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

A deep learning based anomaly detection pipeline for battery fleets

Nabakumar Singh Khongbantabam

KTH ROYAL INSTITUTE OF TECHNOLOGY
ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

Author

Nabakumar Singh Khongbantabam <nskh@kth.se>
Master's Program in ICT innovation, Autonomous Systems major
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 proposes a deep learning anomaly detection pipeline to detect possible anomalies during the operation of a fleet of batteries and presents its development and evaluation. The pipeline employs sensors that connect to each battery in the fleet to remotely collect real-time measurements of their operating characteristics, such as voltage, current, and temperature.

The deep learning based time-series anomaly detection model was developed using Variational Autoencoder (VAE) architecture that utilizes either Long Short-Term Memory (LSTM) or, its cousin, Gated Recurrent Unit (GRU) as the encoder and the decoder networks (LSTM-VAE and GRU-VAE). Both variants were evaluated against three well-known conventional anomaly detection algorithms - Isolation Nearest Neighbour (iNNE), Isolation Forest (iForest), and kth Nearest Neighbour (k-NN) algorithms.

All five models were trained using two variations in the training dataset (full-year dataset and partial recent dataset), producing a total of 10 different model variants. The models were trained using the unsupervised method and the results were evaluated using a test dataset consisting of a few known anomaly days in the past operation of the customer's battery fleet.

The results demonstrated that k-NN and GRU-VAE performed close to each other, outperforming the rest of the models with a notable margin. LSTM-VAE and iForest performed moderately, while the iNNE and iForest variant trained with the full dataset, performed the worst in the evaluation. A general observation also reveals that limiting the training dataset to only a recent period produces better results nearly consistently across all models.

Keywords

Forklift batteries, Battery sensors, Data pipeline, Predictive maintenance, Anomaly detection, Deep learning, Battery failure prediction, Time-series, Variational autoencoder, Long short-term memory, LSTM, Gated recurrent unit, GRU, Isolation nearest neighbor, iNNE, Isolation forest, iForest, kth nearest neighbor, k-NN.

Abstract

Detta examensarbete föreslår en pipeline för djupinlärning av avvikelser för att upptäcka möjliga anomalier under driften av en flotta av batterier och presenterar dess utveckling och utvärdering. Rörledningen använder sensorer som ansluter till varje batteri i flottan för att på distans samla in realtidsmätningar av deras driftsegenskaper, såsom spänning, ström och temperatur.

Den djupinlärningsbaserade tidsserieanomalidetekteringsmodellen utvecklades med VAE-arkitektur som använder antingen LSTM eller, dess kusin, GRU som kodare och avkodarnätverk (LSTM-VAE och GRU) -VAE). Båda varianterna utvärderades mot tre välkända konventionella anomalidetekteringsalgoritmer - iNNE, iForest och k-NN algoritmer.

Alla fem modellerna tränades med hjälp av två varianter av träningsdatauppsättningen (helårsdatauppsättning och delvis färsk datauppsättning), vilket producerade totalt 10 olika modellvarianter. Modellerna tränades med den oövervakade metoden och resultaten utvärderades med hjälp av en testdatauppsättning bestående av några kända anomalidagar under tidigare drift av kundens batteriflotta.

Resultaten visade att k-NN och GRU-VAE presterade nära varandra och överträffade resten av modellerna med en anmärkningsvärd marginal. LSTM-VAE och iForest presterade måttligt, medan varianten iNNE och iForest tränade med hela datasetet presterade sämst i utvärderingen. En allmän observation avslöjar också att en begränsning av träningsdatauppsättningen till endast en ny period ger bättre resultat nästan konsekvent över alla modeller.

Nyckelord

Gaffeltruckbatterier, Batterisensorer, Datapipeline, Prediktivt underhåll, Avvikelsedetektering, Deep learning, Batterifelsprediktion, Tidsserier, Variationsautokodare, Långt korttidsminne, LSTM, Gated recurrent unit, GRU, Isolation närmaste granne, iNNE, Isolation skog, iForest, kth närmaste granne, k-NN.

Acknowledgements

I 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 also like to 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 the 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 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
GAN Generative Adversarial Network
LSTM-VAE Long Short-Term Memory Variational Autoencoder
GRU-VAE Gated Recurrent Unit Variational Autoencoder
iNNE Isolation Nearest Neighbour
iForest Isolation Forest
k-NN kth Nearest Neighbour
ELBO Evidence Lower Bound
KL Kullback–Leibler
RNN Recurrent Neural Network
GRU Gated Recurrent Unit
CNN Convolutional Neural Network
Seq2Seq Sequence to Sequence
OOD Out Of Distribution
SVM Support Vector Machine
ROC Receiver Operating Characteristic
AUC Area Under the Curve

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 [8].

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 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 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 cost-saving measure for the business.

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

This project explores and evaluates 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 and the research question

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. When the batteries operate normally, the said variables change consistently, 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 detectors in the project were developed using the deep-learning-based architectures described in Chapter 4.

This project asks the research question if deep-learning models implemented using Long Short-Term Memory Variational Autoencoder (LSTM-VAE) or Gated Recurrent Unit Variational Autoencoder (GRU-VAE) architectures could be a viable method of anomaly detection for a battery fleet operation.

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

1.3 Purpose

The purpose of the degree project was to develop and evaluate a deep-learning anomaly detection pipeline utilizing the time-series measurements from a customer's battery fleet. The anomaly detection pipeline was implemented using VAE neural network which takes either LSTM or GRU as the encoder and decoder networks.

The purpose of this thesis is to present the development and evaluation 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 deep-learning-based anomaly detectors based on LSTM-VAE and GRU-VAE architectures, and the evaluation results against some well-known anomaly detection algorithms, namely iNNE, iForest, and iNNE.

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 three months.
4. Deploy eFleetly battery maintenance tools (charging room monitors and maintenance user interface)
5. Collect post-deployment data.
6. Design, develop and train the deep-learning-based anomaly detectors using LSTM-VAE and GRU-VAE architectures.
7. Develop the iNNE, iForest, and k-NN anomaly detection algorithms.
8. Analyse and evaluate the performance of the anomaly detectors against each other using the evaluation metrics identified in subsection 1.6 and chapter 3.

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 in a period. Thereby, reducing the environmental impact caused by the customer's operations.

1.6 Methodology

The project follows the engineering method proposed in *"The Electrical and Computer Engineering Design Handbook"*, a book by students of Tufts University as a collaborative effort to design and develop the anomaly detection pipeline, and follows a quantitative experimental method to evaluate the performance of the deep learning anomaly detection methods by answering the research question posed in the section 1.2.

The quantitative method used to evaluate the performance of the anomaly detection methods consists of training and evaluating the GRU-VAE and LSTM-VAE models and three other existing anomaly detection algorithms (iNNE, iForest, and k-NN) by computing and comparing their Area Under the Curve (AUC) and Receiver Operating Characteristic (ROC) plots when tested against a few known anomalies.

All five models were trained using two different training datasets, one utilizing the full year's measurements dataset and the other utilizing only a recent subset of the measurements. Hence, a total of 10 experiments were conducted, evaluating the 10 model variants.

Chapter 3 describes the methodologies of the pipeline development, experiments, and evaluation 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 the 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 is a bit of an open task since anomalies are not explicitly defined upfront. Hence, comprehensive labeled training data are not generally available. The anomaly events used in the evaluation of the models are rather limited in number as well, since anomalies, by definition do not occur very frequently. Nor is the training

dataset free of contamination. Some actual anomalies that haven't been identified likely got included in the training data.

As a result, the interpretation of the results should take these shortfalls into account. The evaluation is most relevant as a relative comparison method rather than an absolute measurement of the performance of the anomaly detection task itself.

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 evaluation of the results.

Chapter 4 ”Work” describes the data pipeline development and the system architecture design, development of the LSTM-VAE and GRU-VAE architectures, development of iNNE, iForest, and k-NN algorithms, mathematical descriptions, their training, and evaluation.

Chapter 5 ”Results” analyses the functioning of the pipeline, functioning and training of anomaly detection methods, and their evaluation results.

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 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 [20]. 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.

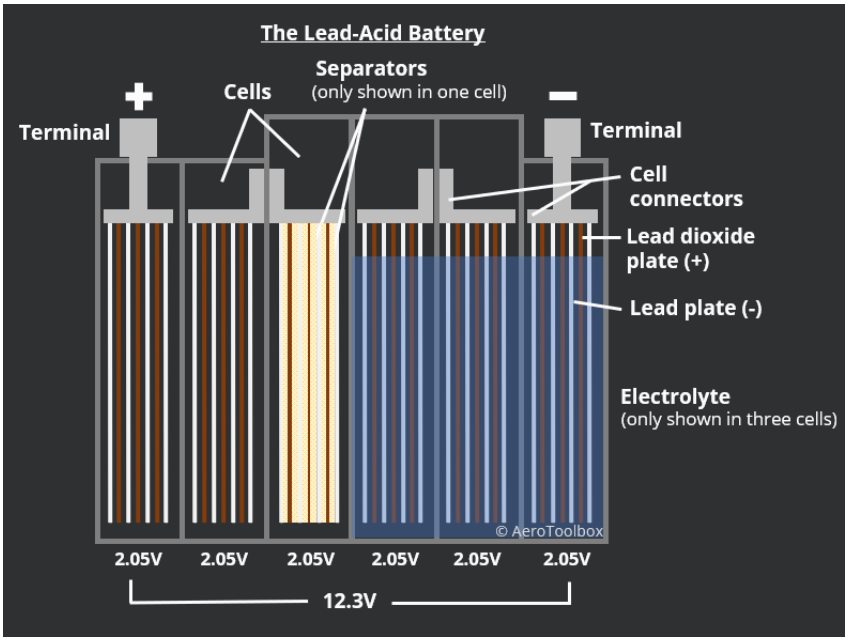


Figure 2.1.1: Lead-acid battery construction

Source: AeroToolbox.com

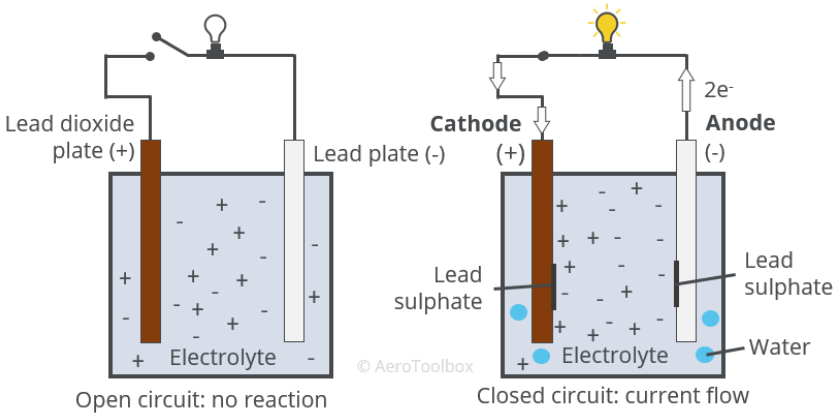
Figure 2.1.1 shows the inside of a lead-acid battery. Each cell is physically isolated and contains positive plates, negative plates, separators, and electrolytes. During a fully discharged state or construction stage (before first charging), the positive plates are made of lead dioxide (PbO_2), but 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-}) [35]. 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 [35].



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 [35].

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 [7].

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 [7].

Sulphuric acid is either consumed during the discharge process, while 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 [7].

2.2.2 Degradation of negative plates

Negative plates have 'expander' materials added to them to improve their performance and life [4]. 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 [7].

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 [31].

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 [7].

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

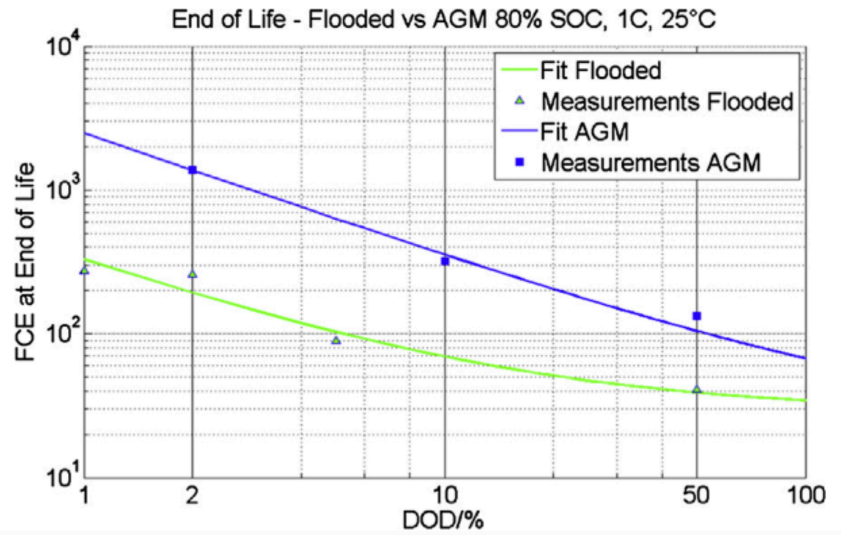


Figure 2.4.1: Impact of dept-of-discharge on the battery life-time.

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: Anomaly detection for batteries

Literature has some related research on machine learning-based anomaly detection methods for batteries.

Garg's master's thesis, 2017 [11], investigated several unsupervised clustering methods for anomaly detection, such as DBSCAN (Density-based spatial clustering of applications with noise), k-Means, Meanshift, Agglomerative and Spectral algorithms. The thesis draws the conclusion that DBSCAN performed the best.

Zhao et al., 2021 [37], evaluated several conventional anomaly detection algorithms applied to aircraft lead-acid battery, including the same three algorithms evaluated in this thesis - iNNE, iForest and k-NN.

Li et al., 2021 [21], propose a clustering-based anomaly detection for electric-vehicle and storage batteries. It primarily focuses on using only their thermal performances for the unsupervised clustering method.

The literature appears to have sparse results when it comes to using deep learning methods for the purpose of anomaly detection for batteries. Li et al., 2019 [22], describe a deep learning method for anomaly detection for lithium-ion batteries (note: this thesis uses lead-acid batteries). They propose a Deep Belief Network to implement an anomaly detection for li-ion spacecraft storage batteries.

2.6 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), Long Short-Term Memory (LSTM) cells, and Gated Recurrent Unit (GRU) cells.

2.6.1 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 2.6.1 shows a general architecture of an autoencoder. The network essentially learns to map higher-dimensional space into a smaller dimensional space.

Mathematically, autoencoders implement the following transformation:

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

Where,

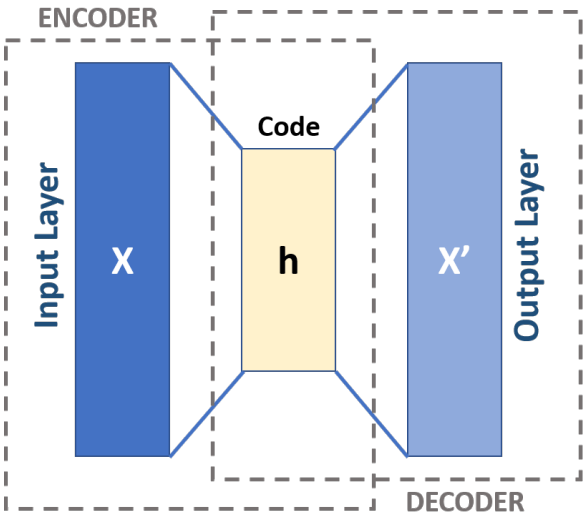


Figure 2.6.1: An illustration of Autoencoder architecture [27]

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:

$$\min_{D,E} ||X' - X||$$

Sakurada and Yairi (2014) demonstrate that autoencoders are able to detect subtle anomalies in which linear PCA fails [34]. 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 [38]. The author names the network ”Robust Deep Autoencoder” (RDA).

2.6.2 Variational Autoencoder (VAE)

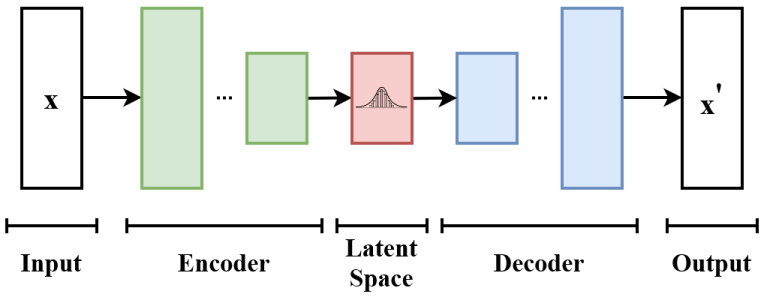


Figure 2.6.2: A simple illustration of Variational Autoencoder architecture [9]. The encoder output is a latent distribution represented by a normal distribution specified by a mean and a variance pair.

VAE is illustrated in the figure 2.6.2. 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 [17]. 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 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*

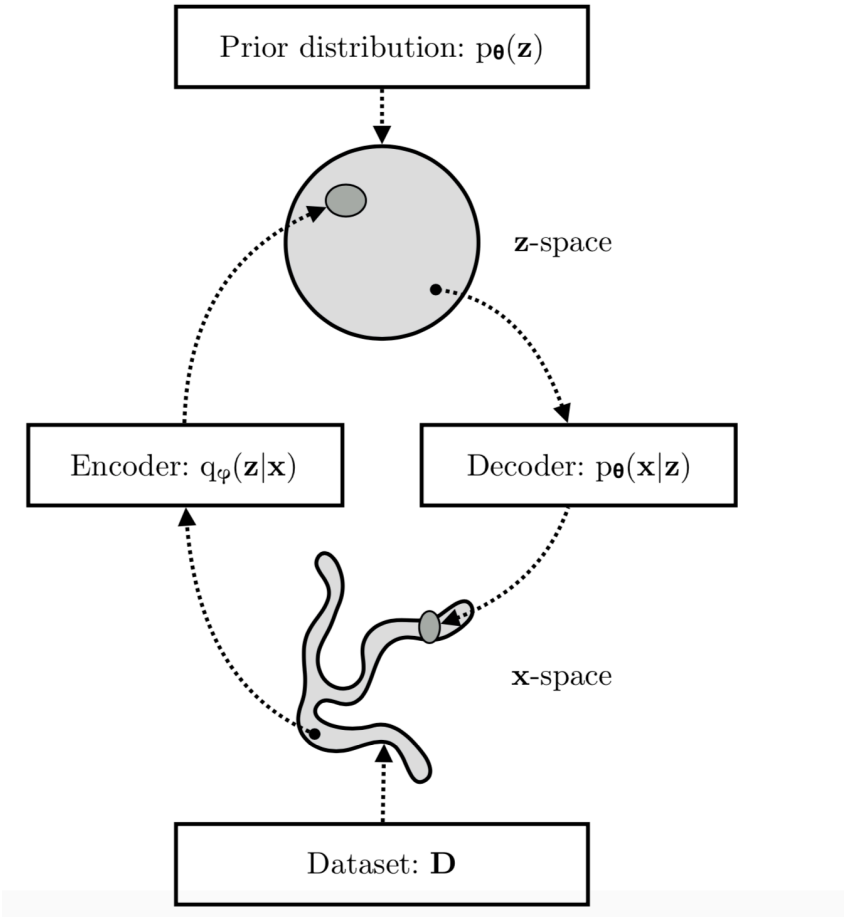


Figure 2.6.3: Variational Autoencoder process, [17]

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 2.6.3. For further detailed reading on VAE, see Kingma and Wellings (2014 [18] and 2019 [17]).

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 [18]. 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 [3]:

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

However, the above integral is also intractable. Instead, we notice that the log-

likelihood of a single sample x^i evaluates as follows [18]:

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

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 [18]. 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 [18]:

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

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.

2.6.3 Long Short-Term Memory (LSTM) cell

The VAE model described earlier can be designed to be temporal aware by utilizing an 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, 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.

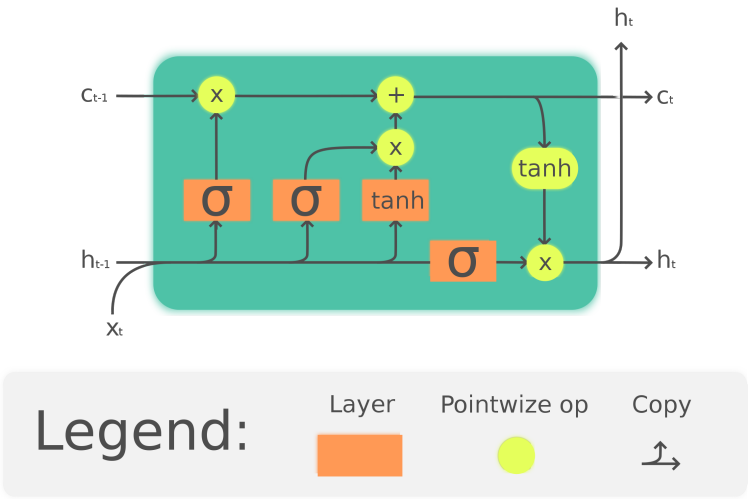


Figure 2.6.4: Long Short-Term Memory Unit [19]

Source: The diagram is courtesy of Guillaume Chevalier, 2018

However, with the advent of LSTM (Sepp Hochreiter, Jürgen Schmidhuber, 1997 [15]), 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 2.6.4 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 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 [6].

There are other variations to LSTM. For example, Morales-Forero1 and S. Bassetto, 2019 [28], 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 [33], proposes an anomaly detection architecture for video surveillance purposes using an autoencoder based on Convolutional Neural Network (CNN).

2.6.4 Gated Recurrent Unit (GRU) cell

GRU is a lot similar to LSTM in function. Cho et al., 2014 [6], originally proposed the first variant of GRU as a simpler network with fewer parameters compared to LSTM and additional consisting of a forget gate first proposed as an improvement to LSTM by Gers et al., 1999 [12]. Several variations of GRU have been proposed in the literature over time, from more complex fully-gated unit to minimally-gated unit (Heck and Salem, 2017 [14]). Figure 2.6.5 shows the fully gated variant of GRU, which is also the variant used in this project.

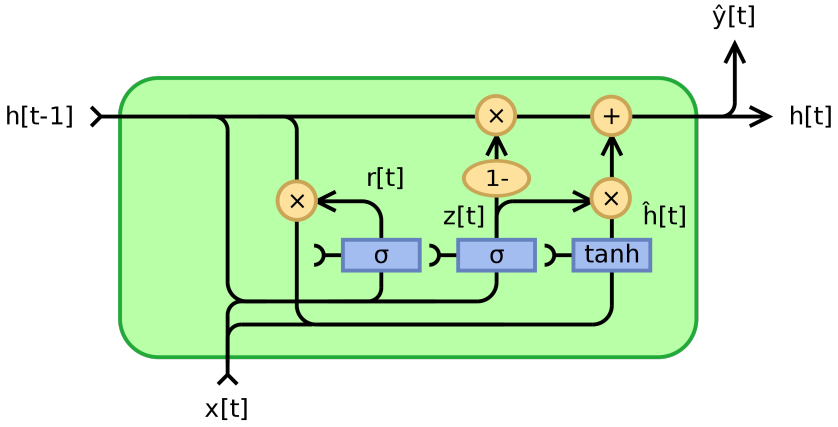


Figure 2.6.5: Gated Recurrent Unit (GRU) network architecture

Source: The diagram is courtesy of Jeblad, Wikipedia, 2018

2.7 Related work: Anomaly detection

This degree project developed a time-series anomaly detection system utilizing previously described VAE, LSTM, and GRU 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, both utilizing deep neural networks and conventional algorithms. The section then further describes the proposed architecture that combines VAE and LSTM / GRU.

2.7.1 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 2.7.1).

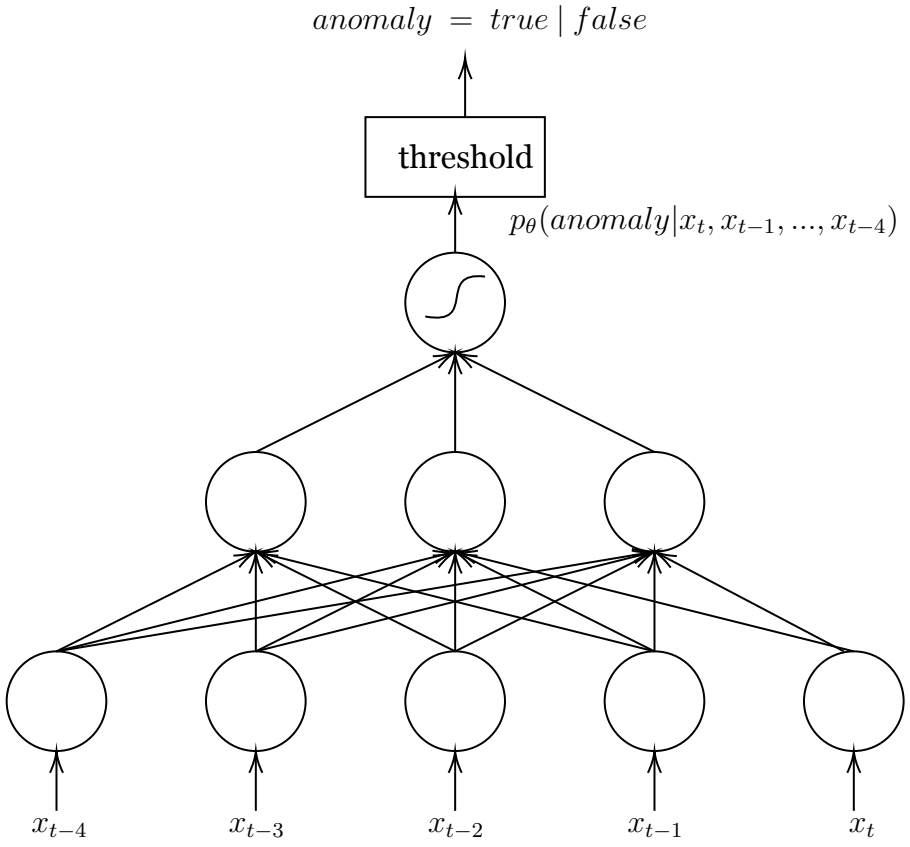


Figure 2.7.1: An example feedforward binary classifier neural network for anomaly detection. The network is trained using known anomalous data and predicts the probability of a given new input being an anomaly during inference.

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.

2.7.2 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 2.7.2. There exists a variety of deep neural network autoencoders which have been adapted for anomaly detection purposes.

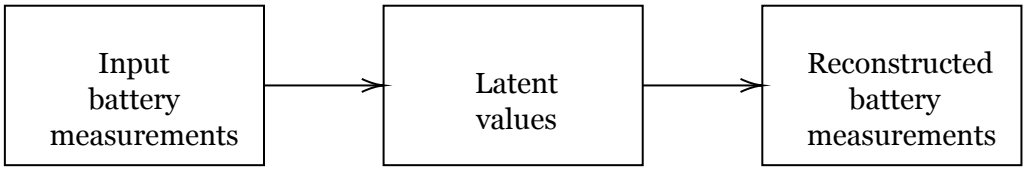


Figure 2.7.2: Reconstruction-based autoencoder.

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.

2.7.3 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 2.6.2 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.

2.7.4 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 [25] [24] [32] [16], GRU [13], or a similar unit as an architecture of choice for a time-series anomaly detector.

Malhotra et al., 2015 [25], 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 2.7.1.

Maleki et al., 2021 [24], 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 [24].

2.7.5 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 propose an anomaly detector for this application that is 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 [32] utilize it in a robot-assisted feeding system. Hsu et al. 2017 [16] also utilize LSTM in a VAE architecture for the purpose of speech recognition. Hsu et al. 2017 [16] implementation is the closest architecture that this project would be utilizing.

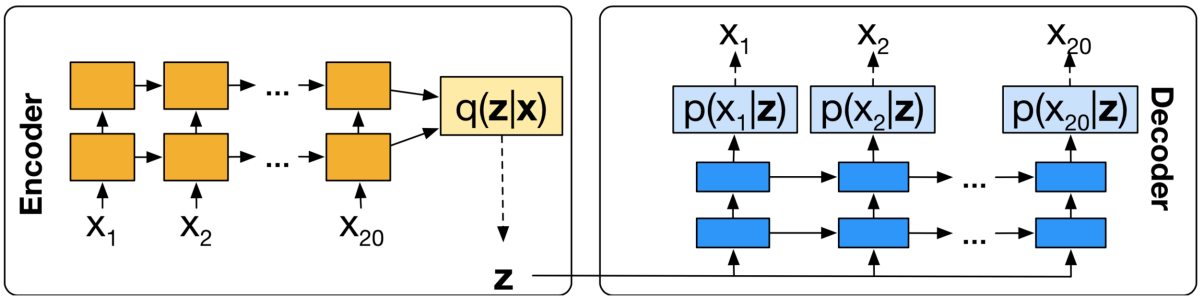


Figure 2.7.3: Variational Autoencoder (VAE) using Long Short-Term Memory (LSTM) units

Source: Hsu et al. 2017 [16]

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 2.7.3.

2.7.6 GRU and VAE based anomaly detection

The proposed GRU-VAE anomaly detection architecture is similar to the LSTM-VAE architecture described in the previous section, except the LSTM cells are replaced with GRU cells. They otherwise function the same way.

2.7.7 Anomaly score from reconstruction error

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, or anomaly, when the score is above the threshold.

2.7.8 Isolation-based anomaly detection

Isolation-based anomaly detection relies on the intuition that anomalous data points are located relatively far from the rest of the normal data points in the input space. Several well-known isolation-based anomaly detection algorithms exist that one way or the other exploit this intuition. Three of the most popular ones, namely k-NN, iForest, and iNNE are developed in this project for evaluation and comparison purposes.

k^{th} Nearest Neighbour (k-NN)

k-NN is the simplest of all isolation-based algorithms discussed in this thesis. There is no training involved in k-NN per se. The entire training dataset is used during inference every time. The anomaly score of a test sample x is simply its distance from its k^{th} neighbor in the training dataset. The longer the distance, the higher the score. The value of k is often found by trial-error.

The time complexity of k-NN is $\mathcal{O}(NM)$ in detecting N input samples using a training dataset of size M. Despite being the simplest method of all the methods discussed in this thesis, k-NN has the worst time complexity. As a result, it is often considered unsuitable for use cases involving very large datasets.

Isolation Nearest Neighbour (iNNE)

iNNE [2] method consists of an ensemble of t models. Each model consists of a set, S, of hyperspheres, centered around n points which are randomly selected subsamples of the training dataset. Each hypersphere centered at their corresponding data point sample is defined as the largest hypersphere which only contains the said data point sample. This in practice means the hypersphere has a radius exactly the distance of its nearest neighbor in the set of subsamples.

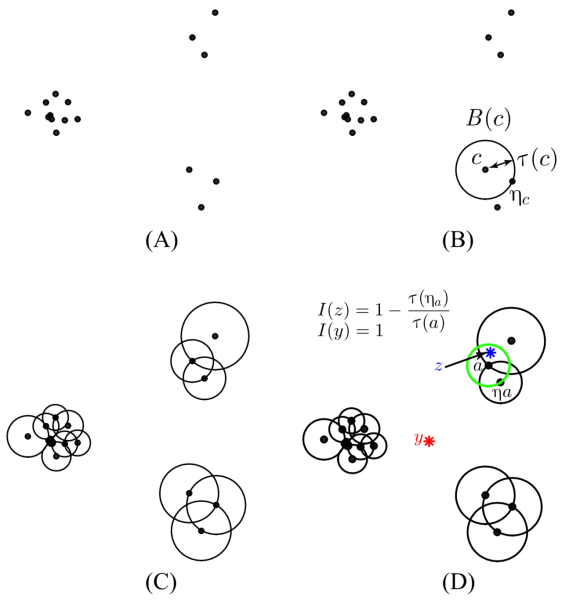


Figure 2.7.4: Isolation Nearest Neighbour Ensemble anomaly detection algorithm

Source: Bandaragoda et al. 2018 [2]

The anomaly score in the range of 0 - 1 for the test sample x, roughly resembling a probability of it being an anomaly, is taken to be the average of its isolation scores obtained from each of the t models in the ensemble.

$$\bar{I}(x) = \frac{1}{t}\sum_{i=1}^{t} I_i(x) \tag{2.10}$$

For each model S in the ensemble, the isolation score I of the test sample x, in the input space R^d, is determined by first searching for the center of the nearest hypersphere, $cnn(x)$, in S that fully contains the test point x and then finding the center of the nearest hypersphere, $\eta_{cnn(x)}$, to the previously found hypersphere center. The ratio of the radii

of the two hyperspheres centered at $\tau(cnn(x))$ and $\tau(\eta_{cnn(x)})$ provides the isolation score using the equation 2.11 below. If x is not found to be part of any hypersphere in S, its isolation score is taken to be 1 (considered to be maximally isolated). This process is also illustrated by figure 2.7.4.

$$I(x) = \begin{cases} 1 - \frac{\tau(\eta_{cnn(x)})}{\tau(cnn(x))}, & \text{if } x \in \cup_{c \in S} B(c) \\ 1, & \text{Otherwise} \end{cases} \tag{2.11}$$

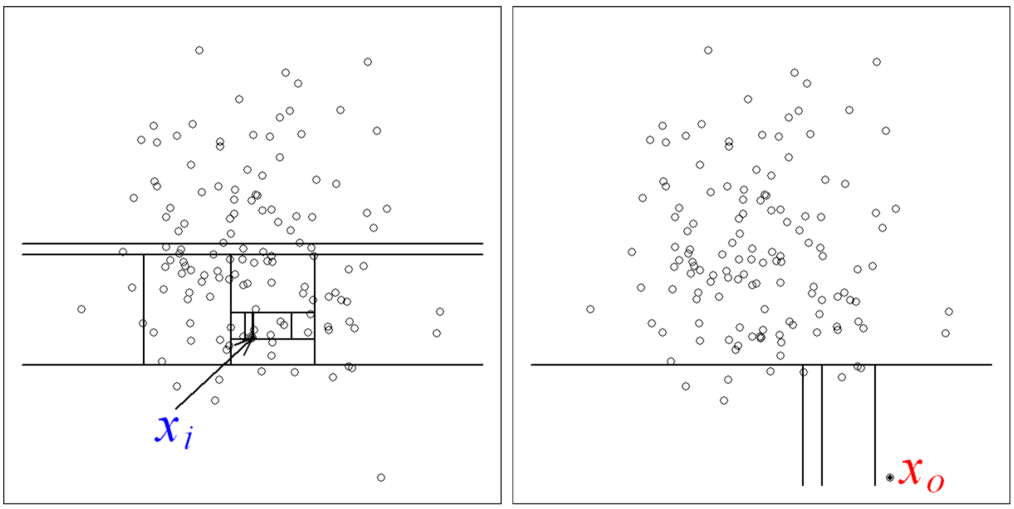
Where, $cnn(x) = arg\ min_{c \in S}\{\tau(c) : x \in B(c)\}$ and $B(c)$ is the hyperspere centered at c with radius $\tau(c)$.

The time complexity of iNNE for detecting N input samples is $\mathcal{O}(N)$. The time complexity is better than that of k-NN.

Isolation Forest (iForest)

Isolation Forest (iForest) [23] is also an ensemble method consisting of a set of t models, known as *Trees* (hence, the name 'Forest' for a collection of trees). Each $Tree$ is a balanced binary tree associated with a set of randomly selected subsamples of the training dataset. The nodes of the tree divide the subsamples using a randomly placed partition value of a randomly selected input feature. Starting from the root node, each node divides the tree into two branches. The left branch takes the samples which have the selected feature value less than the selected partition, while the right branch takes the samples with the feature value greater than the partition. The two branches then continue dividing the training subsamples of the model recursively, until either a leaf node is reached, where there is just one sample left, or a depth of $log(M)$ is reached, where M is the size of the training samples. The number of trees in the forest and the size of the subsamples are the model parameters of the iForest model. The authors recommend 100 and 256, respectively, as reasonable parameters that would work for a wide range of training sizes.

The isolation factor of a test sample x is closely related to its path length $h(x)$. The path length of a sample is the depth of traversal as it traverses a tree until it reaches a leaf node with only one sample (the training sample closest to the test sample). The sample begins the traversal of a tree by entering the root node. If the feature value of x at the



(a) Isolating x_i (b) Isolating x_o

Figure 2.7.5: Isolation Forest anomaly detection algorithm. x_i is deemed normal due to a long traversal path length in the tree, while x_0 is deemed abnormal owing to its short traversal path length.

Source: Liu et el. 2008 [23].

node is below the partition value at the node, the traversal continues on the left branch, otherwise, it continues on the right branch. The traversal continues recursively until it reaches a leaf node. If the leaf node contains only one sample, the depth of the traversal straightforwardly represents the path length. Otherwise, the path length is the sum of depth traversed so far and an estimated path length, $c(n')$ beyond the leaf node, which is estimated from the leftover samples size n' at the leaf node by using equation 2.12 [23]. This traversal is illustrated in figure 2.7.5

$$c(n) = 2H(n-1) - (2(n-1)/n) \tag{2.12}$$

Using the path lengths determined in the manner described above, the anomaly detection score $s(x, n)$ for a given test sample x is given by [23]:

$$s(x, n) = 2^{-\frac{E(h(x))}{c(n)}} \tag{2.13}$$

Where, n is the number of the subsamples in each tree of the forest, $c(n)$ is the average path length of an unsuccessful binary search tree with n nodes and is given by the equation 2.12. $h(x)$ is the path length when traversing one tree. Expected path length $E(h(x))$ is the mean of $h(x)$ traversing all the trees in the forest.

H(i) is the harmonic number and it can be estimated by $ln(i) + 0.5772156649$ (Euler's constant) [23].

The time complexity of iForest for detecting N input samples is $\mathcal{O}(N)$. The time complexity is better than that of k-NN.

2.7.9 Known issues

Recent research has shown 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, generative models such as VAE are considered good in mapping the distribution of the training data so that normal input data that have not been seen before would also correctly map. However, this notion is challenged by Nalisnick et al. 2019 [29]. 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 [29].

Xiao et al. 2020 [36] 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.

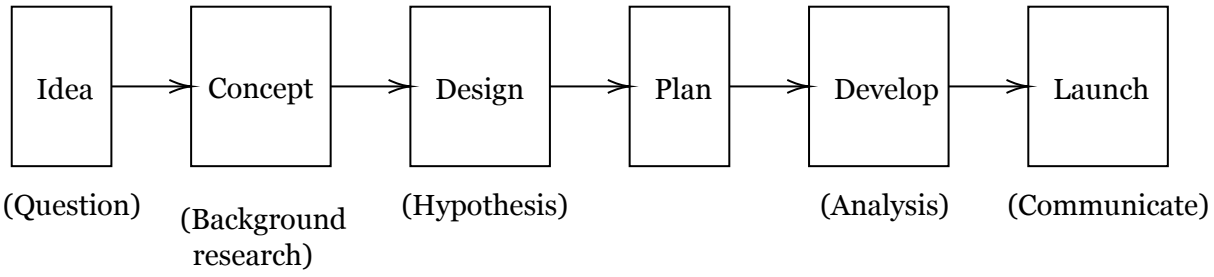


Figure 3.0.1: The engineering method stages followed in the project.

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 method, and analysis of the results.

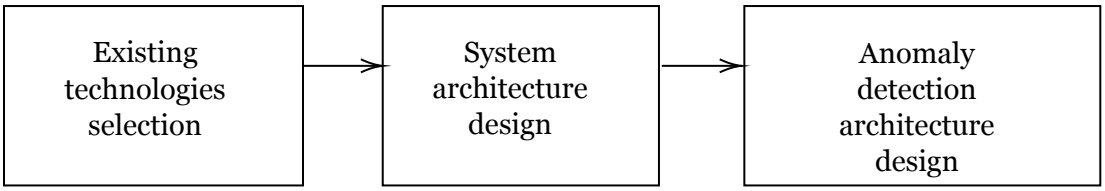


Figure 3.4.1: The design process stages that were conducted in the project.

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 and experiments

The development and the experiments consisted of the bulk of work in the project, namely, the implementation of the data pipeline, the data collection, the implementation of the anomaly detectors, training the models, and analyzing and evaluating the results.

3.5.1 Pipeline development

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.

[1]https://en.wikipedia.org/wiki/Kanban_(development)
[2]https://trello.com/

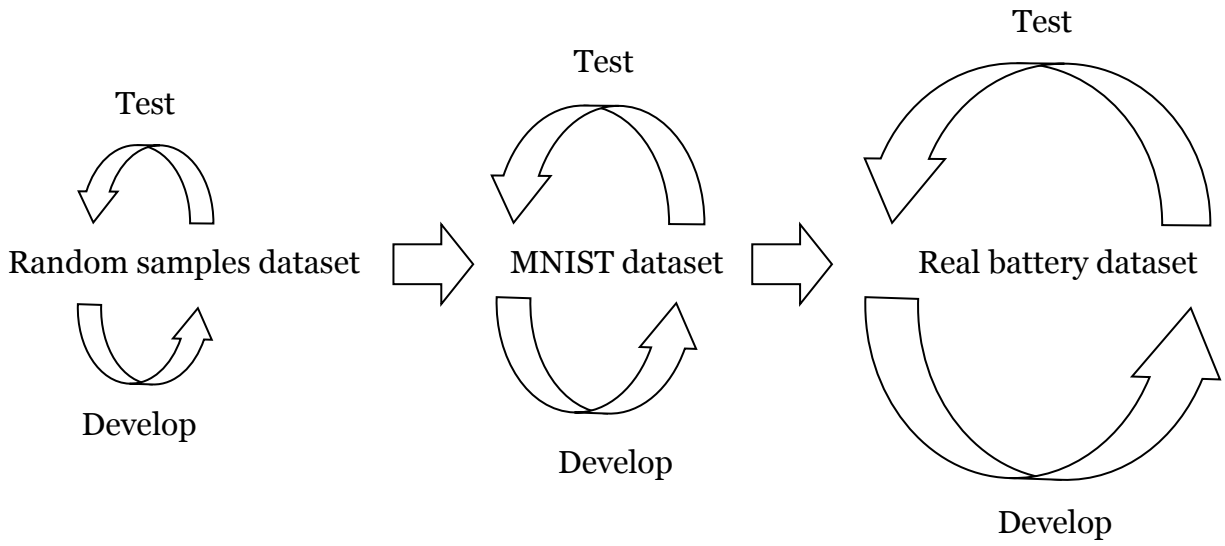


Figure 3.5.1: The iterative process and stages are used in the development of the project. Each stage used progressively more complex data to enable progressive development and debugging.

3.5.2 Tools and resources

The LSTM-VAE and GRU-VAE deep neural networks were developed using the python Keras framework [3], while the conventional models, iNNE, iForest, and k-NN used in the evaluation were developed mostly in bare python. Jupiter Notebook [4] was used as the editing and development platform and Google Colab [5] was used as the execution environment. Keras framework was chosen for its rich community support in 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.

3.5.3 Experiments

The project follows a quantitative experimental method to answer the research questions specified in section 1.2. Ten anomaly detection model variants were evaluated in 10 different experiments. Their performances were evaluated by computing and comparing the AUC values of their respective ROC curves when attempting to classify a set of days in a test dataset as either an anomaly or not.

[3]https://keras.io/
[4]https://jupyter.org/
[5]https://colab.google.com/

Firstly, the large volume of battery measurement samples is reduced by resampling on a day-basis for each battery by computing summary data for each day. The data pre-processing is described in chapter 4, section 4.5.

Secondly, each model is trained using two variants of the training dataset. One utilizes the full year's measurement data samples and the second utilizes only the recent one week of measurement data samples, just prior to the anomalous days being tested. The five anomaly detection models and the two training dataset variants produce a total of 10 different model variants to evaluate. The training datasets are unlabeled since we assume anomalies are rate and unknown by nature.

Thirdly, the model variants are then trained using their corresponding training dataset.

Fourthly, a labeled test dataset is created which consists of a subset of recent days prior to the days when some batteries failed. Since the idea is to detect such failures ahead of time, the last 7 days of each failure day are assumed to generate anomalous data. They formed the true positives of the test dataset. It was further assumed that there are relatively more normal operational data at any given time. As a result, approximately 40x more samples are randomly selected from the same recent days, but excluding the known anomalous data samples. They represent the true negatives of the test dataset.

Lastly, the models attempt to classify each day sample in the test dataset as either an anomaly day or not by computing a probability. The ones with the highest probabilities are considered anomalies.

3.5.4 Model evaluation

The models were evaluated against the labeled dataset using ROC and the corresponding AUC as a metric. The same metric is often used in binary classifier evaluation since it is of a similar nature, except that these models were trained using unsupervised training with no known labels, which is often the case with anomaly detectors.

A ROC is plotted for each model variant using the predicted probabilities of each data sample in the test dataset. The ROC visually illustrates the degree of separation of the true positives and true negatives. The ROC is graphically plotted between

true-positive rate (also known as sensitivity or recall) and false-positive rate (also known as specificity, selectivity, or false alarms) at progressively increasing probability thresholds. The more curved the ROC curve appears the more successful the classifier is in the separation task [10].

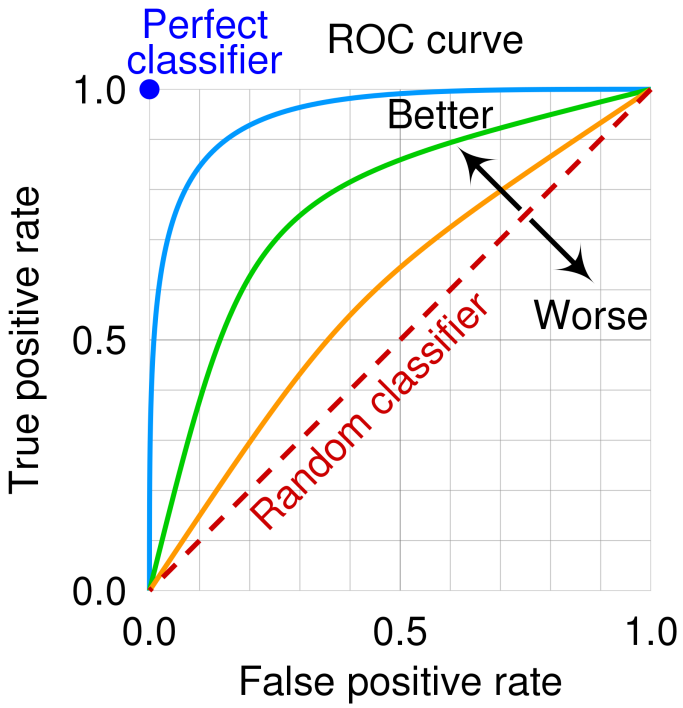


Figure 3.5.2: Receiver Operating Characteristic, ROC.

Source: The figure is courtesy of cmglee, MartinThoma, Wikipedia, 2018.

The AUC value of a ROC curve is the literal area of the space under the curve. This value naturally provides a quantitative measure of the curvature of the ROC. Hence, it also represents a quantitative performance measure of the model in a classification/detection task. The higher is the AUC value of the ROC, the higher is the detection performance.

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

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 loss of productivity and sometimes, loss of equipment life.

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.

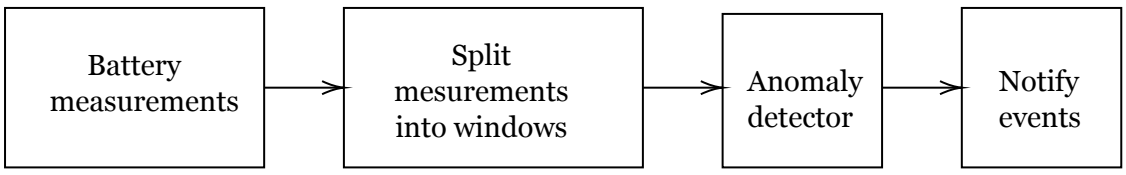


Figure 4.1.1: The high-level concept of battery anomaly detection.

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 major project tasks planned at the beginning of the project. The major tasks were further broken down into minor project tasks that formed the basis of execution during the development. Following the agile development methodology, many project tasks were planned, broken down, and implemented 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 to implement the full anomaly detection pipeline. Figure 4.3.1 illustrates the system architecture.

The customer site consists of multiple separate locations. Each location hosts several 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. It is typical to add one or more WiFi repeaters in such cases.

The measurement samples arrive at a central database where they stay until further processing at more convenient times. Usually, the measurement samples arrive at a much higher frequency than the actual processing. For example, in this project, they arrive in 5 minutes batches while the anomaly detection pipeline processes them every 24 hours. Before the anomaly detection process begins, the measurement data are pre-

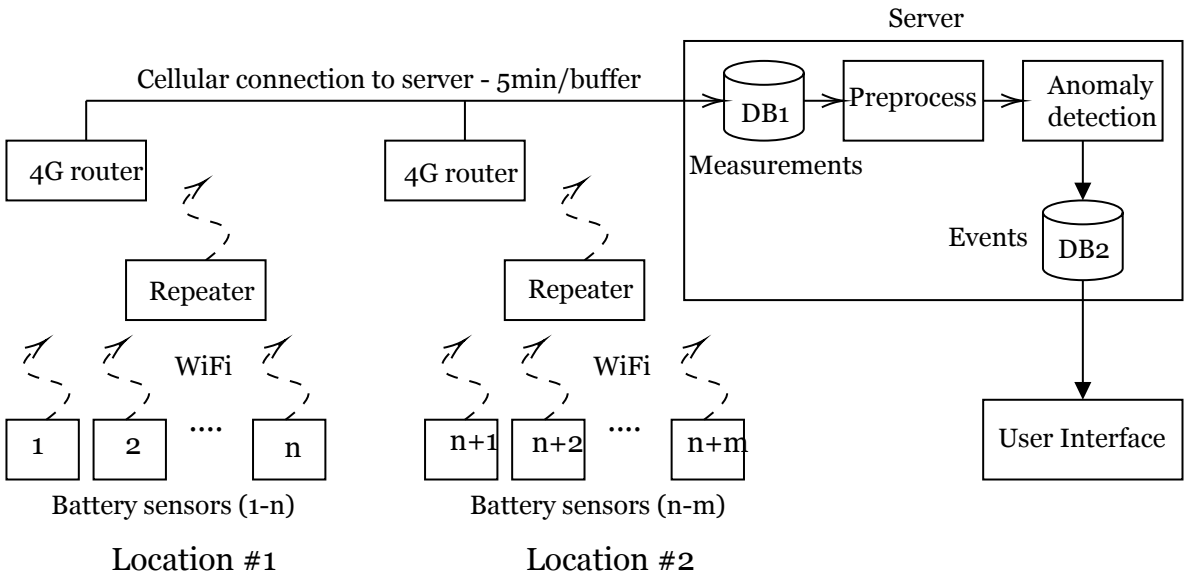


Figure 4.3.1: Data pipeline and system architecture.

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

The following 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 well known in Finland. This particular warehouse is for storing and distributing their final products. The warehouse is located in the central Helsinki region, south of Finland.

The warehouse holds over 100 electric forklifts, all operated by lead-acid batteries. Hundred battery sensors were installed on hundred lead-acid batteries. However, only 97 sensors worked in the end.

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 attached to the top of the battery securely and powered by the battery itself. The sensor's voltage leads and current leads are connected 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. The routers, in turn, connect to the Internet via 4G cellular networks. The pilot data collection used an independent local area network for isolation reasons. The local network for the sensors consisted of several 4G/WiFi routers and WiFi repeaters strategically placed around battery charging rooms (where most battery traffic exists). The independent connectivity network enabled the sensors to connect to the 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 main hubs 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 two months in June and July 2020. The deployment process was slow due to the corona pandemic ongoing at the time. The customer restricted all physical access to the warehouse site due to coronavirus pandemic restrictions. As a result, we remotely train a local technician to install and configure the battery sensors. The setup took more time than would typically take.

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

Following the period of control data collection, there existed about two months (Sept and Oct 2020), during which the eFleetly team deployed other eFleetly battery maintenance features. Unfortunately, that activity 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.

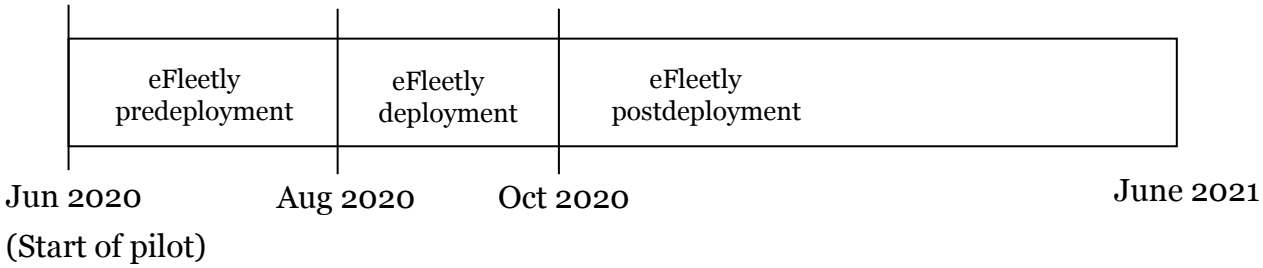


Figure 4.4.2: Collection timeline of battery measurement data during eFleetly pilot.

The periods above were mostly for verifying other predictive maintenance features in eFleetly. This thesis does not distinguish the periods when analyzing the anomaly detection behavior. However, the periods could provide insights frequency of anomaly incidences changes between them.

The sensors collect measurement data every 17 seconds. However, they are programmed to deliver the measurement data every 5 minutes in a batch. The batch delivery reduces needless traffic between the sensors and the cloud server. Furthermore, if some sensors are out of range from the WiFi stations, they buffer the data until they reconnect to the server.

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 simpler tabular forms, such as flat Comma Separated Values (CSV) files, for use by the deep learning network. The models were developed separately inside Google's Collab environment to avoid disruption in the production. 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.

The extraction processes could disrupt the running eFleetly service. The entire eFleetly database was copied to a local computer using backup and restore tools of MongoDB to avoid the risk. 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 challenging. As a result, they were exported separately for each month, each roughly forming a CSV file of approximately 500MB size. They were then compressed, copied to the Google drive, and uncompressed in the project directory.

The measurement samples were separated into individual battery time series to accomplish pre-processing correctly.

4.5.2 Data resampling

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 opportunistic times, the measurements are not assumed to arrive in sorted 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.

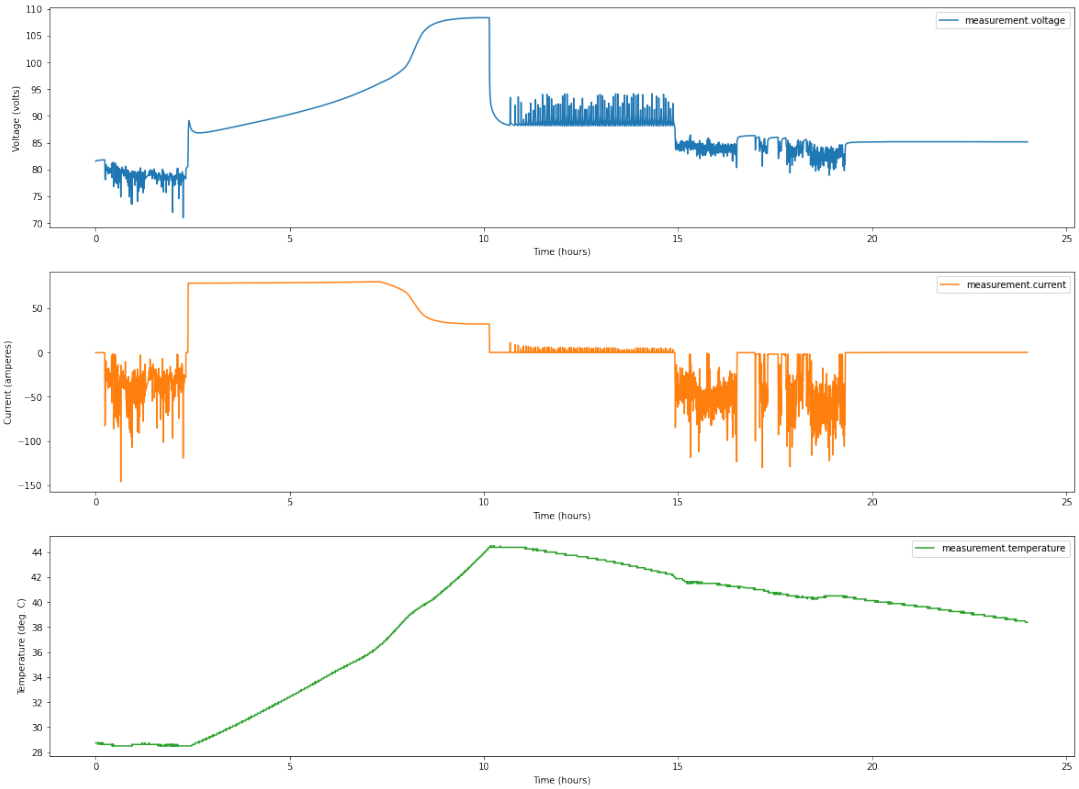


Figure 4.5.1: A plot of battery data on a typical working day)

The sensors measure the batteries every 14 seconds. Consequently, the sensors generated a large volume of data by the 97 battery sensors operating over one year. The data consisted of over 900 million individual measurement data points. A resampling method reduced the data size to enable the experiments in this project using available resources. The resampling method resamples the measurements by computing one data point per day per battery, colloquially known as day summary. The resampling reduced the data size to a manageable size of approximately 146,000 measurement samples. Section 4.5.4 below describes the day summary computation by the resampling method.

4.5.3 Feature augmentation with internal resistance

Voltage, current, and temperature are observable characteristics of a battery operation measured directly by the sensors. However, one particularly unobservable characteristic, known as *Internal Resistance*, is of particular importance. Internal

resistance is the electrical resistance imposed on the electrical circuit by the battery during its operation. All batteries have some internal resistance. However, if high, internal resistance causes significant loss of power and increases the thermal loss.

Internal resistance is generally considered a good indicator of the internal degradation of the battery. Hence, an estimation of the internal resistance, computed for all the samples, could be added as a new feature dimension to the data samples.

In practice, however, internal resistance is not a fixed characteristic. It varies depending on the state of charge, temperature, and other operating factors. It is also often hard to measure on a running system. Usually, dedicated equipment would run several charging and discharging cycles to estimate the internal resistance. Such a method is not practical since our batteries are all operating online.

Electrically, the internal resistance causes a drop in terminal voltage when a load is applied. Hence, an estimation method computes it by measuring three parameters: (i) the float voltage before any load is applied, (ii) the terminal voltage when a known load is applied, and (iii) the current through by the load. The method computes the internal resistance from these three parameters using Ohm's law by dividing the drop in voltage by the load current (equation 4.1). Note, the float voltage itself changes depending on the state of charge.

$$R_{internal} = \frac{(V_{float} - V_{load})}{I_{load}} \tag{4.1}$$

During a live operation, the method estimates the float voltage by taking an average of all voltage measurements where the load current is negligible ($-5A < I_{load} <= 0$; Note: discharge current is negative). This method approximates the estimation of the float voltage for the day.

Once the estimation method computes the average float voltage for the day, all significant discharge samples ($I_{load} < -5A$)) should register a voltage drop. If the voltage drop is lower than the day's estimated float voltage, it uses the difference to compute the internal resistance using the equation 4.1. Since the float voltage is an approximation, the calculated values of the internal resistance are themselves approximate. However, the approximation is considered acceptable for using it as a

new feature dimension for anomaly detection purposes.

4.5.4 Day summary dataset

A resampling method computes the day summary data by taking the average of the voltage, current, temperature, and internal resistance measurements during the discharging of the batteries. It excludes the measurement samples corresponding to the charging of the batteries from summary computation because the chargers overwhelm the charging characteristics in the measurements, overshadowing the ones belonging to the battery. All samples with positive current, which correspond to charging, are removed from the summary computation. It also excludes the data samples without computed internal resistance. That leaves only the samples registered during heavy load discharges as the basis of computing the day summary.

For technical reasons, if resampling method can not compute a day summary, it sets the day summary to a vector of zeros. The day summary computation can fail for reasons such as lack of equipment discharge during the day or inability to compute float voltage for the day's operation. They are still kept in the dataset to maintain the time series positions of the samples. Figure 4.5.2 shows the heat map of the features in the dataset.

4.5.5 Windowing

The LSTM-VAE and GRU-VAE networks training and inference require that the data samples are windowed in time series. As a result, the day summary time-series of each battery were segmented into 7-days time-series records. Hence, an input to the LSTM-VAE or GRU-VAE model consisted of a windowed record of seven days times four features. The 'zero' days mentioned above are left in the windowed records to maintain the time-series positions of all samples.

For other models, iNNE, iForest, and k-NN, there is no windowing in the datasets since they are not time series models. However, all zero-valued samples were removed from their datasets.

4.5.6 Training datasets

Two variants of training datasets were prepared.

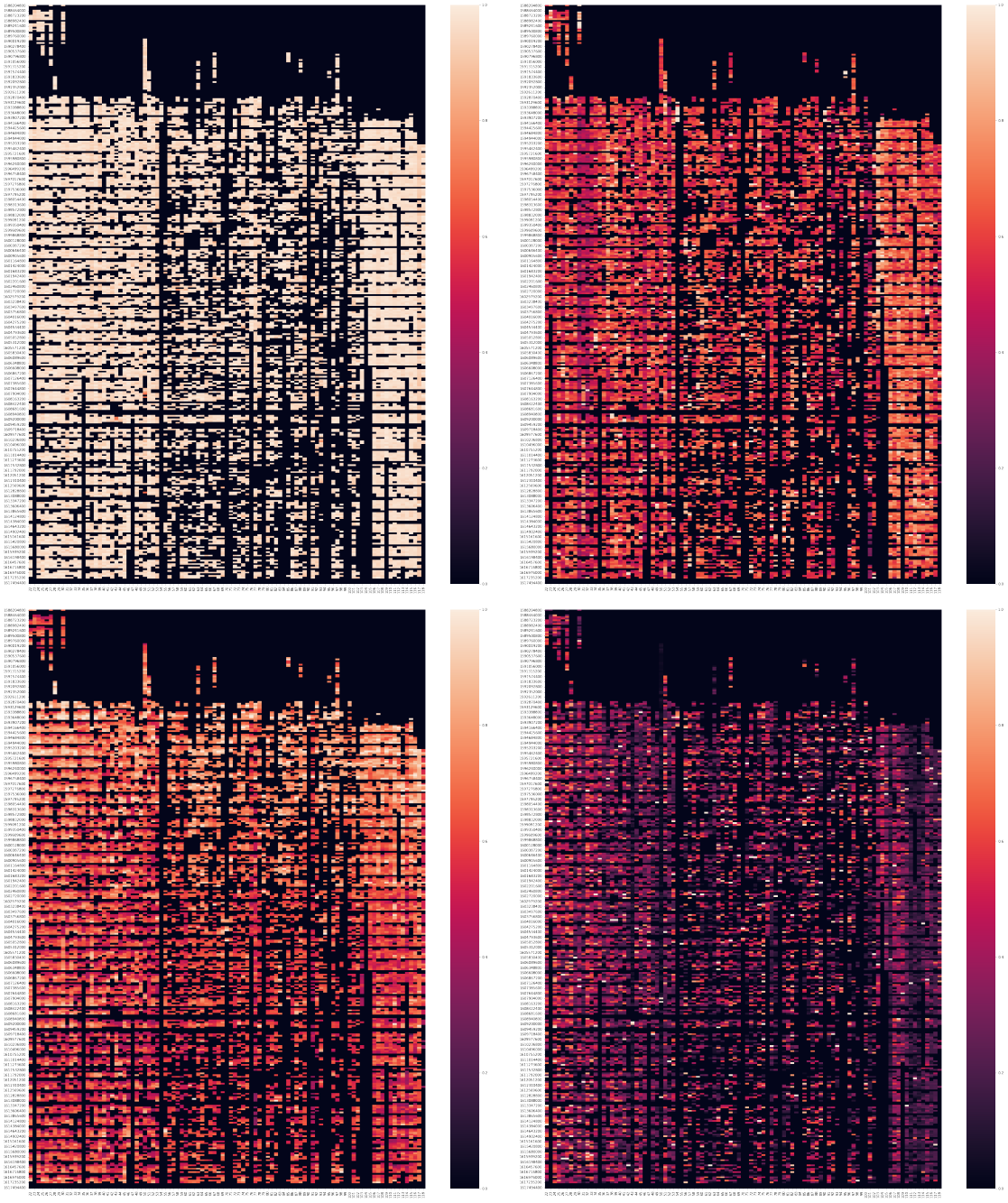


Figure 4.5.2: Heat maps of voltage, current, temperature, and internal resistance of the day summary dataset. the x-axis is battery IDs (1 - 97) and the y-axis is day timestamps covering ca. one year. All features are min-max normalized.

The first one, referred to as the 'full' dataset, consisted of nearly a year of operation of all the 97 batteries. The dataset consisted of ca. 32980 day-samples (including the zero-valued ones) organized into 7-day records for the LSTM-VAE and GRU-VAE networks. The zero-valued samples were removed to leave the dataset comprising only individual samples for the other models. The removal reduced the dataset to 14036 samples.

The second one, referred to as the 'recent' dataset, consisted of only the last seven days of operation immediately before the customer repaired the two batteries (battery #47 and #53). The dataset consisted of 1358 day samples, again organized into 7-day records for the LSTM-VAE and GRU-VAE networks. Similar to the previous dataset, the zero-valued samples were removed, leading to the dataset comprising only individual training samples for the other models. The removal reduced the dataset to 649 samples.

All four features of the dataset, namely, voltage, current, temperature, and internal resistance, were min-max normalized, leading to their values normalizing between 0.0 and 1.0 value range. The normalization helps to account for the different ranges caused by different types of batteries used in the fleet. Since discharges record negative currents, the polarity of the electric current is also reversed just for computational consistency.

4.5.7 Test dataset

The test dataset was a labeled dataset with ground truth binary labels. It was composed of: (a) the previous seven days just before the repair of two batteries each (providing 14-day samples) as "True" (presumed anomaly), and (b) a random 40x samples from the normal operating days as "False" (presumed normal).

The customer serviced battery #47 on Dec 1 and #53 on Dec 30, respectively. During their service, the customer found and replaced one bad cell from each of the two batteries.

We assume the previous seven days of these two repair days would represent anomalous days. The seven days were selected for convenience since it is also the record size of the LSTM-VAE and GRU-VAE models, allowing them to infer as records. Hence, Nov 23 - Nov 30 for battery #47 and Dec 22 - Dec 29 for battery #53 are labeled 'True'. The rest of the test samples represented 'False' samples in the test dataset.

They were sampled from the set of all samples from the same period but excluded the samples from the defective batteries, #47 and #53. The samples represented non-anomalous data. The test dataset consisted of 369 samples.

Evaluation of all 10 model variants used the same test dataset. For the non-deep-learning models, the dataset was pruned by removing the 'zero' samples.

4.6 LSTM-VAE and GRU-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 or GRU layer with several time steps matching the window size of the data sample. The LSTM / GRU layer's output passes through two separate feed-forward networks to produce latent variable's mean μ_z vector and variance vector σ_z^2 (representing diag(Σ) of the covariance discussed in chapter 2).

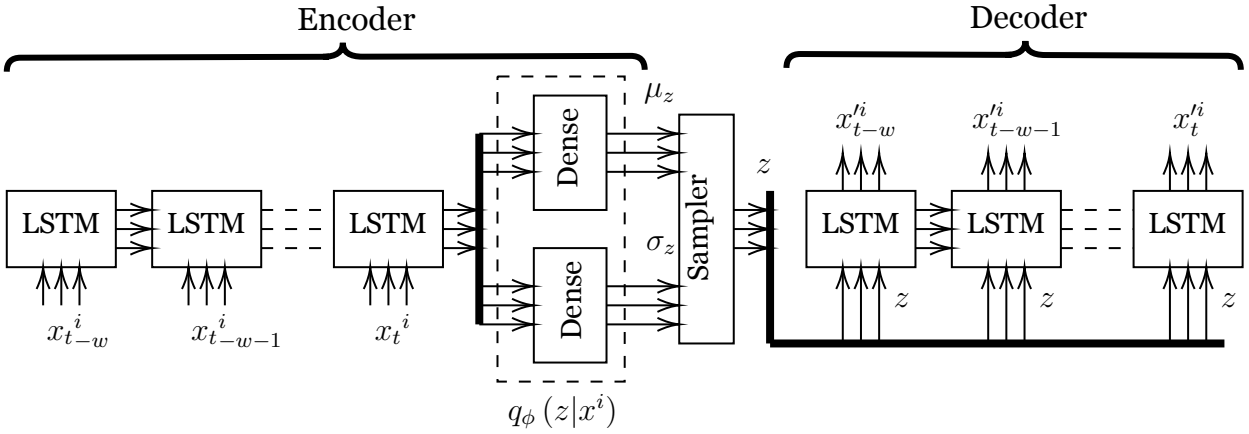


Figure 4.6.1: LSTM-VAE / GRU-VAE network architecture implemented for battery anomaly detection.

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 / GRU layer with several steps equal to the input window size. The steps are to produce an output that has the same size as the input data sample. The decoder LSTM / GRU network still takes input with full window size. For this purpose, the decoder repeats the latent variable z several times equal to window size to form the input data sample (see figure 4.6.1).

The final output of the network is the output of the decoder LSTM / GRU 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 2.6.2. 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.

$$\begin{aligned} Loss &= Reconstruction\ loss + Variational\ loss \\ &= Loss_{MSE} + Loss_{KL} \\ &= \frac{1}{N.W}\sum_{i=1}^{N}\sum_{w=1}^{W}(\hat{x}_{i,w} - x_{i,w})^2 - \frac{1}{2}\sum_{i=0}^{N}(1 + log\ \sigma_{z_i}^2 - \mu_{z_i}^2 - \sigma_{z_i}^2) \end{aligned} \tag{4.2}$$

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 2.6.2 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:

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

The above approach 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.

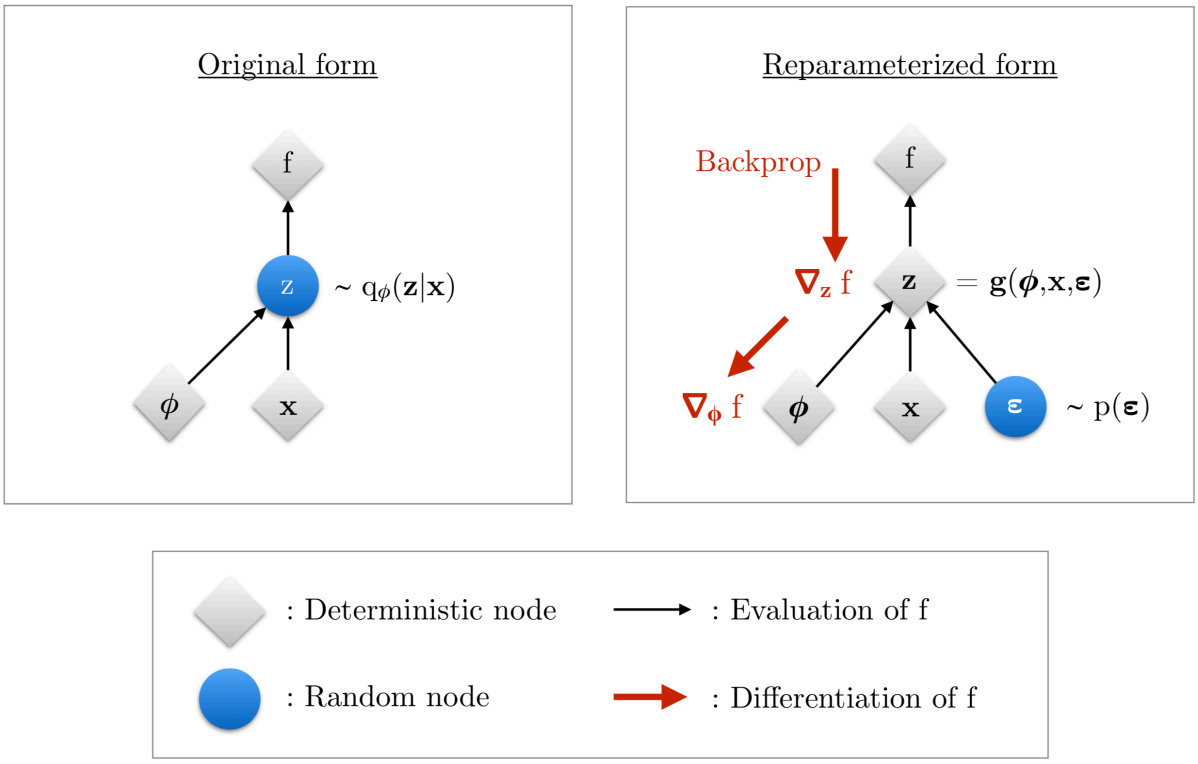


Figure 4.6.2: Illustration of reparameterization trick, [17]

To overcome this limitation, "Reparameterization Trick" is introduced in the implementation of the network [17]. Instead of dealing with a random sampling node, Reparameterization Trick uses a deterministic node, a function taking the latent variable's mean, variance, and an additional input ϵ. The input ϵ is random samples taken from a standard normal distribution N.

$$\begin{aligned} z &= g(\mu, \sigma^2, \epsilon) \\ \epsilon &\sim N(0, I) \\ g &= \mu + \sigma.\epsilon \end{aligned} \tag{4.3}$$

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 LSTM-VAE and GRU-VAE development

The deep neural networks were developed and trained, 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. The 'recent' battery measurement dataset: Used to implement the prepossessing and validate the network operation on the real battery data.
4. The 'full' year battery measurement dataset: Used when the network was confirmed working.

4.8 Hyperparameters search

The LSTM-VAE / GRU-VAE networks developed above can take different model complexities depending on various architecture design factors, such as 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 particular combination of the hyperparameters forms the best combination that yields the best network architecture suited for the target application. The process of searching for 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 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 is generally quite large, often infinite. However, in practice, the search space is reduced by quantizing and limiting the range of each

hyperparameter dimension based on a general understanding of the application. The search then uses the grid formed by the quantized hyperparameters to find an optimal combination. Note that the quantization grid does not have to be linearly spaced. Some applications can utilize logarithmic quantization. Logarithmic quantization helps cover a wide range for a hyper-parameter without increasing the search samples too much.

Randomly sampling the grid rather than brute-forcing all elements in the grid reduces the number of search attempts. Random sampling 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 [30] for hyperparameters search. The limited computing resources available for this project reduced the choices of hyperparameters search space by quantizing the choices for each hyperparameter dimension to fewer numbers. The quantized choices are 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	**Choices**
LSTM/GRU hidden layer size (encoder/decoder)	256, 512, 1024, 2048
Latent variable size	8, 16, 32, 64
Dropout probability (encoder dense layer)	0, 0.1, 0.2, 0.3
Dropout probability (decoder dense layer)	0, 0.1, 0.2, 0.3

Table 4.8.1: LSTM-VAE/GRU-VAE hyperparameters search space.

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

4.9 iNNE, iForest and k-NN algorithms

iNNE and iForest algorithms both involve implementing ensembles of models. iForest authors recommended using 100 trees of 256 subsamples each as generally suitable model parameters [23]. For consistency, iNNE was also developed with an ensemble

of 100 models, each consisting of 256 hyperspheres. The models were trained using the two training dataset variants (full and recent).

For the k-NN algorithm, a few values of k were tried in several experiments. Eventually, $k = 10$ was found to be reasonable. There is no training involved in the k-NN algorithm. The training dataset is straight-up used as the reference dataset to find the k^{th} nearest neighbor data point during the inference of the test dataset.

4.10 Validation and experiments

The pipeline and the network development were verified in three iterations. The first iteration consisted of validating the pipeline and data preprocessing. Data visualization and validation of the results of the intermediate steps in the pipeline validated the measurement inputs and data prepossessing. The second iteration consisted of validating the functioning of the LSTM-VAE and the GRU-VAE networks based on the expectation of the design.

The third iteration consisted of the experiments. Both of the deep learning models were trained using the two variants of the training dataset previously mentioned in section 4.5.6. Both models were trained using hyperparameters search as described in the previous section. Each hyperparameters search consisted of training over 100 epochs with early stopping enabled within ten epochs of stalled performance. Each training epoch used randomly selected 20% of the training dataset as a validation set. Training with the full dataset used a batch size of 128, while training with the recent dataset used a batch size of 5, owing to their relative size difference.

The training performances of the networks were validated from their learning curves and the general observation of the reproduction outputs. The anomaly detection performances of the networks were evaluated by inferring the test dataset and computing the AUC of the ROC metric.

Finally, the last phase of the experimentation consisted of training the iNNE, iForest, and k-NN algorithms and evaluating their performances using the same test dataset and evaluation metric.

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.

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.

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. Table 5.1.1 describes the characteristics of the deployment and the data collected as a result

of the deployment:

5.2 LSTM-VAE and GRU-VAE development results

The hyperparameters search resulted in the best selection of the network hyperparameters for each variant as listed in the table 5.2. They were the best within the search space comprising quantized and randomly selected hyperparameters. The results show latent variable size leaned towards lower values, while the LSTM/GRU hidden layer leaned towards larger values (if not the largest).

A large number of LSTM/GRU hidden cells can be explained by the need of the network to learn a much more complex model than simply the chemical model of the battery because the time series also record seemingly arbitrary forklifts operation and charging operation by the warehouse crews. This results in different charging and discharging sequences.

Hyperparameter	**GRU-VAE (full)**	**GRU-VAE (recent)**	**LSTM-VAE (full)**	**LSTM-VAE (recent)**
LSTM/GRU Hidden layer size	2048	1024	1024	2048
Latent variable size	64	64	64	8
Encoder dropout probability	0.1	0.3	0.3	0.0
Decoder dropout probability	0.2	0.1	0.1	0.2

Table 5.2.1: List of LSTM-VAE and GRU-VAE hyperparameters that were found through random search within hyperparameters space.

5.3 Evaluation of the anomaly detection models

All the model variants were experimented using the methodology described in chapter 3, section 3.5.3 and evaluated using the method described in the same chapter, section 4.10.

Figure 5.3.1 shows ROC plots of all the model variants tested with the test dataset. Table 5.3.1 shows the AUC of the ROC curves. The results indicate that the best performing model is k-NN, followed by GRU-VAE.

Despite its simplicity, the classical k-NN method is seen performing rather well. It seems in the literature, the seminal method is often found to continue performing well

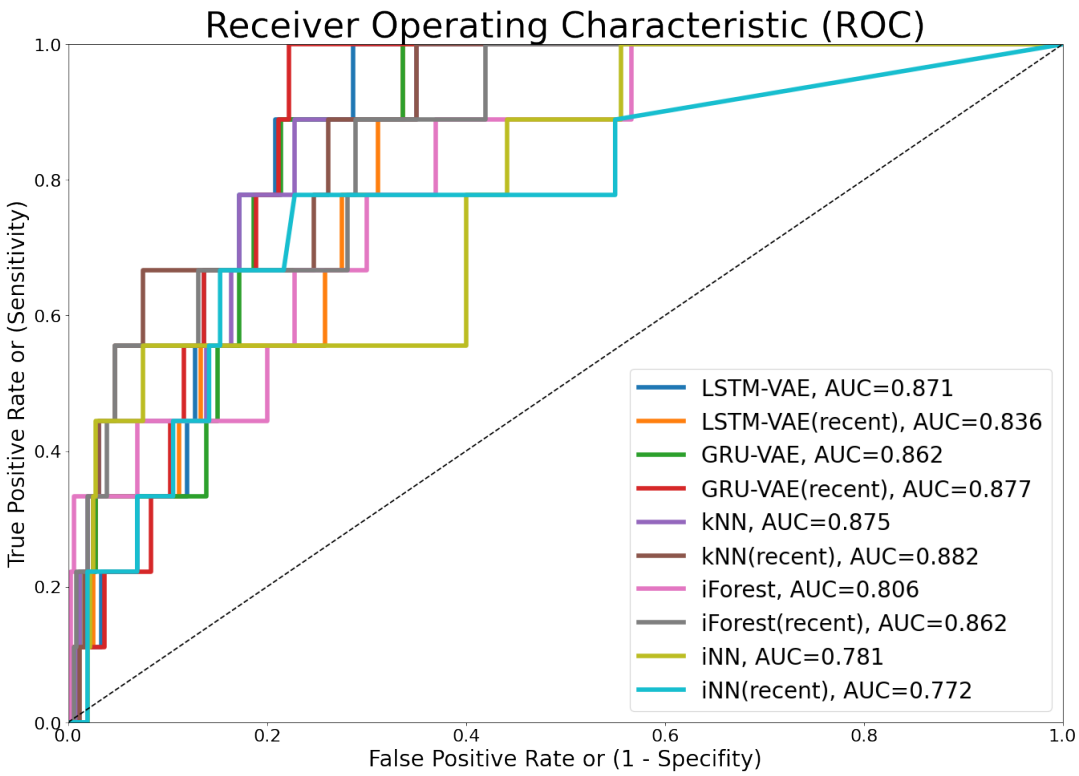


Figure 5.3.1: The ROC plots of all model variants illustrating the classification performance of the models to detect anomalous samples.

in general. For instance, Campos et al., 2016 [5], evaluated several anomaly detection algorithms using several anomaly datasets and reached a conclusion that methods such as k-NN still performed better than the newer methods.

However, the time complexity of the methods needs to be taken into account depending on the size of training and inference datasets. The time complexity of k-NN is $\mathcal{O}(NM)$, which is higher than that of the other methods. All other methods have a time complexity of $\mathcal{O}(N)$.

We further note that both the deep learning methods, GRU-VAE and LSTM-VAE, fared relatively better than the other methods, iForest, and iNNE. In the case of GRU-VAE architecture, the recent dataset fared better, while it was the other way around for the LSTM-VAE architecture. Of the two deep learning models, GRU based VAE model trained on recent dataset performed better than LSTM based model. However, when trained on the full dataset, the LSTM-VAE based model performed better than all other models, except k-NN.

We also note the presence of a general trend where nearly all model variants, except LSTM-VAE, performed better with the recent dataset as opposed to the full dataset.

	Deep learning models		Conventional algorithms		
Time complexity	**GRU-VAE** $\mathcal{O}(N)$	**LSTM-VAE** $\mathcal{O}(N)$	**k-NN** $\mathcal{O}(NM)$	**iNNE** $\mathcal{O}(N)$	**iForest** $\mathcal{O}(N)$
Full dataset	0.862	**0.871**	0.875	0.781	0.806
Recent dataset	**0.877**	0.836	**0.882**	0.772	0.862

Table 5.3.1: AUC values of ROC plots of all five models, each evaluated using two different training dataset variants. The table also highlights the top three models and their better-performing dataset variants.

This observation is curious. Perhaps a logical explanation is that all batteries undergo a similar operational pattern when we observe a limited time window. Hence, the discrimination of an anomalous battery operation is more prominent in the recent dataset. While in the global dataset, the operational patterns are more diverse and, therefore, create a broader generalization.

The next chapter explores the possible conclusions drawn from these results and provides an insight into other future works.

Chapter 6

Conclusions

Chapter 1 described that the purpose of the degree project was to *develop and evaluate* a deep learning 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 chapter.

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. The anomaly detection models were also compared quantitatively with some well-known anomaly detection algorithms.

In chapter 1, we also posed the research question: *if a deep-learning model implemented using LSTM-VAE or GRU-VAE architecture could be a viable method of anomaly detection for a battery fleet operation.*

Drawing from the results observed in the previous chapter, it can be concluded that both the deep learning models appear to be viable methods of anomaly detection for batteries, outperforming some of the existing and well-known methods, such as iForest and iNNE. k-NN method produced the best performance in this experimental setup. However, k-NN has a higher time complexity of $\mathcal{O}(NM)$ compared to the rest of the methods, which have linear time complexity of only $\mathcal{O}(N)$. Hence, on a practical note, it may be more suitable if the application datasets are small. If the datasets are very large, the other methods with lower time complexities may offer better scalability. In those cases, the GRU-VAE or LSTM-VAE could be one of the better choices.

All relevant stakeholders of the project were engaged during the development process to ensure the smooth completion of the project goals. Looking back at those goals, they were all accomplished and described in the previous chapter. Corresponding to each of the objectives listed in section 1.4, we note that:

1. A customer pilot was started jointly with Hartwall Oy and collected the battery operational data.

2. The network of sensors and communication devices were set up and deployed, 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. The project designed and developed a deep learning anomaly detector based on LSTM-VAE and GRU-VAE.

7. Conventional anomaly detection algorithms, iNNE, iForest, and k-NN, were developed.

8. The project evaluated the performances of all the models using an AUC of ROC metric and presented the results.

6.1 Discussion

The project was planned to run for three months. However, due to the COVID-19 pandemic in 2020/2021, the project schedule stretched well beyond a year. Despite this historical disruption, the project managed to deploy the system and evaluate several anomaly detection methods. The data acquisition succeeded in collecting over 6GB of original battery measurement data. The deep learning network development, training, and evaluation were successful post-data acquisition.

6.2 Future Work

6.2.1 Internal evaluation of the models

In addition to the external evaluation of the models using known anomalies, as performed in this project, an ”internal evaluation” can also be performed. The paper *On the Internal Evaluation of Unsupervised Outlier Detection* (Marques et al. 2015 [26]) 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 offers a means to quantify the evaluation. It depends on implementing a nonlinear classier, specifically ones that rely on maximizing margins (examples 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 ROC curve as the evaluation metric. 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.

6.2.2 Evaluating the impact of input record size

To further enhance the capability of the deep learning networks to learn better, the record size could be increased, perhaps significantly, for example, a month.

A wider record size will increase the network complex, resulting in larger hidden layer units and a larger latent variable size. Training such networks inevitably would require a more powerful computing platform. Commercial computing resources developed specifically for such heavy-duty machine-learning purposes could be a choice to consider. Hyperparameters search can also be very resource-intensive during training. Future experiments could use distributed computing to reduce the training time with hyperparameters search.

6.2.3 Finer hyperparameter search space

The limited computing resource also limited the number of hyperparameters sampled from the hyperparameters space. With the help of additional computing resources, the hyperparameters search could increase the number of search samples to find a more optimal combination. Luckily, individual combinations of the hyperparameters can be evaluated independently. As a result, a distributed computing facility with more GPU-

enabled computing nodes can facilitate efficient hyperparameters search. Keras Tuner [30] offers the necessary means to achieve such distributed computation if a cluster of GPU-enabled compute instances are available for such purpose.

6.2.4 Alternative deep neural network architectures

The literature proposes many other deep learning architectures for use with time-series data. Chapter 2 introduced many of them. 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).

An example of a CNN-based architecture is Ribeiro et al., 2018 [33]. 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 some modifications, could also be used for battery time-series measurements.

A GAN-based architecture, on the other hand, is also an attractive 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. The discriminator learns to distinguish such generated output 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 function as a detector of anomalous data on new inputs.

6.3 Final Words

Referring back to the objectives of the project, this thesis presented all related background topics, tools, methods, development processes, evaluation, analysis of the results, and inputs to future work. It also described them in sufficient detail so that future researchers can contribute further to the related field. With this note, the author concludes the thesis by thanking and appreciating all the supports 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] Bandaragoda, Tharindu, Ting, Kai, Albrecht, David, Liu, Fei Tony, Zhu, Ye, and Wells, Jonathan. “Isolation-based anomaly detection using nearest-neighbor ensembles: iNNE”. In: *Computational Intelligence* 34 (Jan. 2018). DOI: 10.1111/coin.12156.

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

[4] 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).

[5] Campos, Guilherme O., Zimek, Arthur, Sander, Jörg, Campello, Ricardo J. G. B., Micenková, Barbora, Schubert, Erich, Assent, Ira, and Houle, Michael E. “On the evaluation of unsupervised outlier detection: measures, datasets, and an empirical study”. en. In: *Data Mining and Knowledge Discovery* 30.4 (July

2016), pp. 891–927. ISSN: 1573-756X. DOI: 10.1007/s10618-015-0444-8. URL: https://doi.org/10.1007/s10618-015-0444-8 (visited on 12/08/2021).

[6] 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).

[7] 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.

[8] 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).

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

[10] Fawcett, Tom. "An introduction to ROC analysis". en. In: *Pattern Recognition Letters* 27.8 (June 2006), pp. 861–874. ISSN: 01678655. DOI: 10.1016/j.patrec.2005.10.010. URL: https://linkinghub.elsevier.com/retrieve/pii/S016786550500303X (visited on 12/05/2021).

[11] Garg, Mayank. "Computer Science and Engineering". en. In: (), p. 70.

[12] Gers, F.A., Schmidhuber, J., and Cummins, F. "Learning to forget: continual prediction with LSTM". In: *1999 Ninth International Conference on Artificial Neural Networks ICANN 99. (Conf. Publ. No. 470)*. Vol. 2. ISSN: 0537-9989. Sept. 1999, 850–855 vol.2. DOI: 10.1049/cp:19991218.

[13] 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).

[14] Heck, Joel and Salem, Fathi M. “Simplified Minimal Gated Unit Variations for Recurrent Neural Networks”. In: *arXiv:1701.03452 [cs, stat]* (Jan. 2017). arXiv: 1701.03452. URL: http://arxiv.org/abs/1701.03452 (visited on 12/12/2021).

[15] 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.

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

[17] 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).

[18] 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).

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

[20] 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).

[21] Li, Xiaojun, Li, Jianwei, Abdollahi, Ali, and Jones, Trevor. “Data-driven Thermal Anomaly Detection for Batteries using Unsupervised Shape Clustering”. In: *arXiv:2103.08796 [cs, eess]* (May 2021). arXiv: 2103.08796. URL: http://arxiv.org/abs/2103.08796 (visited on 11/18/2021).

[22] Li, Xunjia, Zhang, Tao, and Liu, Yajie. “Detection of Voltage Anomalies in Spacecraft Storage Batteries Based on a Deep Belief Network”. en. In: *Sensors* 19.21 (Jan. 2019). Number: 21 Publisher: Multidisciplinary Digital Publishing

Institute, p. 4702. DOI: 10.3390/s19214702. URL: https://www.mdpi.com/1424-8220/19/21/4702 (visited on 11/18/2021).

[23] Liu, Fei Tony, Ting, Kai Ming, and Zhou, Zhi-Hua. "Isolation Forest". In: *2008 Eighth IEEE International Conference on Data Mining*. ISSN: 2374-8486. Dec. 2008, pp. 413–422. DOI: 10.1109/ICDM.2008.17.

[24] 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).

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

[26] 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).

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

[28] 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.

[29] 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).

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

[31] 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).

[32] 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.

[33] 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).

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

[35] 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).

[36] 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).

[37] Zhao, Wenjie, Zhang, Yushu, Zhu, Ye, and Xu, Peng. “Anomaly detection of aircraft lead-acid battery”. en. In: *Quality and Reliability Engineering International* 37.3 (2021). _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/qre.2789, pp. 1186–1197. ISSN: 1099-1638. DOI: 10.1002/qre.2789. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/qre.2789 (visited on 11/18/2021).

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

List of Figures

List of Tables

TRITA -EECS-EX-2021:857