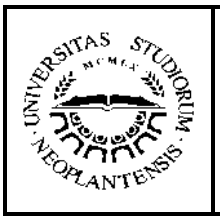


УНИВЕРЗИТЕТ У НОВОМ САДУ
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА

УНИВЕРЗИТЕТ У НОВОМ САДУ
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА
НОВИ САД
Департман за рачунарство и аутоматику
Одсек за рачунарску технику и рачунарске комуникације

ЗАВРШНИ (BACHELOR) РАД

Кандидат: **Витомир Малиновић**
Број индекса: **РА 56/2022**

Тема рада: **Процена стања и оптимизација надзора ИоТ уређаја коришћењем неуронских мрежа**

Ментор рада: **проф. др Мирослав Поповић**

Нови Сад, Септембар, 2026

УНИВЕРЗИТЕТ У НОВОМ САДУ ● **ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА**
21000 НОВИ САД, Трг Доситеја Обрадовића 6

КЉУЧНА ДОКУМЕНТАЦИЈСКА ИНФОРМАЦИЈА

Редни број, **РБР**:		
Идентификациони број, **ИБР**:		
Тип документације, **ТД**:		Монографска документација
Тип записа, **ТЗ**:		Текстуални штампани материјал
Врста рада, **ВР**:		Завршни (Bachelor) рад
Аутор, **АУ**:		**Витомир Малиновић**
Ментор, **МН**:		**проф. др Мирослав Поповић**
Наслов рада, **НР**:		**Процена стања и оптимизација надзора ИоТ уређаја коришћењем науронских мрежа**
Језик публикације, **ЈП**:		Српски / латиница
Језик извода, **ЈИ**:		Српски
Земља публиковања, **ЗП**:		Република Србија
Уже географско подручје, **УГП**:		Војводина
Година, **ГО**:		**2026**
Издавач, **ИЗ**:		Ауторски репринт
Место и адреса, **МА**:		Нови Сад; трг Доситеја Обрадовића 6
Физички опис рада, **ФО**: (поглавља/страна/ цитата/табела/слика/графика/прилога)		**7/46/7/5/22/0/0**
Научна област, **НО**:		Електротехника и рачунарство
Научна дисциплина, **НД**:		Рачунарска техника и рачунарске комуникације
Предметна одредница/Кqучне речи, **ПО**:		ИоТ, ГРУ, неуронске мреже, временски низови података, МКУТТ, дијагностика уређаја, машинско учење, ОННИКС
УДК		
Чува се, **ЧУ**:		У библиотеци Факултета техничких наука, Нови Сад
Важна напомена, **ВН**:		
Извод, **ИЗ**:		У овом раду представљен је систем за интелигентну процену стања ИоТ уређаја на основу дијагностичких временских низова. Развијен је ГРУ модел за класификацију стања уређаја и процену стања његових подсистема. Систем омогућава прикупљање и обраду података путем МКУТТ протокола, анализу у реалном времену и приказ резултата у веб окружењу. Додатно је испитан пренос модела у ОННИКС формат ради ефикаснијег извршавања.
Датум прихватања теме, **ДП**:		
Датум одбране, **ДО**:		
Чланови комисије, **КО**:	Председник:	проф. др Илија Башичевић
	Члан:	ванр. проф. др Марија Антић
	Члан, ментор:	проф. др Мирослав Поповић

Потпис ментора

UNIVERSITY OF NOVI SAD ● **FACULTY OF TECHNICAL SCIENCES**
21000 NOVI SAD, Trg Dositeja Obradovića 6

KEY WORDS DOCUMENTATION

Accession number, **ANO**:		
Identification number, **INO**:		
Document type, **DT**:		Monographic publication
Type of record, **TR**:		Textual printed material
Contents code, **CC**:		Bachelor Thesis
Author, **AU**:		**Vitomir Malinović**
Mentor, **MN**:		**Miroslav Popović, PhD**
Title, **TI**:		**State estimation and monitoring optimization of an IoT device using neural networks**
Language of text, **LT**:		Serbian
Language of abstract, **LA**:		Serbian
Country of publication, **CP**:		Republic of Serbia
Locality of publication, **LP**:		Vojvodina
Publication year, **PY**:		2026
Publisher, **PB**:		Author's reprint
Publication place, **PP**:		Novi Sad, Dositeja Obradovica sq. 6
Physical description, **PD**: (chapters/pages/ref./tables/pictures/graphs/appendixes)		**7/46/7/5/22/0/0**
Scientific field, **SF**:		Electrical Engineering
Scientific discipline, **SD**:		Computer Engineering, Engineering of Computer Based Systems
Subject/Key words, **S/KW**:		IoT, GRU, neural networks, time series data, MQTT, device diagnostics, machine learning, ONNX
UC		
Holding data, **HD**:		The Library of Faculty of Technical Sciences, Novi Sad, Serbia
Note, **N**:		
Abstract, **AB**:		This thesis presents a system for intelligent assessment of IoT device health based on diagnostic time series data. A GRU model was developed for device state classification and estimation of individual subsystem health. The system supports data collection and processing via the MQTT protocol, real-time analysis, and visualization of results in a web environment. In addition, the model was converted to the ONNX format to enable more efficient execution.
Accepted by the Scientific Board on, **ASB**:		
Defended on, **DE**:		
Defended Board, **DB**:	President:	prof. dr Ilija Bašičević
	Member:	vanr. prof. dr Marija Antić — Menthor's sign
	Member, Mentor:	prof. dr Miroslav Popović

	УНИВЕРЗИТЕТ У НОВОМ САДУ ● **ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА** 21000 НОВИ САД, Трг Доситеја Обрадовића 6	Број:
	ЗАДАТАК ЗА ИЗРАДУ ЗАВРШНОГ (BACHELOR) РАДА	Датум:

(Податке уноси предметни наставник - ментор)

Врста студија:	а) Основне академске студије б) Основне струковне студије
Студијски програм:	Рачунарство и аутоматика
Руководилац студијског програма:	проф. др Иван Каштелан

Студент:	Витомир Малиновић	Број индекса:	РА 56/2022
Област:	Електротехника и рачунарство		
Ментор:	проф. др Мирослав Поповић		

НА ОСНОВУ ПОДНЕТЕ ПРИЈАВЕ, ПРИЛОЖЕНЕ ДОКУМЕНТАЦИЈЕ И ОДРЕДБИ СТАТУТА ФАКУЛТЕТА ИЗДАЈЕ СЕ ЗАДАТАК ЗА ЗАВРШНИ (Bachelor) РАД, СА СЛЕДЕЋИМ ЕЛЕМЕНТИМА:

- проблем – тема рада;
- начин решавања проблема и начин практичне провере резултата рада, ако је таква провера неопходна;
- литература

НАСЛОВ ЗАВРШНОГ (BACHELOR) РАДА:

Процена стања и оптимизација надзора ИоТ уређаја коришћењем неуронских мрежа

ТЕКСТ ЗАДАТКА:

У оквиру овог дипломског рада потребно је урадити следеће:

1. Упознати се са релевантном литературом и написати теоријске основе рада,
2. Направити пројекат интелигентног модула за анализу дијагностичких података постојећег ИоТ уређаја, који: (1) прикупља дијагностичке податке путем МКУТТ комуникације, (2) применом модела заснованог на неуронским мрежама процењује стање уређаја и идентификује потенцијалне проблеме, (3) генерише препоруке за оптимизацију надзора уређаја, (4) може примати дијагностичке податке и споља и користити их за фино подешавање свог модела,
3. Имплементирати пројектован и интегрисан модул,
4. Обучити модел интелигентног модула у симулационом окружењу,
5. Евалуирати перформансу модула са обученим моделом у симулационом окружењу,
6. Уочити предности и мане имплементираног решења.

Руководилац студијског програма:	Ментор рада:
проф. др Иван Каштелан	проф. др Мирослав Поповић

Примерак за: ☐ - Студента; ☐ - Ментора

УНИВЕРЗИТЕТ У НОВОМ САДУ ● **ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА**
21000 НОВИ САД, Трг Доситеја Обрадовића 6

ИЗЈАВА О НЕПОСТОЈАЊУ СУКОБА ИНТЕРЕСА

Изјављујем да нисам у сукобу интереса у односу ментор – кандидат и да нисам члан породице (супружник или ванбрачни партнер, родитељ или усвојитељ, дете или усвојеник), повезано лице (крвни сродник ментора/кандидата у правој линији, односно у побочној линији закључно са другим степеном сродства, као ни физичко лице које се према другим основама и околностима може оправдано сматрати интересно повезаним са ментором или кандидатом), односно да нисам зависан/на од ментора/кандидата, да не постоје околности које би могле да утичу на моју непристрасност, нити да стичем било какве користи или погодности за себе или друго лице било позитивним или негативним исходом, као и да немам приватни интерес који утиче, може да утиче или изгледа као да утиче на однос ментор-кандидат.

У Новом Саду, дана _____

Ментор

_____

Кандидат

_____

Захвалност

Велику захвалност упућујем свом ментору, проф. др Мирославу Поповићу, као и техничком ментору Драгану Наранчићу, на издвојеном времену, помоћи и корисним саветима током израде овог рада. Посебно се захваљујем својој породици и пријатељима на подршци током школовања.

САДРЖАЈ

Списак слика

Списак табела

Скраћенице

IoT	-	Internet of Things - Интернет ствари
AI	-	Artificial Intelligence - вештачка интелигенција
GRU	-	Gated Recurrent Unit - рекурентна јединица са капијама
MQTT	-	Message Queuing Telemetry Transport – лаган комуникациони протокол заснован на моделу објави-претплати се
BPTT	-	Backpropagation Through Time – повратно простирање кроз време
TCP/IP	-	Transmission Control Protocol / Internet Protocol – транспортни протокол / Интернет протокол
RNN	-	Recurrent Neural Network – рекурентна неуронска мрежа
LSTM	-	Long Short-Term Memory – дуготрајна краткорочна меморија. Тип рекурентне неуронске мреже са механизмима капија намењен учењу дугорочних временских зависности
MTL	-	Multi-Task Learning – вишезадатно учење
JSON	-	JavaScript Object Notation – формат за размену података
CPU	-	Central Processing Unit – централни процесор
RAM	-	Random Access Memory – радна меморија
CSV	-	Comma-Separated Values – формат текстуалних табеларних података
ONNX	-	Open Neural Network Exchange – отворени формат за представљање неуронских мрежа који омогућава покретање

		обученог модела независно од окружења у ком је обучен
CUDA	-	Compute Unified Device Architecture – NVIDIA платформа за паралелно извршавање рачунарских операција на графичким процесорима

1. Увод

Интернет ствари (Internet of things - IoT) представља једну од најбрже растућих техонологија данашњице која омогућава повезивање милијарди уређаја у јединствени комуникациони систем. Паметни телевизори, сет-топ боксови, кућни апарати, мрежни уређаји, па чак и превозна средства, свакодневно производе огромне количине дијагностичнких података који могу бити искоришћени за праћење стања уређаја и правовремено откривање потенцијалних кварова. Са порастом броја повезаних уређаја, расте и потреба за аутоматизацијом надозора како би се на тај начин смањили време застоја и трошкови одржавања.

Традиционални приступи не пружају могућност предвиђања понашања уређаја што операторима система отежава благовремено реаговање на негативне трендове. Додатни проблем представња и велики број различитих уређаја, што захтева механизме детекције који су у стању прилагодити се специфичним какактеристикама сваког уређаја.

Савремени трендови у развоју индустријских система све више теже ка примени вештачке интелигенције (AI) и машинског учења у циљу унапређивања поступка анализе података. У оквиру тога, неуронске мреже, посебно рекурентне архитектуре попут рекурентне јединице с капијама (Gated Reccurent Unit - GRU) намећу се као идеално решење за обраду временских низова података, захваљујући својој способности да уче дугорочне зависности и препознају обрасце који указују на могуће проблеме.

У наредним поглављима говориће се о целокупном поступку израде оваквог система од симулације рада уређаја, преко припреме скупа података и обучавања модела неуронске мреже, до повезивања обученог модела са апликацијом која у

реалном времену прима, обрађује и приказује дијагностичке податке. Поред самих података могу се видети и процена модела о стању уређаја у коју спадаја класификација стања уређаја која може бити нормално, упозорење и критично, као и бројна оцена из интервала од нула до један. Сваки подсистем (процесор, батерија, радна меморије и стална меморија) има своју оцену и пета заједничка оцена је за цео систем. Приказан је и начин на који се модел може накнадно дообучити када нови подаци постану доступни, као и поступак провере тачности и поузданости самог модела кроз различите сценарије рада.

У другом поглављу изложене су теоријске основе неопходне за разумевање предложеног решења. Обрађене су Интернет ствари и MQTT протокол, временски низови и њихова припрема, основни појмови машинског и дубоког учења, рекурентне неуронске мреже и GRU архитектура, као и поступци вишезадатног учења и финог подешавања модела.

У трећем поглављу представљен је концепт целокупног решења и његове главне функционалне целине: припрема скупа података, обука и фино подешавање неуронске мреже и модул за анализу података у реалном времену. Четврто поглавље детаљније описује програмску реализацију ових целина, од стварања сценарија и прикупљања података, преко поступка обуке модела, до обраде пристиглих мерења и приказа резултата.

У петом поглављу приказани су резултати анализе формираног скупа података, обуке и испитивања више варијанти модела, као и избор коначне архитектуре. Додатно је проверен рад система у реалном времену, исправност појединачних програмских целина помоћу јединичних тестова и могућност извршавања модела у ONNX окружењу уз поређење перформанси. У шестом поглављу сумирани су резултати рада, размотрена ограничења реализованог решења и наведени могући правци његовог даљег унапређења. На крају рада дат је списак коришћене литературе.

2. Теоријске основе

Кроз поглавље обрађени су кључни теоријски концепти, архитектуре неуронских мрежа, процедуре и алгоритми који чине основу за развој система за обраду и анализу података у реалном времену и процену стања уређаја.

2.1 Интернет ствари (IoT) и комуникациони протоколи

2.1.1 Архитектура IoT система

Интернет ствари [1] представљају појам мрежног повезивања физичких уређаја који поседују сензоре, могућност обраде података и комуникационе модуле с циљем размене података са другим системима према дефинисаним процедурама. Архитектура савремених IoT система обично обухвата три основна слоја:

1. Перцептивни (сензорски) слој чине уређаји који прикупљају дијагностичке и оперативне параметре или самог хардвера попут заузетости меморије, температуре батерије или оптерећења процесора.
2. Мрежни слој чини комуникациона инфраструктура која омогућава пренос прикупљених података до централних ресурса или обрадних чворова.
3. Апликативни слој састоји се од сервиса и аналитичких модула за обраду података, детекцију неправилности, доношења управљачких одлука и приказа стања уређаја.

2.1.2 Објави-претплати се (*Publish-Subscribe*) и MQTT протокол

За разлику од традиционалног клијент-сервер модела, Објави-претплати се модел омогућава асинхрону комуникацију и потпуно раздвајање пошиљаоца и примаоца у времену и простору.

MQTT (*Message Queuing Telementry Transport*) [2] је лаки апликативни протокол изграђен на врху TCP/IP слоја, прилагођен за уређаје са ограниченим хардверским ресурсима, мреже са малим пропусним опсегом (throughput) и великим кашењењем.

Основни елементи MQTT архитектуре су:

- Брокер (*Broker*): Централно посредничко чвориште које прима све поруке филтрира их и распоређује их клијентима који су изразили интересовање за одређене теме.
- Теме (*Topics*): Хијарархијски уређени текстуални низови који служе за адресирање и усмеравање података (нпр. *device/id/telementry*).
- Клијенти: Уређаји или сервиси који могу имати улогу издавача (publishers), када шаљу дијагностичке податке на одређене теме, или претплатника (subscribers), када се претплаћују на теме са којих примају резултате анализе или управљачке сигнале.

2.2 Обрада временских серија и припрема података

2.2.1 Анализа временских серија у реалном времену

Временска серија је низ мерних података редоследно индексираних у времену:

$$X = \{x_1, x_2, \dots, x_T\}$$

У контексту IoT дијагностике, вектор у сваком временском кораку $x_t \in R^d$ представља скуп d дијагностичких метрика. Главни изазови ових података у реалном времену укључују асинхроно пристизање резултата мерења, промењиве временске интервале, шум у мерењима и недостајуће вредности.

2.2.2 Техника клизајућег прозора(*Sliding Window Tehnique*)

Како би се непрекидни ток трансформисао у облик погодан за обраду моделима машинског учења, користи се техника клизајућег прозора. Прозор дефинишу два основна параметра:

- Ширина прозора: Временски распон (у секундама или броју одбирака) која дефинише историјски контекст који се анализира.

- Корак клизања: Помаци између два узастопна интервала обраде, који одређују учесталост предвиђања.

2.2.3 Предобрада и нормализација података

Како дијагностички параметри поседују различите вредности, извршена је нормализација улазних података ради побољшања стабилности поступка оптимизације током обучавања модела, решава проблем различитих уређаја који немају исти опсег за исте метрике.

- Мин-Макс скалирање: Трансформише вредности промењиве x у дефинисани опсег [0, 1]:

$$x_{norm} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

- Поред поравнања (*Padding*) низова различитих дужина унутар истог пакета (*Batch*), код временских дијагностичких података може доћи до недостатка појединих мерења услед грешака у систему или честе праксе да се непромењене вредности не шаљу. Овај пробем се решава попуњавањем последњом доступном вредношћу, интерполацијом , попуњавањем средњом или медијанском вредношћу. Честа пракса је и коришћење неутралне вредности, најчешће нуле, при чему се користи бинарна маска $M_т \in \{0,1\}$ како би се знало да се одређене вредности игноришу. Након маскирања, низови се поравнавају, најчешће нулама, како би имали исту дужину и на њих би се могла применити обрада пакета података.

2.3 Основе машинског и дубоког учења

2.3.1 Типови машинског учења

Алгоритми машинског учења се на основу начина обраде података и присуства повратне информације током обучавања примарно деле на три основне категорије: учење под надзором (*supervised learning*), учење без надзора (*unsupervised learning*) и учење уз подстицај (*reinforcement learning*).[4]

У контексту овог рада, кључну улогу има **учење под надзором**, где се модел обучава коришћењем организованог и означеног скупа података који садржи улазне векторе обележја (*features*) и њима одговарајуће тачне излазне ознаке (*labels*). Зависно од природе циљне променљиве, учење под надзором се дели на **класификацију** (предвиђање дискретних категорија, попут врсте аномалије) и **регресију** (предвиђање континуалних вредности, попут процента називног стања уређаја).

2.3.2 Дубоко учење

Дубоко учење (*Deep Learning*) представља подскуп машинског учења заснован на вештачким неуронским мрежама са више слојева обраде. За различите од традиционалних метода које захтевају ручно одређивање карактеристика, модели дубоког учења користе композицију нелинеарних трансформација за аутоматско учење апстрактних хијерархијских репрезентација непосредно из сирових сигнала.[4]

Традиционалне унапред-усмерене неуронске мреже (*Feedforward Neural Networks*) претпостављају да су сви улазни узорци међусобно независни. Међутим, у IoT дијагностици, стање система у временском кораку t непосредно зависи од претходних стања $t-1, t-2, \ldots, t-k$. За разумевање оваквих временских зависности неопходне су рекурентне архитектуре које поседују унутрашњу меморију (скривено стање) која се ажурира кроз време.

2.4 Рекурентне неуронске мреже и GRU архитектура

2.4.1 Ограничења традиционалних рекурентних мрежа

Рекурентне неуронске мреже (RNN) обрађују низове података одржавајући унутрашње скривено стање h_t. Међутим, током обраде дугачких низова путем алгоритма повратног простирања кроз време (*Backpropagation Through Time – BPTT*), долази до проблема нестајања или експлозије градијената.[4] Ова појава онемогућава класичном RNN-у да успешно научи дугорочне зависности у временским серијама.

2.4.2 Рекурентна јединица са капијама (Gated Reccurent Unit - GRU)

GRU архитектура представља оптимизовану варијанту рекурентних мрежа која решава проблем нестајања градијената увођењем механизма капија (*gates*). За разлику од LSTM (*Long Short-Term Memory*) мреже, GRU поседује једноставнију структуру са мање параметара, што је чини рачунарски ефикаснијом и бржом за обучавање. [3]

GRU ћелија управља протоком информација помоћу две основне капије:

- Капија ажурирања (*Update Gate* z_t): Одређује колико претходних информација треба проследити у будућа стања.
- Капија ресетовања (*Reset Gate* r_t): Одређује колико претходних информација треба заборавити.

Математичке функције GRU ћелије дефинисане су следећим једначинама:

$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t] + b_z)$$

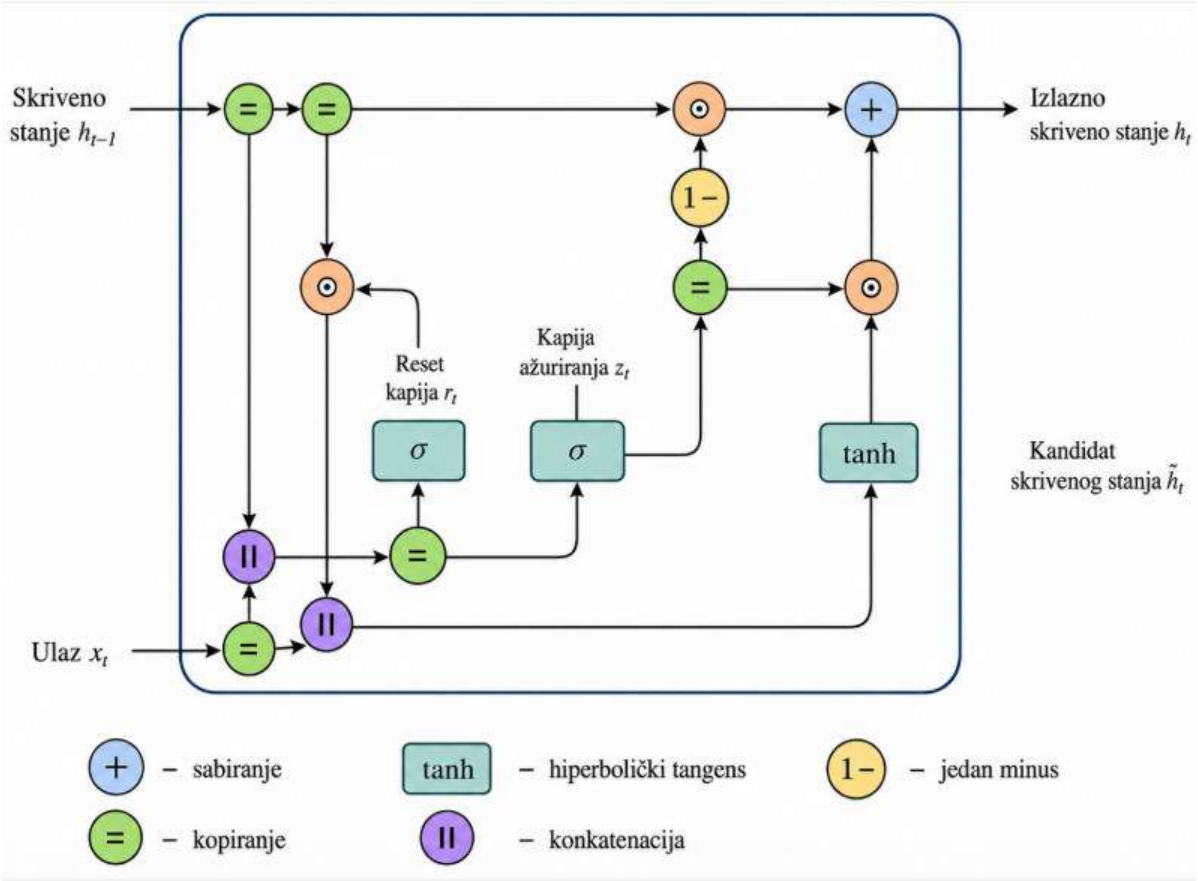
$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t] + b_r)$$

$$\tilde{h_t} = tanh(W_h \cdot [r_t \odot h_{t-1}, x_t] + b_h)$$

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h_t}$$

где су:

- x_t – улазни вектор у временском тренутку *t*
- h_{t-1} – скривено стање из претходног временског корака
- $\tilde{h_t}$ – кандидат за ново скривено стање
- σ – сигмоидна активациона функција
- $\odot$ - Хадамардов (елемент по елемент) производ вектора и матрица



Слика 2. 1 Архитектура GRU ћелије

2.5 Вишезадатно учење

2.5.1 Концепт вишезадатног учења са дељењем параметара

Вишезадатно учење (*Multi-Task Learning – MTL*) је тип машинског учења у ком се више сродних задатака обучава истовремено коришћењем заједничког модела. Овај приступ побољшава способност уопштавања јер заједничка репрезентација приморава мрежу да научи универзалне карактеристике података корисне за све задатке.[5]

У архитектури са дељењем тврдих параметара (*Hard Parameter Sharing*), почетни слојеви (попут GRU слоја) су заједнички за све задатке, док се након њих мрежа грана у специфичне излазне главе (*heads*).

2.5.2 Класификација и регресија дијагностичког стања

У контексту процене стања уређаја, користе се два типа излаза:

- **Категоричка класификација:** Одређивање дискретне класе глобалног стања система.
- **Континуална регресија:** Израчунавање скаларних или векторских вредности у опсегу [0, 1] које описују стање и перформансе појединачних хардверских подсистема.

2.5.3 Комбинована функција губитка

Како би се модел истовремено обучио за класификацију и регресију, користи се хибридна функција губитка.

- Губитак преко ентропије за класификацију (L_{cls}):

$$L_{cls} = -\sum_{c=1}^{C} y_c log(\hat{y_c})$$

- Средња квадратна грешка (L_{reg}):

$$L_{reg} = \frac{-1}{M}\sum_{m=1}^{M}(h_m - \hat{h}_m)^2$$

Укупна функција губитка дефинише се као тежинска сума:

$$L_{total} = L_{cls} + \lambda \cdot L_{reg}$$

где параметар λ представља фактор који уравнотежује утицај регресионог задатка у односу на класификациони.

2.6 Технике обуке, финог подешавања и превенције заборава.

2.6.1 Алгоритам оптимизације и адаптивна стопа учења

За оптимизацију параметара мреже користи се **Adam** (*Adaptive Moment Estimation*) алгоритам. Адам комбинује предности методе Моментума и оптимизатора *RMSProp* тако што израчунава адаптивне стопе учења за сваки параметар појединачно, користећи процене првог момента (средње вредности) и другог момента (нецентриране варијансе)

градијената. Ово га чини изузетно ефикасним за проблеме са великим бројем параметара и асинхроним дијагностичким подацима.[6]

Како би се спречило осциловање у близини локалног минимума, користи се стратегија динамичког смањења стопе учења (*Reduce Learning Rate on Plateau*), која смањује стопу учења када се након унапред дефинисаног броја епоха обуке не примећује побољшање над валидационим скупом, где једна епоха представља један потпун пролаз модела кроз цео скуп података намењен обуци.“

2.6.2 Фино подешавање и концепт катастрофалног заборављања

У реалним системима Интернет Ствари често долази до појаве нових типова кварова који нису били присутни у почетном скупу за обуку. Техника финог подешавања (*fine-tuning*) омогућава ажурирање постојећег обученог модела над новим прикупљеним подацима уместо извођења обуке од самог почетка.

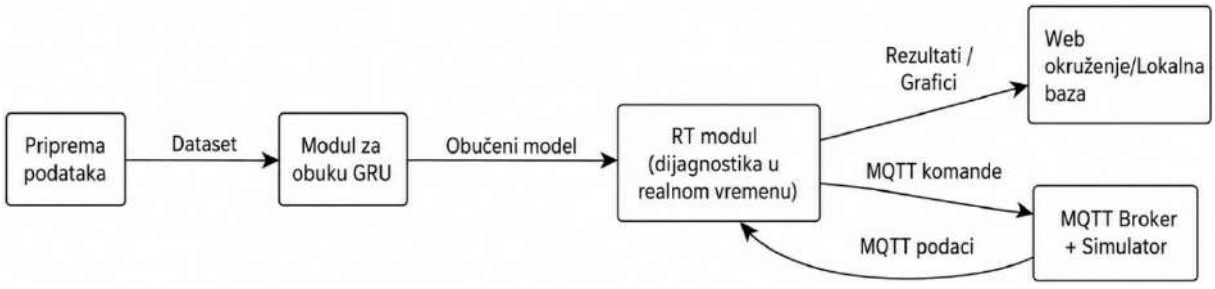
Приликом финог подешавања јавља се изазов **катастрофалног заборава (catastrophic forgetting)**, где модел прилагођавајући се новим подацима потпуно уназађује перформансе над раније наученим сценаријима. Како би се овај проблем спречио, користи се комбиновани скуп података, који током финог подешавања спаја нове приспеле секвенце са избалансираним репрезентативним узорком старих историјских података.

3. Концепт решења

Развој система за интелигентну процену стања уређаја обухвата заокружен процес, започет припремом специфичних сценарија ради формулисања скупа података, а заокружен израдом модула за обуку и оперативну дијагностику у реалном времену. У наставку је изложена целокупана архитектура решења, са посебним фокусом на функционалне целине које омогућавају обраду временских серија и адаптивно реаговање на детекцију кварова.

3.1 Структура и организација система

Затечена инфраструктура ослања се на симулатор уређаја унутар *Docker* окружења који путем *MQTT* посредника шаље дијагностичке метрике попут оптерећења и температуре процесора, заузећа оперативне и складишне меморије, као и стања батерије. Садашњем систему је недостајала компонента способна да препозна трендове у овим подацима и благовремено предвиди потенцијалне кварове. Направљено решење превазилази овај проблем кроз три повезане целине: помоћни процес за израду скупа података за обуку модела, модул за обуку неуронске мреже, као и оперативни модул за анализу у реалном времену.



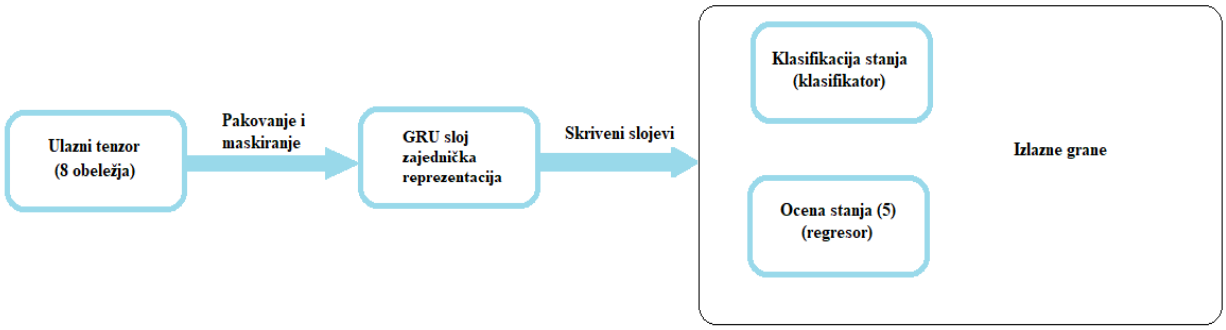
Слика 3. 1 Општа архитектура предложеног система

3.1.1 Изарада и огранизација скупа података

Прва фаза рада била је посвећена формирању репрезентативног скупа података. Путем *MQTT* посредника прикупљене су секвенце од осамсто стекунди рада симулатора који пролази кроз различите сценарије рада. Након чега су дефинисане класе стања (*Нормално*, *Упозорење*, *Критично*) и одговарајуће континуалне оцене стања подсистема. Формирани подаци су груписани у одговарајуће фолдере уз подршку за увоз нових секвенци из *ZIP* архива ради каснијег финог подешавања.

3.1.2 Модул за обуку и фино подешавање неуронске мреже

Главна аналитичка компонента система јесте модул за обуку дубоке неуронске мреже (*GRUSharedRepresentation*), развијен у *PyTorch* окружењу за истовремено решавање задатака класификације и регресије. Овај модул користи адаптивне функције за паковање и попуњавање секвенци нулама (*padding*), чиме се омогућава обрада низова различитих дужина уз стварање одговарајућих маски. Маска се формира за сваки временски корак и обележје и означава да ли је одговарајућа вредност стварно присутна у улазним подацима. Секвенце различитих дужина допуњавају се нулама до дужине најдуже секвенце у пакету, при чему се чувају њихове стварне дужине како додате вредности не би утицале на рад модела. Сами улазни подаци се заједно са маскама прослеђују кроз *GRU*, чије се скривено стање грана на класификациону грану за одређивање стања система и регресиону грану са сигмоидном активацијом за процену оцена система. Мрежа се оптимизује помоћу комбиноване функције губитка, а приликом финог подешавања са новим подацима примењује се комбиновање са старим узорцима како би се спречио катастрофални заборав раније научених сценарија.

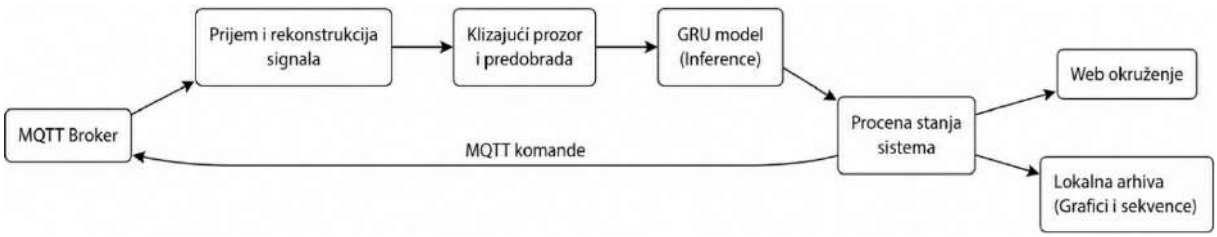


Слика 3. 2 Ток података кроз GRU модел са две излазне гране

3.1.3 Модул за рад у реалном времену

Модул за рад у реалном временује задужен за континуирано преузимање дијагностичких података, формирање клизајућих прозора и процене стања система

помоћу обученог модела. Пристигли подаци се реконструишу по уређајима, попуњавају интерполационим подацима у случају да се чешће врши мерење него слање података, попуњавањем последњом пристиглом вредношћу за случај да подаци нису присутни и пакују у прозоре од осамсто секунди. Модел на основу тих података брзо израчунава тренутно стање и оцене стања система, а у случају откривања проблема модул може аутоматски послати повратну *MQTT* команду ка симулатору (нпр. налог за рестарт или ослобађање меморије) како би се уређај вратио у нормалан режим рада. Поред тога, сви подаци се шаљу на веб графичко окружење ради визуализације и постоји могућност чувања низова података и графика у локалној бази.



Слика 3. 3 Архитектура апликације за рад у реалном времену

4. Програмско решење

У овом поглављу детаљно је описана практична реализација система за процену стања уређаја и детекцију аномалија у реалном времену. Пожељна архитектура обухвата процес који почиње припремом специфичних сценарија кроз симулацију ради формулисања скупа података, а заокружује се израдом модула за обуку и оперативну дијагностику у живом времену.

Архитектура обухвата следеће главне целине:

- **Симулација и прикупљање података:** Симулатор уређаја који преко *MQTT*-а шаље дијагностичке податке створене на основу унапред дефинисаних сценарија, уз помоћне скрипте за сакупљање и паковање сирових података у уређени скуп.
- **Модул за обуку и фино подешавање неуронске мреже:** Модул развијен у *PyTorch* окружењу заснован на *GRU* архитектури који ради симултану класификацију стања и регресиону процену стања подсистема.
- **Модул за рад у реалном времену:** Оперативни асинхрони сервис задужен за континуално преузимање дијагностичких података у живом времену, њихову припрему (интерполација, клизајући прозори), прослеђивање моделу за закључивање (*inference*) и евентуално слање повратних контролних команди ка симулатору.

4.1 Стварање сценарија и формирање скупа података

За потребе развоја и провере модела, било је неопходно обезбедити обиман и репрезентативан скуп података који верније осликава како редован рад, тако и различите типове кварова и деградација постојећег IoT система. Како ови подаци нису лако доступни у аутентичним продукцијским условима у свим границама интервала, систем се ослања на синтетичко стварање помоћу специјализованог симулатора.

4.1.1 Архитектура симулатора и симулација сценарија

Симулатор уређаја је реализован као модуларна Kotlin апликација заснована на савременим обрасцима пројектовања софтвера (попут *Registry* и *Collector* образаца), што омогућава лаку проширивост и независан развој подсистема.

Окосницу симулатора чине три кључна слоја:

- Слој Снабдевача дијагностичких података (Property Providers): Модули попут BatteryProvider, CPUProvider, MemoryProvider и StorageProvider одговорни су за производњу физичких и системских метрика. Сваки од њих садржи унутрашњу софтверску логику за производњу нормалних вредности, али и за симулацију аномалија и деградација стања.
- Централни сакупљач и модел података (*PropertyCollector* и *DataModel*): Сви појединачни снабдевачи се приликом покретања апликације региструју у централни регистар — *PropertyCollector*. Систем затим помоћу *DataModelLoader*-а динамички учитава конфигурациони модел података (у JSON формату, нпр. android-model-full.json) који строго дефинише структуру и валидацију системских метрика које се прате. DataHandler компонује снабдеваче и модел, омогућавајући конзистентно очитавање свих седам кључних системских метрика:
  - CPU: оптерећење и температура,
  - RAM : количина слободне меморије,
  - Storage: количина слободне спољне меморије
  - Battery: проценат напуњености, температура и статус пуњача
- Комуникациони слој и управљање командама (CloudCommunicator и CommandHandler): Апликација користи CloudCommunicator модул који преко MQTT протокола емитује уоквирене метрике ка брокеру унутар дефинисаног именика тема (dms/devices). Поред слања података, овај слој садржи и регистар управљачких команди (CommandHandler) који омогућава пријем асинхроних команди са мреже (нпр. за гашење процеса, чишћење меморије или рестарт уређаја), што систем чини спремним за рад у реалном времену.

Сваки подсистем има могућност симулирања специфичних аномалија. На пример, подсистем који симулира рад батерије поред нормалног стања има и стање повишене термпературе, наглог пражњења батерије и наглих падова процента напуњености батерија за више од једног процента у тренутку. Различити сценарији доводе до

другачије оцене тог подсистема као и целокупног система, као и класификације тренутног стања уређаја зависно од тога колико негативно утичу на рад уређаја.

Сценарио	Класа сценарија	Оцена подсистема
Нормално	Нормално	1.0
Складиште испод 10%	Упозорење	0.5
Нагла промена складишта	Упозорење	0.5
Заузеће радне меморије	Упозорење	0.5
Умерено загревање процесора	Упозорење	0.75
Оптерећење процесора	Упозорење	0.75
Загревање батерије	Упозорење	0.83
Лош контакт пуњача	Упозорење	0.83
Нагли пад процента батерије	Упозорење	0.83
Нагло пражњење батерије	Упозорење	0.83
Преоптерећење процесора	Критично	0.5
Прегревање процесора	Критично	0.5
Засићење радне меморије	Критично	0.0
Критично попуњено складиште	Критично	0.0
Неслагање статуса пуњача и промене процента напуњености батерије	Критично	0.67
Прегревање батерије	Критично	0.67

Табела 4. 1 Преглед симулираних сценарија, класа стања и оцена подсистема

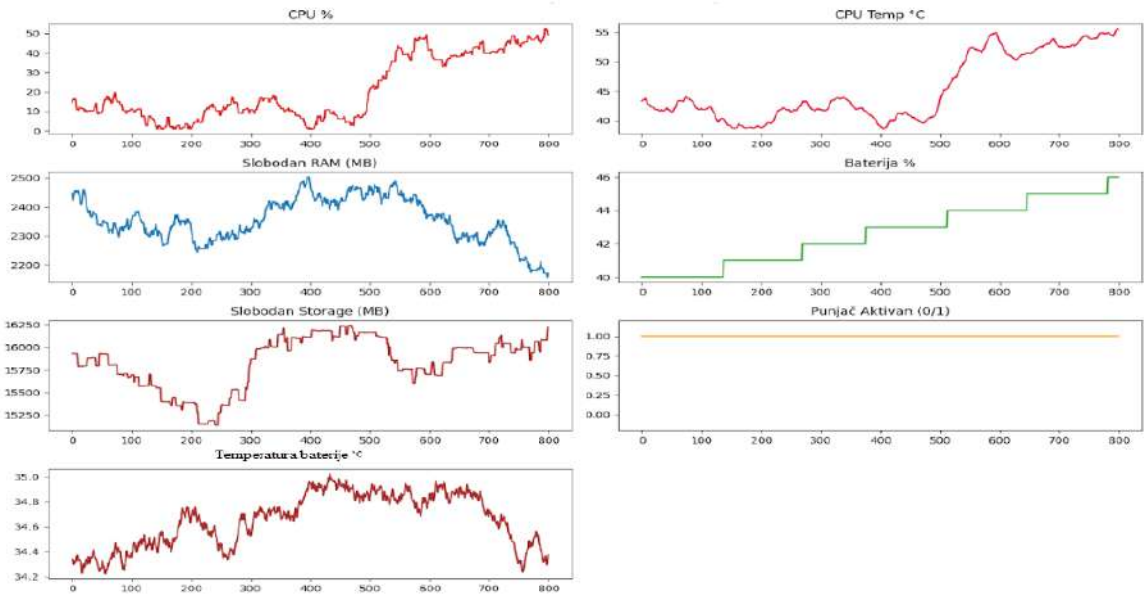
Оцена сваког подсистема формира се на основу броја метрика које му припадају и стања сваке појединачне метрике. Ако подсистем садржи (n) метрика, свака од њих у потпуно исправном стању учествује у укупној оцени са (1/n). Уколико је метрика у стању упозорења, њен допринос се преполовљава, па подсистем губи (1/(2n)) од максималне оцене. Ако је метрика у критичном стању, њен допринос се у потпуности губи, односно оцена подсистема се умањује за (1/n).

На пример, подсистем процесора чине две метрике: оптерећење и температура. Уколико су обе метрике у нормалном стању, оцена подсистема износи 1. Ако је температура у стању упозорења, оцена се умањује за 0,25 и износи 0,75, док у случају критичне температуре губитак износи 0,5, па оцена подсистема износи 0,5. На исти начин формира се и укупна оцена система, при чему се узимају у обзир све присутне метрике.

4.1.2 Прикупљање и паковање узорака

За прикупљање података коришћене су помоћне скрипте. Ове скрипте нису део продукцијског сервиса, већ служе као омотач за грађење скупа података:

- Претплаћују се на MQTT теме симулатора и чувају метрике током рада симулатора у различитим сценаријима.
- Прикупљени временски низови се групишу у прозоре дужине **око 800 секунди** рада.
- Сакупљени сирови узорци чувају се у CSV датотекама и не подвргавају се никаквим препроцесирањима у овом кораку (попут попуњавања или интерполције), већ се задржавају у изворном облику.



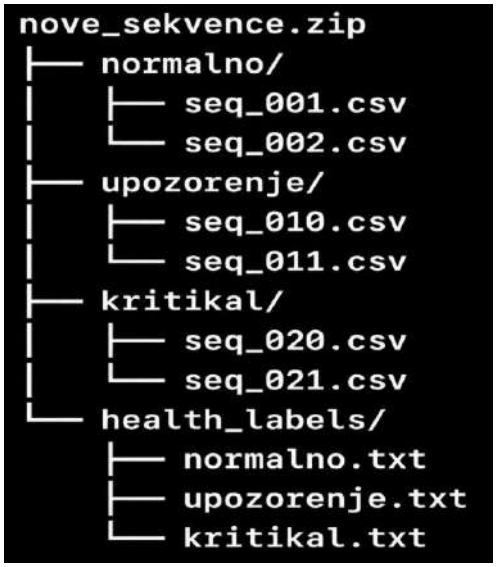
Слика 4. 1 Пример временског прозора у нормалном режиму рада уређаја

4.1.3 Дистрибуција и физичка структура скупа података

Скуп података је направљен тако да осликава реалне односе учесталости аномалија у индустријским системима: нормалан рад је најприсутнији, док су аномалије и критична стања ређа, нешто изнад стварне заступљености, јер су лажни аларми прихватљивији од пропуштених кварова. Недостаци балансираности класа у

обуци могу се решити и механизмом тежина, где се погрешна предвиђања неке класе кажњавају више и тиме се постиже реална репрезентација. Однос узорака у скупу је приближно 16400/10200/9200 за стања нормално/упозорење/критично.

Физичка структура скупа података на диску је организована кроз хијерархију директоријума, што омогућава једноставну интеграцију, учитавање, као и подршку за фино подешавање из нових ZIP архива:



Слика 4. 2 Подразумевана хијарархија скупа података и ZIP архиве за фино подешавање

4.1.4 Организација лабела и оцена подсистема

Поред глобалне класификационе класе (нормално, упозорење или критично), за свака датотека у узорку ствара се и одговарајући запис унутар *health_labels* директоријума *(normalno.txt, upozorenje.txt, kritikal.txt*). Уовим датотекама сваком низу одговарају континуалне [0, 1] оцене стања целокупног система и појединачних подсистема.

4.2 Поступак обуке и финог подешавања модела

Модул за обуку неуронске мреже задужен је за припрему прикупљених временских секвенци, формирање улазних података, обуку модела и проверу његових резултата. Цео поступак реализован је употребом програмског језика *Python* и библиотеке *PyTorch*.

Процес обуке подржава два режима рада. У првом режиму модел се обучава од почетка над основним скупом података. Други режим омогућава фино подешавање већ постојећег модела новим секвенцама, уз задржавање дела старих података ради очувања претходно наученог понашања.

Основни ток обуке обухвата учитавање конфигурације, припрему скупа података, поделу низова на мање узорке уколико је потребно, формирање пакета, пролаз кроз *GRU* модел, израчунавање грешке и ажурирање параметара. Након сваке епохе модел се проверава над валидационим скупом, док се коначна оцена врши над издвојеним тестним подацима.

4.2.1 Конфигурација и режими рада

Параметри обуке издвојени су у конфигурациону датотеку `config.json`. На овај начин понашање система може се мењати без измена у програмском коду.

Конфигурацијом се бира режим рада, одређују путање до података и сачуваног модела и подешавају параметри учитавања података. Такође је могуће мењати број епоха, стопу учења, величину пакета, регуларизацију, стрпљење за рано заустављање и остале параметре поступка обуке.

Структура модела такође се задаје кроз конфигурацију. Могуће је подесити број улазних обележја, величину скривеног стања, број GRU слојева, број излазних класа и степен насумичног искључивања неурона.

Стандардна обука и фино подешавање имају одвојене скупове параметара. Фино подешавање обично користи опрезније измене тежина, јер је његов циљ прилагођавање већ обученог модела, а не поновно учење свих образаца од почетка.

При покретању се проверава доступност графичког процесора. Уколико је доступан, модел и подаци се пребацују на њега, док се у супротном обрада извршава на централном процесору.

4.2.2 Припрема података за фино подешавање

У режиму финог подешавања нови низови достављају се у облику ZIP архиве. Архива се најпре распакује у привремени директоријум, након чега се проверава структура архиве и неопходно је да она буде иста као и структура оригиналног скупа података.

За сваки поступак финог подешавања формира се нови директоријум, чиме се спречава преписивање претходно припремљених података. Нови CSV низови и њихове оцене затим се копирају у одговарајуће директоријуме.

Уколико је укључена употреба старих података, проверава се да ли постоје датотеке са истим називом у новом и старом скупу. У случају подударања поступак се прекида, јер би могло доћи до преписивања датотека или погрешног повезивања ознака.

Нови подаци могу се комбиновати са изабраним старим секвенцама. Број старих узорака одређује се у односу на количину нових аномалних података и њихову класу. Овај поступак представља везивање старим подацима за претходно знање односно научене обрасце и смањује могућност да модел током финог подешавања заборави раније научене обрасце.

За изабране старе секвенце копирају се и одговарајуће оцене стања система. Уколико употреба старих података није омогућена, скуп се формира искључиво од података добијених из нове архиве. Након завршетка припреме привремени директоријум се уклања.

4.2.3 Формирање и обрада временских низова

Подаци су организовани у директоријуме који одговарају класама нормалног стања, упозорења и критичног стања. Свакој датотеци придружена је категоријска ознака и вектор од пет оцена које описују стање целог система и појединачних подсистема.

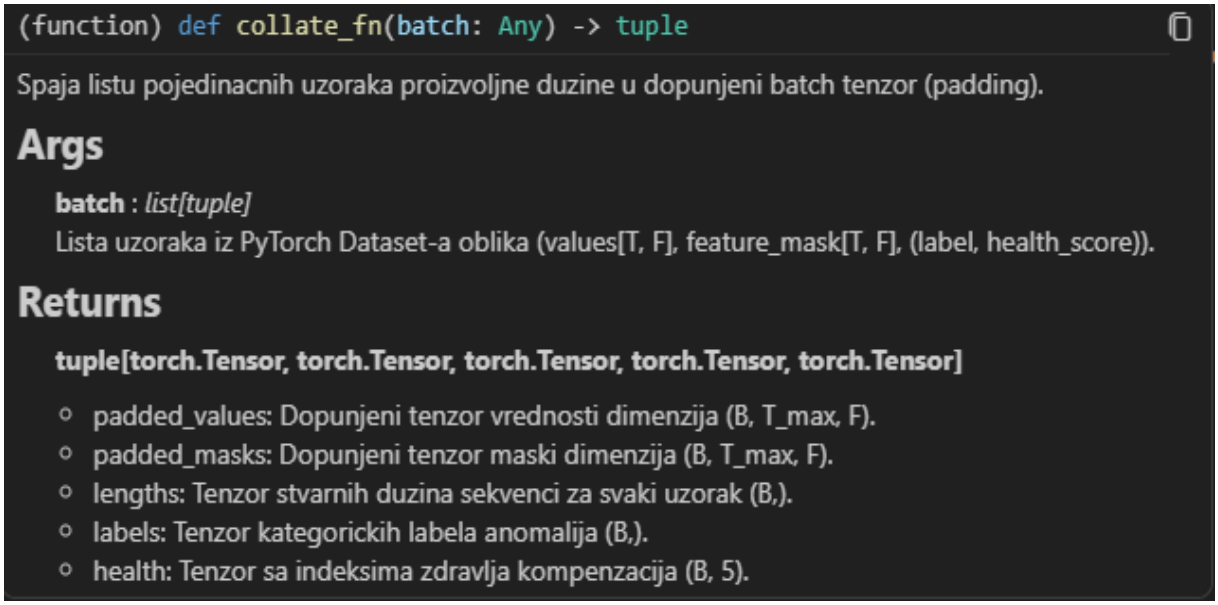
При иницијализацији скупа учитавају се временске ознаке из CSV датотека. Подаци се сортирају хронолошки и деле на временске прозоре. Уколико један прозор садржи више мерења од дозвољеног броја, додатно се дели на мање под-секвенце.За сваки узорак памте се путања до датотеке, почетак и крај прозора, редни број под-секвенце и припадајућа класа. Сви подаци се не чувају стално у меморији, већ се конкретан део CSV датотеке учитава тек када је потребан током обуке.

Након учитавања изабраног прозора, редови се сортирају према времену. Пошто све метрике не морају бити доступне у сваком тренутку, недостајуће вредности се прво попуњавају последњом познатом вредношћу. Преостале празнине попуњавају се нулом.

Статус пуњача, укључен или искључен, претвара се у бинарну вредност, док се остале метрике нормализују према очекиваним опсезима. Тиме се вредности различитих физичких величина доводе на упоредиву размеру, што олакшава поступак учења. Поред нормализованих вредности формира се и маска присутности. Вредност маске означава да ли је одређена метрика доступна и активна у конкретном временском кораку. На овај начин модел може да разликује стварну нулту вредност од недостајућег податка. Из временских ознака рачуна се и размак између два узастопна мерења. Над њим се примењује логаритамска трансформација $\Delta t' = ln(1 + \Delta t)$. Трансформисани временски размак додаје се улазним обележјима, заједно са одговарајућом маском.

4.2.4 Припрема пакета и подела скупа

Временски низови могу да имају различите дужине. Због тога се функцијом ***collate_fn*** више узорака спаја у један пакет и допуњава нулама до дужине најдужег низа у том пакету. На исти начин допуњавају се матрице вредности и маски. Истовремено се чувају стварне дужине свиц низова, како би модел могао да занемари додате елементе. Пакет садржи допуњене вредности, маске, стварне дужине секвенци, категоријске ознаке и векторе стања система. Ознаке класа претварају се у целобројне векторе, док се оцене стања чувају као реалне вредности. Целокупан скуп дели се на скуп за обуку, валидациони и тестни део. Скуп за обуку служи за ажурирање тежина модела, валидациони за праћење напретка и избор најбоље верзије модела, а тестни за коначну проверу.



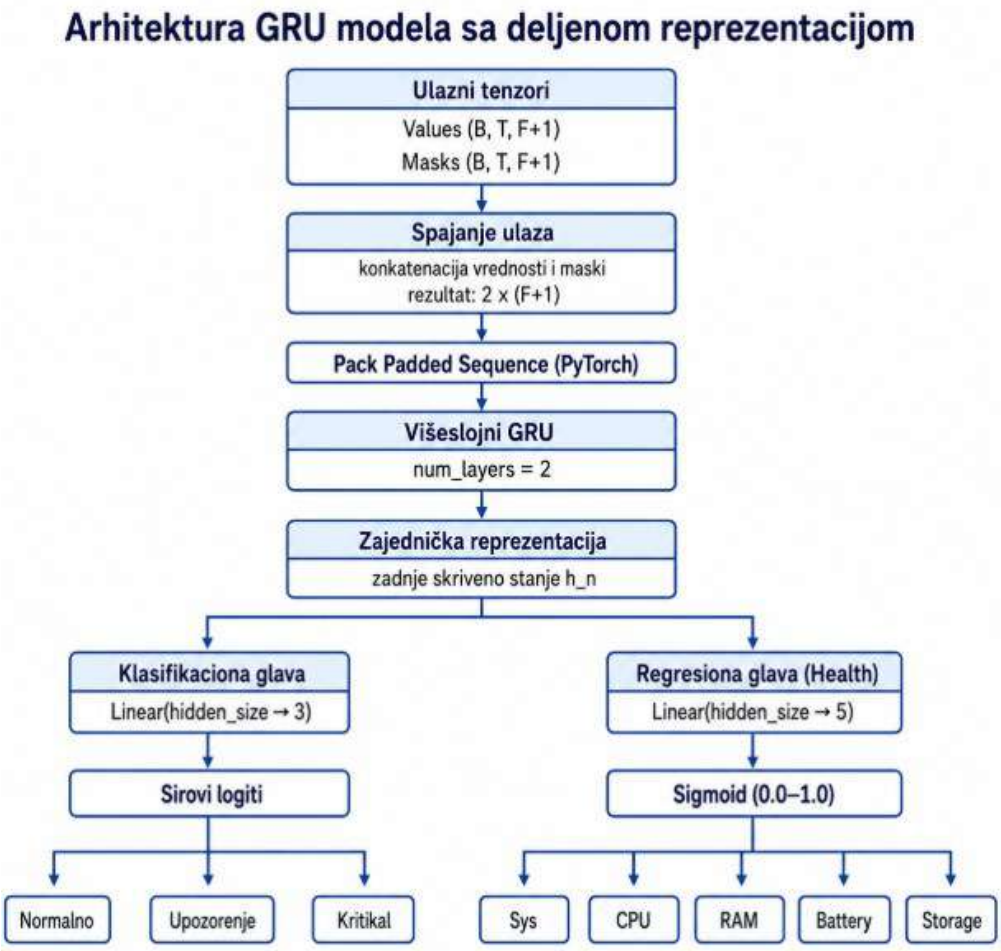
Слика 4. 3 Формирање пакета временских секвенци различитих дужина

4.2.5 Архитектура GRU модела

За обраду временских низова развијен је модел *GRUSharedRepresentation*. Његова основна карактеристика је заједничка скривена репрезентација која се користи за два задатка: класификацију стања система и процену стања подсистема.

На улазу се спајају нормализоване вредности и њихове маске. Добијени тензор затим се пакује функцијом *pack_padded_sequence*, која користи стварне дужине низова и спречава да елементи додати током допуњавања утичу на резултат. Запаковане секвенце пролазе кроз вишеслојну GRU мрежу. Број слојева и величина скривеног стања задају се кроз конфигурацију. Као заједничка репрезентација секвенце користи се последње скривено стање завршног GRU слоја.

Из заједничке репрезентације формирају се две излазне гране. Класификациона грана предвиђа да ли се систем налази у нормалном стању, стању упозорења или критичном стању. Друга грана даје пет оцена стања, које описују целокупан систем, процесор, радну меморију, батерију и складишни простор. Над овим излазима примењује се сигмоидна функција, како би добијене вредности остале у интервалу од 0 до 1.



Слика 4. 4 Архитектура GRU модела са дељеном репрезентацијом

4.2.6 Функција грешке и обука модела

Модел истовремено решава задатак класификације стања уређаја и процене стања његових подсистема. Због тога се укупна функција грешке формира као комбинација класификационе грешке и грешке процене стања $L = L_{cls} + \lambda \cdot L_{health}$. За класификацију се користи унакрсна ентропија, док се за процену стања користи средња квадратна грешка. Утицај регресионог дела контролише се параметром λ, који се задаје у конфигурационој датотеци.

Током сваке епохе модел обрађује пакете временских низова, израчунава грешку и ажурира своје параметре. За оптимизацију се користи *Adam* оптимизатор, уз ограничавање градијената ради стабилнијег поступка обуке.

4.2.7 Валидација, контрола обуке и тестирање модела

Након сваке епохе модел се проверава над валидационим скупом. При томе се прате вредност функције грешке, тачност класификације и матрица забуне, која показује успешност препознавања појединачних класа. Уколико током више епоха не долази до побољшања, стопа учења се смањује. Такође је примењено рано заустављање, чиме се обука прекида када даљи рад више не доноси боље резултате. Модел се чува сваки пут када постигне мању валидациону грешку од претходно најбоље вредности. Резултати обуке по епохама бележе се у посебну датотеку ради касније анализе. По завршетку обуке учитава се верзија модела која је остварила најбољи резултат на валидационом скупу. Она се затим проверава над издвојеним тестним подацима. Током тестирања израчунавају се коначна грешка, тачност класификације и матрица забуне. Пошто тестни скуп није коришћен током обуке и избора најбоље епохе, добијени резултати представљају процену рада модела над претходно непознатим секвенцама.

4.3 Модул за рад у реалном времену

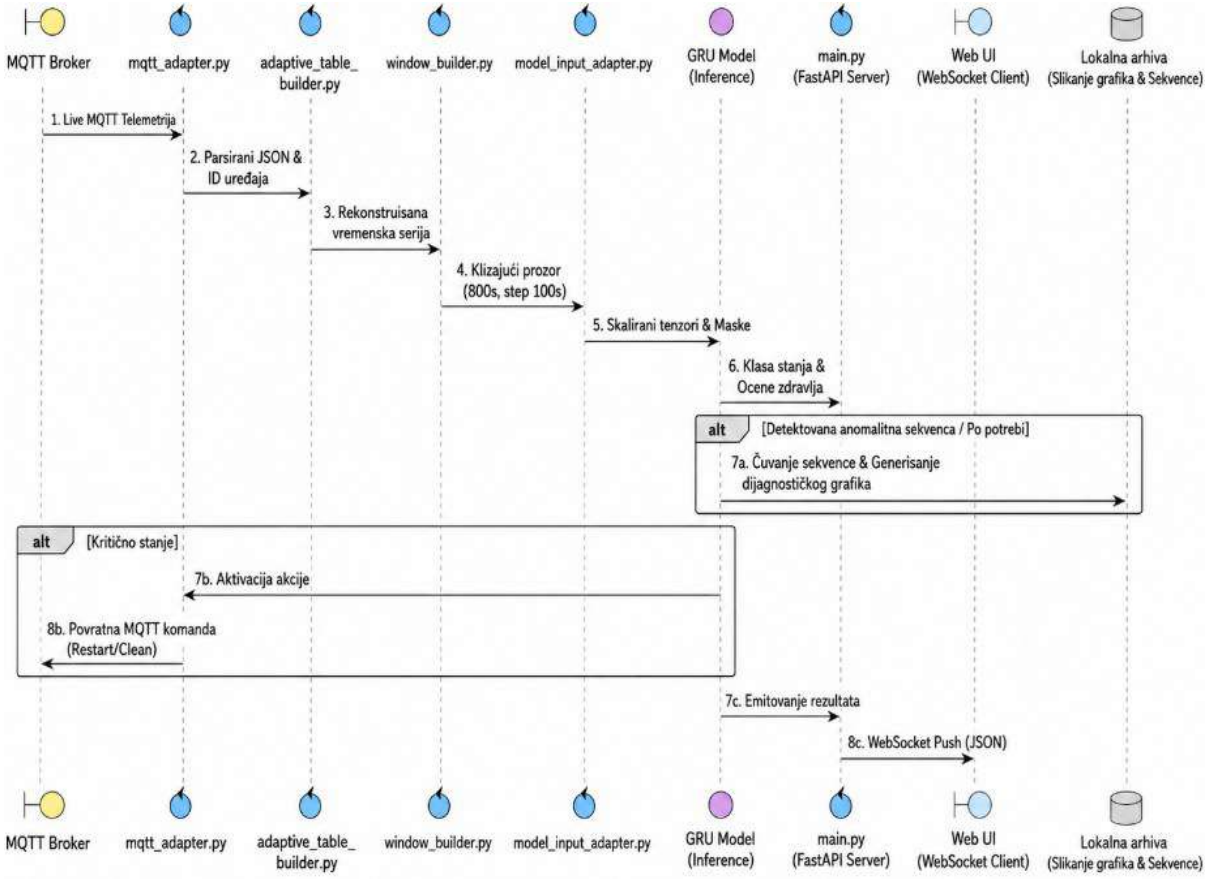
Модул за рад у реалном времену представља оперативни део система који непрекидно прима дијагностичке податке, организује их у временске низове и прослеђује обученом моделу ради процене стања уређаја. Поред извршавања модела, модул је задужен за приказ резултата у веб окружењу, локално чување обрађених секвенци и слање повратних команди у случају откривеног проблема.

Пошто различите метрике не морају пристизати истом учесталошћу, подаци се најпре групишу по уређају и реконструишу у јединствену временску серију. Након тога се формирају клизајући прозори погодни за обраду GRU моделом. Резултат анализе обухвата класу глобалног стања и појединачне оцене стања система, процесора, радне меморије, батерије и складишног простора.

4.3.1 Ток обраде података

Основни параметри оперативног модула задају се у датотекама *system.json* и *devices.json*. Датотека *system.json* садржи списак подржаних метрика, избор метрика које модел користи и параметре клизајућег прозора, односно његово трајање и корак

померања. Датотека *devices.json* садржи регистроване уређаје и њихове референтне хардверске вредности, које се користе приликом нормализације података и одређивања опсега графика. Учитавање и провера ових података обавља се приликом покретања апликације.



Слика 4. 5 Ток обраде података у модулу за рад у реалном времену

Дијагностички подаци стижу преко MQTT посредника до модула *mqtt_adapter*. Примљена JSON порука се затим обрађује у модулу *parser*, који издваја идентификатор уређаја, временски интервал и вредности метрика. На основу идентификатора подаци се усмеравају ка одговарајућој структури у меморији.

Главни проблем приликом реконструкције представља то што је учесталост мерења већа од учесталости слања MQTT порука. Због тога једна порука не мора да садржи само једну вредност сваке метрике, већ листе свих мерења насталих између два тренутка слања. Симулатор гарантује да су вредности из сваке листе настале унутар интервала који припада примљеној поруци. Листе различитих метрика не морају да имају једнаку дужину. На пример, у истом временском интервалу оптерећење процесора може бити измерено више пута, температура батерије само једном, док листа вредности складишног простора може бити празна.

```
"channels": [
  {
    "name": "data",
    "description": "Data channel: periodic with 1s message interval.",
    "messageStrategy": "periodic",
    "messageInterval": "5s",
    "dataItems": [
      { "id": "cpu_usage_pct", "sampling": "periodic", "samplingInterval": "1s", "alwaysSent": true },
      { "id": "battery_pct", "sampling": "periodic", "samplingInterval": "5s", "alwaysSent": true },
      { "id": "battery_temp_c", "sampling": "periodic", "samplingInterval": "10s", "alwaysSent": false },
      { "id": "is_charging", "sampling": "periodic", "samplingInterval": "1s", "alwaysSent": true },
      { "id": "mem_avail_mb", "sampling": "periodic", "samplingInterval": "1s", "alwaysSent": true },
      { "id": "free_storage_mb", "sampling": "periodic", "samplingInterval": "10s", "alwaysSent": true },
      { "id": "cpu_temp_c", "sampling": "periodic", "samplingInterval": "1s", "alwaysSent": true }
    ]
```

Слика 4. 6 Пример конфигурације канала за податке симулатора

MQTT poruka 1

```
{
  "id": "android_student_001",
  "ch": "data",
  "ts": 1785921938213,
  "mi": 5,
  "uc": 0,
  "data": {
    "cpu_usage_pct": [0, 38.16, 1, 38.09, 2, 35.59, 3, 34.67, 4, 32.76],
    "battery_pct": 69,
    "is_charging": [0, false, 1, false, 2, false, 3, false, 4, false],
    "mem_avail_mb": [0, 1266, 1, 1264, 2, 1268, 3, 1269, 4, 1279],
    "free_storage_mb": 10746,
    "cpu_temp_c": [0, 49.5, 1, 49.5, 2, 49.6, 3, 49.5, 4, 49.1]
  }
}
```

MQTT poruka 2

```
{
  "id": "android_student_001",
  "ch": "data",
  "ts": 1785921943215,
  "mi": 5,
  "uc": 0,
  "data": {
    "cpu_usage_pct": [0, 23.49, 1, 23.29, 2, 22.55, 3, 24.43, 4, 24.77],
    "battery_pct": 69,
    "battery_temp_c": 31.9,
    "is_charging": [0, false, 1, false, 2, false, 3, false, 4, false],
    "mem_avail_mb": [0, 1273, 1, 1273, 2, 1267, 3, 1267, 4, 1263],
    "free_storage_mb": 10741,
    "cpu_temp_c": [0, 48.4, 1, 47.9, 2, 47.2, 3, 46.7, 4, 46.2]
  }
}
```

Слика 4. 7 Формат порука за канал конфигурисан по Слици 4.6

Модул adaptive_table_builder овакве податке претвара у заједничку временску серију. Вредности сваке листе распоређују се унутар познатог временског интервала, уз очување њиховог редоследа. На овај начин се мерењима која немају појединачне временске ознаке додељују одговарајући временски положаји. При томе се не стварају нове измерене вредности, већ се постојећа мерења временски распоређују и повезују са осталим метрикама. Пошто метрике имају различиту учесталост мерења, њихове вредности се затим поравнавају у јединственој табели. Ако за одређену метрику у новом интервалу није пристигло мерење, користи се њена последња позната вредност ради одржавања континуитета серије. Може се претпоставити да метрика има вредност претходног мерења јер није могуће наканадно мењање фреквенције слања након покретања одређеног канала за слање и поједине метрике су конфигурисане да се не шаљу уколико се вредност није променила.

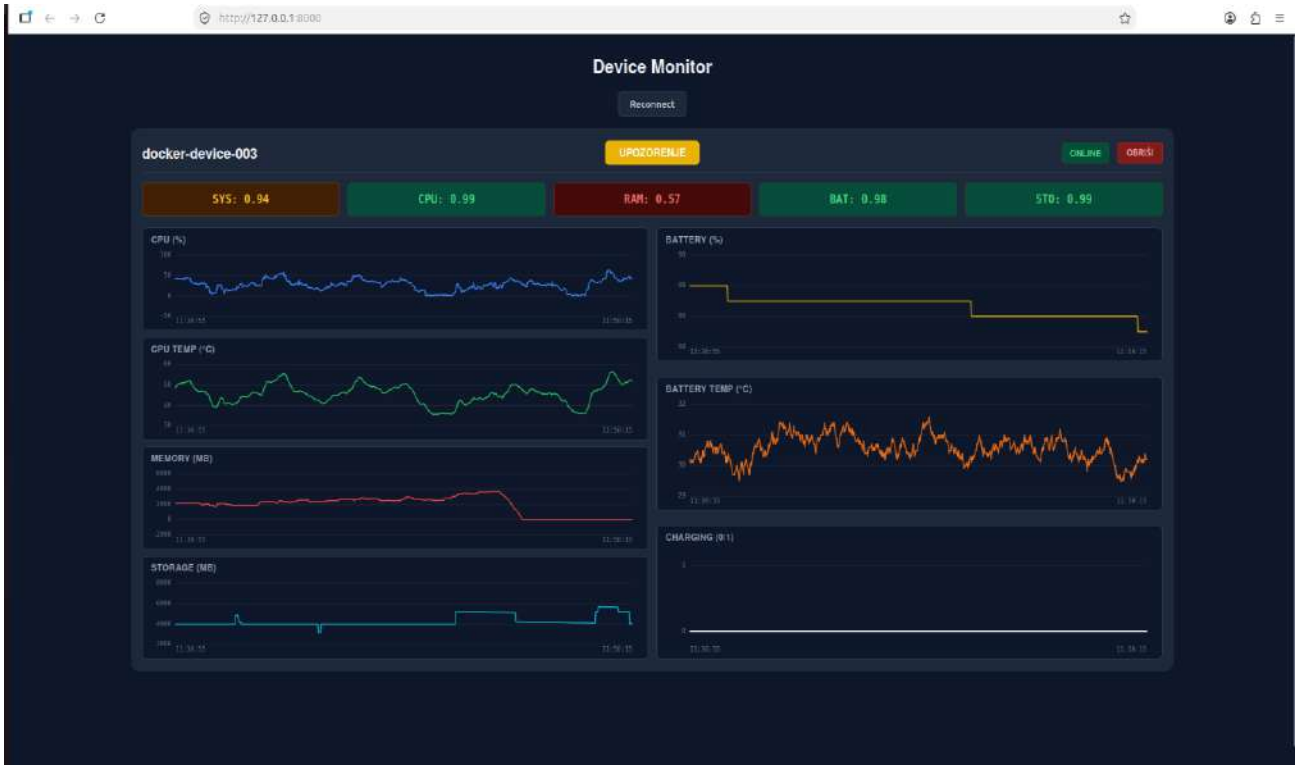
Реконструисани подаци се чувају одвојено за сваки уређај и сортирају према времену. Закаснели подаци који припадају већ обрађеном делу серије се одбацују. Пошто су такви случајеви ретки, поновно отварање и измена већ обрађеног прозора створили би већу сложеност и могли би да доведу до поновљених или међусобно неусаглашених процена.

Након реконструкције, модул *window_builder* издваја временски прозор дефинисаног трајања. Када је прикупљен довољан временски распон, прозор се прослеђује на обраду, а његов почетак се помера за конфигурисани корак. На тај начин се суседни прозори могу делимично преклапати, што омогућава чешће израчунавање стања без губитка претходног контекста. Поред вредности метрика, за сваки временски корак израчунава се и размак у односу на претходно мерење. Над њим се примењује логаритамска трансформација $ln(1+\Delta t)$, како би модел добио информацију о неуједначеном времену пристизања података, а да велики временски размаци не би имали несразмерно велики утицај. Логаритамска трансформација смањује распон вредности временског размака тако што веће интервале сабија знатно више од мањих, при чему се задржава њихов међусобни однос. На тај начин временски размак остаје корисно улазно обележје, али се спречава да његове велике вредности током матричних операција имају несразмерно велики утицај на остала обележја и рад мреже.

Модул model_input_adapter из прозора издваја само метрике омогућене у конфигурацији. Вредности се нормализују у односу на референтне границе конкретног уређаја, формирају се маске присутности и подаци се претварају у тензоре погодне за GRU модел.

Модел као резултат враћа класу глобалног стања и пет оцена стања. Резултат се прослеђује главном модулу, који га путем *WebSocket* везе шаље веб окружењу. По потреби се обрађени времески прозор и одговарајући график чувају у локалној архиви. У случају критичног стања може се покренути корективна акција и преко MQTT адаптера послати команда симулатору, на пример за поновно покретање уређаја или ослобађање меморије.

4.3.2 Приказ резултата у веб окружењу



Слика 4. 8 Веб приказ стања уређаја и дијагностичких метрика

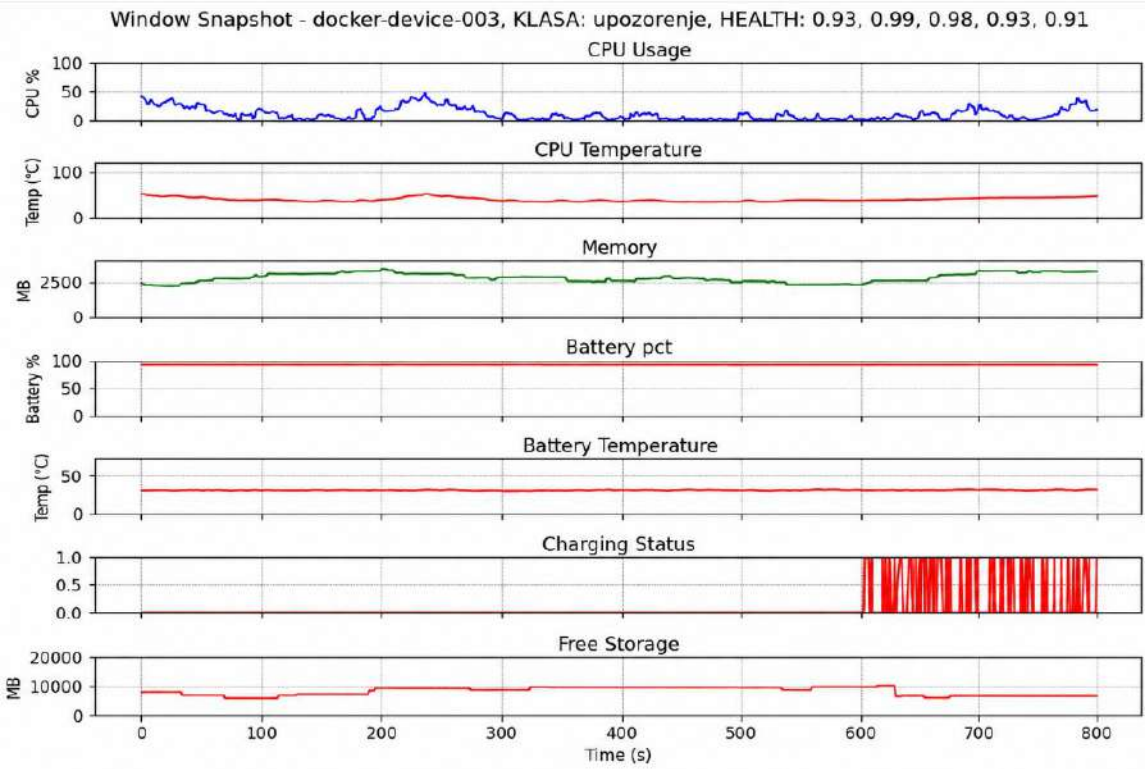
Веб окружење служи за праћење повезаних уређаја и резултата анализе у реалном времену. За сваки уређај приказују се његов идентификатор, статус повезаности, процењена класа стања и оцене система и подсистема. Поред заједничке оцене система, засебно су приказане оцене процесора, радне меморије, батерије и складишног простора. Боје омогућавају брзо уочавање подсистема који има снижену оцену. У овом случају најлошије је оцењена радна меморија, што одговара класи упозорења.

Испод оцена налазе се графици оптерећења и температуре процесора, количине меморије, слободног складишног простора, процента и температуре батерије и статуса пуњења. Графици се допуњавају новим подацима добијеним преко *WebSocket* везе, без потребе за поновним учитавањем странице.

Овај приказ омогућава да се резултат модела не посматра изоловано, већ заједно са вредностима које су довеле до конкретне процене. На тај начин корисник може да утврди да ли је промена класе последица раста температуре, пада расположиве меморије, промене стања батерије или другог негативног тренда.

Треба напоменути да се веб окружење покреће аутоматски на уређајима који имају прикључен екран и могућност брисања надзора уређаја за случај да више не шаље податке.

4.3.3 Локално чување дијагностичких прозора



Слика 4. 9 Локално сачуван дијагностички временски прозор

Поред приказа у веб окружењу, систем може локално да сачува временску секвенцу и њен график која је прослеђена моделу. Сачувани график садржи све улазне метрике за посматрани временски прозор, као и класу и оцене које је модел израчунао. У наслову графика бележе се идентификатор уређаја, предвиђена класа и пет оцена стања. Појединачни графици приказују оптерећење и температуру процесора, стање меморије, проценат и температуру батерије, статус пуњења и слободан складишни простор. Локално чување има две главне намене. Прва је накнадна анализа разлога због којих је модел одредио одређену класу. Друга је издвајање занимљивих или погрешно класификованих секвенци које се касније могу означити и употребити за фино подешавање модела.

5. Резултати

У овом поглављу приказани су резултати испитивања реализованог система и његових главних целина. Најпре су анализирана основна својства формираног скупа података, са циљем провере међусобних зависности метрика и њиховог понашања кроз време.

Након анализе скупа података испитан је рад обучених GRU модела над претходно непознатим сценаријима и упоређени су модели обучени са различитим вредностима појединих параметара мреже. Затим је проверен рад целокупне апликације у реалном времену и исправност појединачних програмских целина помоћу јединичних тестова.

На крају је обучени модел претворен у ONNX формат и покренут без зависности од библиотеке PyTorch. Упоређене су брзина извршавања и потрошња рачунарских ресурса приликом покретања модела у окружењу Python/PyTorch, Python/ONNX Runtime и C++/ONNX Runtime, чиме је испитана погодност модела за примену на уређајима са ограниченим ресурсима.

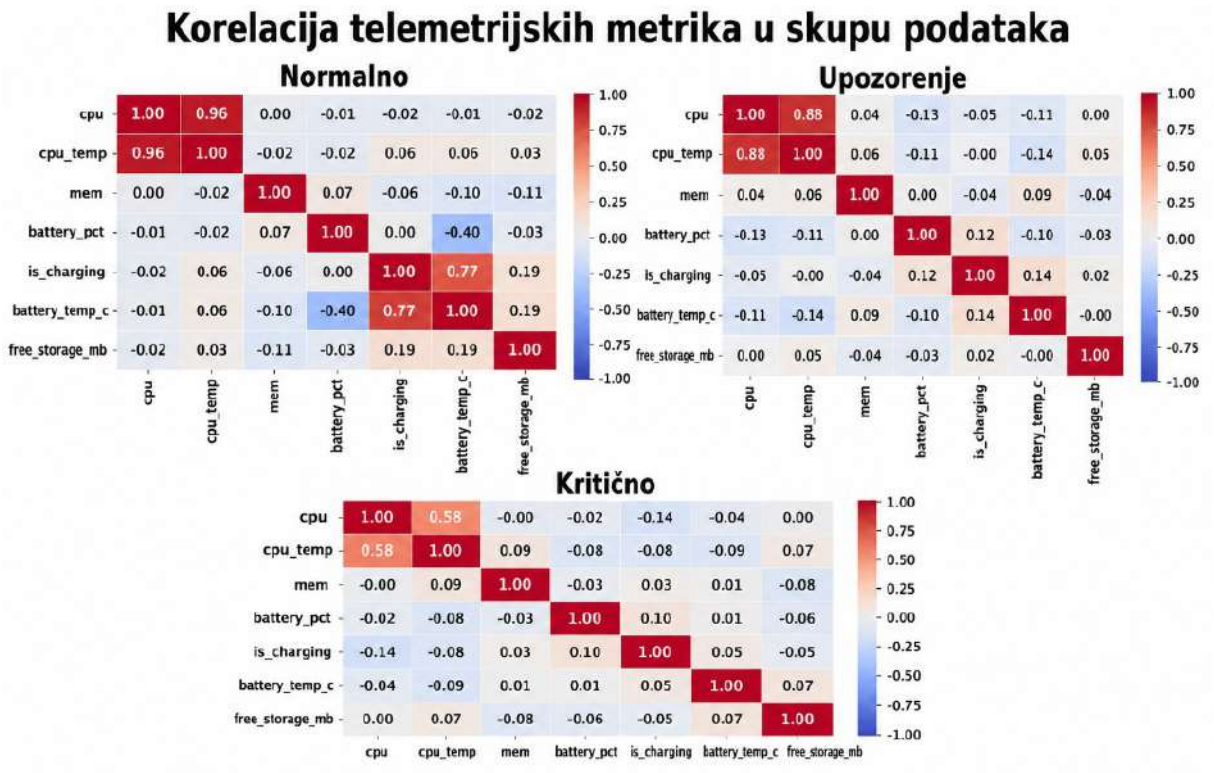
5.1 Анализа скупа података

Пре испитивања обучених модела анализирана су основна статистичка и временска својства скупа података добијеног радом симулатора. Скуп садржи временске низове дијагностичких метрика прикупљене током различитих сценарија рада уређаја, који су разврстани у нормално стање, стање упозорења и критично стање. Пошто је поступак формирања и организације скупа детаљно описан у претходном поглављу, овде је нагласак стављен на проверу својстава самих података.

Анализа је извршена посматрањем међусобне корелације метрика, њихове зависности од претходних вредности и расподеле промена у фреквенцијском домену.

Циљ ових испитивања није потврда да вештачки подаци у потпуности одговарају мерењима стварних уређаја, већ провера да ли симулатор ствара временске низове са смисленим зависностима, трендовима и различитим понашањем у зависности од режима рада.

5.1.1 Корелација дијагностичких метрика



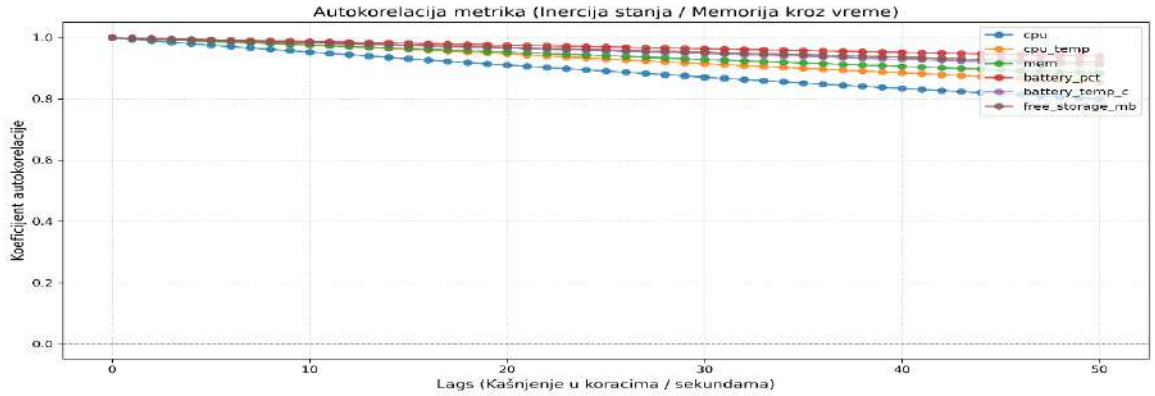
Слика 5. 1 Корелација дијагностичких метрика за различите режиме рада (нормално/упозорење/критично)

Први део анализе односи се на међусобну корелацију дијагностичких метрика. Корелациона анализа омогућава проверу у којој мери се промена једне метрике одражава на промену друге. Вредност коефицијента блиска јединици означава јаку позитивну зависност, вредност блиска минус јединици јаку негативну зависност, док вредности око нуле указују на одсуство изражене линеарне повезаности.

Најизраженија зависност уочава се између оптерећења и температуре процесора. У нормалном режиму коефицијент корелације износи приближно 0,96, у стању упозорења 0,88, док у критичном стању опада на око 0,58. Ово показује да симулатор задржава очекивану повезаност сродних метрика, али да различити сценарији рада могу

мењати њихов међусобни однос. Остале метрике углавном показују слабије зависности, што указује да се промене различитих подсистема не одвијају као један заједнички сигнал.

5.1.2 Временска зависност метрика



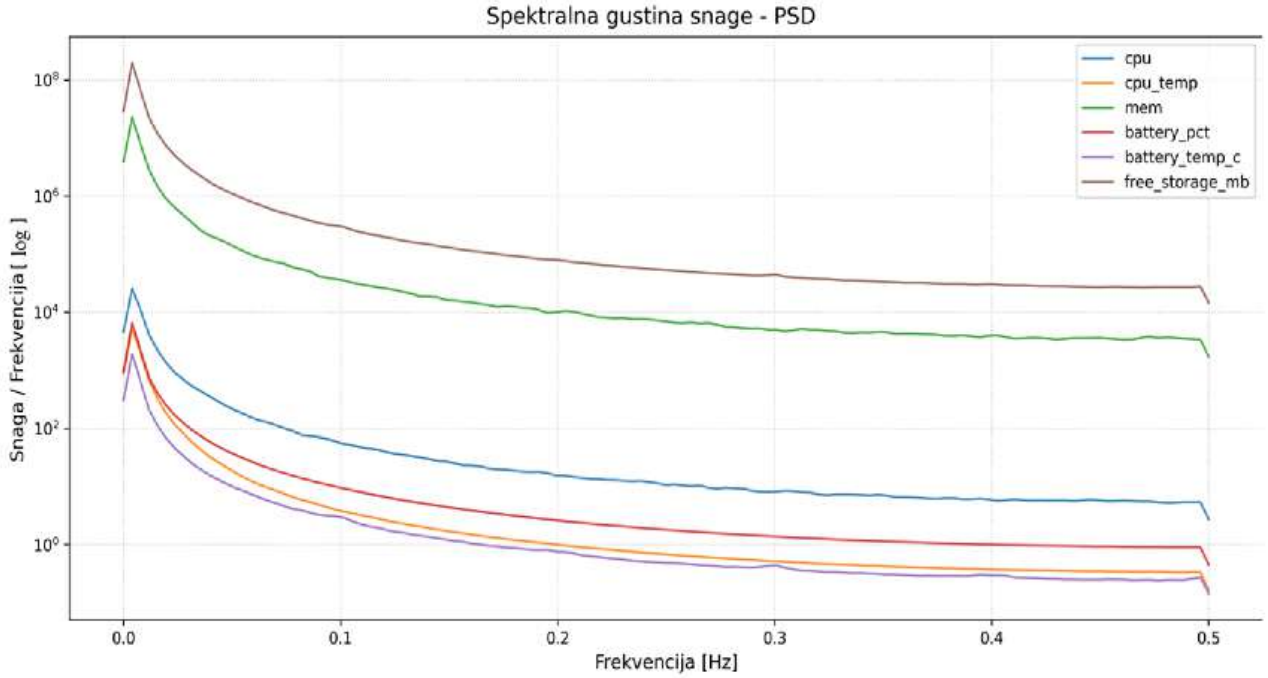
Слика 5. 2 Аутокорелација дијагностичких метрика током промене режима рада уређаја

Временска зависност метрика испитана је применом аутокорелационе анализе. Аутокорелација показује у којој мери је тренутна вредност неке метрике повезана са њеним вредностима из претходних временских тренутака. Висока вредност аутокорелације при већем кашњењу указује да метрика задржава утицај претходног стања и да њене промене нису независне од времена. Ова анализа је корисна за проверу да ли симулирани подаци заиста садрже временску структуру коју рекурентни модел може да искористи.

За анализу је изабран уређај који је током највећег дела посматраног периода радио у нормалном режиму, али је повремено пролазио кроз стања упозорења и критична стања. Учесталост ових промена у симулираном низу намерно је нешто већа него што би се очекивало у уобичајеном раду уређаја, како би скуп садржао довољан број примера различитих стања.

Са Слике 5.2 може се уочити да све посматране метрике задржавају висок степен повезаности са сопственим претходним вредностима и при већим временским кашњењима. Коефицијент аутокорелације опада постепено, а не нагло, што указује на постојање временске инерције у подацима. Овакво понашање је значајно за примену GRU модела, јер показује да информација о стању уређаја није садржана само у тренутној вредности метрике, већ и у њеном развоју током претходних временских корака.

5.1.3 Анализа у фреквентном домену



Слика 5. 3 Спектрална густина снаге дијагностичких метрика током промене режима рада уређаја

Временска својства створених података додатно су анализирана у фреквенцијском домену применом спектралне густине снаге. Ова анализа показује у којим фреквенцијским областима су најизраженије промене посматраних метрика. Ниске фреквенције одговарају споријим и дуготрајнијим променама стања, док више фреквенције представљају брзе промене и осцилације сигнала.

На Слици 5.3 уочава се да је код свих посматраних метрика највећи део снаге концентрисан у области ниских фреквенција, након чега вредности постепено опадају. То указује да су у низовима података најзасупљеније постепене промене и трендови, а не искључиво краткотрајне и насумичне осцилације. Такав резултат је у сагласности са претходно приказаном аутокорелацијом и додатно потврђује да симулирани подаци поседују изражену временску структуру погодну за обраду рекурентном неуронском мрежом.

На основу корелационе, аутокорелационе и спектралне анализе може се закључити да формирани скуп података садржи међусобне и временске зависности метрика, као и различите обрасце понашања током промене режима рада уређаја.

5.2 Испитивање и избор модела

Након анализе својстава скупа података испитан је рад више варијанти GRU модела. Основна архитектура модела задржана је непромењена, док је мењана величина

скривеног стања како би се испитао утицај капацитета мреже на процес обуке, способност уопштавања и брзину извршавања. Испитани су модели са два GRU слоја и величинама скривеног стања од 32, 64 и 128 јединица.

Поређење није засновано искључиво на резултатима валидације током обуке. Модели су додатно испитани на новим тестним сценаријима који нису били део скупа коришћеног за обучавање, као и на сачуваним временским низовима добијеним директним радом симулатора. На овај начин било је могуће упоредити понашање модела и ван самог скупа за обуку и валидацију.

5.2.1 Величина и сложеност испитаних модела

Приликом испитивања модела задржана је иста архитектура мреже, док је мењана величина скривеног стања GRU слојева. Испитане су варијанте са 32, 64 и 128 јединица у скривеном стању, при чему су све мреже имале два GRU слоја и исте излазне гране. На овај начин је било могуће испитати како повећање капацитета мреже утиче на тачност, способност уопштавања и захтеве приликом извршавања.

На улаз модела доводи се осам вредности, које обухватају седам дијагностичких метрика и временски размак између узастопних мерења. За сваку од ових вредности прослеђује се и одговарајућа маска присутности, па је укупан број улазних обележја у GRU мрежу: I = 16. Након два GRU слоја, последње скривено стање представља заједничку репрезентацију временске секвенце и прослеђује се ка две излазне гране. Прва грана врши класификацију у једну од три класе стања система, док друга производи пет континуалних оцена стања система и његових подсистема.

За коришћену архитектуру укупан број параметара модела може се изразити формулом:

$$P = 3 \cdot H \cdot I + 9 \cdot H^2 + 12 \cdot H + H \cdot C + C + H \cdot S + S$$

Где је :

- I – број улазних обележја
- H – величина скривеног стања
- C - број класа класификационог излаза
- S – број континуалних оцена стања

У реализованом моделу важи I=16, C=3 и S=5, па се добија:

$$P = 9 \cdot H^2 + 68 \cdot H + 8$$

Величина скривеног стања	Број параметара	Процесорско време закључивања
32	11 400	107.335 ms
64	41 224	107.600 ms
128	156 168	162.275 ms

Табела 5. 1 Број параметара и време извршавања испитаних GRU модела

Из добијених вредности види се да број параметара расте знатно брже од саме величине скривеног стања. Разлог је присуство члана $9H^2$, који потиче од веза унутар и између GRU слојева. Повећањем скривеног стања са 32 на 64 јединице број параметара расте приближно 3,6 пута, док модел са 128 јединица има око 3,8 пута више параметара од модела са 64 јединице.

Практична разлика између модела уочена је и током саме обуке. Обука је извршавана на процесору, на рачунару са 16 GB RAM меморије, без NVIDIA графичке картице и могућности коришћења CUDA окружења. У зависности од тренутног оптерећења система, модел са 32 јединице обрађивао је приближно 15–18 епоха дневно, модел са 64 јединице око 6–7, док је код модела са 128 јединица брзина износила приближно 3 епохе дневно. Иако ове вредности не представљају контролисано мерење перформанси, оне показују практичан раст времена потребног за обуку са повећањем величине модела.

Већи број параметара даје моделу већи капацитет за представљање сложених зависности у временским низовима, али истовремено повећава меморијске и рачунарске захтеве и не гарантује боље уопштавање на нове податке. Због тога величина модела није посматрана изоловано, већ су у наредним испитивањима упоређени ток обуке, резултати над новим сценаријима и понашање сваке од три варијанте модела.

5.2.2 Ток обуке и избор контролне тачке

Током обуке праћени су губитак над скупом за обуку, губитак над валидационим скупом и тачност класификације над валидационим скупом. Валидациони губитак је посебно значајан јер се користи за чување најбоље контролне тачке модела и за механизам раног заустављања. Укупна функција губитка обухвата класификациони и регресиони део, односно губитак способности класификације и грешку процене оцена стања.

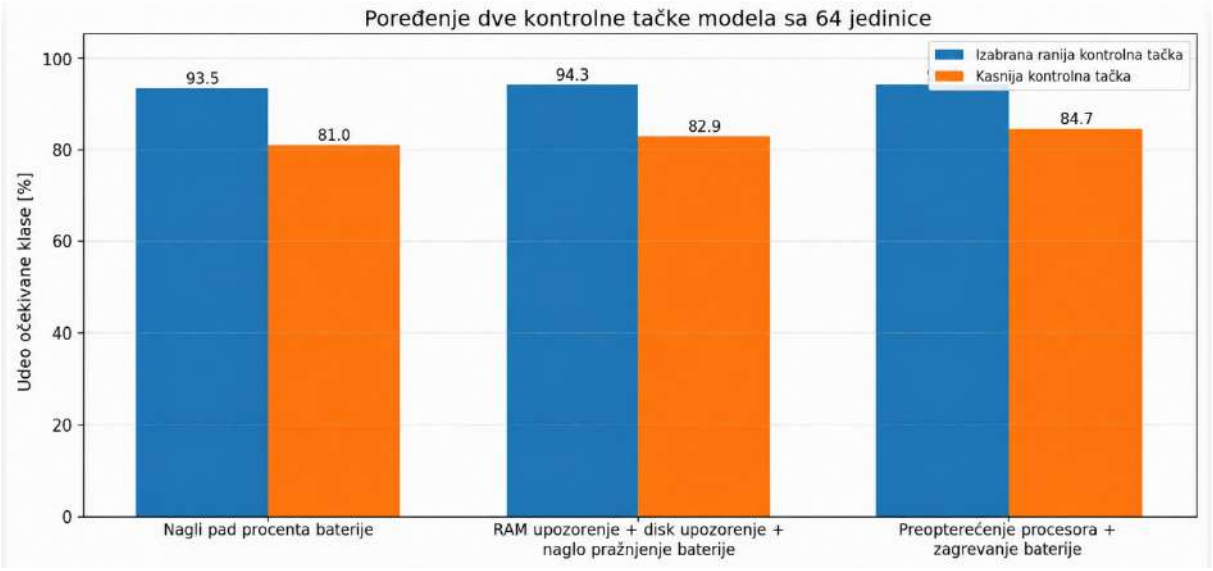
Модел са 32 јединице показао је правилну конвергенцију током 65 епоха. Валидациони губитак се са почетне вредности 0,9960 постепено смањивао, а најнижа вредност од 0,1510 забележена је у 61. епохи, уз валидациону тачност од 98,0%.

Додатно је испитана варијанта истог модела са 32 јединице код које су све класе имале једнаке тежине, а утицај регресионог дела функције губитка био је смањен. Ова варијанта достигла је најмањи валидациони губитак од 0,1595 у 50. епохи. Како у наредних 15 епоха није постигнута мања вредност, механизам раног заустављања прекинуо је даљу обуку. То не представља неуспелу конвергенцију, већ пример ситуације у којој наставак обуке више није доносио побољшање према изабраном валидационом критеријуму.

Модел са 64 јединице показао је већи капацитет за представљање сложенијих временских зависности. Након почетне фазе обуке, обука је настављена над целим скупом при чему је стопа учења смањена са 10^{-3} на $3 \cdot 10^{-4}$, а тежински фактори класа промењени су са [1.0, 1.05, 1.1] на [1.01, 1.03, 1.05], чиме је смањена разлика у кажњавању грешака између различитих класа. Овај поступак не представља фино подешавање новим подацима, већ наставак обуке већ формираног модела са мањим изменама параметара по једној итерацији.

У наставку обуке валидациони губитак је додатно смањен. На једној од ранијих контролних тачака износио је око 0,1286, уз валидациону тачност од 98,66%, док је касније достигао 0,1190 и тачност од 99,01%. На основу самих валидационих показатеља могло би се очекивати да је каснија контролна тачка боља. Међутим, додатни тестови над новим вештачким сценаријима показали су да то није увек случај.

На пример, код сценарија наглог пада нивоа попуњености батерије ранија контролна тачка исправно је препознавала стање упозорења у 93,5% секвенци, док је код касније контролне тачке тај удео пао на 81,0%. Код комбинације упозорења радне меморије, складишта и умереног пада нивоа попуњености батерије резултат се смањио са 94,3% на 82,9%, а код комбинације критичног оптерећења процесора и упозорења температуре батерије са 99,7% на 84,7%.

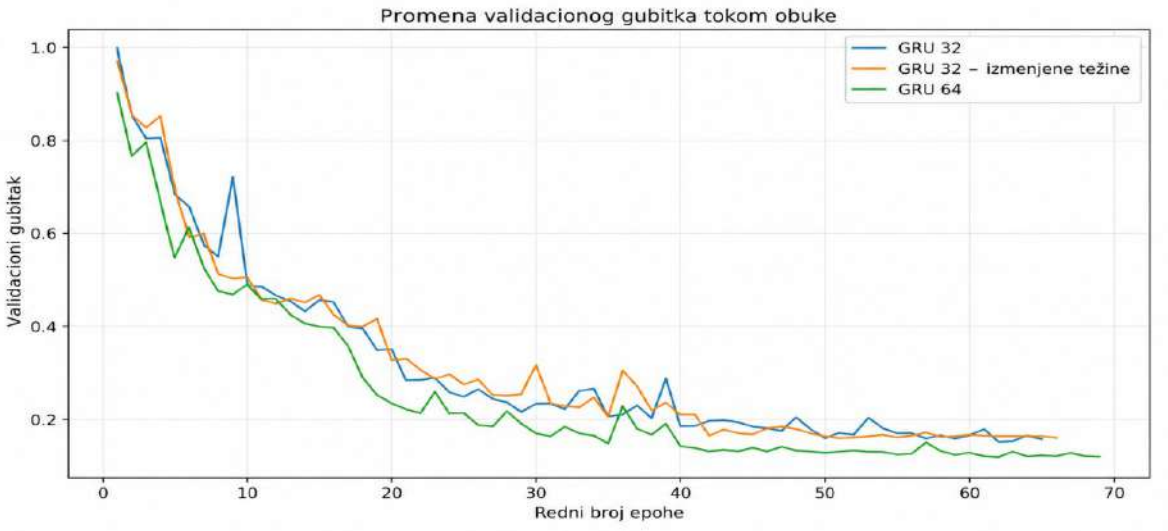


Слика 5. 4 Поређење изабране и касније контролне тачке модела са 64 јединице по слоју

Овај резултат показује да мањи губитак над валидационим скупом није сам по себи довољан критеријум за избор модела. Валидациони скуп припада истој основној расподели као подаци за обуку, па се током даље обуке модел може све боље прилагођавати специфичностима тог скупа, без одговарајућег побољшања над новим реализацијама сценарија. Додатно, пошто валидациони губитак представља комбинацију класификационог и регресионог губитка, његово смањење не мора значити да се истовремено побољшава свака појединачна класификациона одлука.

Због тога је као коначна варијанта модела са 64 јединице изабрана ранија контролна тачка која је показала стабилније понашање у независним тестовима, иако није имала апсолутно најмањи валидациони губитак.

Модел са 128 јединица такође је успешно обучен и у бројним тест сценаријима постигао је веома добре резултате. Ипак, већи капацитет није донео систематско побољшање у свим случајевима. Пошто је ова варијанта обучавана краће, око 40 епоха, на основу спроведеног експеримента није могуће закључити како би се понашала при знатно дужој обуци. За потребе реализованог система није показана довољна практична предност која би оправдала готово четири пута већи број параметара у односу на модел са 64 јединице.



Слика 5. 5 Промена валидационог губитка током обуке испитаних модела

5.2.3 Испитивање на тестним сценаријима

Поред резултата добијених током обуке, модели су испитани помоћу посебне тест скрипте која ствара нове временске низове за унапред дефинисане сценарије. За свако испитивање ствара се 1000 независних секвенци дужине 800 временских корака, при чему се због случајних компоненти симулације појединачне секвенце међусобно разликују. Тиме се проверава да ли модел препознаје општи образац одређеног стања, а не само конкретне примере из скупа за обуку. У тест скрипти аномалије обухватају цео посматрани прозор како би се изоловала способност модела да препозна облик и понашање одговарајућих метрика.

Сценарио	GRU 32	GRU 64	GRU 128
Нормално стање	98.1%	97.0%	98.4%
Упозорење диск	96.1%	97.8%	94%
Нагли пад процента батерије	75.0%	93.5%	72.8%
RAM упозорење + диск упозорење + нагло пражњење батерије	99.5%	94.3%	99.9%
Лош контакт пуњача	20.9%	99.4%	100%
Преоптерећење процесора + загревање батерије	99.3%	99.7%	100%

Табела 5. 2 Резултати модела на одабраним тестним сценаријима

Вредности у Табели 5.2 представљају удео секвенци које су сврстане у очекивану класу за конкретан сценарио. Међутим, приликом поређења модела није посматран само укупан проценат исправних предвиђања, већ и начин на који су распоређене погрешно класификоване секвенце. Код система за дијагностику није исто да ли је стање упозорења погрешно проглашено нормалним или критичним. Први случај представља пропуштање потенцијалног проблема, док други доводи до конзервативније процене стања и могућег непотребног аларма или реакције система.

Разлика се може уочити код више испитаних сценарија. На пример, за сценарио упозорења на диску модел са 32 јединице исправно класификује 96,1% секвенци, при чему 2,4% проглашава нормалним, а 1,5% критичним. Модел са 64 јединице постиже нешто бољих 97,8%, док само 0,1% секвенци погрешно проглашава критичним. Слично понашање се уочава и код других појединачних стања упозорења, где модел са 64 јединице углавном задржава веома мали број прелаза у погрешну суседну класу.

Посебно је занимљив сценарио у ком се истовремено RAM упозорење, диск упозорење и нагло пражњење батерије. Модел са 64 јединице исправно одређује класу упозорења у 94,3% случајева, што је мање од 99,5% код модела са 32 и 99,9% код модела са 128 јединица. Међутим, свих преосталих 5,7% секвенци модел са 64 јединице сврстава у критично стање, док ниједну не проглашава нормалном. За систем чији је задатак откривање проблема овакав тип грешке је повољнији од пропуштања аномалије, иако може довести до већег броја непотребних критичних упозорења.

Највеће разлике између модела јављају се код сложенијих и граничних сценарија. Код лошег контакта пуњача модел са 32 јединице препознаје очекивано стање упозорења у само 20,9% случајева, док модел са 64 јединице достиже 99,4%. Са друге стране, модел са 128 јединица није систематски бољи. Код сценарија наглог пада батерије модел са 64 јединице исправно класификује 93,5% секвенци, док модел са 128 јединица достиже 72,8%; код њега се 18,9% секвенци проглашава критичним, а 8,3% нормалним.

Резултати зато показују да повећање величине скривеног стања не доводи аутоматски до бољег понашања модела. Приликом избора коначне варијанте узети су у обзир укупан проценат исправних класификација, расподела погрешних предвиђања између суседних класа, стабилност на различитим типовима сценарија и сложеност самог модела. На основу ових критеријума модел са 64 јединице показао је најповољнији укупни однос.

Као додатна провера способности уопштавања формиран је и временски образац који није био присутан међу сценаријима коришћеним током обуке. Унутар секвенце

извршено је нагло ослобађање целокупног складишног простора, које се по облику разликовало од промена на којима је модел обучаван. Модел је више од 70% оваквих секвенци сврстао у критично стање. Овај резултат не представља меру тачности, јер за овај нови образац није постојала унапред дефинисана циљна класа, али показује да модел реагује на изражено неуобичајено временско понашање које није непосредно било присутно у скупу за обуку.

5.2.4 Испитивање на сачуваним записима симулатора

Као додатна провера, модели су примењени на 906 сачуваних временских низова добијених континуираним радом симулираног уређаја. Уређај је радио претежно у нормалном режиму, али су услед стохастичког карактера симулације поједини прозори могли оправдано да садрже вредности које одговарају стању упозорења. Због тога резултати нису посматрани као класична мера тачности, већ као провера стабилности предвиђања током дужег периода рада.

Модел	Нормално	Упозорење	Критично
GRU 32	74.72%	25.28%	0%
GRU 32 – измењене тежине	98.57%	1.21%	0.22%
GRU 64	97.68%	2.32%	0%
GRU 128	99.45%	0.22%	0.33%

Табела 5. 3 Расподела предвиђених класа над сачуваним записима симулатора

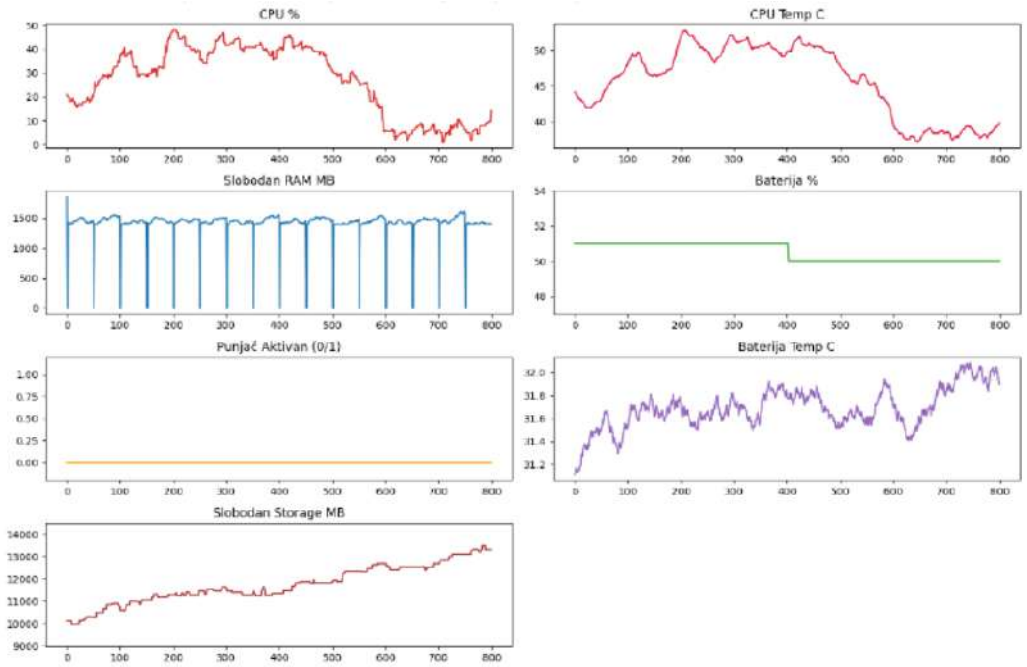
Основни модел са 32 јединице показао је знатно већу осетљивост, док су остале варијанте давале стабилнија предвиђања. Модел са 64 јединице је 97,68% прозора означио као нормалне, а преостале као упозорење, без иједног критичног предвиђања. Иако је модел са 128 јединица имао већи удео нормалних предвиђања, поједине прозоре је сврстао у критично стање.

При избору модела зато није посматран само укупан проценат нормалних предвиђања, већ и врста грешке. Модел са 64 јединице показао је добро понашање и код сложенијих сценарија повезаних са променом нивоа батерије и стањем пуњача, који су код других варијанти чешће доводили до погрешне класификације. Узимајући у обзир резултате обуке, сценарије из тест скрипте, сачуване записе симулатора и сложеност модела, за коначну примену изабран је модел са два GRU слоја и 64 јединице у скривеном стању. Изабрана је ранија контролна тачка, јер је у додатним

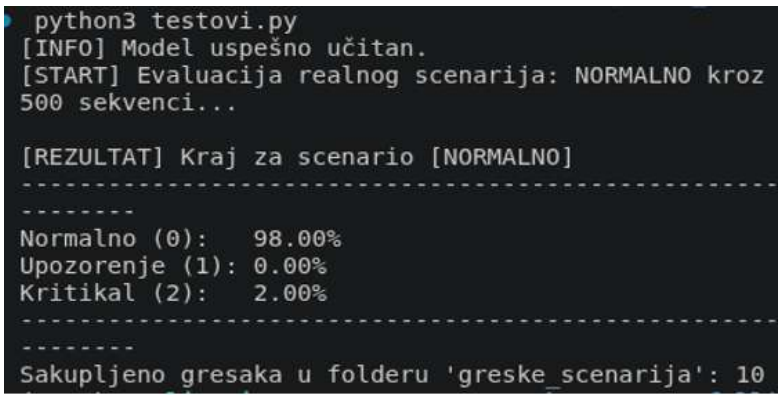
тестовима показала стабилније уопштавање од каснијих контролних тачака са нешто мањим валидационим губитком.

5.2.5 Додатно испитивање робустности модела

Додатно је испитано понашање изабраног модела у присуству краткотрајних наглих одступања улазних података. У нормалне временске секвенце уношени су периодични краткотрајни падови вредности расположиве RAM меморије, приближно на сваких 50 временских корака. Овакав образац није био присутан међу сценаријима коришћеним током обуке. Над 500 овако измењених секвенци модел је задржао понашање упоредиво са одговарајућим секвенцама без додатог шума. Резултат указује да изолована екстремна вредност не одређује непосредно излаз модела, већ да GRU приликом процене користи шири временски контекст.



Слика 5. 5 Пример временске секвенце са периодичним краткотрајним падовима RAM меморије



Слика 5. 6 Резултати рада модела над секвенцама са периодичним краткотрајним падовима RAM меморије

5.3 Испитивање модула за рад у реалном времену

Након избора обученог модела испитан је рад целокупног модула за дијагностику у реалном времену. Испитивање је извршено повезивањем апликације са MQTT посредником и симулатором уређаја, чиме је проверен целокупан ток података од пријема до приказа резултата модела.

Током рада симулатора проверено је да се пристигла мерења исправно издвајају из MQTT порука, реконструишу у јединствен временски низ и групишу у клизајуће прозоре. Након формирања довољног временског интервала, подаци се припремају за модел, након чега се добијена класа и оцене стања подсистема прослеђују веб окружењу. Променом режима рада симулираног уређаја проверено је да се приказано стање мења у складу са понашањем пристиглих метрика.

Посебно је проверен случај детекције критичног стања, при ком модул, поред приказа резултата, може да формира и пошаље одговарајућу MQTT команду ка уређају. Механизам временског ограничења поновног слања спречава вишеструко узастопно слање исте команде. На овај начин потврђен је рад целог пута од пријема телеметрије, преко анализе моделом, до приказа резултата и могућег повратног реаговања система.

Слика 5. 7 Веб приказ детектованог критичног стања уређаја

```
LOGITS: tensor([[-1.5427, -0.7940,  3.0962]])
HEALTH: [0.8076928  0.9262134  0.9892592  0.9847808
 0.03017146]
KLASA: kritikal

[MQTT POSLATO] -> Kritikal poslat na temu: dms/devic
es/docker-device-003/commands
[COMMAND SENT] app_clear_data -> docker-device-003

[MQTT POTVRDA] Simulator javlja sa teme [dms/devices
/docker-device-003/responses]: {
    "commandId": "cmd_app_clear_data_1786524984655",
    "status": "completed",
    "message": "App data cleared",
    "respondedAt": "2026-08-12T08:56:24.660639921Z"
}
```

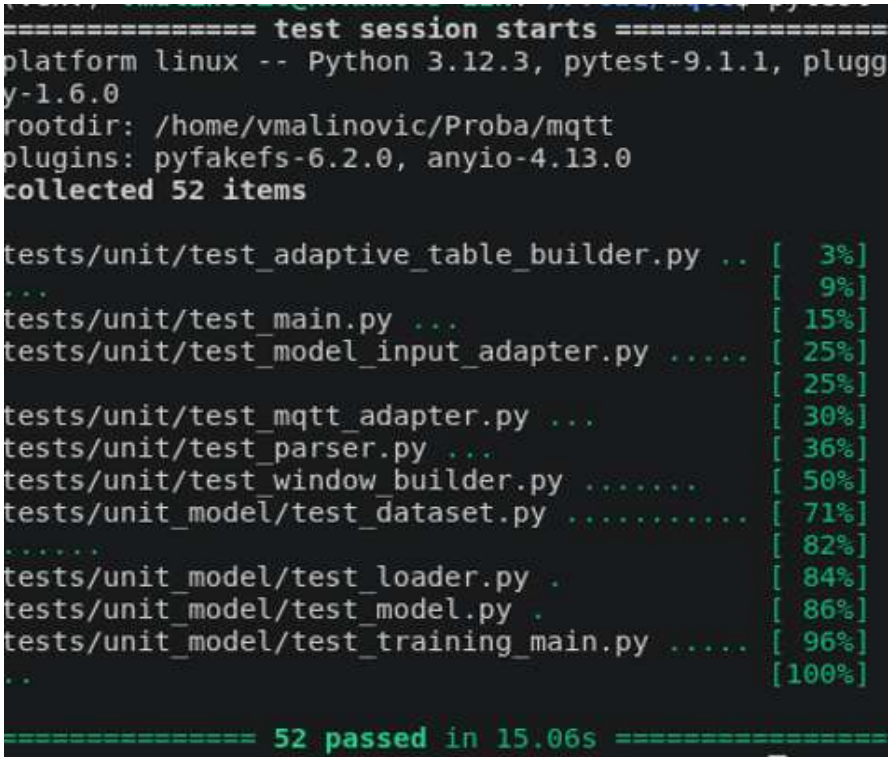
Слика 5. 8 Резултат слања и потврде MQTT команде након детекције критичног стања

5.4 Јединични тестови

Поред испитивања система као целине, исправност појединачних програмских компоненти проверена је јединичним тестовима изведеним помоћу оквира *pytest*. Тестови су обухватили компоненте за пријем и тумачење MQTT порука, реконструкцију временских података, стварање клизајућих прозора, припрему улаза за модел, саму структуру неуронске мреже, припрему пакета података и поступак стварања скупа за фино подешавање.

Посебна пажња посвећена је граничним и неисправним случајевима. Проверавано је понашање при недостатку појединих метрика, пријему више мерења у једној MQTT поруци, закаснелим или поновљеним временским ознакама, недовољној временској покривености прозора и празним улазним секвенцама. Такође је проверено исправно попуњавање секвенци различитих дужина, формирање маски и димензија улазних и излазних тензора модела.

Тестовима су обухваћене и помоћне функције апликације, као што су формирање управљачких MQTT команди и ограничење њиховог поновљеног слања, као и припрема нових података за накнадну обуку модела. Укупно су покренута 52 јединична теста и сви су успешно завршени, при чему је целокупно извршавање трајало 15,06 секунди.



Слика 5. 9 Резултат извршавања јединичних тестова

5.5 Пренос модела у ONNX формат и анализа перформанси

Након избора коначног модела испитана је могућност његовог покретања независно од библиотеке PyTorch. Модел је извезен у ONNX формат, који омогућава употребу обучене неуронске мреже у различитим програмским окружењима без потребе за присуством целокупног окружења у ком је модел обучаван [7]. Ово је посебно значајно у случају примене на уређајима са ограниченим процесорским и меморијским ресурсима.

За потребе извоза направљен је помоћни омотач око обученог модела. Оригинална реализација током обуке користи паковање низова различитих дужина, док је за ONNX извршавање задржан непосредан пролаз вредности и одговарајућих маски кроз GRU слој. На овај начин избегнута је зависност од PyTorch-специфичне структуре за паковане секвенце, а задржани су обучени GRU слој и обе излазне гране модела. Извезени модел има два улаза, вредности и маске, и два излаза, класификационе вредности и пет оцена стања. Приликом извоза задржана је и могућност променљиве дужине временске секвенце.

Након конверзије проверено је да ONNX модел даје практично исте резултате као оригинални PyTorch модел. Над 500 насумично створених улазних секвенци поређени су класификациони и регресиони излази оба модела. Просечна средња квадратна разлика износила је приближно $6.8 \cdot 10^{-12}$, што показује да конверзија није довела до значајне промене резултата модела.

За даљу проверу направљена је и C++ компонента заснована на ONNX извршној библиотеци. Она формира улазне тензоре директно из низова вредности и маски, покреће модел и враћа три класификациона излаза и пет оцена стања. Извршавање је реализовано на процесору, уз укључене оптимизације ONNX извршног графа и ограничење обраде на једну нит унутар операција.

Перформансе су упоређене покретањем истог типа временског прозора дужине 800 корака 500 пута у три окружења: Python/PyTorch, Python/ONNX Runtime и C++/ONNX Runtime. Пре мерења извршено је по десет пробних покретања како иницијализација не би утицала на резултат.

Окружење	Просечно време	CPU окружење	RAM
Python / PyTorch	96.617 ms	209.540 ms	496.26 MB
Python / ONNX Runtime	5.082 ms	4.980 ms	60.47 MB
C++ / ONNX Runtime	4.233 ms	4.232 ms	20.55 MB

Табела 5. 4 Поређење перформанси различитих окружења за извршавање модела

Резултате из различитих табела не треба директно поредити, јер мерења нису обављена под истим оптерећењем система. Зато се свака табела користи за међусобно поређење варијанти у оквиру истог огледа, а не за поређење апсолутних времена између различитих огледа.

Резултати показују значајно смањење времена извршавања и меморијских захтева након преласка са PyTorch-а на ONNX Runtime. Најмање просечно кашњење измерено

је у C++ реализацији и износило је око 4,23 ms по једном пролазу модела, уз заузеће процеса од око 20,55 MB RAM меморије. Python варијанта ONNX Runtime-а постигла је слично време од око 5,08 ms, али са већом меморијском потрошњом услед присуства Python окружења. Највеће захтеве имао је оригинални PyTorch модел, са око 96,6 ms по пролазу и приближно 496 MB заузете меморије. Добијени резултати показују да је ONNX/C++ варијанта погоднија за интеграцију модела у систем у ком су брзина обраде и расположиви ресурси ограничени.

6. Закључак

Резултати овог рада показују да је могуће реализовати систем за аутоматску процену стања уређаја на основу дијагностичких временских низова. Развијено решење обухвата прикупљање и припрему података, обуку GRU модела, анализу података у реалном времену, приказ резултата у веб окружењу и могућност слања повратних MQTT команди у случају откривеног критичног стања. Модел истовремено одређује класу глобалног стања уређаја и процењује стање појединачних подсистема.

Испитивањем више варијанти модела показано је да повећање величине мреже не доводи нужно до бољих резултата. Модел са 64 јединице у скривеном стању показао је најповољнији однос тачности, стабилности и сложености и због тога је изабран за коначну примену. Током обуке уочено је и да смањење валидационог губитка није увек праћено бољим резултатима на новим тестним сценаријима, због чега је избор коначне контролне тачке заснован и на додатним испитивањима способности уопштавања.

Додатно, пренос обученог модела у ONNX формат показао је да се његово извршавање може значајно убрзати и реализовати са знатно мањом потрошњом меморије у односу на директно коришћење PyTorch окружења. Тиме је показано да развијено решење не представља само огледни модел, већ основу за даљу интеграцију интелигентне дијагностике у IoT системе и уређаје са ограниченим рачунарским ресурсима.

Једна од предности приступа заснованог на GRU мрежи у односу на методе засноване на унапред дефинисаним правилима јесте могућност учења временских зависности између више дијагностичких метрика, без потребе да се за сваку могућу ситуацију појединачно дефинише правило. Код система заснованог искључиво на унапред задатим праговима, нови тип понашања захтевао би додавање новог правила,

док обучени модел може да реагује и на обрасце који нису непосредно били присутни током обуке. Такво понашање уочено је и приликом испитивања нагле промене расположивог складишног простора, где је модел већину оваквих секвенци препознао као значајно одступање и сврстао у критично стање. Иако овај резултат не гарантује препознавање произвољне непознате аномалије, указује на способност модела да научене временске зависности примени и ван строго дефинисаних сценарија обуке.

Највеће ограничење рада представља скуп података коришћен за обуку и испитивање модела. Формирани скуп садржи приближно 35.800 временских секвенци распоређених између нормалног стања, упозорења и критичног стања, што по самом броју узорака није занемарљиво. Ипак, сви подаци потичу из истог симулационог окружења и ограниченог скупа унапред дефинисаних сценарија, због чега њихова стварна разноврсност може бити мања него што укупан број секвенци сугерише. Ово постаје посебно значајно код модела већег капацитета, који могу све боље да се прилагођавају специфичностима синтетичких података без одговарајућег побољшања над новим реализацијама сценарија. Такав ефекат уочен је и код модела са 64 јединице, где је наставак обуке смањивао валидациони губитак, али није увек побољшавао резултате независних тестова.

Поред прикупљања стварних података, могуће је унапредити и сам начин формирања скупа за обуку. Један од праваца је стварање нових временских секвенци комбиновањем делова већ постојећих примера, на пример преузимањем појединих метрика или временских интервала из различитих секвенци исте или сродне класе. Додатна разноврсност могла би се добити и променљивим узорковањем података, тако да се у сваком примеру не користи сваки расположиви временски корак, већ се део мерења прескаче или задржава са различитом учесталошћу. На насумично изабраним местима могуће је додавати и контролисани шум, краткотрајне скокове или губитак појединих мерења.

Комбиновањем ових поступака од ограниченог броја основних секвенци могао би се формирати знатно већи број различитих примера за обуку, при чему би свака нова секвенца представљала нешто другачији временски ток. На тај начин би се повећала разноврсност скупа без потребе за једноставним стварањем великог броја веома сличних примера и смањила могућност да модел научи специфичне обрасце симулатора уместо општих зависности између метрика. При оваквом комбиновању било би неопходно водити рачуна да новоформирана секвенца задржи смислену класу и одговарајуће оцене стања.

Као даљи правац развоја саме архитектуре могу се испитати и механизам пажње, којим би се различитим деловима временског прозора додељивао различит значај, као и енкодер–декодер приступ за испитивање аномалија које нису обухваћене означеним скупом за обуку.

Због тога најзначајнији правац даљег унапређења није само повећање броја секвенци, већ пре свега повећање њихове разноврсности и реалистичности. Прикупљање података са стварних уређаја, различитих типова хардвера и различитих услова рада омогућило би поузданију проверу способности уопштавања модела и боље искоришћење већих архитектура. Додатно би било корисно проширити број граничних и комбинованих сценарија и формирати засебан тестни скуп који не потиче из истих правила стварања као подаци коришћени током обуке.

7. Литература

[1] **AWS Documentation:** *"What is Internet of Things (IoT)?"*, Amazon Web Services. Available: https://docs.aws.amazon.com/iot/latest/developerguide/what-is-aws-iot.html

[2] **EMQX Guides:** *"MQTT in Python with Paho Client: Beginner's Guide"*, EMQX Distributed MQTT Broker Documentation, 2025. Available: https://www.emqx.com/en/blog/how-to-use-mqtt-in-python

[3] **Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y.:** "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation" Available: https://aclanthology.org/D14-1179/

[4] **Goodfellow, I., Bengio, Y., & Courville, A.:** *"Deep Learning"*, MIT Press, 2016. Available: https://www.deeplearningbook.org/

[5] **Caruana, R.:** *"Multitask Learning"*, Machine Learning, vol. 28, pp. 41–75, 1997. Available: https://arxiv.org/abs/1412.6980

[6] **Kingma, D. P., & Ba, J.:** *"Adam: A Method for Stochastic Optimization"*, 2014. Available: https://arxiv.org/abs/1412.6980

[7] **ONNX Documentation:** *"Introduction to ONNX"*, Open Neural Network Exchange Available: https://onnx.ai/onnx/intro