Perancangan *Software* untuk Mengestimasikan Nilai *State-of-Health* (SoH) pada Baterai Lithium berdasarkan Nilai Temperatur

[1] Nandhito Adiyatma Rahadi, [2] Oyas Wahyunggoro, Ir., M.T., Ph.D., [3] Dzuhri Radityo Utomo, S.T., M.E., Ph.D.
[1,2,3]*Departemen Teknik Elektro dan Teknologi Informasi, Fakultas Teknik, Universitas Gadjah Mada*

***Intisari*— Capstone Project ini dilaksanakan untuk mengatasi masalah mengenai estimasi parameter kesehatan baterai (*State-of-Health*) berbahan lithium. Baterai yang merupakan komponen penyimpan daya berbahan kimia menyimpan dan mengeluarkan energi berdasarkan reaksi bolak balik yang terjadi di dalamnya, yang pada titik tertentu bahan kimia tersebut dapat menjadi jenuh atau rusak pada penggunaan berulang yang mengakibatkan menurunnya kapasitas penyimpanan energi dari baterai itu sendiri. Penggunaan baterai dengan SoH yang rendah pada perangkat elektronik dapat meningkatkan resiko kerusakan dari perangkat tersebut. Kerusakan bisa terjadi karena kebocoran elektrolit pada baterai, suhu baterai yang terlampau tinggi, atau kegagalan fungsi perangkat akibat dari penurunan kapasitas baterai *(State-of-Charge* menurun drastis secara tiba-tiba yang mengakibatkan perangkat mati tanpa terencana). Meskipun SoH baterai merupakan hal yang penting untuk diperhatikan, namun tidak tersedianya alat/metode untuk mengukurnya secara *cost-efficient* membuat fitur pengukuran SoH masih belum tersedia pada kebanyakan perangkat elektronik.**

Untuk mengatasi permasalahan tersebut, diusulkan solusi untuk mengestimasikan SoH baterai hanya dengan variabel temperatur. Dengan hanya menggunakan satu variabel, maka implementasi pada perangkat akan menjadi lebih sederhana yang dengan demikian akan mengurangi biaya produksi. Solusi yang ditawarkan adalah sebuah program/aplikasi yang digunakan untuk mengestimasikan tingkat SoH dan kapasitas baterai berdasarkan data temperatur yang diunggah. Program bekerja berdasarkan model *Long Short-Term Memory* (LSTM) *Neural Network* yang sudah dilatih menggunakan dataset dari baterai lithium yang telah diuji dan diukur pada kondisi baru (100% SoH) sampai *End of Life* (EoL). Berdasarkan hasil simulasi pengujian, program dapat mengestimasikan nilai kapasitas baterai dengan akurasi yang cukup tinggi (>80%) dengan cara penggunaan yang sederhana.

***Istilah Kunci*—Perangkat Lunak, Estimasi, Baterai Lithium, *State-of-Health*, *Battery Management System*, Temperatur**

I. Pengantar

State of Health (SoH) merupakan salah satu parameter yang digunakan untuk menilai tingkat kesehatan baterai berdasarkan kemampuannya untuk menyimpan energi dibandingkan dengan kondisinya saat baru [1]. Informasi mengenai SoH dapat digunakan untuk melihat kapasitas baterai saat ini, sehingga pengguna dapat menyesuaikan paramater *charging* dan mencegah terjadinya *overcharge*. Kondisi *overcharge* pada baterai dapat menyebabkan berkurangnya umur baterai secara signifikan dan eksitasi panas berlebih yang pada kondisi ekstrim dapat memicu percikan api dan ledakan, atau yang sering disebut dengan istilah *thermal runaway* [2].

Meskipun dianggap penting, implementasi SoH pada perangkat elektronik masih jarang ditemui, karena terhambat oleh kompleksitas perancangan perangkat keras dan biaya, karena metode estimasi SoH konvensional memerlukan banyak sensor dan perangkat *Battery Management System* (BMS). Oleh sebab itu diusulkan metode yang lebih sederhana dengan hanya menggunakan informasi tentang perubahan informasi sebagai estimator tunggal untuk mengestimasikan nilai SoH. Untuk mengimplementasikan solusi tersebut, digunakan metode *Neural Network*, khususnya *Long Short-Term Memory* (LSTM) yang dinilai dapat menjadi solusi alternatif dari metode SoH lain, seperti yang disinggung pada [3].

II. Dasar Teori Pendukung

A. Baterai Lithium-Ion

Baterai Lithium-Ion merupakan salah satu jenis baterai berbahan dasar lithium yang memiliki elektrolit padat dan menggunakan *Lithium Cobalt Oxide* sebagai katoda, dan grafit sebagai anoda. Baterai jenis ini bekerja dengan melakukan proses lithiumisasi dan delithiumisasi seperti yang dapat dilihat berdasrkan persamaan [4]:

$$xLiCoO_2 \leftrightarrow xCoO_2 + xLiC_6 \qquad (2\text{-}1)$$

Karena reaksi tersebut, Baterai Lithium-Ion memiliki kecenderungan untuk menghasilkan panas berdasarkan dengan tingkat reaksi lithiumisasi/delithiumisasi.

B. Pengaruh Temperatur terhadap Baterai

Karena reaksi elektrokimia yang disebutkan berdasarkan Persamaan (2-1), peningkatan suhu dan penggunaan berulang dapat memicu kristalisasi lithium yang *irreversible*. Lithium yang sudah terkristalisasi akan kehilangan kemampuan untuk menyimpan energi dan akan menyebabkan berkurangnya kapasitas baterai, atau sering disebut dengan istilah *Capacity Fading* [2]. Paparan suhu tinggi secara terus menerus juga

Makalah ini dibuat berdasarkan dokumen C-501 *Capstone Project*.

Nandhito Adiyatma Rahadi, Mahasiswa Sarjana Teknik Elektro, Departemen Teknik Elektro dan Teknologi Informasi, Fakultas Teknik, Universitas Gadjah Mada (surel: nandhito.adiyatma@mail.ugm.ac.id)

diketahui dapat memberi dampak kepada elektroda, seperti menumpuknya lapisan *Solid Electrolyte Interphase* (SEI) dan meningkatnya porositas elektroda yang mengurangi luas kontak elektroda baterai dengan beban atau dengan catu daya, sehingga dapat meningkatkan resistansi baterai [1].

C. Battery Management System

Battery Management System (BMS) merupakan hasil implementasi pada sistem tertanam yang digunakan dalam upaya untuk menjaga parameter kerja baterai seperti arus, tegangan, dan suhu berada pada titik aman dan bekerja dalam kondisi yang ideal agar baterai dapat digunakan dalam jangka waktu yang lebih panjang [1]. Untuk mewujudkan hal tersebut BMS bekerja dengan melakukan proses monitoring, dan pengendalian sebagai respon data monitor dengan menyesuaikan parameter-parameter kerja yang berpengaruh dalam proses penggunaan maupun pengisian ulang baterai [4].

D. State of Health

State of Health (SoH) adalah parameter BMS yang mewakili derajat penuaan (*aging*) pada baterai. Dengan C_a adalah kapasitas maksimum baterai pada saat ini, C_r adalah kapasitas maksimum baterai pada *cycle* ke-0 (*design capacity*), R_a adalah hambatan dalam baterai pada saat ini, dan R_r adalah hambatan dalam baterai pada *cycle* ke-0, maka SoH dapat diestimasikan dalam persentase berdasarkan kedua Persamaan (2-2) dan Persamaan (2-3) [5]:

$$SoH = \frac{C_a}{C_r} \times 100\% \quad (2\text{-}2)$$

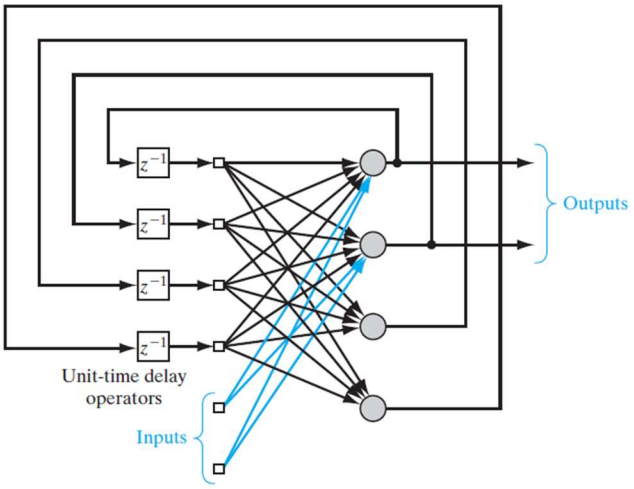
$$SoH = \frac{R_a - R_r}{R_r} \times 100\% \quad (2\text{-}3)$$

F. Neural Network

Artificial Neural Network (ANN/NN) atau yang disebut juga sebagai Jaringan Syaraf Tiruan (JST) merupakan sebuah algoritma yang dirancang berdasarkan cara kerja jaringan syaraf manusia. Informasi yang diterima akan masuk melewati *input layer*, yang kemudian akan diproses pada *hidden layer* yang terdiri atas *neuron,* lapisan *dropout* dan lapisan pemroses akhir seperti *fully-connected layer* dan *softmax layer*, hasil komputasi kemudian akan diproses pada *output layer* untuk memberikan hasil akhirnya. *Input layer* pada NN sebanding dengan dendrit pada sel syaraf, berisi cabang-cabang (*nodes*) yang menerima masukan. *Hidden layer* sebanding dengan nukleus dan batang akson yang berisikan model untuk mengolah *synapse* untuk kemudian dikirimkan pada *output layer* yang sebanding dengan cabang akson pada sel syaraf [6].

G. Recurrent Neural Network

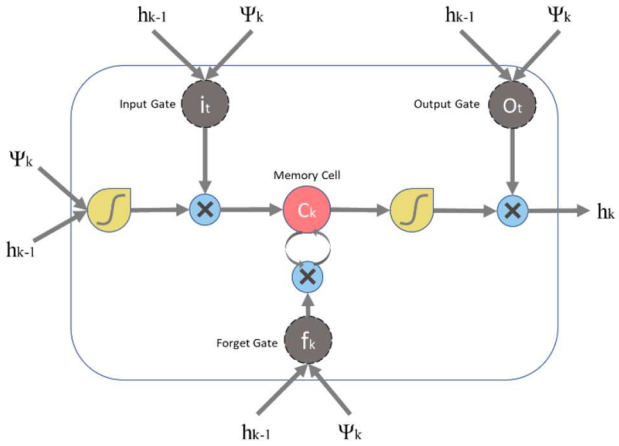
Recurrent Neural Network (RNN) adalah salah satu metode NN yang menggunakan operator tunda (z-1) pada keluaran *node* yang kemudian dialihkan kembali ke masukan *node* sebagai umpan balik. Dengan demikian, keluaran baru akan dipengaruhi oleh keluaran sebelumnya, sehingga dapat di implementasikan pada sistem yang berjalan secara kontinu dan tanpa pengawasan (*unsupervised*)), yang ilustrasinya dapat dilihat pada Gambar 2.1 [7].



Gambar 2.1. Ilustrasi jaringan RNN dengan lapisan operator tunda, lapisan masukan, satu lapisan tersembunyi sebagai pemroses lanjutan nilai masukan, dan lapisan keluaran [7].

H. Long Short-Term Memory

Berdasarkan metode RNN, dikembangkanlah metode *Long-Short-Term Memory* (LSTM) dengan menambahkan fitur untuk menghapus memori (keluaran terdahulu) pada titik tertentu pada proses estimasi untuk mencegah penurunan akurasi yang diakibatkan oleh keluaran terdahulu yang sudah tidak relevan lagi dengan kondisi saat ini [8]. Contoh ilustrasi jaringan LSTM dapat dilihat pada Gambar 2.2.



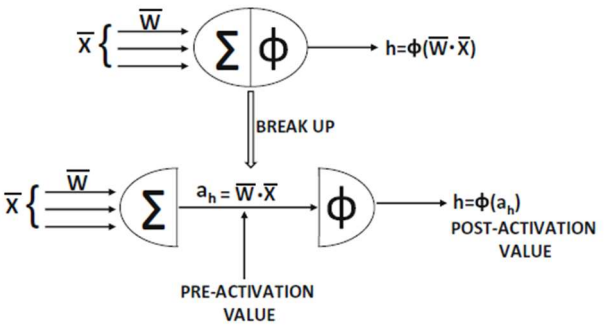
Gambar 2.2. Ilustrasi *Long-Short-Term Memory* (LSTM) pada *Recurrent Neural Network* (RNN) dengan gerbang tambahan untuk mengaktifkan lapisan masukan, masukan keluaran, dan fungsi untuk menghapus memori [8].

I. Activation Function

Activation function dalam konfigurasi NN berfungsi untuk memperkenalkan non-linearitas ke lapisan keluaran NN. Dengan adanya kapabilitas untuk mengestimasikan nilai yang non-linear, fitur dan setiap perubahan dari hasil keluaran dapat menjadi lebih terlihat. Untuk membentuk keluaran seperti yang dimaksud, beban dari fungsi aktivasi akan ditambahkan nilai

estimasi sebelum sampai di lapisan keluaran berdasarkan Persamaan (2-4) dengan Φ adalah fungsi aktivasi. Konfigurasi jaringan dengan fungsi aktivasi dapat dilihat pada Gambar 2.3. Beberapa fungsi aktivasi yang digunakan adalah *linear, sign, sigmoid*, *hyperbolic tangent* (tanh), dan *Rectified Linear Unit* (ReLU) [6].

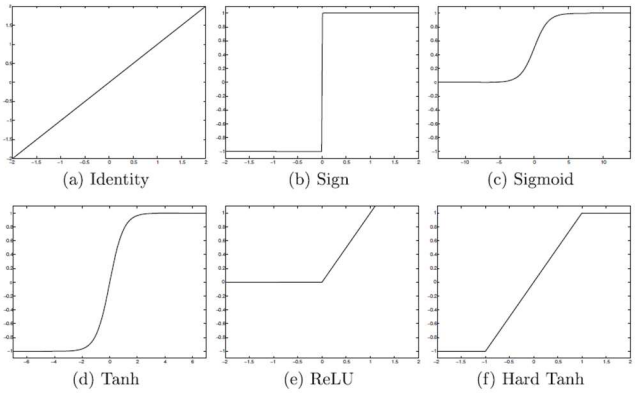
$$\hat{y} = \Phi\left(\overline{W} \cdot \overline{X}\right) \tag{2-4}$$



Gambar 2.3. Implementasi fungsi aktivasi pada NN [6].

TABEL 2.1
MACAM FUNGSI AKTIVASI [6]

Name	Range	Activation Function
Linear/Identity	(-∞,∞)	-
Sign	(0/1)	$\Phi(v) = sign(v)$
Sigmoid	(0,1)	$\Phi(v) = \frac{1}{1+e^{-v}}$
Hyperbolic Tangent (tanh)	[-1,1]	$\Phi(v) = \frac{e^{2v}-1}{e^{2v}+1}$
Rectified Linear Unit (ReLU)	(0,∞)	$\Phi(v) = \max\{v, 0\}$
Hard Tanh	[-1,1]	$\Phi(v) = \max\{\min[v, 1], -1\}$



Gambar 2.4. Gambaran berbagai fungsi aktivasi [6].

III. ANALISIS STUDI PUSTAKA KUNCI DAN PEMILIHAN METODE

A. Pemilihan Metode Estimasi

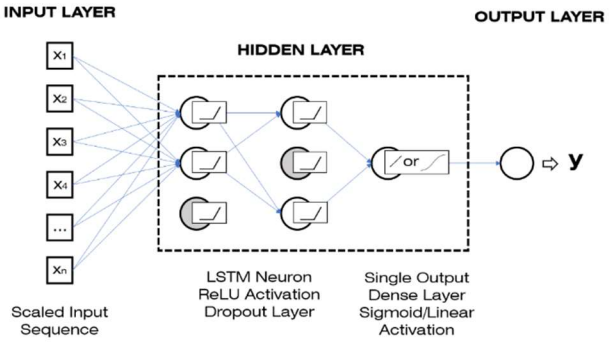
Berdasarkan informasi yang sudah dijelaskan pada bagian Dasar Teori Pendukung, diketahui bahwa untuk melakukan estimasi, perhitungan berdasarkan kapasitas baterai, seperti yang dapat dilihat pada Persamaan (2-2), memiliki akurasi yang lebih tinggi dibandingkan dengan perhitungan menggunakan nilai hambatan dalam seperti yang dapat dilihat pada Persamaan (2-3) [9]. Sehingga metode estimasi menggunakan perbedaan nilai kapasitas digunakan untuk mengestimasikan nilai SoH.

Sementara itu, keseluruhan proses estimasi akan berjalan menggunakan metode *Recurrent Neural Network* (RNN) khususnya LSTM, karena dianggap cocok untuk digunakan dalam menjalankan fungsi regresi untuk mengestimasikan nilai perubahan temperatur untuk mengestimasikan jumlah *cycle* RUL baterai. Selain itu, fungsi tambahan LSTM yang berupa *forget gate* dapat dimanfaatkan untuk proses estimasi yang berjalan secara kontinu atau untuk masukan dengan deret informasi yang panjang, karena memiliki fitur untuk menghapus memori pada titik tertentu pada proses estimasi untuk mencegah penurunan akurasi yang diakibatkan oleh keluaran lama yang sudah tidak relevan [8].

B. Pemilihan Konfigurasi Lapisan Neural Network

Secara umum, jaringan akan terdiri atas lapisan masukan, lapisan tersembunyi (*neuron* dan *fully-connected layer*), dan lapisan keluaran. Pada lapisan masukan, jumlah *input nodes* akan disesuaikan secara bertahap secara eksperimental untuk menghasilkan bentuk keluaran yang diingingkan. Pada lapisan tersembunyi, terdiri atas *neuron* dengan metode RNN-LSTM dengan jumlah nodes yang disesuaikan dengan jumlah nodes pada lapisan masukan, metode tersebut dipilih berdasarkan analisa dasar teori pada bagian yang sebelumnya dan dianggap sebagai metode yang paling cocok untuk digunakan untuk mengestimasikan nilai dengan masukan berbentuk data sekuensial. Lapisan tersembunyi selain memuat *neuron*, akan ditambahkan lapisan *Dropout* dengan persentase yang minimal namun fungsional untuk mengenalkan nonlinearistas selama masa training yang diharapkan akan meningkatkan *robustness* dari jaringan.

Data yang akan digunakan dalam proses latihan akan berbentuk deret dengan panjang yang disesuaikan dengan jumlah *nodes* pada lapisan masukan, dan dibuat kedalam skala 0 sampai 1 untuk mencegah *Gradient Explosion* yang umum terjadi pada jaringan yang memproses nilai masukan yang besar. Sehingga untuk menyesuaikan dengan bentuk masukan, fungsi aktivasi ReLU juga akan digunakan sebagai fungsi aktivasi pada lapisan tersembunyi karena kecenderungannya untuk dapat konvergen dengan lebih cepat. Sementara itu *fully-connected layer* atau *dense layer* juga akan ditambahkan sebelum lapisan keluaran untuk menyatukan keseluruhan hasil proses dari lapisan tersembunyi yang di sisi lain juga digunakan untuk menyesuaikan bentuk keluaran menjadi keluaran tunggal dengan fungsi aktivasi linear (karena cocok digunakan untuk regresi), namun fungsi aktivasi *sigmoid* juga akan dipertimbangkan (karena memiliki bentuk keluaran yang sama dengan data masukan yang sudah diskala dan bentuk keluaran yang diinginkan) [10].



Gambar 3.1. Ilustrasi rancangan *Neural Network* berdasarkan metode yang dipilih.

IV. Detail Implementasi

A. Luaran Capstone Project beserta Spesifikasinya

TABEL 4.1
LUARAN YANG DIJANJIKAN

Jenis Luaran	Deskripsi Luaran
Dataset parameter baterai (*dataset* untuk melatih jaringan LSTM)	*Dataset* parameter akan berisi informasi mengenai tegangan, arus, temperatur, dan kapasitas baterai yang di-*sampling* tiap detik. *Dataset* terpisah untuk masing-masing *cycle,* dan diujikan sebanyan 600 *cycle* dengan format CSV. *Dataset* yang didapat sebagai hasil pengujian akan diolah untuk mengambil parameter kapasitas maksimum dan temperatur maksimum untuk masing-masing *cycle*, membentuk satu file CSV baru dengan panjang 600 baris dengan 2 kolom (temperatur maksimum dan kapasitas maksimum).
Model LSTM yang sudah dilatih dengan menggunakan *dataset* latih	Rancangan model *Neural Network* yang menggunakan metode LSTM dan sudah dilatih menggunakan *dataset* latih. *Pre-trained model* akan disimpan dalam file terpisah dengan format TensorFlow SavedModel.
Prototipe s*oftware* untuk mengestimasikan nilai SoH berdasarkan nilai temperatur	*Software* yang dapat digunakan untuk mengestimasikan nilai SoH suatu baterai dengan menggunakan catatan nilai temperaturnya. *Software* berbasis Python dengan memanfaatkan *Library* Keras
(lanjutan) **Jenis Luaran**	**(lanjutan)** **Deskripsi Luaran**
Prototipe s*oftware* untuk mengestimasikan nilai SoH berdasarkan nilai temperatur	(lanjutan) sebagai *Neural Network Toolbox* nya. GUI untuk *software* ini dirancang menggunakan Gradio API dengan mengemas keseluruhan algoritma estimasi kedalam satu fungsi yang kemudian ditampilkan pada GUI sebagai fitur.
Hasil simulasi lengkap	Hasil simulasi pengujian prototipe *software* dengan menggunakan beberapa *dataset* uji. Hasil simulasi memuat hasil estimasi, akurasi estimasi, dan visualisasi hasil estimasi dan *error* pada *software.* Disimulasikan di Jupyter Notebook.

TABEL 4.2
SPESIFIKASI SOFTWARE YANG DIHARAPKAN

Spesifikasi	Satuan	Standar	Keterangan
Ukuran program/*software* (Jupyter Notebook)	Megabytes (MB)	< 1 MB	Tidak termasuk paket dan *library* penunjang yang terpasang di komputer
Kecepatan Estimasi	Detik (s)	< 30 s	Ditampilkan di GUI selama proses berlangsung
Akurasi Estimasi	Persen (%)	> 80 %	Akurasi estimasi kapasitas maksimum

B. Spesifikasi Lingkungan Pengujian dan Objek Pengujian

Berdasarkan luaran yang dijanjikan yang dapat dilihat pada Tabel 4.1, akan dijabarkan dengan lebih detail mengenai spesifikasi objek uji (baterai lithium-ion XTAR 16340), parameter dan kondisi pengujian (pengaturan cycling pada data logger SKYRC MC3000), serta spesifikasi perangkat pengujiannya.

TABEL 4.3
SPESIFIKASI KOMPUTER DAN VERSI LIBRARY

Spesifikasi	Satuan	Standar	Keterangan
Prosesor (Intel® Core™ i5-10210U)	*Clock Speed* (GHz)	1,6 GHz *Base*, 4,2 GHz *Turbo*	4 *Core* – 8 *Thread*
RAM	Gigabytes (GB)	8 GB	DDR4
Memory	Gigabytes (GB)	512 GB	SSD
Python Version	-	3.9.4	-
TensorFlow Version	-	2.1	Versi harus sesuai
Keras Version	-	2.3.1	Versi harus sesuai
h11 Version	-	0.12	Versi harus sesuai
httpcore Version	-	0.15.0	Versi harus sesuai
Pandas, glob2, grpcio, numpy, statsmodels, matplotlib, sklearn, fsspec, gradio	-	Terbaru	Tidak memerlukan versi tertentu secara spesifik, versi yang digunakan adalah versi terbaru (2022)
Jupyter Notebook	-	Terbaru	(2022)

TABEL 4.4
KONFIGURASI CYCLING SKYRC MC3000

Parameter	Satuan	Standar	Keterangan
Tegangan maksimal	Volt (V)	4,2 V	Jika nilai tercapai, maka proses berhenti dan berlanjut ke fase selanjutnya.
Tegangan minimal	Volt (V)	3,3 V	Jika nilai tercapai, maka proses berhenti dan berlanjut ke fase selanjutnya.
Arus keluar minimum	Ampere (A)	0,05 A	Jika nilai tercapai, maka proses berhenti dan berlanjut ke fase selanjutnya.
Arus masuk minimum	Ampere (A)	0,02 A	Jika nilai tercapai, maka proses berhenti dan berlanjut ke fase selanjutnya.
Arus keluar maksimum	Ampere (A)	1 A	-
Arus masuk maksimum	Ampere (A)	3 A	-

TABEL 4.5
SPESIFIKASI BATERAI LITHIUM-ION XTAR 16340

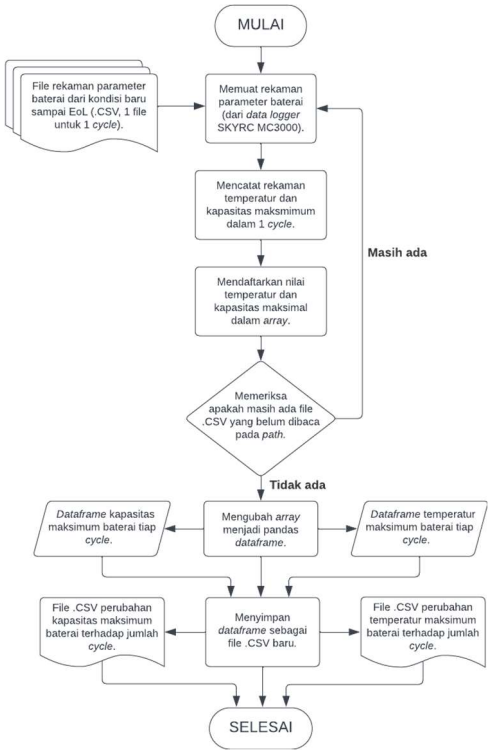
Spesifikasi	Satuan	Standar	Keterangan
Dimensi	Milimeter (mm)	16 mm × 340 mm	Baterai silinder, *button top*
Kapasitas desain	Miliampere Hour (mAh)	650 mAh	-
Tegangan minimum	Volt (V)	3,7 V	-
Tegangan maksimum	Volt (V)	4,2 V	-
Arus maksimum	Ampere (A)	1,5 A	*PCB Protected*
Bobot	Gram (gr)	18,2 gr	-

C. Batasan Masalah

Pada proyek yang dilakasanakan mengenai metode estimasi SoH menggunakan *Neural Network*, secara spesifik LSTM, akan ditentukan beberapa batasan masalah. Adapun batasan-batasan dan potensi masalah yang sudah disinggung akan disampaikan dalam beberapa poin berikut ini:

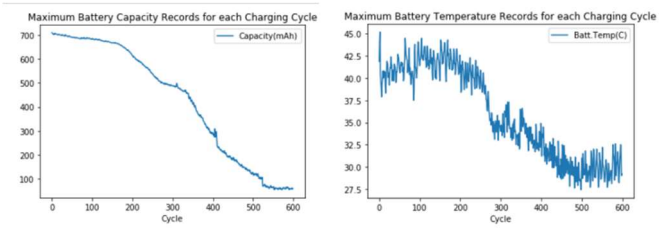
1. Pengambilan dataset parameter kerja baterai dilakukan untuk objek uji coba dan perangkat pengujian yang spesifik, yaitu menggunakan baterai lithium-ion XTAR 16340 (650 mAh) sebagai objek uji coba dan SKYRC MC3000 sebagai perangkat pengujian (*data logger*). Sehingga segala informasi yang dimuat dalam dataset adalah parameter yang berhasil diukur dengan kondisi-kondisi yang spesifik.
2. Dataset diharapkan akan memiliki rekaman temperatur maksimal yang terus menurun setiap untuk setiap *cycle* nya, kenaikan temperatur akan dianggap sebagai galat dan bagian dari dataset dimana temperatur baterai naik dapat dianggap sebagai bagian penyesuaian (*conditioning*) baterai yang tidak disertakan dalam pengujian.
3. Terkait dengan poin 2, apabila dataset yang digunakan dalam pengujian memiliki nilai yang fluktuatif, maka harus dilakukan proses dekomposisi dan mengambil komponen tren sebagai dataset yang siap digunakan untuk proses pengujian.
4. Dataset pengujian disusun atas potongan informasi dari dataset yang digunakan dalam proses pelatihan.
5. Seluruh versi aplikasi, plug-in, maupun library yang digunakan harus sesuai dengan apa yang ditampilkan pada Tabel 4.3.

D. Detail Rancangan – Pengumpulan Dataset



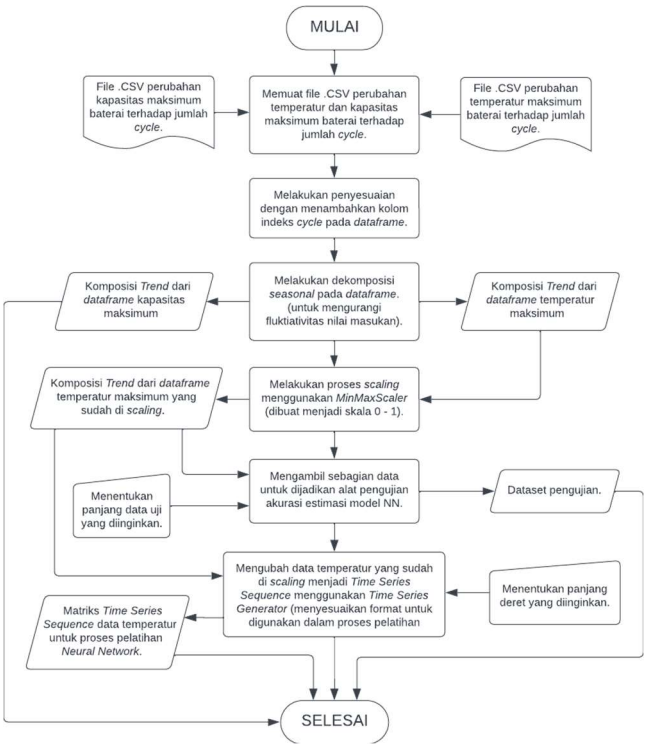
Gambar 4.1. Diagram alur pemrosesan awal *dataset*.

Metode pengumpulan *dataset* dirancang untuk menyatukan file CSV hasil dari proses *data logging* yang dilakukan dengan perangkat SKYRC MC3000 dan aplikasi *data logging* SKYRC MC3000 Monitoring V1.05. Masing-masing dari file CSV mewakili setiap *cycle* rekaman parameter kerja baterai (tegangan, arus, kapasitas, dan temperatur). Namun dari keempat parameter yang sudah disebutkan, yang akan digunakan dalam proses estimasi adalah parameter temperatur dan kapasitas pada sesi *charging* saja. *Dataset* yang berhasil di kumpulkan seperti yang dapat dilihat pada Gambar 4.2.



Gambar 4.2. Perubahan kapasitas maksimum baterai (kiri) dan perubahan temperatur maksimum baterai (kanan) pada proses *charging* seiring dengan bertambahnya jumlah *cycle*.

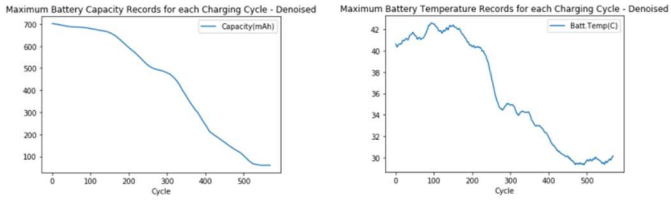
E. Detail Rancangan – Pengolahan Awal Dataset



Gambar 4.3. Diagram alur pemrosesan lanjutan *dataset* menjadi matriks *Time Series Sequence*.

Dari *dataset* yang berhasil disusun pada bagian Pengumpulan *Dataset*, kemudian dapat diolah untuk menghasilkan *dataset* yang bisa digunakan untuk melatih jaringan LSTM. Untuk merubah *dataset* sumber menjadi *dataset* latih, pengguna harus

mengurangi komponen yang membuat nilai menjadi fluktuatif, sehingga *dataset* akan diubah menjadi *dataframe* untuk di dekomposisi menggunakan fungsi seasonal_decompose dari *library* statsmodels untuk digunakan komponen tren nya. Hasil dekomposisi *dataframe* temperatur dan kapasitas (tren) dapat dilihat pada Gambar 4.4. Tahap selanjutnya adalah untuk membuat nilai pada *dataframe* menjadi skala dengan jangkauan 0 sampai 1 mengunakan fungsi minmaxscaler dari *library* sklearn. Tahapan ini diperlukan untuk mencegah terjadinya *Gradient Explosion* pada proses dengan masukan bernilai besar untuk jaringan LSTM. Pada tahapan yang terakhir, *dataframe* yang sudah berbentuk skala dengan format *array* akan dibuat menjadi *Time Series Sequence* dengan menggunakan fungsi TimeseriesGenerator dari *library* Keras.



Gambar 4.4. Komponen tren dari *dataframe* perubahan kapasitas (kiri) dan *dataframe* perubahan temperatur (kanan).

F. Detail Rancangan – Perancangan Model LSTM

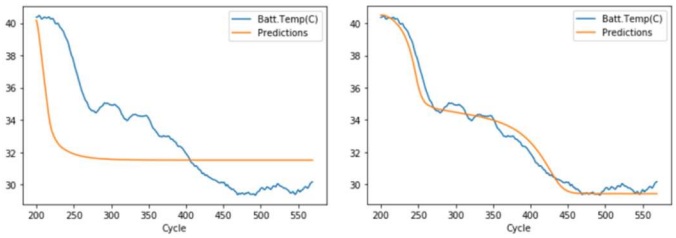
Dengan menggunakan *library* dari Keras, model *Neural Network* dapat dirancang berdasarkan konfigurasi yang sudah disampaikan pada bagian Analisis Studi Pustaka dan Pemilihan Metode. Model LSTM dirancang dengan konfigurasi yang dapat dilihat pada Gambar 4.5 dan dilatih menggunakan *dataset* yang sudah selesai di proses pada bagian Pengolahan Awal Dataset sebanyak 30 *epochs*.

```
Model: "sequential_5"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_5 (LSTM)                (None, 20)                1760
_________________________________________________________________
dropout_5 (Dropout)          (None, 20)                0
_________________________________________________________________
dense_5 (Dense)              (None, 1)                 21
=================================================================
Total params: 1,781
Trainable params: 1,781
Non-trainable params: 0
_________________________________________________________________
```

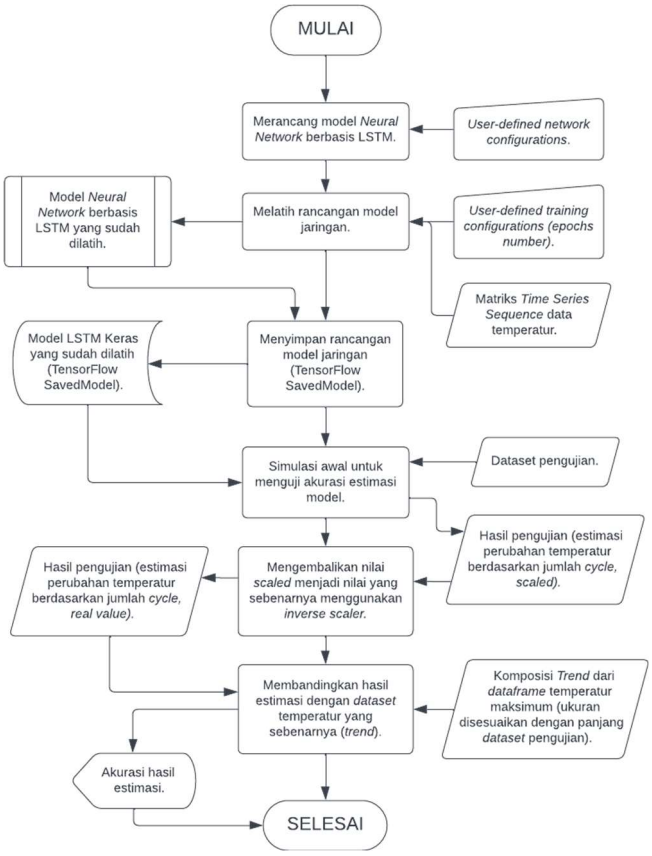
Gambar 4.5. Konfigurasi Model LSTM

Model LSTM yang sudah kemudian diujikan menggunakan *dataset* simulasi awal yang terdiri atas 370 *cycle* terakhir dari *dataset* temperatur (sumber). Model dicobakan dua kali, yaitu saat model menggunakan fungsi aktivasi linear dan *sigmoid* untuk lapisan *dense* nya. Hasil menunjukkan bahwa fungsi aktivasi *sigmoid* lebih cocok untuk digunakan dalam model ini, seperti yang dapat dilihat pada Gambar 4.6. Dengan menggunakan metode *Mean Absolute Percentage Error* (MAPE), dapat diketahui bahwa model dapat mengestimasikan kurva perubahan nilai temperatur dengan *error* sebesar 1,15%.

Keseluruhan proses yang telah dideskripsikan, diilustrasikan pada diagram alir yang dapat dilihat pada Gambar 4.7.



Gambar 4.6. Grafik perbandingan nilai temperatur hasil estimasi pada model yang menggunakan fungsi aktivasi linear (kiri) dan sigmoid (kanan) pada lapisan *dense* nya.

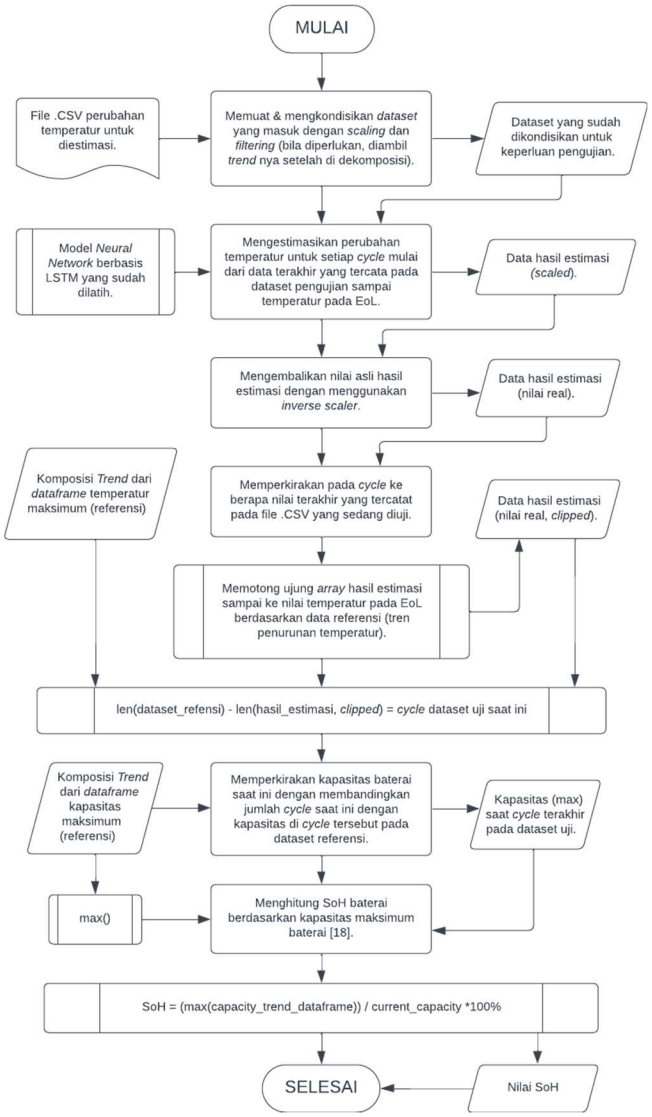


Gambar 4.7. Diagram alur keseluruhan proses perancangan, pelatihan, dan simulasi awal jaringan LSTM untuk mengestimasikan kurva perubahan temperatur.

G. Detail Rancangan – Pengaplikasian Model LSTM

Berdasarkan estimasi kurva perubahan nilai temperatur pada bagian yang sebelumnya, proses yang sama dapat dilakukan untuk iterasi yang lebih panjang, cukup panjang untuk dapat mencakup keseluruhan panjang *dataset* sumber. Hasil estimasi berlebih kemudian dapat dipotong sesuai dengan nilai minimum bacaan temperatur pada *dataset* sumber yang dianggap mewakili titik *End-of-Life* (EoL) dari baterai yang dijadikan sebagai objek pengujian. Panjang *dataframe* yang

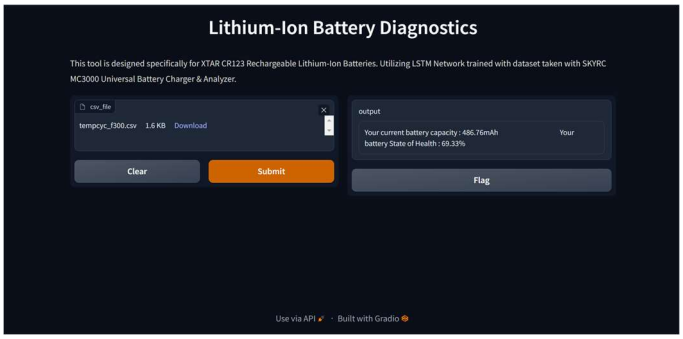
sudah dipotong tersebut mewakili jumlah *cycle* yang tersisa, atau disebut dengan istilah *Remaining Useful Life* (RUL). Dengan mengurangi jumlah *cycle* pada *dataset* sumber dengan estimasi nilai *cycle* RUL, pengguna dapat mengetahui jumlah *cycle* baterai saat ini, dan jika dibandingkan dengan *dataset* kapasitas, maka besaran nilai kapasitas aktual baterai dapat diketahui, yang dengan demikian pula SoH dapat diketahui berdasarkan Persamaan (2-2) dengan menggunakan nilai bacaan kapasitas tertinggi pada *dataset* kapasitas sebagai kapasitas desain. Keseluruhan proses yang sudah dideskripsikan diilustasikan pada diagram alur yang dapat dilihat pada Gambar 4.8.



Gambar 4.8. Diagram alur keseluruhan proses estimasi nilai SoH dari hasil estimasi RUL *cycle* baterai.

G. Detail Rancangan – Perancangan GUI

Graphical User Interface (GUI) untuk *software* estimasi SoH dirancang menggunakan Gradio API, sebagai salah satu *toolkit* perancangan antarmuka yang terkenal untuk proyek yang berkaitan dengan *Neural Network*. Dengan menggunakan Gradio, pengguna dapat merancang antarmuka sesedarhana mungkin. Pada *software* estimasi SoH, antarmuka akan dirancang untuk dapat bekerja hanya dengan satu tombol saja. Pengguna dapat mengunggah file CSV sebagai masukan dan menekan tombol "Submit" untuk menjalankan proses estimasi. Gradio API bekerja dengan menjalankan satu fungsi yang berisikan keseluruhan algoritma estimasi SoH yang sudah dirancang sebelumnya. Hasil pemrosesan akan menampilkan nilai estimasi SoH dan kapasitas seperti yang dapat dilihat pada Gambar 4.9.



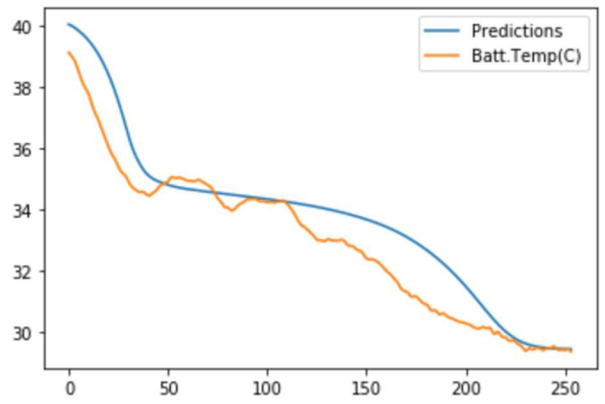
Gambar 4.9. Tampilan antarmuka pengguna *software* estimasi SoH pada Gradio API.

V. PENGUJIAN DAN PEMBAHASAN

Pada bagian Pengujian dan Pembahasan ini, akan dilakukan pengujian terhadap algoritma estimasi SoH, dan apabila pengujian menunjukkan performa yang baik, maka akan dilanjutkan dengan pengujian *software* secara keseluruhan dengan menggunakan fungsi yang digunakan pada GUI.

A. Pengujian Algoritma SoH

Pengujian Algoritma SoH akan dilakukan dengan metode yang sama yang telah dijelaskan pada bagian Detail Rancangan mengenai Pengaplikasian Model LSTM, yaitu dengan mengestimasikan kurva perubahan nilai temperatur terhadap jumlah *cycle* yang telah dilalui baterai. Estimasi dilakukan dengan jumlah iterasi sepanjang jumlah *cycle* pada *dataset* sumber. Hasil estimasi akan dipotong pada titik minimum nya yang dianggap sebagai temperatur pada EoL, dan jumlah *cycle* pada hasil estimasi yang telah dipotong mewakili jumlah *cycle* RUL. Dari jumlah *cycle* RUL, penguji dapat mencari titik *cycle* aktual baterai dan menghubungkannya dengan *dataset* kapasitas untuk mengetahui besar kapasitas aktual baterai. Dengan menggunakan potongan dari *dataset* sumber dengan mengambil sampel pada *cycle* ke-201 sampai ke-221. Dari hasil pengujian, didapati bahwa hasil estimasi memiliki akurasi mencapai 94,43% dengan perhitungan akurasi didasarkan pada perbedaan bacaan nilai kapasitas baterai. Adapun perbandingan antara hasil estimasi dan *dataset* sumber dapat dilihat pada Gambar 5.1. Dengan hasil akurasi diatas standar yang ditentukan (>80%), maka pengujian bisa dilanjutkan ke tahap yang selanjutnya.



Gambar 5.1. Grafik perbandingan hasil estimasi (yang sudah dipotong, garis biru) dengan nilai dari *dataset* sumber (dipotong pada bagian akhir sesuai panjang *dataframe* hasil estimasi yang sudah dipotong, garis oranye).

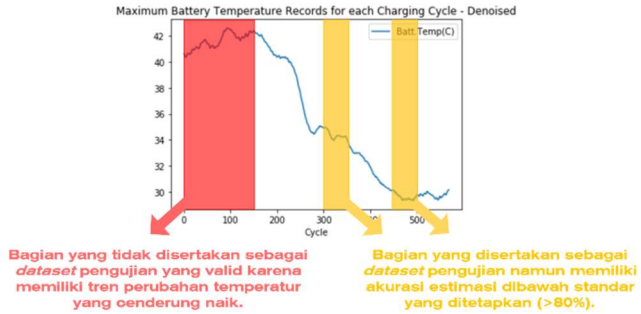
B. Implementasi Algoritma pada Software

Pada skenario pengujian *software* secara keseluruhan, *dataset* yang akan diujikan dibagi menjadi 9 *dataset* uji yang dianggap mewakili setiap fase pada perubahan bacaan temperatur pada *dataset* sumber. *Dataset* uji tersebut, sama dengan bagian yang sebelumnya, merupakan potongan dari *dataset* sumber dengan pemotongan pada interval *cycle* berikut: 1-101, 51-151, 101-201, 151-251, 201-301, 251-351, 301-401, 351-451, dan 401-501 (berbeda 50 *cycle* dengan panjang 100 *cycle*). *Dataset* uji tersebut kemudian digunakan untuk mengestimasikan nilai SoH dan kapasitas dengan *software* estimasi SoH yang sudah dirancang pada bagian Detail Rancangan dengan desain antarmuka seperti yang dapat dilihat pada Gambar 4.9. Untuk mengunggah file CSV ke antarmuka, pengguna dapat menyeret file ke kotak di sebelah kiri atau menekan kotak di sebelah kiri untuk membuka direktori. Setelah file CSV berhasil diunggah, pengguna dapat menekan tombol "Submit" untuk memulai proses estimasi. Adapun hasil pengujian terhadap 9 *dataset* uji seperti yang sudah disebutkan dapat dilihat pada Tabel 5.1.

Informasi pada Tabel 5.1 disajikan dalam 3 warna, yaitu hitam untuk hasil pengujian yang dianggap sesuai dengan harapan, oranye/kuning untuk hasil pengujian dengan akurasi estimasi dibawah standar (>80%), dan merah untuk hasil estimasi dari *dataset* yang dianggap tidak valid berdasarkan poin ke-2 dari batasan masalah yang telah ditentukan. Sebaran hasil pengujian berdasarkan interval *cycle* pada *dataset* sumber dapat dilihat pada Gambar 5.2.

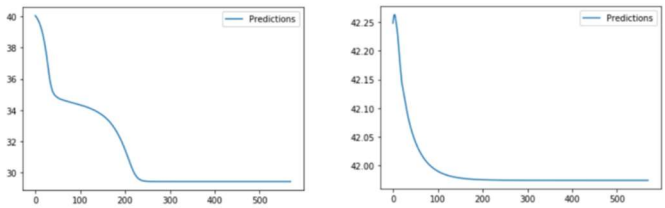
TABEL 5.1
PERBANDINGAN HASIL ESTIMASI DENGAN HASIL YANG DIHARAPKAN

Interval (Cycle)	Temp. Awal (°C)	Temp. Akhir (°C)	Kapasitas Asli (mAh)	Kapasitas Est. (mAh)	SoH Asli (%)	SoH Est. (%)	Est. Waktu (s)	Est. Akurasi (%)
1-101	40,57	42,48	679,13	114,83	96,73	16,36	24,4	16,91
51-151	41,37	42,38	660,17	114,83	94,03	16,36	23,0	17,39
101-201	42,47	40,36	591,28	548,49	84,21	78,12	22,6	92,76
151-251	42,38	37,58	511,86	498,19	72,90	70,96	21,8	97,33
201-301	40,38	34,94	477,97	486,96	68,08	69,33	21,3	98,16
251-351	37,37	34,10	370,83	352,9	52,75	50,26	21,2	95,28
301-401	34,93	31,93	240,79	183,67	34,29	26,16	21,0	76,28
351-451	34,02	30,11	163,55	147,55	23,22	21,01	20,9	90,49
401-501	31,88	29,76	101,72	132,33	14,49	18,85	20,8	69,91



Gambar 5.2. Visualisasi interval *dataset* yang dikecualikan dari hasil pengujian (diwarnai merah) dan interval *dataset* yang memiliki akurasi dibawah standar (diwarnai oranye).

Berdasarkan hasil pengujian yang dapat dilihat pada Tabel 5.1, dapat diketahui bahwa rata-rata akurasi ada dikisaran 72,72%. Namun hasil perhitungan tersebut meliputi dua *dataset* pada bagian awal yang di cetak yang memiliki tren peningkatan temperatur. *dataset* dengan tren naik menunjukkan akurasi estimasi kapasitas yang sangat buruk di angka 114,83 mAh. Hal ini disebabkan karena keluaran dari segala masukan yang berada di interval tersebut akan menunjukkan karakteristik estimasi yang sama seperti yang dapat dilihat pada Gambar 5.3 bagian kanan.



Gambar 5.3. Visualisasi karakteristik hasil estimasi yang diharapkan (kiri) dan hasil estimasi dari masukan yang berupa *dataset* dengan tren naik (kanan).

Seperti yang dapat dilihat dari Gambar 5.2, jika dibandingkan dengan wilayah interval lain yang memiliki tren penurunan nilai temperatur yang cukup drastis dan konsisten, wilayah interval yang ditandai dengan warna oranye memiliki karakteristik perubahan nilai yang cenderung fluktuatif. Hal tersebut diyakini berkontribusi dalam menentukan arah estimasi yang ditandai dengan turun nya nilai akurasi estimasi jika dibandingkan dengan hasil estimasi pada interval yang sebelumnya. Sehingga dapat dikatakan bahwa semakin drastis dan konsisten tren penurunan dari *dataset* yang digunakan sebagai masukan, maka akan semakin tinggi pula akurasi estimasinya. Sebaliknya, semakin kecil perubahan dari titik awal ke titik akhir, dan semakin fluktuatif karakteristik dari nilai perubahannya, maka hasil estimasi akan menjadi kurang deterministik yang ditandai dengan akurasi estimasi yang tidak setinggi di wilayah interval lain.

Dengan menentukan bahwa dua hasil estimasi pertama dikecualikan dari keseluruhan hasil estimasi karena tidak memenuhi syarat yang sudah disampaikan pada bagian batasan masalah, maka rata-rata akurasi estimasi berubah menjadi 88,60% dan dianggap sudah memenuhi standar yang dijanjikan, yaitu >80%. Selain itu, dari Tabel 5.1 juga diketahui bahwa semakin sering *software* digunakan (tanpa melakukan *reset* pada kernel) maka akan semakin cepat pula proses estimasinya. Hal tersebut dapat disebabkan karena komputer menyimpan konfigurasi dan muatan *library* yang dilakukan selama proses (*cache*), sehingga alih-alih melakukan seluruh proses komputasi dari awal, komputer dapat memuat *cache* untuk melewati beberapa tahapan pada proses estimasi yang dengan demikian dapat memangkas waktu proses. Adapun keseluruhan *project* dirancang menggunakan *editor* Jupyter Notebook dengan besar file (ipynb) 494 KB.

VI. Analisis Mengenai Pengaruh Solusi Engineering Design

Meskipun tingkat kesehatan baterai dianggap penting, perangkat untuk mengestimasikan nilai SoH masih jarang ditemui pada perangkat elektronik yang lebih sederhana. Salah satu alasannya adalah karena masalah harga. Jumlah komponen tambahan yang harus dipasang pada sebuah perangkat untuk dapat mengestimasikan nilai SoH akan mempengaruhi kerumitan rancangan perangkat keras, yang akan mempersulit proses fabrikasi, dan pada akhirnya akan meningkatkan modal produksi dari produk tersebut. Dengan mengembangkan metode estimasi SoH yang hanya memerlukan informasi perubahan temperatur, pengguna tidak memerlukan *battery analyzer* seperti yang digunakan sebagai perangkat pengujian da hanya akan memerlukan *data logger* temperatur yang harganya jauh lebih murah untuk dapat mengestimasikan kapasitas dan SoH baterai.

Dengan mengetahui kapasitas maksimal baterai, pengguna juga dapat menentukan konfigurasi pengisian ulang baterai dengan baik untuk mencegah *overcharging* dengan menyesuaikan arus dan waktu *charging* dengan kapasitas baterai. Dengan demikian, kemungkinan kerusakan baterai terjadi akan berkurang, selain memaksimalkan umur baterai dan menambah interval penggantian baterai, juga mengurangi resiko berbahaya seperti munculnya percikan api yang ditimbulkan oleh kenaikan temperatur ekstrim atau yang sering disebut sebagai *thermal runaway*.

VII. Kesimpulan dan Saran

Kesimpulan yang diambil adalah bahwa *dataset* uji berhasil diperoleh dengan merekam dan mengambil dua parameter pengujian yang akan digunakan dalam keseluruhan proses, yaitu parameter temperatur dan kapasitas baterai yang dilakukan sebanyak 600 *cycle* untuk mengamati perubahan parameter baterai dari kondisi baru sampai *End of Life* (EoL) nya. *Dataset* tersebut kemudian digunakan untuk melatih jaringan LSTM (sebanyak 20 *nodes*) yang dirancang dengan fungsi aktivasi ReLU dengan tambahan lapisan *Dense* keluaran tunggal dengan fungsi aktivasi Sigmoid sebanyak 30 *epoch*. Dengan menggunakan model LSTM tersebut, masukan yang berupa nilai bacaan temperatur akan digunakan untuk mengestimasikan jumlah *Remaining Useful Life* (RUL) *cycle*. Dari estimasi jumlah *cycle* yang tersisa dibandingkan dengan jumlah *cycle* lengkap baterai dari baru sampai EoL, maka estimasi *cycle* baterai saat ini dapat diketahui, demikian halnya dengan nilai kapasitas dan estimasi SoH pada *cycle* tersebut dengan membandingkannya dengan *dataset* sumber. Untuk dapat menggunakan algoritma estimasi SoH, kemudian dirancang antarmuka berbasis web sederhana menggunakan *toolbox* yang disediakan oleh Gradio. Setelah melakukan simulasi dengan menguji *software* dengan menggunakan beberapa *dataset* uji yang merupakan potongan dari *dataset* sumber, didapati *software* dapat mengestimasikan nilai kapasitas dan SoH dengan akurasi rata-rata 88,60%, melebihi standar minimal yang ditetapkan pada proposal yaitu 80%. Adapun keseluruhan *project* dirancang menggunakan *editor* Jupyter Notebook dengan besar file 494 KB.

Saran yang dapat diberikan untuk pengembangan lanjutan dari *capstone project* ini adalah dengan melakukan pencatatan parameter baterai dengan menggunakan peralatan yang lebih profesional pada lingkungan yang lebih terkendali untuk meningkatkan kualitas *dataset*. Karena performa model sangat bergantung dengan *dataset* , maka metode estimasi hanya efektif untuk masukan dari peralatan pengukuran dan baterai tertentu saja. Oleh sebab itu, pengembangan dengan melakukan pencatatan dan pelatihan dengan menggunakan peralatan untuk jenis dan ukuran baterai yang lebih beragam juga dapat dilakukan untuk membuat metode efektif untuk lebih banyak sampel uji. Untuk menunjang fungsionalitas tambahan yang sudah disebutkan tadi, akan lebih baik jika model dilengkapi dengan fungsi klasifikasi untuk menentukan jenis baterai dan memilih model mana yang paling tepat untuk melakukan proses estimasi, sehingga diharapkan dapat meningkatkan akurasi estimasi secara signifikan.

Referensi

[1] G. L. Plett, *Battery Management Systems*, vol. 2. Norwood, Massachusetts: Artech House, 2016.

[2] S. Ma, M. Jiang, P. Tao, C. Song, J. Wu, J. Wang, T. Deng, and W. Shang, “Temperature effect and thermal impact in

lithium-ion batteries: A Review," *Progress in Natural Science: Materials International*, vol. 28, no. 6, pp. 653–666, 2018.

[3] N. Noura, L. Boulon, and S. Jemeï, "A review of Battery State of Health Estimation Methods: Hybrid electric vehicle challenges," *World Electric Vehicle Journal*, vol. 11, no. 4, p. 66, 2020.

[4] H. J. Bergveld, W. S. Kruijt, and N. P. H. L., *Battery management systems: Design by modelling*, Dordrecht, Eindhoven: Springer, 2011.

[5] X. Hu, F. Feng, K. Liu, L. Zhang, J. Xie, and B. Liu, "State Estimation for Advanced Battery Management: Key Challenges and future trends," *Renewable and Sustainable Energy Reviews*, vol. 114, p. 109334, 2019.

[6] C. C. Aggarwal, *Neural networks and deep learning: A textbook*. Yorktown Heights, New York: Springer, 2019.

[7] S. S. Haykin, *Neural networks and learning machines*, 3rd ed. Upper Saddle River, New Jersey: Pearson, 2016.

[8] C. Vidal, P. Malysz, P. Kollmeyer, and A. Emadi, "Machine learning applied to electrified vehicle battery state of charge and State of Health Estimation: State-of-the-art," *IEEE Access*, vol. 8, pp. 52796–52814, 2020.

[9] M. N. Ramadan, B. A. Pramana, S. A. Widayat, L. K. Amifia, A. Cahyadi, and O. Wahyunggoro, "Comparative study between internal ohmic resistance and capacity for Battery State of Health Estimation," *Journal of Mechatronics, Electrical Power, and Vehicular Technology*, vol. 6, no. 2, pp. 113–122, 2015.

[10] C. Ranjan, *Understanding deep learning application in rare event prediction*. 2020.