point, where the backpropagation can not flow back.

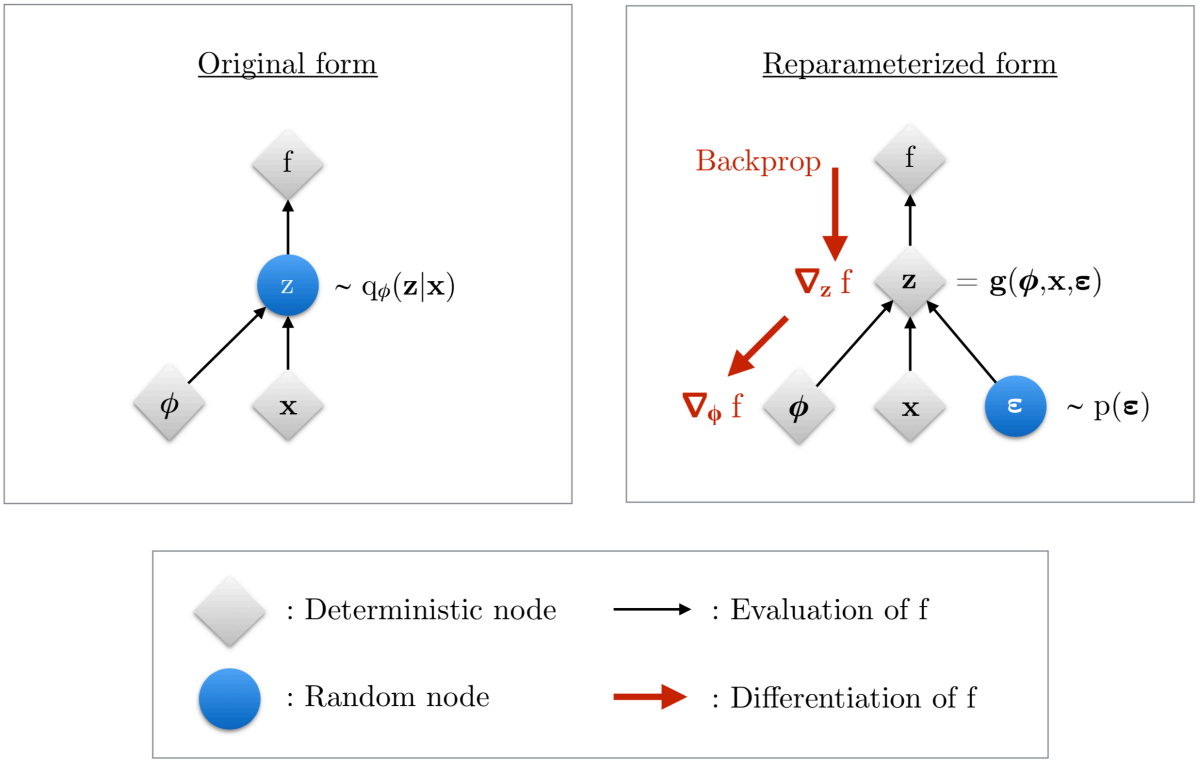


Figure 4.6.2: Illustration of reparameterization trick, [11]

To overcome this limitation, "Reparameterization Trick" is introduced in the implementation of the network [11]. Instead of dealing with a random sampling node, a deterministic node is introduced which is a function taking the latent variable mean and variance, as well as an additional input ϵ. The additional input ϵ is simply random

samples taken from a unit normal distribution N.

$$\begin{aligned} z &= g(\mu, \sigma^2, \epsilon) \\ \epsilon &\sim N(0, I) \\ g &= \mu + \sigma.\epsilon \end{aligned} \tag{4.2}$$

The node simply outputs the result of modifying the random sample input by the mean x and standard deviation σ. Figure 4.6.2 illustrates this construct.

4.7 Network development and training

The deep neural network development and training were performed at the same time in an iterative process. The datasets used in the development were progressively increased in complexity so that the iterative development progressed in the following order (in the given order):

1. A small dataset of random samples: Used to develop the architecture of the network.
2. MNIST dataset: Used to debug and troubleshoot the training. MNIST being a known dataset, it provided a good basis to debug and troubleshoot the implementation.
3. A small subset of the real battery data (one-month data): Used to implement the prepossessing and validate the network operation on the real battery data.
4. The full one-year battery dataset: Used for the final training of the network and analysis.

The network architecture was developed using a test dataset consisting of 6-dimensional input samples with elements taking only uniformly generated random values between 0.0 - 1.0. This enabled rapid development and troubleshooting of the network architecture and validation of its model processing. Figure 4.7.1 shows the training result. The test network had a latent variable of dimension two and the input record consisted of 32 time-steps.

The figure 4.7.1 shows that the network is learned as expected by converging to a

possible model with minimum loss. At the end of the training, it would actually approximate itself as a uniform random samples generator of the same parameters as the random generator used to generate the input. Obviously, the samples themselves can not be differentiated among themselves. This is seen as a fairly random variation in the training and validation losses. The network was trained using Adam optimizer [] with a learning rate of 0.0001 and a decay rate of $1 \times 10^{-}6$. The training was conducted using 100 input samples in batches 5 samples, and iterated in 200 epochs. However, it can be noted from the figure that the network has not fully converged yet by then. This is not considered an issue since the purpose of the test was mainly to assist in developing the first iteration of the network.

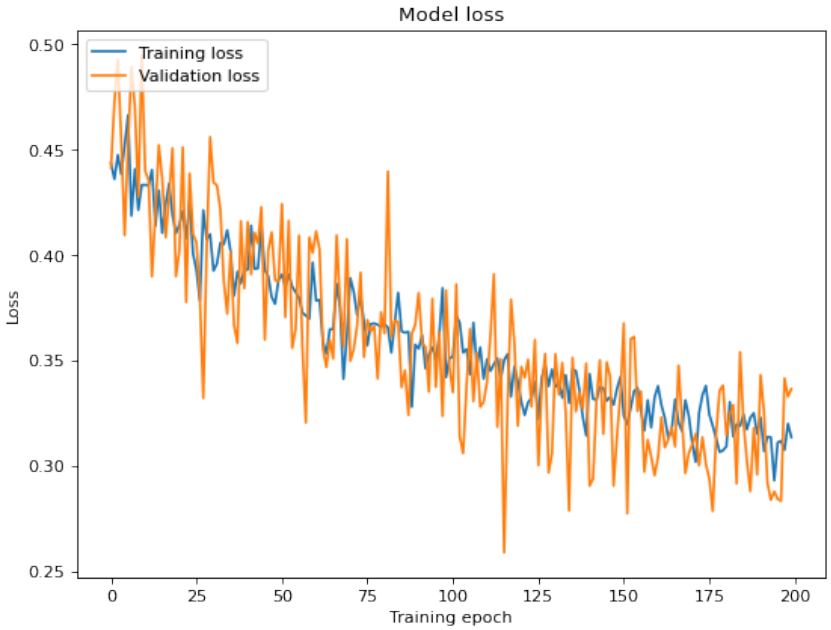


Figure 4.7.1: Result of training the LSTM-VAE network with a dataset of random samples. Validation loss progressing wildly demonstrates the network's randomness in the model.

The next iteration developed the network further using the well know handwritten digits dataset by Modified National Institute of Standards and Technology database (MNIST) [15]. The MNIST dataset is a popular dataset containing 70,000 images of various handwritten digits, each sized 28x28 pixels and mono-colored. The database is often used by researchers to develop models involving pattern recognition. In the development, each individual digit in the MNIST dataset was taken as one windowed data record. The MNIST dataset was selected as a test dataset because of its well-recognized performance in the literature.

The rows of a digit in the dataset form the time-series values of the input record, while the columns are the dimensions of the input variable. Therefore, each training input record representing each digit in the dataset consists of m time-series values of n dimensional value, where m is the number of rows and n is the number of columns of the digit. The rows are ordered top to bottom in the time-series input.

The digit images in the MNIST dataset may not appear to be typical time-series records. However, the order of the rows in a digit is important in forming the complete image of the digit. As a result, a time-series record formed using the rows serves a useful purpose to test the temporal nature of the network architecture. Note, we could have reversed the rows and columns in the input record formation. The same reasoning applies.

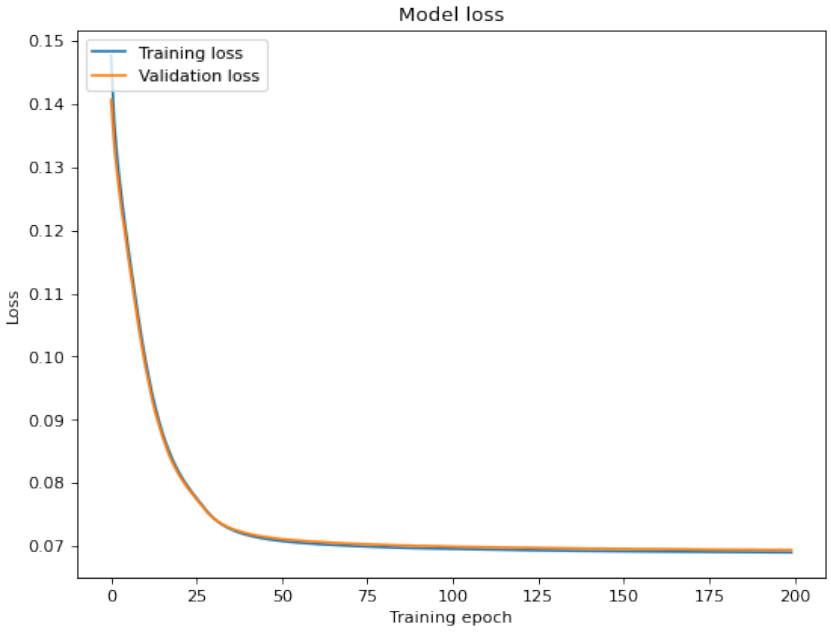


Figure 4.7.2: Result of training the LSTM-VAE network with the MNIST dataset. Validation loss reduction rapidly demonstrates that the network has learned to encode and decode the underlying model.

The LSTM-VAE network architecture was constructed with a hidden layer dimension of 6, latent variable dimension of 2, input dimension of 28 (the width of each digit image), and 28 time-steps (the height of each digit image). The network training used an Adam optimizer with a learning rate of $1 \times 10^{-}5$ and a decay rate of 1×10^{6} and was trained in batches of 128 input samples over 200 epochs. Figure 4.7.2 shows the result of training with the MNIST dataset. The figure shows a rapid reduction in the validation error, which demonstrates that the LSTM-VAE autoencoder is able to learn

the generative model using just a 2-dimensional latent variable. This demonstration validates the functioning of the network's VAE architecture. This further implies the loss implementation of the network is functioning as well.

We finally arrive at the third iteration of development where we used the measurement samples from one battery, captured over a period of one year (the duration of the pilot). This iteration took the longest so far due to the size of the data used in the training, and the fact that the network architecture parameters were the most numerous so far. By this iteration, the network is considered fully functioning

4.8 Hyperparameters search

The LSTM-VAE network developed above can take different model complexities depending on various architecture design factors, such as LSTM hidden layer dimension, latent variable dimension, dropout probability in encoder's dense layers, and dropout probability in decoder's dense layers. These design factors are known as *hyperparameters*. We assume a certain combination of the hyperparameters forms an ideal combination that will yield the best network architecture suited specifically for the target application. The process of searching for the best combination, the best hyperparameters combination, is known as hyperparameters search. In practice, this search is often a brute force search, where many possible combinations are trained, evaluated for performance, and compared. As a result of such a search method, hyperparameters search can be quite resource-intensive and often requires large clusters of parallel computing facilities.

Hyperparameters search space is a multidimensional space with each hyperparameter representing a dimension. This space can be normally quite large, often infinite. However, in practice, the search space can be reduced by quantizing and limiting the range of each hyperparameter dimension based on a general understanding of the application. The search is then performed on the grid formed by the quantized hyperparameters. Note that the quantization grid does not have to be linearly spaced. Some applications can utilize logarithmic quantization, which helps to cover a large range for a hyper-parameter without increasing the search samples too much.

The number of search attempts can be further reduced by randomly sampling the grid rather than brute-forcing all elements of the grid. This works on the assumption that a

combination of hyperparameters nearby an ideal combination would still constitute a good choice, even if not the best choice, thereby reducing the search choices significantly.

This degree project used Keras Tuner [22] for hyperparameters search. Due to limited computing resources available for this project, the hyperparameters search space was limited to following the ranges and increments for each hyperparameter as listed in table 4.8. Furthermore, the search was limited to only 25 combinations of hyperparameters which were uniformly sampled from the search grid. Chapter 6 addresses the possibility of increasing the search space by adding adequate computing resources as one of the proposed future works.

Hyperparameter	**Min value**	**Inc.**	**Max value**
LSTM hidden layer size (encoder/decoder)	8	16	512
Latent variable size	8	16	512
Dropout probability (encoder dense layer)	0	0.1	0.5
Dropout probability (decoder dense layer)	0	0.1	0.5

Table 4.8.1: LSTM-VAE hyperparameters search space.

The network was created, trained, and evaluated for each of the 25 randomly selected combinations. The following chapter presents the best hyperparameters combination and discusses the results derived from this search.

4.9 Validation and analysis

The model is first tested with the MNIST dataset to verify that the autoencoder is working correctly as expected. Being a smaller dataset, it also helped speed up the development during model implementation.

The pipeline and the network development were verified in three iterations. The first iteration consisted of validating the pipeline and data preprocessing. The measurement inputs and data prepossessing were validated through data visualization and validation of the results of the intermediate steps in the pipeline. The second iteration consisted of validating the functioning of the LSTM-VAE network as per the design. For this purpose, only a small subset of the battery measurements was utilized. The network training was validated from the learning curve and general observation

of the intermediate outputs. The last development iteration consisted of training the network using one full year of measurements of one battery and analyzing the performance of the network. The performance of the network as a potential anomaly detector was analyzed by studying the learning curve, the reproduction of the inputs, and the reproduction errors.

The results and analysis of the project are described in the following chapter.

Chapter 5

Results and Analysis

5.1 Sensors deployment, data acquisition result

The sensors deployment and the resulting data acquisition were two of the key parts of the degree project. At the end of the project, 100 battery sensors were eventually deployed to nearly all of the batteries in the warehouse fleet. The sensors and the accompanying networking systems streamed data to the online cloud storage for most of the time without interruption. The original goal of the degree project was to deploy the system in a period not taking more than a month. However, due to COVID-19 2020 pandemic, the deployment of the sensors extended over a period of several months.

On the flip side, due to the same pandemic reasons, some sensors managed to acquire data over 12 months, managing to collect over 12GB of original battery measurement data. This provided an additional opportunity to utilize operational data for an entire year. Considering warehouse operation is rather seasonal, having access to data from the entire year was particularly useful during the study of anomaly detection. The following table describes the characteristics of the deployment and the data collected as a result of the deployment:

5.2 LSTM-VAE network development result

The
hyperparameters search resulted in the best selection of the network hyperparameters

Acquisition attribute	Result
Number of sensors deployed	100 batteries.
Number of forklifts monitored	50 approx.
Measurement data per battery	1.38 million samples, 500MB+.
Measurement data for the whole fleet	138 million samples, 12GB+.

Table 5.1.1: Results of sensors deployment and data acquisition.

as listed in the table 5.2 - best within the search space, through random sampling. It can be observed from the search result that it leaned towards the lowest latent variable size and larger (if not the largest) LSTM hidden variable size in the range provided for each hyperparameter.

A large number of LSTM hidden cells can be explained from the need of the network to learn a much more complex model than simply the chemical model of the battery, because the battery's time-series data also record seemingly arbitrary operation of the forklifts and battery charging by the warehouse crews, resulting in different charging and discharging sequences.

Hyperparameter	Best value
LSTM hidden layer size (both encoder and decoder)	312
Latent variable size	8
Dropout probability (encoder dense layer)	0.2
Dropout probability (decoder dense layer)	0.5

Table 5.2.1: LSTM-VAE hyperparameters found through random search within hyperparameters space.

The training progress as seen in figure 5.2.1 is intriguing. Both the training loss and validation loss converge close to each other. This observation indicates a strong bias in the network model and an inability to learn to fully reproduce the training data.

Given that the hidden layer size during the hyperparameters search was leaning towards a larger value of 312 (found within its search range of 8 - 512 cells) and a strong indication of a high bias in its learned behavior as seen in the learning curve in figure 5.2.1, there is a good possibility that the network complexity may have been limited by the search range. There is also the possibility that the search samples did not include higher ranged values because the number of samples chosen was too few (25 in the experiment – limited by the available training resources). There may not have been a sample with a value close to the maximum (512 cells).

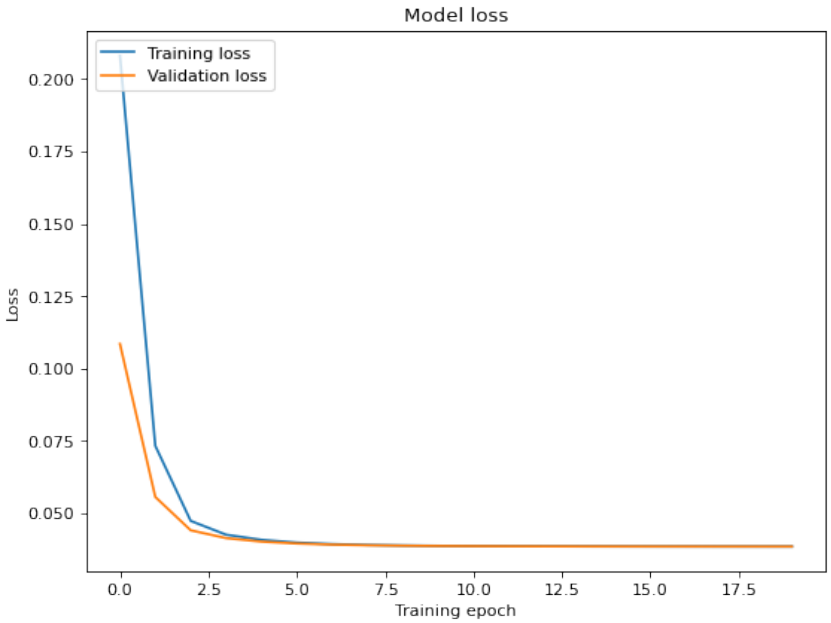


Figure 5.2.1: Result of training the LSTM-VAE network using one year of measurement data acquired from battery #26.

To investigate this possibility, the network was trained again using an arbitrarily high complexity by setting the LSTM hidden layer dimension to 2048 cells. All other hyperparameters were left unchanged. This network constituted a total of 36 million parameters. The result of the training is as seen in figure 5.2.2. The observation of the new learning curve still shows the network continues to hold a high bias. As a result, one could conclude the limitation in the learning comes from factors outside the design of the LSTM-VAE architecture. Further analysis of time-series reproduction is performed in the following section to investigate the nature of the performance of the network.

5.3 Analysis of the time-series reproduction

Reproducibility of the input time-series data is a key element in the functioning of the LSTM-VAE network as an anomaly detector. To evaluate the reproducibility, the trained network was used to infer and reproduce the entire 12 months data. Figure 5.3.1 shows the voltage, current, and temperature reproduction. The three plots on the left-hand side show zoomed-out time-series representing about four days of forklift operation. The blue plot is the original time series, while the orange plot represents reproduction. It can be observed from these plots that the network, as a high-level generalization, mostly learned the means of the input and ignored the low-frequency

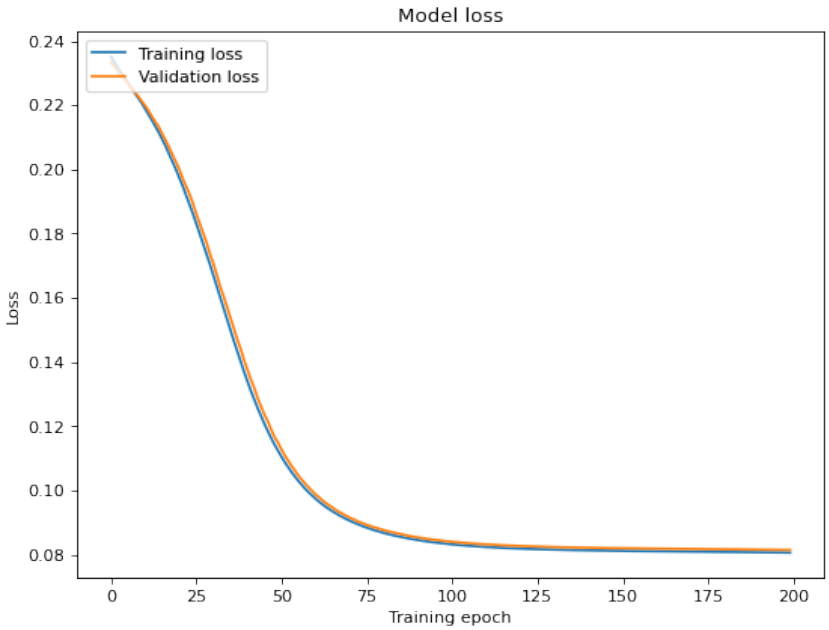


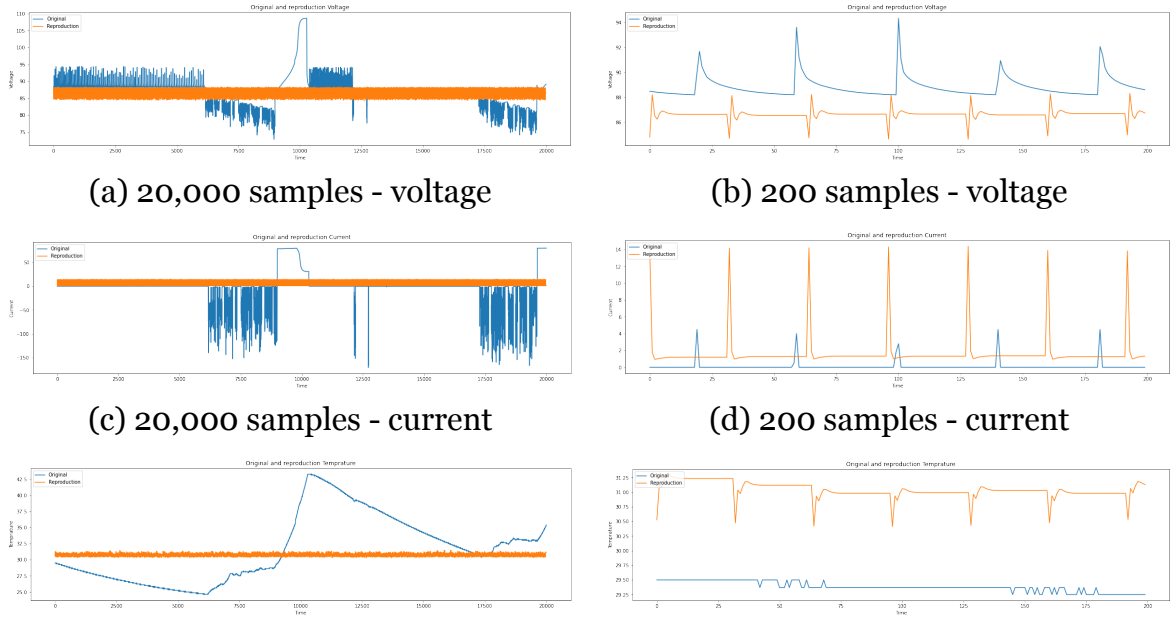
Figure 5.2.2: Result of training the LSTM-VAE network with a high network complexity to analyze the high-bias observation.

variations that run hours in length.

On the other hand, the three plots on the right-hand side show a zoomed-in section representing about an hour of operation. The patterns observed in the original input time-series are periodic top-up charging while the battery is on a charger and it's already fully charged. We observe that the network performs a reasonable attempt to reproduce the high-frequency patterns in the input, albeit with some noticeable time-shift.

We draw a remark from these two observations that the network attempted to learn the reproduction of high-frequency patterns with reasonable success while failing to notice the low-frequency patterns. From the experimental verification with the previous over-parameterized network, we also draw a related remark that the limitation itself does not arise from a bottleneck in the network's architecture.

As a result, one may consider factors that lie outside the architecture itself. Perhaps, the small input record size, 32 samples or roughly 10 minutes, limits the network's ability to capture and learn the multi-records pattern - the aforementioned low-frequency patterns? Those low-frequency patterns run in the range of 1000s of records. It should be noted that the temporal relation between the records is not directly taken into account in the LSTM-VAE architecture. In fact, they are randomly shuffled as part of



(a) 20,000 samples - voltage

(b) 200 samples - voltage

(c) 20,000 samples - current

(d) 200 samples - current

(c) 20,000 samples - temperature

(d) 200 samples - temperature

Figure 5.3.1: Analysis of the time-series reproduction by the LSTM-VAE network

the Stochastic Gradient Decent (SGD) training procedure. With this learning in place, a future experiment could involve training with much larger record size. However, due to a limited compute time and resources during the project, this experiment is left as one of the future works as listed in the next chapter.

5.4 Analysis of anomaly detection by the network

Figure 5.4.1 shows mean squared error between the reproduction and original input of the same samples shown in figure 5.3.1. Anomaly detection is often implemented as a threshold to this error, as described in chapter 2.

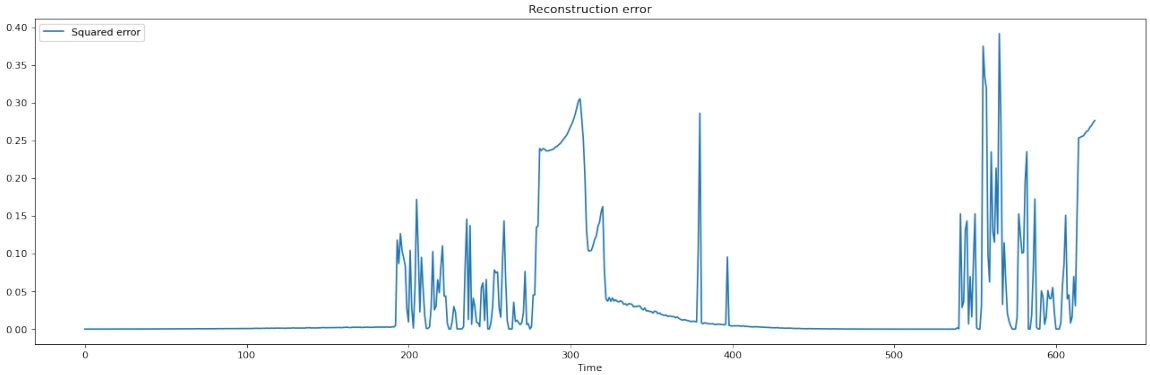


Figure 5.4.1: Reproduction error

The previous section established that the network misses the reproduction of low-frequency changes in the measurement time-series. This can also be observed in the reproduction error in the figure, as expected. The thick areas with high error in the plot correspond to such low-frequency changes. On the other hand, the plot shows significant sections with low error, indicating better reproducibility in those sections. However, there are also high-error sections as indicated by the spikes in the plot. This observation is an obvious manifestation of the previously observed high-bias learning in the network.

For the reproduction error to be used as an anomaly detector in practice, the reproduction error would need to be lower and more consistent. Hence, addressing the high bias of the network and investigating further enhancements in the network model and training structure would be logical next steps. The next chapter, section 6.2, describes further development and studies of the system by (1) evaluating the impact of the input record size, (2) extending the training dataset size to include all 100 batteries, (3) Expanding the hyperparameters search space, and (4) evaluating alternative deep neural network architectures.

Chapter 6

Conclusions

Chapter 1 described that the purpose of the degree project was to develop and evaluate an anomaly detection pipeline utilizing the time-series measurements from battery fleets. Reflecting on this, we see the cumulative results of the whole project presented in the previous section.

The project was implemented following the engineering methodology described in chapter 3, resulting in a successful pilot deployment for the customer, with ample volume in data acquisition, and functional implementation of the deep neural network. All relevant stakeholders were engaged during the development process of the project to ensure the goals of the project are achieved smoothly.

The goals of the project were achieved and described in previous sections to the extent that someone knowledgeable in the field can reproduce them and take the research further in this area. Corresponding to each of the objectives listed in section 1.4, we note that:

1. A customer pilot was started together with Hartwall Oy and collected the battery operational data.
2. The entire network of sensors and communication devices were set up enabling the collection of the raw battery measurement data.
3. The first part of the pilot collected the baseline data.
4. eFleetly battery maintenance tool, charging room monitors, and maintenance.
5. The second part of data representing post-deployment were collected.

6. A deep learning anomaly detector based on LSTM-VAE was designed, developed, and analyzed.
7. Results from the operation of the anomaly detector were analyzed to draw some conclusions and identify limitations in its effectiveness.

Furthermore, the anomaly detector pipeline was developed following engineering best practices. More specifically, by:

1. Utilizing iterative and systematic implementation approach in steps introducing incremental complexity,
2. Analysing network learning to deduce model behaviors,
3. Introducing hyperparameters search to eliminate limitations due resulting from network complexity,
4. Deeper analysis of the learned behaviors, and
5. Based on analysis and evaluation, recommending future work to possibly overcome the limitations.

6.1 Discussion

The project was planned to run for 3 months. However, due to the COVID-19 pandemic in 2020/2021, the project got stretched well beyond a year. Despite this historical disruption, the project was successfully executed resulting in the deployment of the system. The data acquisition succeeded in collecting over 6GB of original battery measurement data. The deep learning network development and training were successful post-data acquisition. A thorough analysis of the behavior of the network was performed. A conclusion was drawn that due to the nature of the time-series in battery measurement, the anomaly detector architecture requires further enhancements to overcome the limitations in reproducing all aspects of the measurement time series. We see that the network is able to reproduce higher frequency features of the time series, but struggled to reproduce low-frequency features. Further investigation was conducted by over-parameterizing the network to eliminate the possibility of a high bias caused by low network complexity. Based on the evaluation of the results and the limitations found, I identify and list various future works in the following section.

6.1.1 Discussion on model validation

From the analysis of the reproduction error in chapter 5, we know that the LSTM-VAE model developed doesn't work for this specific application. The reproduction of the signal should be near perfect in order to be usable as an anomaly detection model. The reproduction error (5.3.1) shows that this is not the case, hence verifying that the model doesn't work as anticipated. Chapter 5 performed an in-depth analysis of reasons for this case and suggested several follow-up steps to perform as additional experiments. It appeared that the development of the model itself was not the issue, which seemed correctly functioning from test data and training performances. But the choice of the architecture seemed to be unsuitable for the task, hence the development require further experiments.

The anomalies in battery operation are rare in the given time frame of the degree project since batteries run for typical 3-7 years before breaking down. The battery data acquired are unlabeled since the operation doesn't really keep track of or follow any known anomalies before the batteries or cells within just die. As a matter of fact, no anomaly has been identified and documented explicitly for battery #26 that was used in the experiment.

For this reason, this system was developed as an unsupervised system due to the lack of labeled data. To help with the customer's situation, this system was developed where it was meant to either:

1. Fully reproduce the measurements, hence detect no anomaly, or
2. Detect a very rare anomaly/outlier. The reproduction error would be mostly flat except for a bump or two somewhere. This should then prompt a physical and electrical field investigation to verify the detected anomaly. This would come as the next level of verification.

Given the unsupervised nature of the system, the only practical way to verify an anomaly is to get to state (2) described above. However, the model never gets past the prerequisite of perfect reproduction of the input measurements.

6.1.2 Discussion on internal evaluation of the model

On the topic of verification of unsupervised anomaly detection, a further investigation yielded a possibility to consider a systematic way to handle it. In addition to physical

investigation post-detection described in the previous section 6.1.1, point (2), an ”internal evaluation” can be performed as well. This excellent paper *On the Internal Evaluation of Unsupervised Outlier Detection* (Marques et al. 2015 [18]) describes a well suited method for this purpose. It relies on the intuition that anomalies are rare and therefore their data separability from the rest provides a means to quantify the evaluation. It depends on implementing a nonlinear classier, specifically ones that rely on maximizing margins (e.g. they proposed: Nonlinear Support Vector Machine (SVM) or Kernel Logistic Regression). The paper considers classifying the detected anomalies from the rest of the data and using the performance of the Receiver Operating Characteristic (ROC) curve as the evaluation metric, i.e. the sharper the curve is, the more separated the anomaly data points are from the rest. As a baseline, they compare the evaluation against the NULL hypothesis.

Their approach sounds fitting to this project, where there are no labeled data and unsupervised training is used for anomaly detection. However, implementing this feels like another thesis project on its own. This certainly seems like a good next step in the project when a working model is later found.

6.1.3 Discussion on the time constraints of the project

This degree project aimed to develop a full system pipeline of the anomaly detection system – starting from real-time sensors, network communication, cloud database, anomaly detection modeled on real data, and all the way to the user interface. The LSTM-VAE model at the end of the pipeline was only one element of the project, albeit the main component.

This degree project ended up being a little too big. There was not enough time available to work on multiple model architectures and the data volume was relatively big. Additionally, the user-interface integration could not be completed either. However, other parts of the data pipeline were implemented (thanks to support from a technician in the field and the eFleetly engineers). For the detection model, it became apparent that many different presumably relevant models or their variations need to be tried before a suitable one is found.

It’s unfortunate that the project could not be completed by finding the right model architecture in time and having the full pipeline in place. As a result, the project is still incomplete. For this reason, the goal of this thesis is to describe everything that

has been performed to develop this system so that the next person can continue and hopefully complete it.

6.2 Future Work

6.2.1 Evaluating the impact of input record size

The analysis in the previous chapter has shown that the network did not learn long-running patterns across the time series. One reason hypothesized is that the record size is too small to account for low-frequency patterns (the slow changes). This is a plausible scenario given that all the records in the input were randomly shuffled before feeding to the network during its training, which loses any inherent time order of the records. To further enhance the capability of the network to learn lower frequency patterns, the record size could be increased, perhaps significantly, such as a day, a week, or even a month.

The longer record sizes will call for more complex networks, with a much higher number of LSTM hidden layer units and a larger latent variable size. Training such networks inevitably requires a more powerful computing platform. Powerful commercial computing resources developed specifically for such heavy-duty machine-learning purposes could be a good choice to consider. Hyperparameters search can also be very resource-intensive, therefore, distributed computing is recommended to lessen the training time.

6.2.2 Extended training data size and length

Due to the large volume of data and limited available hardware resources for the deep learning network training, the project utilized measurement data from only one battery. This project can be further extended by training the network using the data from all 100 sensors.

The project encountered several disruptions while running long training sessions in the Google Colab environment. The Golab platform has restrictions on resource usage, especially for long-running training sessions. As a result, to accomplish training with a larger dataset, dedicated hardware with a state-of-the-art GPU would be necessary to train and evaluate a single combination of hyperparameters uninterrupted for days.

As also mentioned in the previous section, there are commercial services for online GPU instances that can provide a similar dedicated compute instance if an additional budget is available.

The number of epochs for each training round was limited to a small number. However, with the use of dedicated hardware as described above, the number of epochs can be increased to ensure the network gets to learn most optimally, such as by reaching early stopping naturally.

6.2.3 Finer hyperparameter search space

The number of hyperparameters sampled from the hyperparameters space was also considerably limited due to the limited computing time in Google's Colab platform. To succeed in the search for a more optimal hyperparameters combination, the number of samples could be increased sufficiently. Luckily, an individual combination of the hyperparameters can be evaluated independently. As a result, a distributed computing facility with a large number of GPU-enabled computing nodes can be utilized to facilitate more efficient hyperparameters search. Keras Tuner [22] provides the necessary means to achieve such distributed computation if a cluster of GPU-enabled compute instances can be configured for this purpose.

6.2.4 Alternative deep neural network architectures

This degree project has developed and evaluated only an LSTM-VAE architecture for anomaly detection. There are many other deep learning architectures in the literature developed for use with time-series data. Many of them have been highlighted in chapter 2. Future work can also consider evaluating some of them and comparing their performances against each other. Some choices of suitable architectures for further study include architectures based on CNN and Generative Adversarial Network (GAN).

A good example to follow for a CNN-based architecture is Ribeiro et al, 2018 [25]. It proposes an anomaly detection architecture for video surveillance purposes that uses an autoencoder based on CNN. A video stream is a time-series input. Hence, the same architecture, perhaps with small modifications, could be implemented for battery time-series measurements as well.

A GAN-based architecture, on the other hand, is also an interesting choice to evaluate. A GAN network learns to generate probable outputs based on the inputs. A GAN consists of two major components - a generator and a discriminator. The generator learns to generate a realistic output similar to the input, while the discriminator learns to detect generated output and distinguish them from the real inputs. It is the discriminator component that is of interest for an anomaly detector. Once a GAN is trained on the input time-series records, the discriminator could likely be used for detecting anomalous data on new inputs.

6.3 Final Words

The purpose of this thesis, recapping from chapter 1, is *to present the development and analysis of an anomaly detection pipeline so that someone knowledgeable in the area can implement a similar pipeline and further the research in predictive maintenance of battery fleets*.

Keeping the said goal in mind, this thesis has presented all related background topics, tools, and methods, development processes, analysis of the results, and inputs to future work. This thesis presented them in sufficient detail so that other researchers can contribute further in the related field. With this, the author closes the thesis with a warm thank you and appreciation for all the supports he has received during the degree project.

Bibliography

[1] Badeda, J., Huck, M., Sauer, D. U., Kabzinski, J., and Wirth, J. “16 - Basics of lead–acid battery modelling and simulation”. In: *Lead-Acid Batteries for Future Automobiles*. Ed. by Jürgen Garche, Eckhard Karden, Patrick T. Moseley, and David A. J. Rand. Amsterdam: Elsevier, 2017, pp. 463–507. ISBN: 978-0-444-63700-0. DOI: https://doi.org/10.1016/B978-0-444-63700-0.00016-7. URL: https://www.sciencedirect.com/science/article/pii/B9780444637000000167.

[2] Bishop, Christopher M. *Pattern recognition and machine learning*. en. Information science and statistics. New York: Springer, 2006. ISBN: 978-0-387-31073-2.

[3] Boden, D. P. “Comparison of methods for adding expander to lead-acid battery plates—advantages and disadvantages”. en. In: *Journal of Power Sources*. Proceedings of the Tenth Asian Battery Conference 133.1 (May 2004), pp. 47–51. ISSN: 0378-7753. DOI: 10.1016/j.jpowsour.2003.12.006. URL: https://www.sciencedirect.com/science/article/pii/S0378775303012011 (visited on 08/19/2021).

[4] Cho, Kyunghyun, Merrienboer, Bart van, Gulcehre, Caglar, Bahdanau, Dzmitry, Bougares, Fethi, Schwenk, Holger, and Bengio, Yoshua. “Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation”. In: *arXiv:1406.1078 [cs, stat]* (Sept. 2014). arXiv: 1406.1078. URL: http://arxiv.org/abs/1406.1078 (visited on 08/16/2021).

[5] Culpin, B. and Rand, David. “Failure modes of lead/acid batteries”. In: *Journal of Power Sources* 36 (Dec. 1991), pp. 415–438. DOI: 10.1016/0378-7753(91)80069-A.

[6] Dive, Research. *Forklift Battery Market, by Type (Lead–Acid, Lithium ion (Li-ion), and Others), by Application (Construction, Manufacturing, Warehouses, Retail and Wholesale Stores, and Others), Regional Analysis (North America, Europe, Asia-Pacific, LAMEA), Global Opportunity Analysis and Industry Forecast, 2019–2026*. 2020. URL: https://www.researchdive.com/70/forklift-battery-market (visited on 10/25/2021).

[7] EugenioTL. *VAE Basic*. 2021. URL: https://commons.wikimedia.org/wiki/File:VAE_Basic.png (visited on 10/24/2021).

[8] Guo, Yifan, Liao, Weixian, Wang, Qianlong, Yu, Lixing, Ji, Tianxi, and Li, Pan. "Multidimensional Time Series Anomaly Detection: A GRU-based Gaussian Mixture Variational Autoencoder Approach". en. In: *Asian Conference on Machine Learning*. ISSN: 2640-3498. PMLR, Nov. 2018, pp. 97–112. URL: http://proceedings.mlr.press/v95/guo18a.html (visited on 08/18/2021).

[9] Hochreiter, Sepp and Schmidhuber, Jürgen. "Long Short-term Memory". In: *Neural computation* 9 (Dec. 1997), pp. 1735–80. DOI: 10.1162/neco.1997.9.8.1735.

[10] Hsu, Wei-Ning, Zhang, Yu, and Glass, James. "Unsupervised Domain Adaptation for Robust Speech Recognition via Variational Autoencoder-Based Data Augmentation". In: (July 2017).

[11] Kingma, Diederik P. and Welling, Max. "An Introduction to Variational Autoencoders". In: *Foundations and Trends® in Machine Learning* 12.4 (2019). arXiv: 1906.02691, pp. 307–392. ISSN: 1935-8237, 1935-8245. DOI: 10.1561/2200000056. URL: http://arxiv.org/abs/1906.02691 (visited on 04/05/2021).

[12] Kingma, Diederik P. and Welling, Max. "Auto-Encoding Variational Bayes". In: *arXiv:1312.6114 [cs, stat]* (May 2014). arXiv: 1312.6114. URL: http://arxiv.org/abs/1312.6114 (visited on 08/06/2021).

[13] Kremer, Stefan C. and Kolen, John F. *Field Guide to Dynamical Recurrent Networks*. 1st. Wiley-IEEE Press, 2001. ISBN: 0780353692.

[14] Kurzweil, P. "Gaston Planté and his invention of the lead–acid battery—The genesis of the first practical rechargeable battery". en. In: *Journal of Power Sources*. Celebration of lead-acid 150 years 195.14 (July 2010), pp. 4424–4434. ISSN: 0378-7753. DOI: 10.1016/j.jpowsour.2009.12.126. URL: https://

www.sciencedirect.com/science/article/pii/S0378775310000546 (visited on 04/29/2021).

[15] LeCun, Yann, Cortes, Corinna, and Burges, Christopher J.C. *MNIST handwritten digit database, Yann LeCun, Corinna Cortes and Chris Burges*. URL: http://yann.lecun.com/exdb/mnist/ (visited on 09/03/2021).

[16] Maleki, Sepehr, Maleki, Sasan, and Jennings, Nicholas R. "Unsupervised anomaly detection with LSTM autoencoders using statistical data-filtering". en. In: *Applied Soft Computing* 108 (Sept. 2021), p. 107443. ISSN: 1568-4946. DOI: 10.1016/j.asoc.2021.107443. URL: https://www.sciencedirect.com/science/article/pii/S1568494621003665 (visited on 08/13/2021).

[17] Malhotra, Pankaj, Vig, Lovekesh, Shroff, Gautam, and Agarwal, Puneet. "Long Short Term Memory Networks for Anomaly Detection in Time Series". In: Apr. 2015.

[18] Marques, Henrique O., Campello, Ricardo J. G. B., Zimek, Arthur, and Sander, Jörg. "On the internal evaluation of unsupervised outlier detection". In: *Proceedings of the 27th International Conference on Scientific and Statistical Database Management*. SSDBM '15. New York, NY, USA: Association for Computing Machinery, June 2015, pp. 1–12. ISBN: 978-1-4503-3709-0. DOI: 10.1145/2791347.2791352. URL: https://doi.org/10.1145/2791347.2791352 (visited on 11/02/2021).

[19] Massi, Michela. *Autoencoder schema*. 2019. URL: https://commons.wikimedia.org/wiki/File:Autoencoder_schema.png (visited on 10/24/2021).

[20] Morales-
Forero, A. and Bassetto, S. "Case Study: A Semi-Supervised Methodology for Anomaly Detection and Diagnosis". In: *2019 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*. ISSN: 2157-362X. Dec. 2019, pp. 1031–1037. DOI: 10.1109/IEEM44572.2019.8978509.

[21] Nalisnick, Eric, Matsukawa, Akihiro, Teh, Yee Whye, Gorur, Dilan, and Lakshminarayanan, Balaji. "Do Deep Generative Models Know What They Don't Know?" In: *arXiv:1810.09136 [cs, stat]* (Feb. 2019). arXiv: 1810.09136. URL: http://arxiv.org/abs/1810.09136 (visited on 08/13/2021).

[22] O'Malley, Tom, Bursztein, Elie, Long, James, Chollet, François, Jin, Haifeng, Invernizzi, Luca, et al. *KerasTuner*. https://github.com/keras-team/keras-tuner. 2019.

[23] Papazov, G., Pavlov, D., and Monahov, B. "Influence of temperature on expander stability and on the cycle life of negative plates". en. In: *Journal of Power Sources*. Proceedings of the International Conference on Lead-Acid Batteries, LABAT '02 113.2 (Jan. 2003), pp. 335–344. ISSN: 0378-7753. DOI: 10.1016/S0378-7753(02)00546-3. URL: https://www.sciencedirect.com/science/article/pii/S0378775302005463 (visited on 08/19/2021).

[24] Park, Daehyung, Hoshi, Yuuna, and Kemp, Charles C. "A Multimodal Anomaly Detector for Robot-Assisted Feeding Using an LSTM-Based Variational Autoencoder". In: *IEEE Robotics and Automation Letters* 3.3 (July 2018). Conference Name: IEEE Robotics and Automation Letters, pp. 1544–1551. ISSN: 2377-3766. DOI: 10.1109/LRA.2018.2801475.

[25] Ribeiro, Manassés, Lazzaretti, André Eugênio, and Lopes, Heitor Silvério. "A study of deep convolutional auto-encoders for anomaly detection in videos". en. In: *Pattern Recognition Letters*. Machine Learning and Applications in Artificial Intelligence 105 (Apr. 2018), pp. 13–22. ISSN: 0167-8655. DOI: 10.1016/j.patrec.2017.07.016. URL: https://www.sciencedirect.com/science/article/pii/S0167865517302489 (visited on 08/13/2021).

[26] Sakurada, Mayu and Yairi, Takehisa. "Anomaly Detection Using Autoencoders with Nonlinear Dimensionality Reduction". In: Dec. 2014, pp. 4–11. DOI: 10.1145/2689746.2689747.

[27] Schmidt-Rohr, Klaus. "How Batteries Store and Release Energy: Explaining Basic Electrochemistry". In: *Journal of Chemical Education* 95.10 (Oct. 2018). Publisher: American Chemical Society, pp. 1801–1810. ISSN: 0021-9584. DOI: 10.1021/acs.jchemed.8b00479. URL: https://doi.org/10.1021/acs.jchemed.8b00479 (visited on 08/20/2021).

[28] Xiao, Zhisheng, Yan, Qing, and Amit, Yali. "Likelihood Regret: An Out-of-Distribution Detection Score For Variational Auto-encoder". In: *arXiv:2003.02977 [cs, stat]* (Oct. 2020). arXiv: 2003.02977. URL: http://arxiv.org/abs/2003.02977 (visited on 08/13/2021).

[29] Zhou, Chong and Paffenroth, Randy. “Anomaly Detection with Robust Deep Autoencoders”. In: Aug. 2017, pp. 665–674. DOI: 10.1145/3097983.3098052.

Appendix A

LSTM-VAE network layers

```
Model: "encoder"
_________________________________________________________________________________
Layer (type)                 Output Shape              Param #     Connected to
=================================================================================
input (InputLayer)           [(None, 32, 3)]           0

_________________________________________________________________________________
lstm_encoder (GRU)           (None, 2048)              12613632    input[0][0]

_________________________________________________________________________________
z_mean (Dense)               (None, 8)                 16392       lstm_encoder[0][0]

_________________________________________________________________________________
z_log_var (Dense)            (None, 8)                 16392       lstm_encoder[0][0]

_________________________________________________________________________________
z (Lambda)                   (None, 8)                 0           z_mean[0][0]
                                                                   z_log_var[0][0]
=================================================================================
Total params: 12,646,416
Trainable params: 12,646,416
Non-trainable params: 0
_________________________________________________________________________________
Model: "lstm_vae"
_________________________________________________________________________________
Layer (type)                 Output Shape              Param #     Connected to
=================================================================================
input (InputLayer)           [(None, 32, 3)]           0
```

lstm_encoder (GRU)	(None, 2048)	12613632	input[0][0]
z_mean (Dense)	(None, 8)	16392	lstm_encoder[0][0]
z_log_var (Dense)	(None, 8)	16392	lstm_encoder[0][0]
z (Lambda)	(None, 8)	0	z_mean[0][0] z_log_var[0][0]
repeat_vector (RepeatVector)	(None, 32, 8)	0	z[0][0]
lstm_decoder (GRU)	(None, 32, 2048)	12644352	repeat_vector[0][0]
time_distributed (TimeDist)	(None, 32, 3)	6147	lstm_decoder[0][0]

Total params: 25,296,915

Trainable params: 25,296,915

Non-trainable params: 0

Appendix B

Project plan

Task	Description
Deploy sensors	Order and deploy the battery sensors and network equipment at the customer site.
Setup data capture	The data arriving from the sensors are captured in the cloud.
Research battery technology	Research the background of lead-acid battery technology, failure modes, and effect on operation.
Deploy eFleetly	Deploy the eFleetly service at customer site (not directly relevant to this project, but needed for data collection).
Connect sensors and network infra-structure	The battery sensors are installed on each battery at the customer site, the communication network infrastructure is set up and data upload to the cloud is established.
Collect post-eFleetly data	Post eFleetly deployment, the battery fleet should receive significantly reduced abuses.
Research deep learning technology	Research relevant deep-learning technologies to implement the right anomaly detector.
Acquire training data	eFleetly battery data arrive at a central location in a database. However, it requires exporting in the right format to acquire them.
Develop data pre-processing	Implement a data pipeline to clean up, format, and organize the data in a suitable form for the neural network to process.
Develop LSTM-VAE neural network	Develop the neural network using intermediate datasets in multiple iterations.
Train neural network	Train the neural network in iterations and perform hyperparameters search.
Analyze results	Analyze the reproduction of battery measurement data and reproduction error to establish anomaly detection function.
Write thesis	Document all relevant background, engineering methodologies, development process, training process, and results analysis of the project.

Table B.0.1: Project tasks list planned at the beginning of the project.