Continuous Time Networks for Adaptive Process Control

Thesis to be submitted in partial fulfillment of the requirements for the degree

of

Master of Technology

by

Abhinash Anilkumar
20ME31056

Under the guidance of

Dr. Surjya Kanta Pal

MECHANICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR

Department of Mechanical Engineering
Indian Institute of Technology,
Kharagpur
India - 721302

CERTIFICATE

This is to certify that we have examined the thesis entitled **Continuous Time Networks for Adaptive Process Control**, submitted by **Abhinash Anilkumar**(Roll Number: *20ME31056*) a postgraduate student of **Department of Mechanical Engineering** in partial fulfilment for the award of degree of Master of Technology. We hereby accord our approval of it as a study carried out and presented in a manner required for its acceptance in partial fulfilment for the Post Graduate Degree for which it has been submitted. The thesis has fulfilled all the requirements as per the regulations of the Institute and has reached the standard needed for submission.

Supervisor

Department of Mechanical Engineering
Indian Institute of Technology,
Kharagpur

Place: Kharagpur
Date: 10.11.2024

ACKNOWLEDGEMENTS

I express my sincerest gratitude to **Dr. Surjya Kanta Pal** for his invaluable insights and guidance that helped me in formulating the research methodology and addressing the various aspects of the project. It provided me an opportunity to sharpen my thinking skills and deal with numerous technicalities that arose, due to his perceptive inputs. He motivated me to dig deep into the details of the problem which allowed me to explore a variety of interesting flow phenomena.

I would also like to thank the **Department of Mechanical Engineering, IIT Kharagpur** for providing me the opportunity to take up this project.

I would also like to thank **Mr. Avishek Mukherjee** for his support and guidance that helped me complete the project.

Abhinash Anilkumar

IIT Kharagpur

Date: 10.11.2024

ABSTRACT

The rising demand for reliable and efficient energy storage has placed lithium-ion batteries at the forefront of technological advancement, emphasizing the need for accurate modelling of their discharge behaviour. This thesis presents a comparative analysis of general sequential neural network models, including Recurrent Neural Networks (RNNs), Long Short-Term Memory networks (LSTMs), alongside more dynamic, continuous-time architectures, such as Neural Ordinary Differential Equations (Neural ODEs) and Neural ODEs with Liquid Time Constant (LTC) layers. Utilizing a dataset detailing the discharge cycles of lithium-ion batteries, this study evaluates each model's capacity to capture the complex sequential patterns inherent in battery degradation. The comparison focuses on model accuracy, computational efficiency, and adaptability to long-term dependencies, aiming to identify the optimal approach for accurate discharge predictions. This research underscores the potential of Neural ODE-based architectures, particularly with LTC enhancements, to outperform traditional sequential models, contributing valuable insights for battery health prediction and broader applications in sequential data modelling.

Contents

List of Figures

Chapter 1

Introduction

The world is becoming more and more driven by technology, thus it's critical to foresee and adjust to changes quickly. Industries of all stripes are looking for creative ways to react proactively to invisible changes and avoid expensive failures, as well as ways to improve the safety, effectiveness, and longevity of their operations. The goal of adaptive process control, which allows systems to self-monitor, learn from continuous data, and make real-time modifications to maximise performance, is at the centre of this shift.

With the development of science and technology, the number of electronic devices is increasing, and the requirements for power supply are getting higher and higher. Lithium-ion batteries are widely used in various fields because of their advantages such as high energy density, small self-discharge, wide operating temperature range, long service life and green environment protection. However, in the working process of lithium-ion batteries, wear and aging are inevitable, and the degradation of equipment performance reduces the reliability of the system, and even cause major accidents. Therefore, the health management of batteries is very important.

Battery monitoring is a crucial use case for this technology, since even minute changes in discharge cycles might signal upcoming inefficiencies or maintenance requirements. Accurate performance monitoring and early deterioration detection are not just operational requirements for lithium-ion batteries, which power everything from home gadgets to electric cars and industrial gear, but they are also

essential for sustainability and safety. Battery system failures may lead to decreased capacity, a shorter lifetime, and in extreme scenarios, dangerous circumstances.
Since the lithium-ion battery capacity degradation data is time series data, how to better simulate the dynamics of lithium-ion battery data in the time dimension with a simple structure network remains to be studied.

The goal of this project is to create and implement an adaptive process control model designed especially to track the discharge behaviour of lithium-ion batteries. A system may optimise battery consumption, predict maintenance requirements, and send out timely alarms by gathering comprehensive performance data. By providing a scalable solution that helps enterprises achieve improved operational resilience and product dependability, the benefit of implementing an intelligent, self-regulating model for battery monitoring goes far beyond the laboratory.

Chapter 2

Literature Review and Problem Statement

Deep structural neural networks have been used across several domains and are extensively employed in the health management of lithium-ion batteries. Zhang et al. introduced a hybrid prediction model that integrates random forest (RF), Artificial Bee Colony (ABC), and general regression neural network (GRNN), which efficiently identifies significant characteristics and facilitates timely, accurate predictions. Tagade et al. introduced a deep Gaussian process method for the health monitoring of lithium-ion batteries. The technique immediately utilises partial charge and discharge time series data, including voltage, temperature, and current, to estimate capacity without the need of extracting input attributes.

The data-driven approach primarily relies on entering typical characteristics from the charging and discharging cycle curves into the model, including voltage, current, internal resistance, temperature, and other relevant metrics. The predominant methods include machine learning techniques such as Support Vector Machines, Random Forests, Gaussian Process Regression, and Artificial Neural Networks, as well as deep learning approaches like Long Short-Term Memory networks, Convolutional Neural Networks, and Graph Neural Networks. Certain researchers have used charge cycle curves to derive distinctive characteristics for capacity estimate.

Li developed an electrochemical model to extract internal ageing characteristics, used various situations in charging and discharging modes to derive external features, and employed a machine learning method to choose both internal and external features for capacity estimation. Yao used partial segments from charging and discharging to construct a deep transfer convolutional neural network model for capacity estimate, achieving an estimation error of just 0.022. Xiong retrieved voltage feature values of varying widths during charging segments and integrated them with diverse machine-learning techniques to assess lithium-ion batteries. Wei devised graph convolutional neural networks using several attention methods for predicting battery capacity by recognising feature parameters, including current, voltage, and temperature that exhibit high similarity during constant current charging periods.

Certain researchers have used discharge cycle curves to derive characteristics for capacity estimate. Hu introduced a technique for extracting discharge voltage profile features using a voltage segmentation approach and developed a data-driven model including support vector machines and Gaussian regression for estimating battery capacity. Deng examined several ageing battery identification methods and transfer learning, retrieved four distinctive variables from the battery's discharge capacity curve, and developed a capacity estimate model using LSTM to enhance accuracy.

State Space Models (SSMs) are recognised frameworks for the analysis of deterministic and stochastic dynamical systems (KALMAN, 1960). Their state and input transition matrices may be directly acquired by gradient descent to model sequences of observations (Gu et al., 2021; Hasani et al., 2021b; Lechner et al., 2020b). In a seminal study, Gu et al. (2022a) demonstrated that through a few essential algorithmic techniques for memorisation and computation of input sequences, SSMs can evolve into the most potent sequence modelling framework to date, surpassing sophisticated RNNs, temporal and continuous CNNs (Cheng et al., 2022; Romero et al., 2021a, b), and a diverse array of Transformers (Vaswani et al., 2017).

We have implemented the LTC-based Neural ODE, derived from a more expressive state-space model, namely the liquid time-constant (LTC) representation (Hasani et al., 2021b), which outperforms both generic Neural ODEs and sequential models such as LSTMs.

Chapter 3

Methodology

In this thesis, we will develop and evaluate a series of sequential models, including Long Short-Term Memory (LSTM) networks, Neural Ordinary Differential Equations (Neural ODEs), and Neural ODEs with a Liquid Time Constant (LTC) layer, to predict capacity values over discharge cycles for lithium-ion batteries. The methodology involves pre-processing the battery discharge dataset to structure the time-series data and selecting relevant features for model input. Each model will be trained on historical data to capture temporal dependencies in capacity degradation. To assess model performance, we will utilize the R-squared metric, providing a measure of predictive accuracy for each model. Additionally, we will create plots comparing actual versus predicted capacity values across discharge cycles, visually demonstrating each model's ability to track the true capacity trajectory.

3.1 Lithium-Ion Batteries

Lithium-ion (Li-ion) batteries are widely used in applications that demand high energy density, lightweight design, and extended cycle life, such as consumer electronics, electric vehicles, and renewable energy storage.

- High Energy Density: Li-ion batteries offer a superior energy-to-weight ratio, making them suitable for applications where space and weight are at a premium.
- Efficiency: These batteries have a high charge and discharge efficiency, typically above 90%, making them suitable for applications requiring frequent charge and discharge cycles.

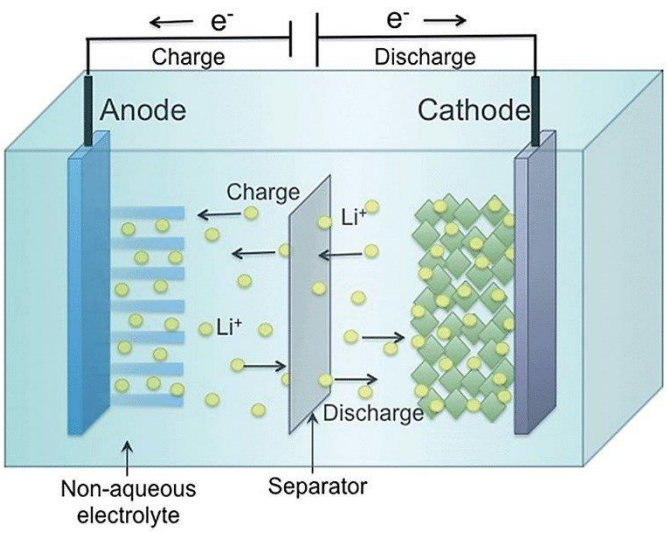

Figure 3.1: Li Battery

3.2 Data

The dataset used in this study originates from Electrochemical Impedance Spectroscopy (EIS) tests on lithium-ion batteries, providing a detailed view of the battery's electrochemical behaviour over multiple discharge cycles. EIS is a powerful technique that measures a battery's impedance across a range of frequencies, offering insights into its internal resistances, capacitances, and inductances. This impedance data enables the identification of changes within the battery's structure, such as degradation processes, electrolyte resistance, and charge transfer resistance, which are critical to understanding and predicting battery capacity over time.

Each feature in this dataset represents a specific real or imaginary impedance value at different frequency points, labelled as "Re" for real components and "Im" for imaginary components. These features capture the complex, frequency-dependent behaviour of the battery and provide a detailed profile of its internal electrochemical processes. The dataset includes:

The significance of EIS in accurate data collection lies in its sensitivity to small structural and chemical changes within the battery, allowing this dataset to offer a high-resolution view of capacity degradation. This granularity provides valuable

input features for sequential models, enabling them to track and predict capacity changes over discharge cycles more accurately.

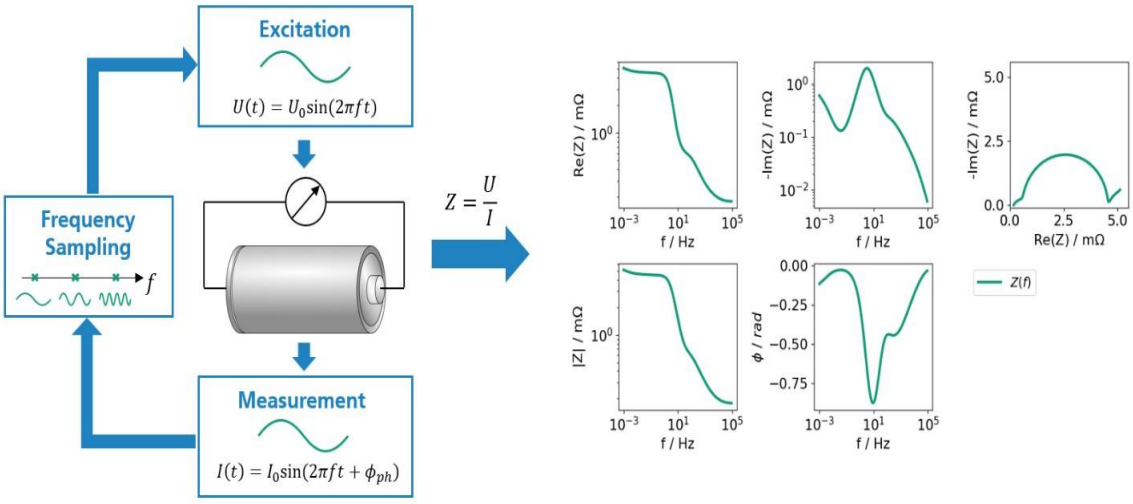

Figure 3.2: Implementation of EIS

3.3 Time-Series Data and Modelling

Time-series data reflects changes over time, with each observation sequentially linked to the preceding one. In predictive modelling, capturing temporal dependencies within this data is essential, as previous values often influence future ones. Time-series models, such as Long Short-Term Memory (LSTM) networks and Neural Ordinary Differential Equations (Neural ODEs), are designed to recognize these dependencies, making them effective for predicting future values based on historical sequences. By learning patterns like trends, seasonality, and temporal dependencies, these models can make informed forecasts for dynamic data, such as battery discharge cycles.

The lithium-ion battery dataset used in this study included capacity and impedance values across multiple discharge cycles, covering various battery types. To enable individualized analysis and enhance model accuracy, the data was segmented by

battery type. Each segment represented a distinct battery, capturing its unique discharge and degradation characteristics. This segmentation approach facilitated type-specific training, allowing the model to better capture the behaviour and degradation patterns of each battery type.

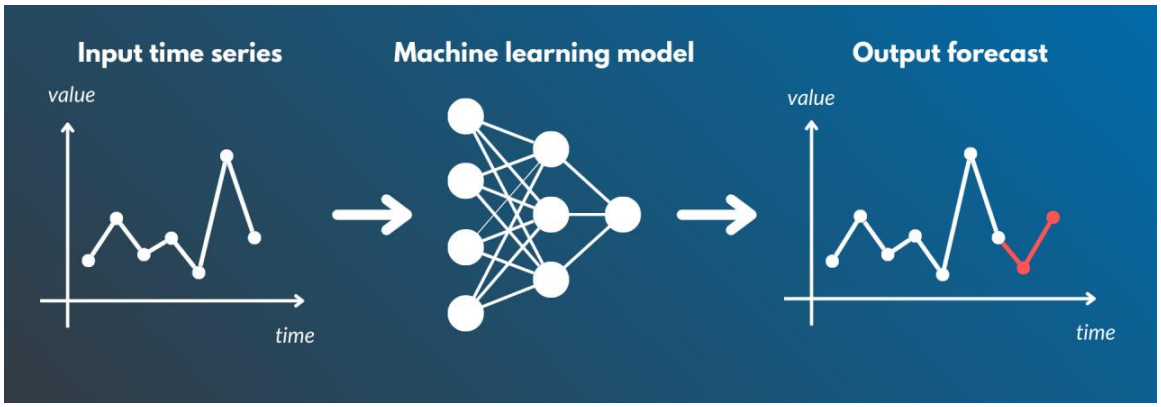

Figure 3.3: Time-Series prediction

3.4 Feature Selection and Data Filtering

The dataset consists of 125 columns:

- Label Column: A categorical column, which identifies the battery type, such as "25C01", "25C02", "25C03" ,, "25C04" ,, "25C05" ,, "25C06" ,, "25C07" ,, "25C08" ,, "35C01", "35C02" ,, "45C01" ,, "45C02".
- Cycle Number: The cycle number column specifies the discharge cycle for each observation, marking the stage in the battery's life. This column is essential for tracking capacity changes across cycles.

Total number of cycle for each battery type :

25C01 = 349
25C02 = 180
25C03 = 201
25C04 = 34
25C05 = 328
25C06 = 212
25C07 = 140
25C08 = 36
35C01 = 325
35C02 = 317
45C01 = 298
45C02 = 309

Figure 3.4 (a): Battery Cycles

- Capacity (Target Column): The Capacity/mA.h column represents the battery's capacity in milliampere-hours at each cycle, serving as the target variable in the predictive modelling process. This capacity value indicates the battery's charge-holding capability, which naturally degrades over time and usage.
- Impedance Columns (Feature Columns): The remaining columns include impedance measurements, obtained at various frequencies, represented as real (Re) and imaginary (Im) components. Each real (Re) and imaginary (Im) value at specific frequencies serves as an input feature, encapsulating the frequency response of the battery. The real values relate to resistance elements, while imaginary values represent capacitive or inductive elements within the battery There are 60 real impedance columns (e.g., 20004.453 Re, 15829.126 Re, etc.) and 60 imaginary impedance columns (e.g., 20004.453 Im, 15829.126 Im, etc.).

After segmentation, non-essential columns such as identifiers and metadata (e.g., battery labels and cycle numbers) were removed, focusing on the electrochemical features, particularly impedance and capacity values (121 columns are left including the target column).

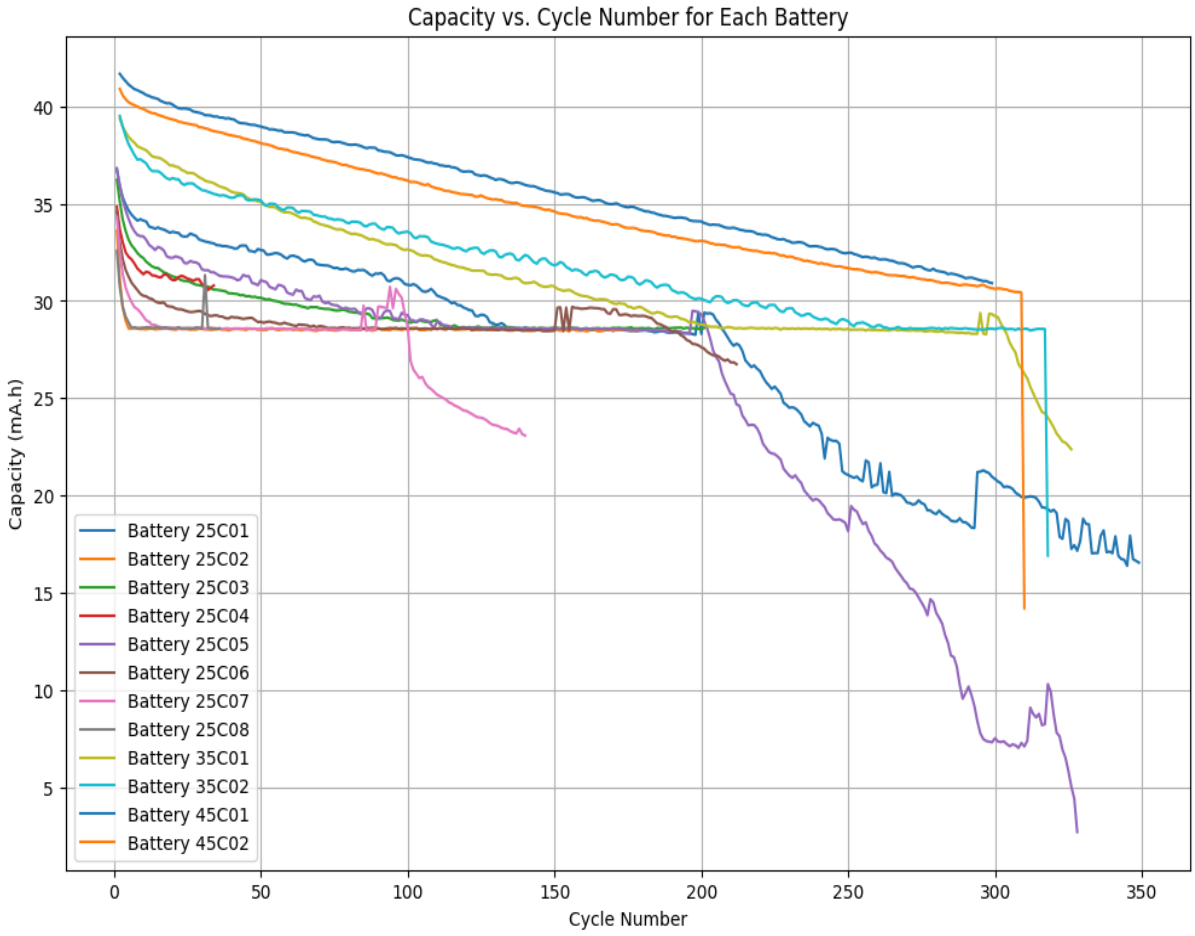

Figure 3.4 (b): Capacity vs. Cycle number

3.5 Sequence Generation for Time-Series Modelling

To prepare the data for sequential models, the filtered data was restructured into sequences of a fixed length (sequence length = 5). Each sequence included a set of historical observations across consecutive cycles, with the target set as the next cycle's capacity value. The sequence-based input-output pairs were then reshaped into formats compatible with time-series neural networks, creating structured datasets for training. This final transformation ensured that the models could fully leverage the temporal structure of the battery discharge data, facilitating accurate predictions of capacity degradation over cycles.

Chapter 4

Deep Learning Algorithms

A Neural Network is a part of Artificial Intelligence that trains the computer, allowing it to recognize patterns like a human brain. It comprises various layers, including input, hidden, and output. The activation of any node depends on the threshold value.

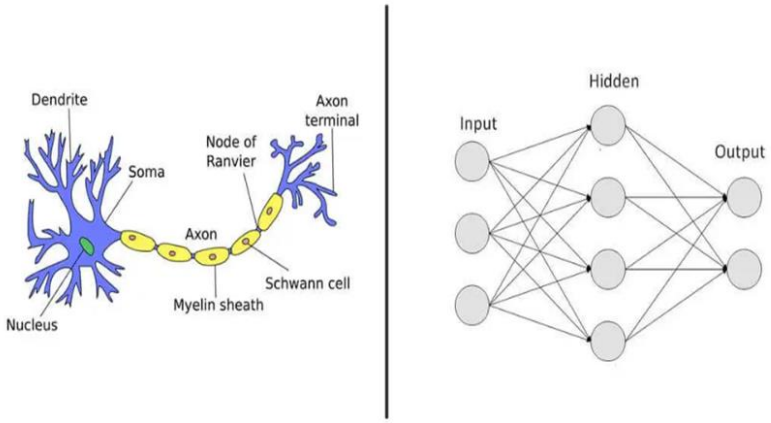

Figure 4: Neural Network Intuition

4.1 Deep learning in Time series modelling

Deep learning, a subset of machine learning, has gained immense popularity in time series forecasting due to its ability to model complex, non-linear relationships in data. Unlike traditional statistical methods, deep learning doesn't require explicit

programming to recognize patterns. Instead, it learns these patterns directly from the data, making it particularly adept at handling the intricacies of time series data.

4.2 Recurrent Neural Networks (RNN)

Recurrent Neural Network (RNN) is a type of Neural Network where the output from the previous step is fed as input to the current step. In traditional neural networks, all the inputs and outputs are independent of each other. For generating sequences, there is a need to remember or store the previous information from the network. Thus RNN came into existence, which solved this issue with the help of a Hidden Layer. The main and most important feature of RNN is its Hidden state, which remembers some information about a sequence. The state is also referred to as Memory State since it remembers the previous input to the network. It uses the same parameters for each input as it performs the same task on all the inputs or hidden layers to produce the output. This reduces the complexity of parameters, unlike other neural networks.

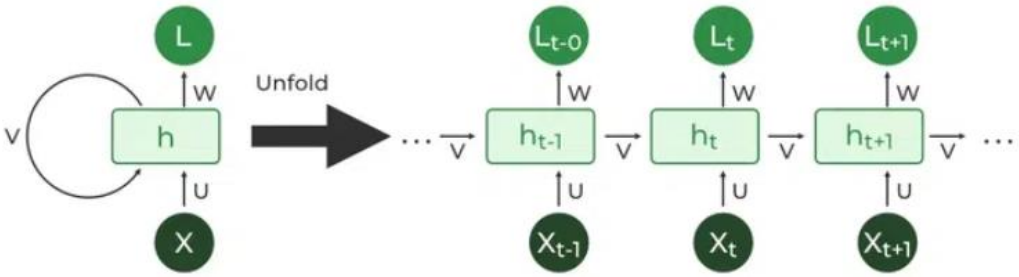

Figure 4.2 (a): RNN architecture

One of the biggest challenges with RNNs is the problem of vanishing or exploding gradients. This occurs when the gradients of the loss function with respect to the parameters of the RNN become very small or very large as they propagate through time. This can make it difficult to train the RNN effectively, as the updates to the parameters will be very small or very large, and the network will not learn effectively.

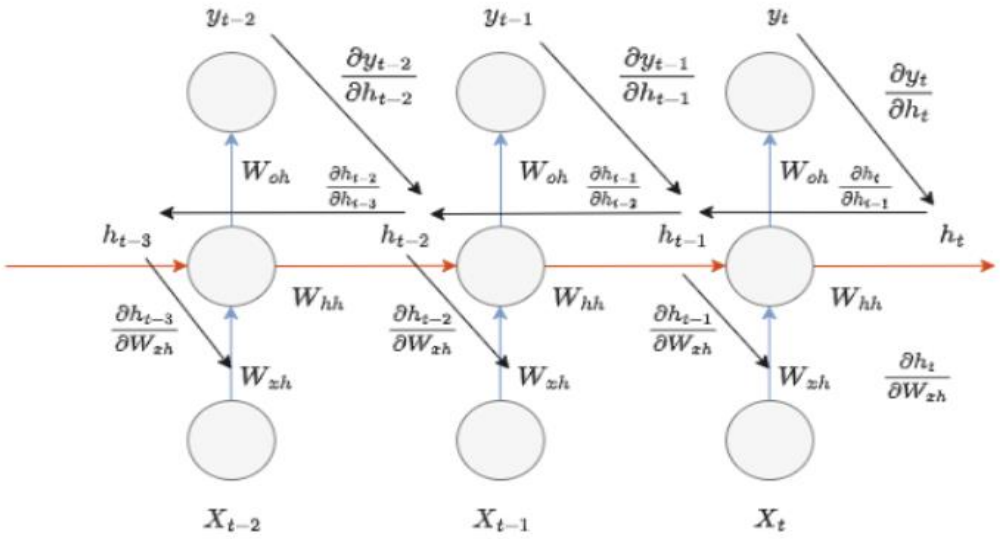

Figure 4.2 (b): RNN BPTT

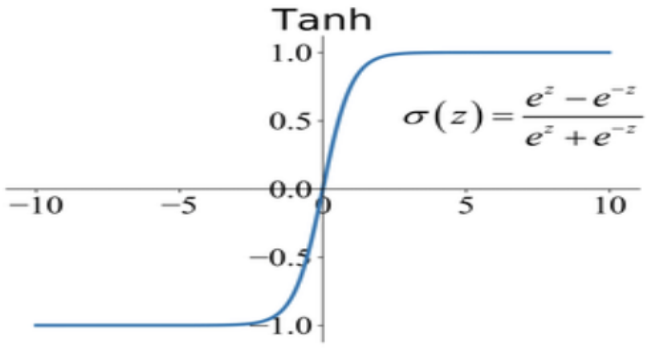

Figure 4.2 (c): Tanh function associated with the hidden state (h)

Derivative of Loss With Respect to Weights in Input Layer W_{xh},

$$\frac{\partial L}{\partial W_{xh}} = \sum_{i=0}^{t} \frac{\partial L}{\partial y_{t-i}} \frac{\partial y_{t-i}}{\partial h_{t-i}} \left(\prod_{j=t-i+1}^{t} \frac{\partial h_{t-j+1}}{\partial h_{t-j}} \right) \frac{\partial h_{t-i-1}}{\partial W_{xh}}$$

Derivative of Loss With Respect to Weigths in the Hidden States W_{hh},

$$\frac{\partial L}{\partial W_{hh}} = \sum_{i=0}^{t} \frac{\partial L}{\partial y_{t-i}} \frac{\partial y_{t-i}}{\partial h_{t-i}} \left(\prod_{j=t-i+1}^{t} \frac{\partial h_{t-j+1}}{\partial h_{t-j}} \right) \frac{\partial h_{t-i-1}}{\partial W_{hh}}$$

Derivative of Loss With Respect to Weights in the Output Layer W_{oh},

$$\frac{\partial L}{\partial W_{oh}} = \sum_{i=0}^{t} \frac{\partial L}{\partial y_{t-i}} \frac{\partial y_{t-i}}{\partial h_{t-i}} \left(\prod_{j=t-i+1}^{t} \frac{\partial h_{t-j+1}}{\partial h_{t-j}} \right) \frac{\partial h_{t-i-1}}{\partial W_{oh}}$$

Derivative of Loss With Respect to Bias in the Hidden States b_h,

$$\frac{\partial L}{\partial b_h} = \sum_{i=0}^{t} \frac{\partial L}{\partial y_{t-i}} \cdot \frac{\partial y_{t-i}}{\partial h_{t-i}} \cdot \frac{\partial h_{t-i}}{\partial b_h}$$

Derivative of Loss With Respect to Bias in the Output layer b_o,

$$\frac{\partial L}{\partial b_o} = \sum_{i=0}^{t} \frac{\partial L}{\partial y_{t-i}} \cdot \frac{\partial y_{t-i}}{\partial b_o}$$

Figure 4.2 (d): Equations associated with BPTT in RNN

4.3 Long – Short Term Memory RNN (LSTM - RNN)

A traditional RNN has a single hidden state that is passed through time, which can make it difficult for the network to learn long-term dependencies. LSTMs model address this problem by introducing a memory cell, which is a container that can hold information for an extended period.

The LSTM architectures involves the memory cell which is controlled by three gates: the input gate, the forget gate, and the output gate. These gates decide what information to add to, remove from, and output from the memory cell.

- The input gate controls what information is added to the memory cell.
- The forget gate controls what information is removed from the memory cell.
- The output gate controls what information is output from the memory cell.

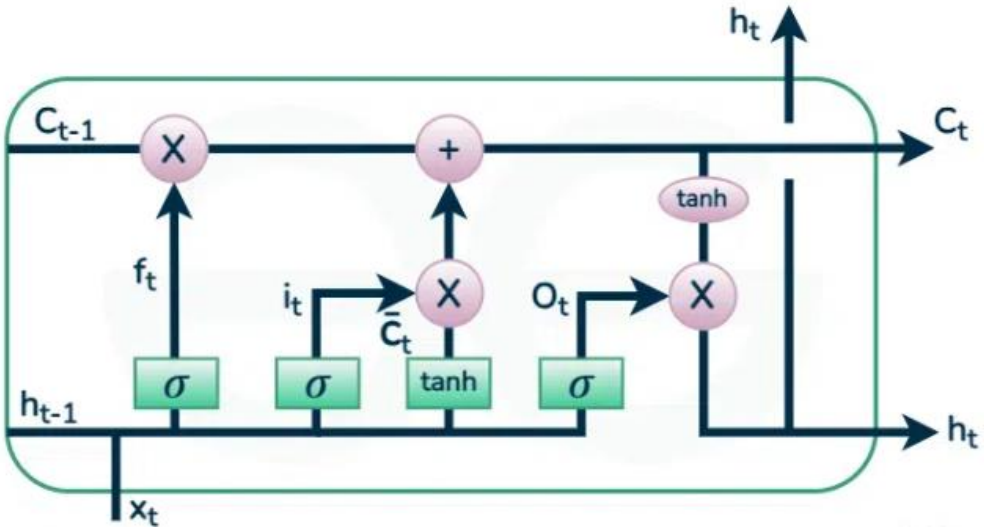

Figure 4.3(a): LSTM cell

Forget Gate

The information that is no longer useful in the cell state is removed with the forget gate. Two inputs *xt* (input at the particular time) and *ht-1* (previous cell output) are fed to the gate and multiplied with weight matrices followed by the addition of bias. The resultant is passed through an activation function which gives a binary output. If for a particular cell state the output is 0, the piece of information is forgotten and for output 1, the information is retained for future use. The equation for the forget gate is:

$$f_t = (W_f[h_{t-1}, x_t] + b_f)$$

Figure 4.3(b): Forget Gate equation

Input gate

The addition of useful information to the cell state is done by the input gate. First, the information is regulated using the sigmoid function and filter the values to be remembered similar to the forget gate using inputs *ht-1* and *xt*. . Then, a vector is created using *tanh* function that gives an output from -1 to +1, which contains all the possible values from ht-1 and *xt*. At last, the values of the vector and the regulated values are multiplied to obtain the useful information. The equation for the input gate is:

$$i_t = (W_i[h_{t-1}, x_t] + b_i)$$

$$\hat{C}_t = tanh(W_c[h_{t-1}, x_t] + b_c)$$

Figure 4.3(c): Input Gate equation (1)

We multiply the previous state by ft, disregarding the information we had previously chosen to ignore. Next, we include it∗Ct which represents the updated candidate values, adjusted for the amount that we chose to update each state value.

$$C_t = f_t C_{t-1} + i_t \hat{C}_t$$

Figure 4.3(d): Input Gate equation (2)

Output gate

The task of extracting useful information from the current cell state to be presented as output is done by the output gate. First, a vector is generated by applying tanh function on the cell. Then, the information is regulated using the sigmoid function and filter by the values to be remembered using inputs h t-1 and x t. At last, the values of the vector and the regulated values are multiplied to be sent as an output and input to the next cell. The equation for the output gate is:

$$o_t = (W_o[h_{t-1}, x_t] + b_o)$$

Figure 4.3 (e): Output Gate equation

Despite these advancements, LSTMs still have some drawbacks which are:

- Data Hungry: Require large labeled datasets
- Over fitting Risk: May not generalize well to unseen data.
- Lack of Interpretability: Considered "black boxes."
- Computationally expensive

4.4 Neural Ordinary Differential Equations (Neural ODEs)

Neural Ordinary Differential Equations (Neural ODEs) offer a fresh perspective on designing neural network architectures by treating network depth as a continuous variable rather than discrete layers. Traditional neural networks process inputs through discrete layers, each performing a distinct transformation on the data as it

passes through. In contrast, Neural ODEs use a continuous model to describe these transformations.

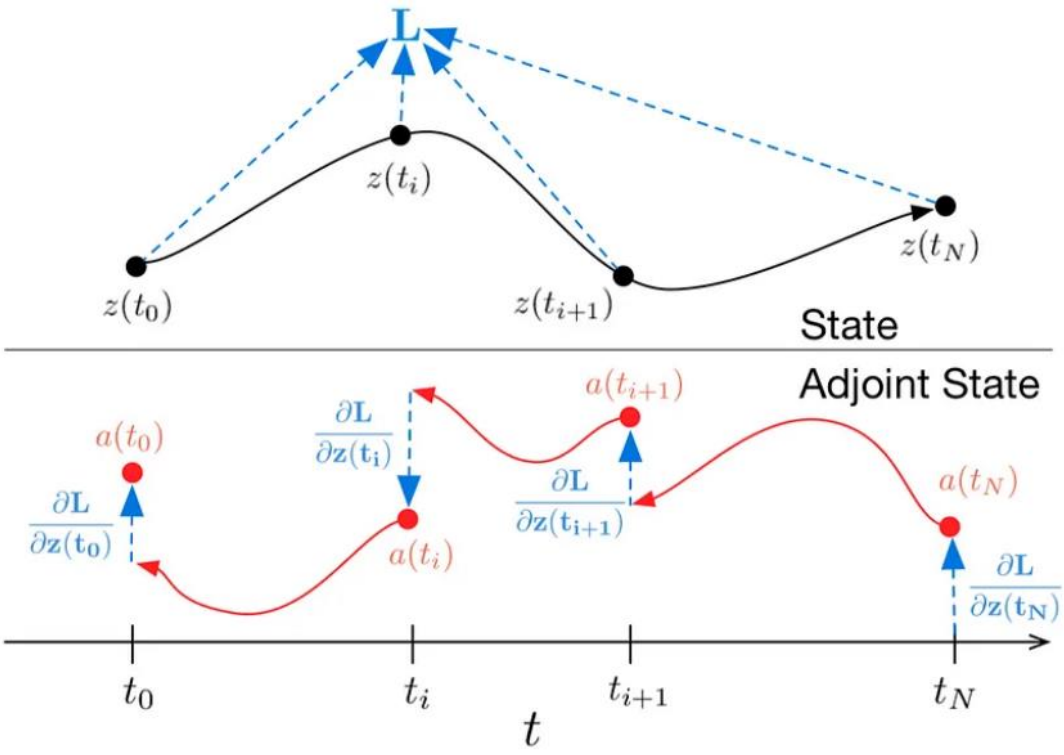

Figure 4.4 (a): Neural ODE architecture

Instead of defining transformations as occurring at set layer intervals, Neural ODEs model the transformation from input to output as a continuous, dynamic process governed by a differential equation. This approach draws on the theory of dynamical systems and treats the evolution of data through the network as a trajectory in a function space, guided by time-dependent changes.

The core of Neural ODEs lies in their primary governing equation:

$$\frac{d\mathbf{h}(t)}{dt} = f(\mathbf{h}(t), t, \theta)$$

Figure 4.4 (b): Neural ODE governing equation

- $h(t)$ represents the state of the system (or the "hidden state") at any time tt. In the context of neural networks, this can be thought of as the activations at any "depth" or point in the processing of the input data.

- f is a function parameterized by θ (the parameters or weights of the neural network), which specifies the rate of change of the hidden state. This function is typically modelled using neural network layers themselves, which are designed to be differentiable with respect to both the hidden state and the parameters.

- t explicitly denotes time or depth within the model, introducing a notion of continuous transformation depth, unlike traditional models where depth is implicitly defined by discrete layers.

Dynamics of Neural ODEs

The dynamics of the hidden state $h(t)$ can be further dissected by examining the specifics of the function f. For instance, if f is modeled as a linear transformation, we have:

$$f(\mathbf{h}(t), t, \theta) = A(t)\mathbf{h}(t) + b(t)$$

Figure 4.4 (c): Hidden State Dynamics

Here $A(t)$ and $b(t)$ can vary with time, providing a linear but time-dependent transformation of the hidden state.

Role of ODE Solvers

ODE solvers play a crucial role in implementing Neural ODEs. They numerically integrate the function f over time from the input to the desired output. This process involves solving the differential equation from an initial state $h(t0)$ (corresponding to the input layer in traditional networks) to $h(t1)$ (corresponding to the output layer), where $t0$ and $t1$ represent the bounds of the integration interval.

Unlike traditional networks that compute outputs layer by layer, ODE solvers adjust the "depth" dynamically:

- **Adaptive Computation:** ODE solvers can adapt the number and size of computational steps they take based on the desired accuracy of the solution. This adaptability can lead to efficiency gains, as the solver can take larger steps in parts of the domain where the solution varies slowly, and smaller steps where the solution changes rapidly.

$$\text{Step size adjustment: } \Delta t = t_{\text{next}} - t$$

This adaptability can lead to efficiency gains, as the solver can take larger steps where the solution varies slowly and smaller steps where rapid changes occur.

Figure 4.4 (d): Step size adjustment

- **Backpropagation Through Solvers:** Training Neural ODEs involves back propagating through these solvers. This is typically achieved via the adjoint

method, which efficiently computes gradients by solving a second, augmented ODE backwards in time, simultaneously with the forward pass.

$$\text{Let } \mathbf{a}(t) = \frac{\partial L}{\partial \mathbf{h}(t)}, \text{ then}$$

$$-\frac{d\mathbf{a}(t)}{dt} = \frac{\partial f(\mathbf{h}(t), t, \theta)}{\partial \mathbf{h}(t)}^T \mathbf{a}(t)$$

Figure 4.4 (e): Calculation of gradients

4.5 Liquid Time Constant (LTC)

Liquid neural nets are an evolution of neural ODEs, which model system dynamics using a series of first order ordinary differential equations (ODEs) coordinated via nonlinear interlinked gates. This is different from normal neural nets, which represent systems via a series of implicit nonlinearities (activation functions).

The derivative of a typical neural ODE's hidden state can be expressed as the following equation:

$$dx(t)/dt = f(x(t)I(t), t, \theta)$$

Figure 4.5 (a): Neural ODE hidden state equation

f is the output of the neural net with parameters *theta*, *x(t)* is the current state, *I(t)* are the inputs at that time *t*

However, they are also susceptible to instability and gradient explosion, which can derail training and make the networks useless. Neural ODEs and other continuous time recurrent architectures have been used widely in the past, and LNN aims to build on their successes and address their failures.

LNNs package linear ODEs a little differently, introducing the liquid time constant (*tau*) and a new bias parameter:

$$dx(t)/dt = -[1/\tau + f(x(t), I(t), t, \theta)]x(t) + f(x(t), I(t), t, \theta)A$$

Figure 4.5 (b): Liquid Neural Network governing equation

The term containing the time constant and the term containing the bias counterbalance each other, helping create the stable behavior characteristic of LNNs. Also note how the time constant term acts directly on this hidden state of the neural net, affecting and bounding the strength of the weights between nodes.

$$\tau_{sys} = \tau/1 + \tau f(x(t), I(t), t, \theta))$$

Figure 4.5 (c): Liquid Time Constant equation

The updated LTC is the previous LTC over 1 plus the LTC multiplied by the neural net output at a specific time step. The LTC "characterizes the speed and coupling sensitivity of an ODE", essentially a constant term that determines how strong connections between various nodes are and how sharp the gradients in each ODE node are. Because the LTC changes with input, LNNs can create new systems to adapt to new shifts in input over time.

4.6 Coefficient of Determination (R-Squared)

R-squared (R^2) is defined as a number that tells you how well the independent variable(s) in a statistical model explains the variation in the dependent variable. It ranges from 0 to 1, where 1 indicates a perfect fit of the model to the data.

$$\text{R-Squared} = 1-\left(\frac{SSR}{SST}\right)=1-\frac{\sum_{i=1}^{n}(yi-\widehat{yi})^2}{\sum_{i=1}^{n}(yi-\overline{yi})^2}$$

where:

SSR is the sum of squared residuals (i.e., the sum of squared errors)

SST is the total sum of squares (i.e., the sum of squared deviations from the mean)

Figure 4.6: R-Squared equation

1. '0' R-Squared value represents a model that does not explain any of the variation in the response variable around its mean. The mean of the dependent variable predicts the dependent variable as well as the regression model.
2. '1' R-Squared value represents a model that explains all the variation in the response variable around its mean

Chapter 5

Results and Conclusion

The models—LSTM, Neural ODE, and Neural ODE with LTC—were trained for 250 epochs and were evaluated on their ability to predict capacity values across different lithium-ion battery types. The R-squared metric was used to assess each model's performance, with higher R-squared values indicating a better fit to the observed data.

LSTM Model with total parameters = 95329 (372.38 KB space)

- R-squared for 25C01: 0.8116213442188918
- R-squared for 25C02: -2.249969310740224
- R-squared for 25C03: 0.9220701494344158
- R-squared for 25C04: 0.8846314622250739
- R-squared for 25C05: 0.7871019924623952
- R-squared for 25C06: -3.0258410357614487
- R-squared for 25C07: 0.6259620391553309
- R-squared for 25C08: 0.5060268132785087
- R-squared for 35C01: 0.38693299341134924
- R-squared for 35C02: 0.5162240518573302
- R-squared for 45C01: 0.9181573912539158
- R-squared for 45C02: 0.8385637939240064

Neural ODE Model with total parameters = 6017 (23.50 KB space)

- R-squared for 25C01: 0.9403743470251013
- R-squared for 25C02: 0.851300733780944
- R-squared for 25C03: 0.6213505222028429
- R-squared for 25C04: 0.18225645357123077
- R-squared for 25C05: 0.986778428637118
- R-squared for 25C06: 0.5626068374885229
- R-squared for 25C07: 0.21491218019955316
- R-squared for 25C08: 0.423122585797846
- R-squared for 35C01: 0.8953666083002053
- R-squared for 35C02: 0.8863437862028222
- R-squared for 45C01: 0.9423434826149811
- R-squared for 45C02: 0.6622459308494442

Neural ODE with LTC Model with total parameters = 19924 (77.83 KB space)

- R-squared for 25C01: 0.9937464434814521
- R-squared for 25C02: 0.9623289668388555
- R-squared for 25C03: 0.8486119009931611
- R-squared for 25C04: -6.987134043466714
- R-squared for 25C05: 0.997959188275109
- R-squared for 25C06: 0.8847557171488794
- R-squared for 25C07: 0.9640858227352456
- R-squared for 25C08: 0.5428551024065432
- R-squared for 35C01: 0.9973647797751999
- R-squared for 35C02: 0.949461517729117
- R-squared for 45C01: 0.9957811638649253
- R-squared for 45C02: 0.8846603395558577

Some of the R-Squared values are negative for the models, so these are assumed to have R-Squared value of 0 for the easiness of comparing.

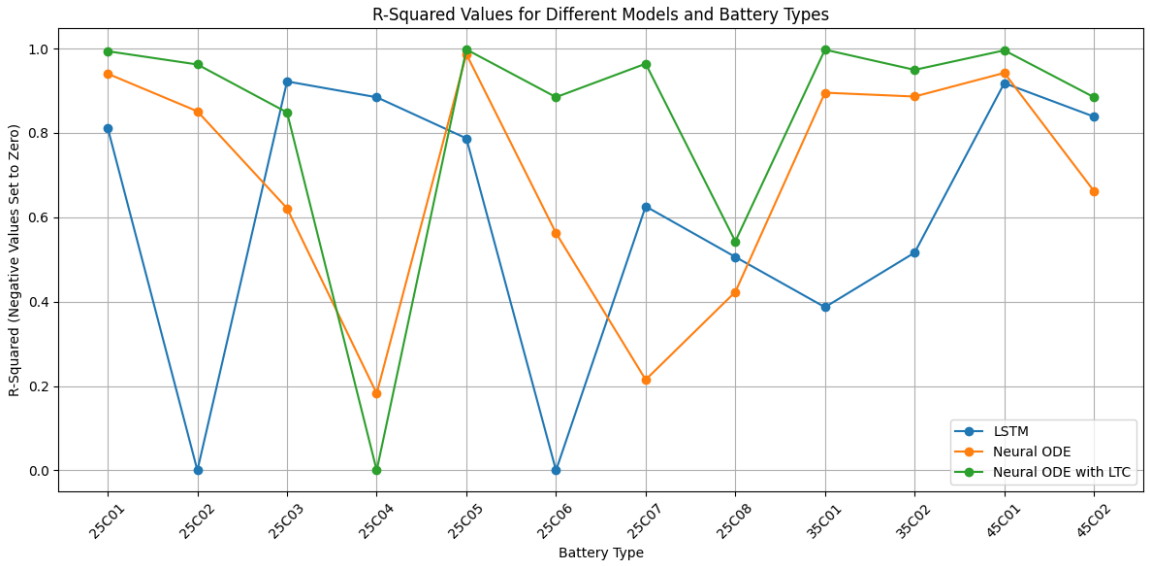

Figure 5: Results

LSTM Results: The LSTM model showed high R-squared values for certain battery types, such as 25C03 (0.92), 45C01 (0.92), and 25C04 (0.88), indicating strong predictive accuracy for these specific types. However, performance varied significantly across batteries, with some batteries, like 25C02 and 25C06, showing negative R-squared values, indicating poor fit and lack of predictive power for those cases.

Neural ODE Results: The Neural ODE model demonstrated moderate to high predictive accuracy, achieving R-squared values above 0.9 for several battery types, including 25C05 (0.99), 45C01 (0.94), and 35C01 (0.89). Compared to the LSTM

model, Neural ODE showed more consistent performance across battery types, with fewer cases of drastically low or negative R-squared values, suggesting better generalization to different battery behaviours.

Neural ODE with LTC Results: The Neural ODE with LTC model yielded the highest R-squared values overall, outperforming both the LSTM and the basic Neural ODE models. For batteries such as 45C01 (0.99), 25C01 (0.99), and 35C01 (0.99), the model demonstrated near-perfect fit, indicating strong predictive capability. However, there was one notable exception in battery 25C04, where the model's performance was poor, with an R-squared of -6.98. Despite this anomaly, the Neural ODE with LTC model showed the best general performance, with consistently high R-squared values across most battery types.

Chapter 6

Bibliography

Meng Zhang, Tao Hu c, Lifeng Wu, Guoqing Kang, Yong Guan - ***A method for capacity estimation of lithium-ion batteries based on adaptive time-shifting broad learning system***. URL: https://doi.org/10.1016/j.energy.2021.120959

Zhipeng Jiao, Jian Ma, Xuan Zhao, Kai Zhang, Qi Han, Zhao Zhang - ***Capacity estimation for lithium-ion batteries with short-time working condition in specific voltage ranges***. URL: https://doi.org/10.1016/j.est.2023.109603

Ralf C. Staudemeyer, Eric Rothstein Morris - ***– Understanding LSTM – a tutorial into Long Short-Term Memory Recurrent Neural Networks.***
URL: https://arxiv.org/pdf/1909.09586

Ramin Hasani, Mathias Lechner, Tsun-Hsuan Wang, Makram Chahine, Alexander Amini, Daniela Rus - ***Liquid Structural State-Space Models.***
URL: arXiv:2209.12951v1

Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, David Duvenaud - ***Neural Ordinary Differential Equation****s*. URL : https://papers.nips.cc/paper/7892-neural-ordinary-differential-equations.pdf

Ramin Hasani,1,3 Mathias Lechner,2* Alexander Amini,1 Daniela Rus,1 Radu Grosu3* - ***Liquid Time-constant Networks****.* URL : https://arxiv.org/abs/2006.04439