УНИВЕРЗИТЕТ У КРАГУЈЕВЦУ
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА У ЧАЧКУ

Душан Б. Марковић

ХАРДВЕРСКО-СОФТВЕРСКА ПОДРШКА ПРОЦЕСИРАЊУ НИЗОВА ПОДАТАКА У ОКВИРУ КОНЦЕПТА CLOUD–FOG РАЧУНАРСТВА

докторска дисертација

Чачак, 2025.

UNIVERSITY OF KRAGUJEVAC
FACULTY OF TECHNICAL SCIENCES ČAČAK

Dušan B. Marković

HARDWARE-SOFTWARE SUPPORT FOR DATA STREAM PROCESSING IN CLOUD–FOG COMPUTING CONCEPT

Doctoral Dissertation

Čačak, 2025

Идентификациона страница докторске дисертације

Аутор
Име и презиме: Душан Б. Марковић
Датум и место рођења: 3.12.1982. године, Чачак
Садашње запослење: Асистент, Агрономски факултет у Чачку Универзитета у Крагујевцу
Докторска дисертација
Наслов: ХАРДВЕРСКО-СОФТВЕРСКА ПОДРШКА ПРОЦЕСИРАЊУ НИЗОВА ПОДАТАКА У ОКВИРУ КОНЦЕПТА CLOUD – FOG РАЧУНАРСТВА
Број страница: 111
Број слика: 33
Број графикона: 15
Број библиографских података: 160
Установа и место где је рад израђен: Факултет техничких наука у Чачку Универзитета у Крагујевцу
Научна област (УДК): Техничко-технолошке науке, Електротехничко и рачунарско инжењерство (према Правилнику редни број 2.5), уже научне области: - Рачунарска техника (према Правилнику редни број 2.5.6) - Примењено рачунарство (према Правилнику редни број 2.5.11)
Ментор: др Борислав Ђорђевић, виши научни сарадник, Институт „Михајло Пупин“ Универзитета у Београду.
Број и датум одлуке Већа универзитета о прихватању теме докторске дисертације: IV-04-884/12 од 10.11.2021. године.

Резиме

Велика заступљеност и популарност IoT уређаја, на светском нивоу, довели су до генерисања огромне количине података, док се са друге стране појавило Cloud рачунарство које поседује снажну подршка за процесирање тих података. Прослеђивањем података од IoT уређаја до Cloud окружења формирана је јако погодна комбинација, при чему се подразумева да је обрада података сасвим централизована.

Мотивација и главни циљ у овој дисертацији јесте пружање додатног увида у другачији концепт који се ослања на децентрализовану обраду података. Као погодан приступ коришћено је Fog рачунарство како би се задаци процесирања података пренели са нивоа Cloud рачунарства на ниво ближи изворима података. Као пример обраде података коришћене су апликације које се заснивају на моделима дубоког учења, где је конкретно реализован модел који припада типу конволуцијских неуронских мрежа. Наведене апликације су служиле за класификацију слика у области паметне пољопривреде (Smart Agriculture) са циљем да се укаже на могућности ефикасније детекције стања посматраног објекта и примене одговарајућих мера.

Представљени су поступци за тренирање и проверу модела класификације, као и његово конвертовање у оптимизоване моделе који могу да се извршавају на уређајима са ограничним хардверским ресурсима. Главна варијанта је подразумевала припрему модела за извршавање на FPGA колу формирајући тако хардверске акцелераторе за класификацију слика.

Са примњеним моделом класификације који је пренет за извршавање у оквиру Fog рачунарстава добијене су вредности са прихватљивим кашњењем. Што значи да су остварене обраде података у реалном времену, док су редуковане вредности оптерећења сервера у погледу обима пренетих података и енергетске потрошње.

У случају потребе за проценом система Fog рачунарства већег обима моделована је Cloud–Fog структура коришћењем iFogSim пројекта за симулацију. При томе су узети у обзир приоритети апликација и вршене су процене о прихватљивом оптерећењу корисничких апликација које омогућава добијање резултата у реалном времену.

Кључне речи: Cloud рачунарство, Fog рачунарство, Интернет ствари (IoT), Класификација слика, Дубоко учење, Конволуцијске неуронске мреже, Паметна пољопривреда, FPGA, Акцелератори.

Abstract

The large presence and popularity of IoT devices, at the global level, have led to the generation of a huge amount of data, also Cloud computing has emerged with strong support for processing that data. By forwarding data from IoT devices to the Cloud environment, a very suitable combination has been formed, whereby it is assumed that data processing is completely centralized.

The motivation and main goal of this dissertation is to provide additional insight into a different concept that relies on decentralized data processing. Fog computing was used as a suitable approach to transfer data processing tasks from the Cloud computing level to a level closer to the data sources. As an example of data processing, applications based on deep learning models were used, where specifically the type of convolutional neural networks was used for model implementation. The above applications were used for image classification as a segment of Smart Agriculture to indicate the possibilities of more efficient detection of the state of the observed object and the application of appropriate measures.

Procedures for training and testing the classification model are presented, as well as its conversion into optimized models that can be executed on devices with limited hardware resources. The main variant involved preparing the model for execution on an FPGA circuit, thus forming hardware accelerators for image classification.

With the pre-trained classification model, executed within Fog computing, acceptable result delay has been obtained. This means that data processing was achieved in real time, while the load on the server was reduced such as the volume of transferred data and energy consumption.

In case of need for evaluation of a larger scale Fog computing system, a Cloud-Fog structure was modeled using the iFogSim simulation project. Application priorities were taken into account and estimates were made of the acceptable load of user applications that enable results in real time.

Keywords: Cloud computing, Fog computing, Internet of Things (IoT), Image classification, Deep learning, Convolutional neural networks, Smart agriculture, FPGA, Accelerators.

Садржај

Списак скраћеница

AdaGrad	Adaptive Gradient
Adam	Adaptive Moment Estimation
AI	Artificial Intelligence (Вештачка интелигенција)
ANN	Artificial Neural Network (Вештачке неуронске мреже)
ARM	Advanced RISC Machines (Микроконтролерски систем)
APIs	Application Programming Interfaces (Апликациони програмски интерфејси)
ASIC	Application-Specific Integrated Circuit
BCE	Binary Cross-Entropy
BGD	Batch Gradient Descent
CCE	Categorical Cross-Entropy
CEP	Complex Event Processing (Обрада сложених догађаја)
CNN	Convolutional Neural Network (Конволуцијска неуронска мрежа)
CPU	Central Processing Unit (Централна процесорска јединица)
CW-03	Corn Weed - 03 (Корови у кукурузу)
DL	Deep Learning (Дубоко учење)
Ed	End device (Крајњи уређај)
FCNN	Fully Connected Neural Network (Потпуно повезана неуронска мрежа)
Fds	Fog devices (Fog уређаји)
FNN	Feedforward Neural Network (Директна вештачка неуронска мрежа)
FPGA	Field-Programmable Gate Array
GD	Gradient Descent (Градијентно опадање)
GPU	Graphics Processing Unit (Графичка процесорска јединица)
HDL	Hardware Description Language (Језик за опис хардвера)
IaaS	Infrastructure as a Service (Инфраструктура као сервис)
IoT	Internet of Things (Интернет ствари)
IP-02	Insect Pests - 02 (Инсекти штеточине)
JSON	JavaScript Object Notation (Формат)
LUT	LookUp Table (Табела претраживања)
ML	Machine Learning (Машинско учење)
MAE	Mean Absolute Error (Средња апсолутна грешка)
Mini-BGD	Mini-Batch Gradient Descent
MIPS	Million Instructions Per Second (Милион инструкција по секунди код процесора)
MSE	Mean Squared Error (Средња квадратна грешка)
MQTT	Message Queuing Telemetry Transport (Протокол за комуникацију)
PaaS	Platform as a Service (Платформа као сервис)
PLB	Programmable Logic Block (Програмабилни логички блок)

PSO	Particle Swarm Optimization (Алгоритам оптимизације појем честица)
PYNQ	Python productivity on Zynq
ReLU	Rectified Linear Unit (Исправљена линеарна функција)
ResNet	Residual Network (Резидуална мрежа)
RMSProp	Root Mean Square Propagation
RTL	Register-Transfer Level
SoC	System On Chip (Систем на чипу)
SaaS	Software as a Service (Софтвер као сервис)
SGD	Stochastic Gradient Descent
SGDM	SGD са Моментумом
TL-01	Tomato Leaf - 01 (Листови парадајза)
XML	Extensible Markup Language
VHDL	VHSIC Hardware Description Language
ViTs	Vision Transformers

Списак слика

Списак графикона

Списак табела

1. УВОДНА РАЗМАТРАЊА

Cloud рачунарство (Cloud Computing – CC) са својом снажном хадрверско/софтверском подршком постаје врло брзо атрактивно и веома погодно решење у виду подршке за многе корисничке апликације и податаке. Корисници су били у прилици да чувају своје податке на ценрализованим складишним просторима, или да користе само одређене платформе или услуге сходно својим потребама. Један од бенефита Cloud рачунарства може се посматрати кроз комбинацију са концептом Интернета ствари (Internet of Things - IoT) [1], [2]. Посебно је то занимљива комбинације јер су често IoT уређаји са ограниченим рачунарским ресурсима, тако да је подршка за преузимање и процесирање података била веома значајна.

Количина података коју генеришу IoT уређаји може да варира у зависности од типа сензора који су прикључени, учесталости прикупљања података и наравно од броја уређаја који се користе. Један IoT уређај може да генерише од неколико килобајта до неколико гигабајта података дневно. Ови подаци могу представљати очитавања сензора, информације о локацији и друге врсте података које прикупљају IoT уређаји који су саставни део корисничких апликација.

Захваљујући великој заступљености технологија на којима се заснива IoT концепт, број примењених IoT уређаја је био у порасту претходних година, а очекује се да се тај тренд настави сходно динамици развоја нових технологија. Управо на ту тему у раду [3] представљен је број IoT уређаја на светском нивоу, за период од 2019. до 2030. године, на основу података са онлајн платформе [4] која пружа увид у адекватне прикупљене тржишне податке. Према наведеној процени очекивана је употреба 13,50 милијарди IoT уређаја у 2023. години, а предвиђен је раст броја IoT уређаја на 25,44 милијарде до 2030. године.

Подаци који се генеришу од IoT уређаја према дефинисаним учесталостима представљају непрекидне низове података које је потребно процесирати. Наведени термин за низове података који континуирано пристижу, био би превод од првобитног назива на енглеском језику Data Stream, иако се често среће још један термин – токови података. Тако да ће се термин низ података користити за ток података од IoT уређаја, док ће термин низ, у зависности од контекста, бити коришћен када се говори о сложеном типу података који се састоји од колекције других података или променљивих. На пример, дигитални запис слике се може представити као дводимензионални или тродимензионални низ, а где основни елемент низа представља вредност једног пиксела слике.

Карактеристично за такве низове података које генеришу IoT уређаји јесте да се они не морају прво сачувати како би се накнадно започела њихова анализа. Већ се такви подаци могу процесирати редом како пристижу од свог извора. Овакви захтеви су често јасно истакнути како би се остварило директно процесирање података и обезбедили одговори у реалном времену. Како би се избегло превелико кашњење код

апликација које захтевају извршење у реалном времену може се користити концепт Fog рачунарства (Fog Computing). На тај начин остварују се проширење карактеристичних функционалности Cloud рачунарства до крајева комуникационе мреже. Према томе одређени задаци процесирања података могу се пренети на слој ближи уређајима који генеришу податке.

Анализа података може да подразумева употребу различитих техника, а у последњем периоду јако је изражена свеобухватна примена вештачке интелигентције, односно метода машинског учења и вештачких неуронских мрежа. Овакве мреже карактерише један или више скривених слојева који би означавали дубину креираних модела. Отуда назив мреже дубоког учења (Deep Learning - DL) у случају постојања већег броја скривених слојева. Постоје многи примери реализације и примене DL модела у оквиру Cloud рачунарства, али њихова примена на нивоу крајева мреже може бити изазовна пошто се ту обично налазе уређаји са ограниченим хардверским ресурсима. Посебно изазовно може бити процесирање података који захевају одређену интезивнију рачунарску подршку, као што је анализа слика. У ту сврху могу се реализовати модели за препознавање односно класификацију слика засновни на конволуцијским неуронским мрежама (Convolutional Neural Network - CNN).

Намера је да се дефинисане апликације засноване на моделу класификације искористе за проверу понашања ситема приликом процесирања података на Cloud оркужењу али и на нивоу Fog рачунарства. А у случају потребе за системом већег обима вршило би се моделовање и симулација његове структуре у циљу процене перформанси за различита оптерећења корисничких апликација.

1.1. Мотивација и циљ дисертације

Истакнута комбинација IoT концепта и Cloud рачунарства постала је веома популарна у савремено доба које карактеришу све веће размене података са тенденцијом њиховог складиштења и обраде у Cloud окружењу. Овакав приступ указивао ја на постојање тренда да се процесирање података одвија на потпуно централизован начин. Мотивација дисертације била је да се размотре и другачији приступи односно да се укаже на услове под којима је могуће спроводити више дистрибуирани приступ у обради података.

Предмет истраживања у оквиру предложене докторске дисертације односи се на оптимизацију процесирања података са IoT уређаја и њихово измештање из Cloud окружења. Циљ је да се део обраде изворних података приближи сензорским уређајима и на тај начин смањи потреба за њиховом обрадом на нивоу Cloud рачунарства. У овом случају процесирање података би се пренело са нивоа сервера (Cloud рачунарство) на посредне уређаје до којих подаци директно стижу са IoT сензорских уређаја.

У варијанти када је укључено Fog рачунарство, где се процесирање извршава на платформама које су непосредно повезане са IoT уређајима, могу се очекивати резулати без значајног кашњења, односно одзиви у реалном времену. Обрадом података на нивоу Fog уређаја смањио би се проток информација према Cloud систему, а тиме и ниво обраде који би морао да се врши унутар Cloud система. Такође, истраживање треба да покаже могућности подизања нивоа обраде података на процесним јединицама са ограниченим ресурсима, које чине рачунарску основу система Fog рачунарства. Полазну претпоставку овог дела истраживања чини оцена да се процесна снага

рачунарских компонената у оквиру система Fog рачунарства може повећати кроз додавање хардверских акцелератора.

Повећање броја IoT уређаја захтева прилагођавање комуникационих и процесних перформанси Cloud окружења да би се обезбедили оптимални услови његовог рада. Ово је посебно значајно код обраде података у реалном времену.

Циљ истраживања је био реализација хардверско/софтверске подршке на плану добијања новог модела процесирања низова података укључивањем концепта Fog рачунарства. Основна идеја предложеног приступа подразумева да део апликација за обраду улазних података буде реализован на уређајима којима се успоставља оквир Fog рачунарства. Реално је очекивати да овим буде покривена обрада података који потичу са блиске локације. То значи да би овај модел био базиран на дистрибуираном процесирању података, са циљем да целокупни систем прикупљања и обраде података искаже значајне показатеље перформанси у односу на моделе базиране искључиво на Cloud рачунарству. Критеријуми који би се користили приликом разматрања модела за процесирање података би подразумевали величину заузећа пропусног опсега при прихватању података, кашњење при обради података, као и потрошњу електричне енергије уређаја и главног сервера. Имајући у виду да би процесирање података на Fog уређајима било подржано хардверским акцелераторима додатно би се анализирале перформансе система за тај случај.

Примена IoT апликација често може да подразумева захтеве за пружање одговора у реалном времену. Остварење оваквог захева може бити упитно ако се процесирање података врши у Cloud окружењу због могућих кашњења у преносу података. Такође, на испоруку резултата може утицати повећање броја IoT уређаја у оквиру истог система што доводи до повећања обима долазних података, односно оптерећења платформе где се врши процесирање података.

Применом динамичког приступа у извршавању процеса над подацима, део обраде се може и даље извршавати на централном рачунарском систему, док се део обраде може пренети на посредне Fog уређаје. У току формирања захтева за појединим процесима они се динамички распоређују на уређаје где ће се извршавати. Идеја је да се оствари пренос процеса обраде података на уређаје сходно њиховим ресурсима и да се испита који утицај имају на укупни пренос података.

На тај начин, уместо централизоване обраде података на страни сервера могуће је извршити прилагођену обраду података на бази посредних уређаја уз коришћење хардверских акцелератора. Увођењем акцелератора на страни Fog уређаја процесирање података би било брже и потрошња енергије би била мања што би довело до могућности прихватања већег обима података и са мање кашњења у њиховој обради.

У циљу реализације процесирања података на Fog уређајима узимају се у обзир различити модели обраде података, са и без акцелератора. Идеја је била да се у тим варијантама врши процесирање података на посредним уређајима и да се при томе прати потрошња енергије. Претпоставља се да је могуће остварити оптималну раподелу процеса обраде над подацима тако да се на сервер (Cloud окружење) прослеђује мања количина података, што би довело до растерећења централног рачунарског система и донело знатне уштеде по питању њихових ресурса.

Модели који се могу обезбедити у Cloud окружењу, могу се прилагодити и пренети на ниво Fog рачунарства. Пренос ових модела остварује се тако што се они конвертују у верзије које би могле да се извршавају на уређајима са ограниченим ресурсима. Према томе тренирани прилагођени модели за класификацију могу се

превести на верзије модела за покретање на одговарујућим платформама. То могу бити платформе засноване на општим процесорима или хардверска подршка у виду програмабилне логике. Као хардверска подршка користе се FPGA (Field-Programmable Gate Array) кола у којима се могу реализовати хардверски акцелератори.

FPGA кола представљају платформу која омогућава пројектантима да креирају високо оптимизоване и ефикасне системе за широк спектар апликација. Њихова програмабилна природа, могућност паралелне обраде и реконфигурација чине их вредним алатом за пројектанте хардвера који желе да имплементирају сложене системе високих перформанси. Пребацивањем рачунарски интензивних задатака на FPGA коло, могу се постићи веће брзине обраде, мање кашњење и боља енергетску ефикасност у корисничким апликацијама. Ова комбинација Fog рачунарства, машинског учења на крајевима мреже и FPGA хардверског убрзања може помоћи у оптимизацији перформанси и омогућити доношење одлука у реалном времену у различитим случајевима примене.

Пошто је за процесирање података одабран пример препознавања, односно класификацију слика, потребно је њихове моделе класификација превести у варијанте које се могу покретати на FPGA акцелераторима. Препознавање слика у оквиру Fog рачунарства помоћу FPGA кола, подразумева коришћење FPGA технологије за убрзавање задатака препознавања слике на крајевима мреже. Наведени приступ омогућава обраду података, односно процесирање слика, у реалном времену са малим кашњењем и смањеном потрошњом енергије, што га чини идеалним за апликације где је кључно брзо доношење одлука.

Целокупни предстаљени модел би могао наћи примену код разних апликација у оквиру Cloud–Fog рачунарства, али је за показни пример одабрана паметна пољопривреда (Smart agriculture). Управо због важног утицаја који примена DL модела може имати на квалитет пољопривредне производње кроз максимизирање производње, умањења утицаја средине или примену оптимизационих алгоритама [5]. Паметна пољопривреда (Smart agriculture) представља коришћење предности информационих и комуникационих технологија као што су IoT и Cloud рачунарство за поспешивање развоја нових процеса управљања у пољопривредној производњи. Поменуте технологије и у области паметне пољопривреде генеришу велике количине података који би могли да се чувају и користе за анализу и доношење одлука. Обично на фармама постојећи ресурси, које чине уређаји на крајевима мреже (Edge уређаји), нису тако хардверски садржајни као што би то потенцијално могли бити системи у урбаним срединама. Тако да је важно под тим условима обезбедити прилагођене апликације које ће моћи да одговоре захтевима и дају поуздане одговоре како би се омогућило одржавање квалитетне производње.

Паметна пољопривреда, као што је приказано у [6] описује нова достигнућа у системима заснованим на IoT уређајима за добијање неопходних података. Њихова сврха је била да се оптимизују процеси као што су сејање, жетва, суочавање са штеточинама, откривање корова или друге сличне примене у реалном времену коришћењем IoT уређаја, бежичне комуникације, вештачке интелигенције или машинског учења. Преглед нових технологија и одговарајућих истраживачких напора у оквиру паметне пољопривреде представљен је у радовима [7], [8] и [9].

Апликације које се могу сврстати у оквир паметне пољопривреде и које су намењене за Cloud и Fog рачунарство могу се сагледати кроз следеће примере: предикција појаве одређене врсте инсеката [10], детекција корова [11], идентификација котрљајних лежајева код пољопривредне механизације [12], детекција губитака семена

уљане репице [13], праћење активности у пчелињацима [14], као и праћење штетних инсеката у пољопривреди [15].

У општем случају за потребе процене систем Fog рачунарства потребно је узети у обзир потенцијалне конфигурације код нових система ширег обима. Да би се одговорило захтевима за проценом нових варијанти може се користити концепт моделовања и симулације Cloud–Fog структура. За спровођење процеса провере различитих варијанти система може се користити iFogSim пројекат, који је представљао средство многих истраживача за симулацију система Fog рачунарства. Помоћу наведеног iFogSim пројекта, као адекватног средстава, могу се разматрати перформансе задатих конфигурација, као што је посматрано време кашњења код апликација за различита оптерећења. Наведени iFogSim пројекат би послужио као основа за реализацију система помоћу кога би се вршиле симулације дефинисаних Cloud–Fog структура, процењивало њихово оптерећење, као и перформансе за различите корисничке апликације.

1.2. Преглед стања у области истраживања

Приказана истраживања у наставку, односе се на моделе акцелератора који се могу изводити на уређајима у близини извора података, а које карактеришу ограничени ресурси. Такође у наставку представљена су истраживања која указују на широке могућности примене класификације слика у области пољопривреде, као и на значај њеног спровођења у реалном времену.

CNN и акцелератори на крајевима мреже

Аутори [16] нуде свеобухватан преглед најновијих достигнућа код CNN мрежа, истичући побољшања у дефинисању слојева, активационим функцијама, функцијама губитка, регуларизацији, техникама оптимизације и рачунарској ефикасности. Такође разматрају широк спектар апликација у којима су CNN мреже показале значајан успех, у распону од рачунарске визије до говора и обраде природног језика.

CNN мреже су постигли значајан успех у рачунарској визији, али њихова сложеност и захтеви за рачунарским ресурсима могу бити ограничавајући фактор код практичне примене, посебно са увећањем величина података. Аутори [17] разматрају различите методе компресије, фокусирајући се на уклањање редундантних параметара и квантизацију, како би побољшали применљивост CNN мрежа у окружењима са ограниченим ресурсима. При томе предлажу и нову платформу за решавање изазова компресије CNN мрежа великих размера.

Модели дубоког учења (Deep Learning - DL) остварили су значајан напредак у различитим доменима, али због величине тренираних модела, проблем је њихова примена на уређајима са ограниченим ресурсима, као што су мобилни телефони и IoT уређаји. Да би се омогућиле апликације у реалном времену на таквим уређајима, кључно је компримовати и убрзати ове моделе без угрожавања тачности; ово покреће истраживање различитих техника за компресију и убрзање модела [18].

Модели дубоког учења постали су саставни део различитих услуга и апликација, при чему се њихово тренирање и коришћење највише ослањају на високе перформансе Cloud рачунарства. У случају захтева за испоруком резултата, код којих не постоји проблем са кашњењем уз спречавање преоптерећења са подацима из IoT уређаја,

потребан је алтернативни приступ за коришћење DL модела. Edge рачунарство се појављује као могуће решење, али његова ограничења у снази и ресурсима захтевају нове, енергетски ефикасне моделе дубоког учења [19].

Висока енергетска ефикасност и могућност реконфигурисања су обећавајуће карактеристике које чине FPGA кола као кључну платформу за CNN хардверске акцелераторе. Аутори [20] пружају свеобухватан преглед техника за имплементацију и оптимизацију CNN алгоритама на FPGA колима, који служе као вредан ресурс за истраживаче у области вештачке интелигенције, хардверске архитектуре и пројектовања система.

Широко распрострањена употреба CNN мрежа у различитим задацима, посебно на Edge уређајима са ограниченим рачунарским ресурсима, истакнута је у [21]. Такође, аутори предлажу прилагођене рачунарске архитектуре засноване на FPGA колима као решење за побољшање перформанси CNN закључивања, уз задржавање тачности.

Аутори [22] дају преглед постојећих радних варијанти од CNN модела до FPGA кола, нудећи увид у њихове кључне карактеристике као што су подржане апликације, избор архитектуре и перформансе. Док аутори у [23] истражују технике оптимизације за CNN моделе на алгоритамском и хардверском нивоу, што је од суштинског значаја за ефикасну имплементацију на уређајима са ограниченим ресурсима, посебно FPGA колима. Њихов циљ је био да одговоре на изазов постављања већих и дубљих CNN модела на ограничене хардверске ресурсе, испитивањем различитих стратегија оптимизације.

Преглед изазова примене DL модела на Edge уређајима због њихових ограничених ресурса, као и побољшања процеса за доношења одлука у реалном времену, дат је у [24]. Да би се одговорило на ове изазове, развијене су технике оптимизације и на хардверском и на софтверском нивоу, фокусирајући се на нову DL архитектуру као и пројекат ефикасног акцелератора.

Препознавање слика употребом мање сложених CNN модела, омогућавају покретање алгоритама високих перформанси на уређајима са ограниченим ресурсима [25]. Аутори [26] представљају свеобухватан преглед технологија за оптимизацију неуронске мреже засноване на FPGA колима, истичући њен значај и предности у убрзавању задатака код DL модела.

Изазови код крајњих уређаја који се односе на бројност параметара и упитну рачунарску подршку представљали су повод да аутори развију лагану архитектуру вештачке неуронске мреже под називом SparkNet [27]. У тој варијанти успели су да значајно редукују број параметара тежина и потребе за рачунарским ресурисма, при чему SparkNet има могућност да вишеструко компримује CNN мрежу. Њена архитектура је коципирана тако да је повољна за мапирање високо паралелног конволуцијског рачунарања на FPGA кола. SparkNet мрежни модел и одговарајућа архитектура акцелератора су развијени за FPGA кола, који након имплементације показују да по питању брзине и енергетске ефикасности надмашују претходне методе.

Аутори [28] нуде детаљан преглед најновијих достигнућа у алгоритмима рачунарске визије и њихове хардверске имплементације. Они се фокусирају на задатке као што су класификација слика, детекција објеката и сегментација слике омогућене техникама дубоког учења. Они разматрају методе за оптимизацију и имплементацију ових алгоритама на различитим хардверским акцелераторима као што су GPU процесори (Graphic Processing Unit), FPGA кола и нове процесорске архитектуре, са циљем да омогуће рад у реалном времену и енергетски ефикасне операције.

CNN и Vision Transformers

Vision Transformers (ViTs) као модерне методе показали су одличне перформансе у задацима рачунарске визије као што је класификација слика. Vision Transformers представљају тип модела дубоког учења који примењује Transformer архитектуру претходно пројектовану и коришћену за обраду природног језика (Natural language processing - NLP). ViTs се примењују у рачунарској визији где се слика третира као низ сегмената, на исти начин као што се речи третирају у реченици када се ради о задацима код NLP [29].

ViTs системи су нашли важне примене у многим областима, тако да се могу користити код дијагностиковања у медицини [30], дронова у случају лета на малим висинама [31], или код сегментације тродимензионих слика.

Такође, постоје примене ViTs у пољопривреди, као што су технике за рано откривање болести биљака [32], идентификација здравих и болесних биљака [33] или откривање штетних инсеката [34].

Избор између модела CNN и ViTs може зависити од посебних захтева у конкретном случају с обзиром на својства као што су расположиви ресурси и величина података, као и очекивани компромис између тачности, сложености модела и перформанси.

CNN и даље може да има ефикасне примене за многе задатке рачунарске визије због њиховог успеха у различитим областима, посебно када раде са мањим скуповима података, ограниченим рачунарским ресурсима или потребом за поузданим решењима [35].

Класификација слика и паметна пољопривреда

Аутори [36] испитују типове великих података (Big Data) који се генеришу у паметној пољопривреди, дискутују њихове примене, као што су предвиђање приноса и управљање фармама. При томе истражују и могућности примене машинског учења које се користе у анализи података. Аутори [37] указују на значајан утицај IoT концепта и других паметних технологија на пољопривреду, укључујући Cloud рачунарство, машинско учење и вештачку интелигенцију.

Аутори у [38] представљају детаљан преглед кључних области у паметној пољопривреди, фокусирајући се на технологије као што су Big Data, машинско учење, IoT, роботика и друге. На исту групу технологија су усмерена истраживања у раду [39], при чему су разматране постојеће апликације, као што су беспилотне летелице за надгледање усева и оптимизацију производње.

Утицај дигитализације и аутоматизације на традиционалну пољопривредну праксу, коришћењем рачунарске визије и вештачке интелигенције (AI), испитују аутори у раду [40]. Истакнуто је како ове технологије могу побољшати продуктивност и економски раст и одговорити на глобалне изазове као што су сигурност хране и одрживост.

Аутоматизација у анализи слике помоћу рачунарске визије и дубоког учења побољшава прецизност у мапирању поља и приноса. Аутори у [41] приказују различите апликације за паметну пољопривреду, као што су аутоматизована жетва и откривање корова, док разматрају ограничења и будућа истраживања у техникама дубоког учења.

Биљне болести значајно угрожавају производњу усева и сигурност хране, због чега је неопходно њихово прецизно откривање. Док традиционалне методе попут визуелног посматрања и лабораторијских тестова имају ограничења, методе дубоког учења, посебно CNN мреже, су се појавиле као најсавременија решења у класификацији биљних болести. Преглед [42] наглашава најновији напредак у примени CNN мрежа за откривање биљних болести.

IoT концепт унапређује паметну пољопривреду омогућавајући прикупљање података у реалном времену са локација од интереса за пољопривредну производњу. На овај начин прикупљене слике и сензорски подаци, уз помоћ дубоких неуронских мрежа омогућавају аутоматизовано препознавање болести биљака. Представљени систем [43] постиже 96% тачности у откривању и класификацији нпр. болести листа биљака користећи CNN мреже.

Рано откривање и лечење болести парадајза може значајно повећати принос, ефикасност и квалитет, што доводи до приступачнијих цена за потрошаче и спречавања губитака на страни произвођача. Рана идентификација и класификација ових болести су од кључне важности за побољшање приноса усева, а употреба CNN модела се показала као обећавајућа у овој области. Управо у експерименту представљеном у [44] са скупом података од 3000 слика, развијен је CNN модел за класификацију болести парадајза са 98,49% тачности.

Аутори у [45] представљају прилагођени CNN модел дизајниран да класификује болести парадајза, нудећи побољшану тачност и смањене рачунске трошкове у поређењу са постојећим моделима као што су AlexNet и VGG-16. Ефикасност прилагођеног CNN модела, посматрано кроз перформансе у класификацији 10 категорија болести, чини га погодним за коришћење и на паметним технологијама. Предложена техника у [46] користи CNN модел, посебно модел Visual Geometry Group (VGG), да класификује оболеле и здраве листове код парадајза са високом прецизношћу од 95,71%.

Аутори [47] предлажу компактни CNN модел са само шест слојева за идентификацију болести парадајза, који је трениран на скупу података које представљају слике листова парадајза. Предложена мрежа надмашује добро познате унапред обучене дубоке мреже, показујући да мање сложена варијанта и даље може дати одличне резултате у идентификацији болести парадајза. Док аутори у [48] представљају алгоритам за обраду слике који користи вештачке неуронске мреже за откривање и праћење четири болести усева парадајза. Алгоритам категорише слике на основу боје, текстуре и морфологије, при чему је на основу морфологије постигнута највећа тачност од 93%.

Инсекти штеточине значајно утичу на принос и квалитет усева на глобалном нивоу, чинећи брзо и прецизно праћење неопходним за ефикасну контролу штеточина. Традиционалне методе идентификације инсеката ослањају се на квалификоване стручњаке и провере замки за инсекте, што је често временски захтевно и мање ефикасно у поређењу са применама савремених техника машинског учења.

Аутори у раду [49] испитују употребу технологије дубоког учења (DL) у паметном праћењу штеточина, фокусирајући се на откривање штетних инсеката помоћу слика са терена. Истраживање [50] представља аутоматизовани метод који користи CNN класификатор у циљу унапређења стратегије за контролу инсеката штеточина у пластеницима. Док је још један алгоритам за детекцију инсеката штеточина заснован на CNN моделима приказан у раду [51], постижући високу тачност за различите скупове података.

Аутори [52] предлажу коришћење резидуалног CNN модела са трансферним учењем за тачну идентификацију штеточина, користећи технике за увећање скупа да би побољшали робусност модела. Студија показује да трансферно учење значајно побољшава тачност класификације и скраћује време тренирања модела. Рад истраживача [53] представља дубоки CNN модел за детекцију 102 уобичајене врсте штеточина. Коначни модел интегрисан је у мобилну апликацију и класификација инсеката штеточина може се вршити путем снимања или одабира претходних слика, радећи у онлајн и офлајн режиму.

Препознавање корова и прецизна примена хербицида су од суштинског значаја, што захтева поуздану идентификацију усева и корова. Аутори у раду [54] говоре о традиционалној обради слике и методама заснованим на дубоком учењу за откривање корова. Напредак у машинској визији и обради слике нуди обећавајућа решења за откривање корова у реалном времену на терену. Рад [55] описује процедуре за откривање корова, укључујући претходну обраду, сегментацију, екстракцију карактеристика и класификацију, фокусирајући се на технике као што су индекси боја и машинско учење.

Аутори у [56] разматрају технике за откривање и класификацију корова засноване на DL моделу, фокусирајући се на прикупљање података, припрему скупова података, методе детекције и метрику евалуације. Конкретно у раду [57] представљен је систем за детекцију корова заснован на визији користећи моделе дубоког учења за ефикасну идентификацију корова у плантажама соје. Међу пет тестираних модела, укључујући MobileNetV2, ResNet50 и три прилагођена CNN модела, прилагођени петослојни CNN модел постигао је високу тачност детекције од 97,7%. При томе је тај модел са најмањим кашњењем и употребом меморије када се користи на Raspberri PI уређају. Још једно истраживање из ове тематике може се сагледати у [58] где аутори уводе приступ граф конволуцијске мреже (Graph Convolutional Network - GCN) за препознавање корова и усева. Ова метода је постигла високу тачност и испунила захтеве за добијање одговора у реалном времену.

Предложени пример у оквиру дисертације представља свеобухватно решење које садржи избор алгоритама, тренирање модела и имплементацију на крајњим уређајима. Истовремено, могуће је директно имплементирати нове моделе класификације на крајњим уређајима који би били добијени оптимизацијом постојећих већ тренираних модела. При томе се користи одговарајући алат за превођење модела, као што је Tensil, без потребе да се знају сви детаљи имплементације FPGA кола. Такође, у овако дефинисаном тестном окружењу могуће је утврдити компромис између типа и величине модела с једне стране и прихватљиве тачности приликом класификације.

Истраживања везана за поставку модула апликације у оквиру Cloud–Fog рачунарства

Постављање апликативних модула у оквиру Cloud и Fog рачунарства био је истраживачки циљ многих студија. Доступна литература предлаже решења заснована на алгоритмима оптимизације, при чему су често коришћени алгоритми који узимају у обзир ресурсе уређаја или ограничења временских рокова.

Аутори су у раду [59] представили алгоритам за мапирање модула помоћу којег је спроведена оптимална поставка апликативних модула на мрежне ресурсе Cloud–Fog инфраструктуре. Чворови ресурса су сортирани према њиховом капацитету, а модули апликације су сортирани према њиховим захтевима у растућем редоследу. Затим се користи одговарајући метод за проналажење места за сваки модул апликације. Овај

процес се имплементира тако што се листа чворова подели на пола, а затим се ресурси чворова из листе проверавају у складу са захтевима модула. Процес се наставља све док се не пронађе чвор са адекватним ресурсима. У овом случају, за проналазак прихватљивог чвора постојала је логаритамска временска сложеност. Предложена поставка модула довела је до смањења употребе мреже и кашњења апликације у испоруци резултата. У раду [60] предложен је сценарио са IoT апликацијом и подршком заснованом на Cloud–Fog рачунарству за моделовање кашњења услуге. Уведена је политика за Fog чворове да би се минимизирало кашњење услуге у апликацији која садржи IoT чворове. Дакле, растерећење рачунања је узето у обзир и за типове захтева који се примењују са различитим временима обраде, што указује на разлике између лаке и тешке обраде података.

У раду [61] аутори су предложили алгоритам који узима у обзир кашњење за оптималну поставку апликативних модула. Постигли су мање кашњења смањењем удаљености између уређаја који садрже модуле. Постављање модула почиње на крајњем уређају близу корисника. Ако постављање није било могуће извршити на тренутном уређају, онда се суседни уређај бира са листе која садржи уређаје сортиране растућим редоследом на основу временског кашњења. Овај процес реализације алгоритма се наставља све док сви модули не буду постављени на ниво Fog уређаја.

Још један алгоритам за поставку апликативних модула заснован на истом приступу који узима у обзир кашњење представљен је у раду [62]. Одлука о поставци за сваки модул се ажурира узимајући у обзир локацију модула који је последњи постављен, а све то са циљем да се минимизира комуникација између њихових чворова на који су постављени. Суштина алгоритма је била да задржи модуле најосетљивије на кашњење што ближе један другом. Дакле, циљ овог алгоритма је био да наведне модуле постави близу, што може да укључује промену локације модула са једног чвора на други, да би се оптимизирало коришћење ресурса чвора и смањила употреба пропусног опсега.

Аутори у раду [63] су предложили модел управљања ресурсима за поставку апликативних модула за више задатака у Fog рачунарству. Циљ је био да се смањи комуникација између уређаја, који би били потенцијални кандидати за прихватање модула апликације тако што би се извршило груписање чворова који су близу један другом. Мапирање ресурса се врши методом заснованом на Greedy методи коју карактерише висока скалабилност и перформансе близу оптималних за мапирање ресурса. Сличан приступ који укључује груписање блиских чворова представљен је у раду [64]. Предложен је метод који се састоји од две фазе, прва садржи постављање критичних модула, а друга подразумева постављање преосталих некритичних модула. Идеја је била да се међузависни модули са високом брзином размене података поставе на исти чвор, ако је могуће, како би се смањила потреба за преносом података, а самим тим и коришћење пропусног опсега и кашњење.

У раду [65] представљен је планер за мапирање модула апликације који узима у обзир ресурсе. Планер има могућност да максимално искористи расположиве ресурсе уз минималну употребу пропусног опсега. Приказани алгоритам се састоји из два дела. У првом делу модули апликације су категорисани према какви су захтеви за рачунарске ресурсе и пропусни опсег, а Fog уређаји су сортирани према расположивим ресурсима у растућем редоследу. У другом методу, категорисани модули се проверавају на Fog уређајима; ако има довољно слободних ресурса, модул се поставља на Fog уређај. У супротном се бира следећи уређај, а претходни се уклања са листе доступних уређаја.

У раду [66] временски рок је сматран једним од најзначајнијих изазова приликом извршавања апликација у Fog рачунарству. У овом случају, да би се испунили захтеви

корисника везани за рокове, предлажу се два алгоритма. Први алгоритам врши рангирање ресурса, а други обезбеђује ресурсе користећи хијерархијски и хибридни режим. Они се пореде са другим алгоритмима на основу стања ресурса и вредности кашњења. У првом алгоритму, расположива процесорска снага, пропусни опсег и време одзива уређаја узимају се у обзир за рангирање ресурса. У другом алгоритму, који се односи на обезбеђивање ресурса, разматрају се захтеви апликације и динамичке промене корисничких рокова.

Циљ аутора у раду [67] био је да прикажу модел за мапирање ресурса у Fog рачунарству у односу на перформансе Fog уређаја. Предложене су две методе за доношење одлука за процену карактеристичних перформанси Fog уређаја. У првом методу се израчунавају тежине приоритета, а затим се користи други алгоритам за рангирање Fog уређаја. Након тога, Fog уређаји се додељују постојећем задатку према свом рангу. Резултати су показали да је представљени приступ дао боље перформансе од других алгоритама, тако што укључује перформансе Fog уређаја и метрику трошкова приликом доношења одлуке о распоређивању ресурса.

Аутори у раду [68] су истраживали како се методе машинског учења за анализу токова података (Data Streams) могу применити у Cloud–Fog архитектури. Они су се фокусирали на боље испуњавање критичних и временски критичних захтева који се могу појавити у сајбер-физичким системима.

Постојеће технике представљају остварена достигнућа у оптимизацији дистрибуције постојећих апликативних модула на расположивим ресурсима и према дефинисаним захтевима. Техника која је предложена у овом раду, осим дистрибуције модула, има за циљ да израчуна услове оптерећења под којима ће расположиви ресурси одговорити на дато очекивано кашњење. Ова техника није само узела у обзир апликације са нижим приоритетом које се могу преместити у Cloud, већ и случај када главни модули апликације морају да се изврше на нивоу Fog уређаја. Главни фокус је био на процени колико учесника са њиховим уређајима може бити прихваћено, за апликације са различитим приоритетима, на доступним ресурсима без угрожавања очекиваног кашњења апликације.

2. ТЕОРИЈСКЕ ОСНОВЕ

У поглављу Теоријске основе представљени су теоретски прикази свих коришћених елемената и концепата у реализацији докторске дисертације. Описани су низови података, IoT концепт, затим Cloud и Fog рачунарство, процесирање слика, вештачке неуронске мреже и FPGA.

2.1. Токови (низови) података (Data Streams)

Низ података (Data Streams) се односи на податке који стижу непрекидно током времена, често у реалном времену. За разлику од традиционалних скупова података, овакви низови података континуирано се генеришу, потенцијално су бесконачни и обично захтевају тренутну обраду.

Оваква поставка се може схватити као континуирани ток, где подаци стижу у реалном времену и захтевајући обраду у ходу. Често их карактерише и временска осетљивост јер подаци могу изгубити своју релевантност како време одмиче. На овај начин подаци у низовима пристижу великим брзинама и редослед у неким случајевима такође може бити важан тако да су потребни ефикасни системи за обраду.

Генерисање низова података остварује се у следећим применама:

- IoT уређаји – код којих прикључени сензори генеришу континуиране податке (нпр. температура, влажност ваздуха, GPS подаци и друго).
- Финансијски системи – где се прикупљају подаци за потребе анализе цена акција у реалном времену или у другом случају за откривање превара.
- Друштвени медији – код којих се врши анализирање активности корисника, трендова или расположења у реалном времену.
- Сигурност мреже – код система за откривање упада.
- Видео стримови – приликом обрада у реалном времену за надзор или видео аналитику.

Постоје одређени изазови који се тичу континуираних низова података, на које је потребно одговорити, како би се успешно реализовале одговарајуће апликације. При томе јавља се питање ограниченог складишног простора, пошто је често немогуће сачувати све могуће пристигле податке. Више је реално задржавање делимичних података или њихова компресија. Пошто подаци пристижу континуирано током временског периода могуће их је обрадити само једном или у малим серијама. Битно је ограничење које се односи на процесирање података у реалном времену, што значи да постоје строги захтеви за добијање резултата са малим кашњењем. Системи који су предодређени за рад са низовима података треба да одговоре на захтеве високе скалабилности, што значи да треба да имају могућност да прихвате велике количине података [69].

Сложена обрада догађаја (Complex Event Processing - CEP) и опште процесирање низова података су две технике које се користе за анализу података у реалном времену, али служе различитим сврхама и имају различите карактеристике.

Обрада сложених догађаја (CEP) се фокусира на идентификацију и анализу образаца и односа унутар низа догађаја у реалном времену, како би се открили сложени догађаји или ситуације. CEP системи обично користе правила, упите и обрасце за корелацију и анализу догађаја како се дешавају, омогућавајући организацијама да донесу тренутне одлуке или предузму акције на основу откривених образаца [70].

С друге стране, процесирање низова података укључује обраду и анализу континуираних токова података у реалном времену како би се извели нови закључци, открили аномалије и генерисале информације које су од значаја за кориснике. Системи за процесирање низова података су пројектовани да раде са великим количинама података и обављају прорачуне у ходу, док подаци пролазе кроз систем, пружајући могућност да се стекну увиди у реалном времену и да се адекватно реагује на промењљиве услове [71].

Док се сложена обрада догађаја фокусира на откривање сложених образаца и односа унутар низова догађаја, процесирање низова података је у ширем смислу фокусирано на обраду и анализу континуираних токова података, како би се извукли закључци и покренуло доношење одлука у реалном времену. Обе технологије играју важну улогу у анализи података и могу се користити заједно за решавање различитих аспеката захтева за обраду података у реалном времену .

Примери платформи које се могу користити за преузимање података са IoT уређаја и процесирање низова података:

- Apache Kafka – представља дистрибуирану стриминг платформу за управљање долазним низовима података у реалном времену и њихово процесирање. Карактерише је висока пропусност и отпорност на грешке приликом преноса низова података [72].
- Apache Flink – је платформа пројектована за процесирање континуираних низова и формираних серија података. Омогућава процесирање временских догађаја и израчунавања стања [73].
- Apache Spark Streaming – платформа изграђена је на Apache Spark платформи за процесирање података у микро серијама. Карактерише је добра интеграција са већ постојећим екосистемима великих података [74].
- AWS IoT Core – може се дефинисати као управљива платформа у Cloud окружењу за повезивање IoT уређаја и процесирање низова података. Омогућава прикупљање података са IoT уређаја и њихово слање у апликације на Cloud платформи за даљу обраду, складиштење и аналитику [75].
- Google Cloud Dataflow – представља сервис за процесирање низова и готових серија података. Може се користити за аналитику у реалном времену низова података са IoT уређаја за апликације као што су праћење у реалном времену или откривање аномалија [76].
- Microsoft Azure IoT чвориште – Cloud сервис за управљање IoT уређајима и процесирање низовима података. Може се употребити за процесирање података са IoT уређаја, а пружа могућност и Edge рачунарства и аналитике [77].

Низови података и велики подаци су међусобно повезана поља у науци о подацима. Оба поља се баве великим количинама података, али се њихова природа,

захтеви за обраду и апликације разликују. Велики подаци се фокусирају на складиштење и групну обраду великих скупова података. Низови података дају приоритет обради у реалном времену, често одбацујући податке након извлачења информација.

2.2. Интернет ствари (IoT)

Интернет ствари (Internet of Things - IoT) представља концепт који се односи на мрежу физичких објеката, односно уређаја, који су рачунарски подржани и омогућено им је повезивање на рачунарску мрежу и пренос података. На тај начин умрежени уређаји уз помоћ прикључених сензора могу да прикупљају податке и преносе их до других IoT уређаја или до удаљених сервера преко Интернета. Све већа примена IoT концепта ствара могућност за нове примене и унапређења у индустрији и многим другим областима људског животног и радног простора [78].

Значај IoT концепта може се сагледати кроз IoT апликације, које као саставни део управо имају повезане уређаје за прикупљање и размену података у циљу побољшања продуктивности и ефикасности у различитим областима примене. Утицај IoT апликација је постао велики захваљујући широкој примени, што је допринело трансформацији у индустрији и свакодневном животу кроз предности које нуди спој повезаних уређаја и аналитике података.

Примери IoT апликација обухватају следеће примене: Паметне куће (Smart homes) [79], [80], Паметни градови [81], [82], Индустријски IoT [83], [84], Здравствена заштита [85], Паметне зграде [86], Паметна пољопривреда [87],[88], Управљање енергијом [89], Транспорт и логистика [90], Мониторинг животне средине [91] и друге сличне примене.

IoT концепт подразумева мрежу међусобно повезаних уређаја који прикупљају, преносе и обрађују податке у реалном времену. IoT уређаји континуирано генеришу токове података, често захтевајући специјализоване системе за обраду и анализу. Ови токови података у реалном времену са IoT уређаја чине језгро многих савремених апликација. Ток података у IoT контексту односи се на континуирани ток података које генеришу IoT уређаји (нпр. сензори, паметни мерачи, преносиви уређаји). Ови уређаји генеришу податке у реалном времену или скоро у реалном времену, често великом брзином, а којима ће можда бити потребна правовремена обрада или анализа. IoT уређаји долазе од различитих произвођача и могу генерисати различите типове података (нпр. структурирани, неструктурирани, подаци временских серија). Често имају ограничене хардверске ресурсе односно рачунарску снагу, меморију и енергетску потрошњу, тако да обрада података директно на самом уређају може бити доведена у питање.

IoT уређаји и анализа података иду једно са другим као погодна комбинација, пошто се огромна количина података коју генеришу IoT уређаји могу искористити за добијање нових информација, а могу представљати и основ за доношење нових одлука. Уз употребу различитих аналитичких техника, као што су примене машинског учења и аналитика у реалном времену, могу да се открију обрасци понашања и трендови у IoT подацима у циљу постизања сврсисходне оптимизације и унапређења ефикасности код корисничких апликација [92].

Неке уобичајене технике које се користе за анализу IoT тока података укључују:

- Откривање аномалија - Надгледањем низова података за откривање необичних образаца (нпр. превелики скок температуре или изненадно кретање у сигурносним системима).
- Анализа временских серија - Анализирање података из сензора током времена, што је уобичајено у апликацијама као што су предиктивно одржавање или праћење потрошње енергије.
- Откривање догађаја - Идентификовање специфичних догађаја или граничних вредности у низовима података, као што је на пример детекција оних вредности које указују на квар уређаја због прекомерне употребе.
- Edge AI - Примена модела машинског учења на нивоу Edge уређаја ради предвиђања и класификација у реалном времену на низовима података из IoT уређаја.

Приликом процесирања низова података, перформансе су кључне за ефикасно добијање резултата у реалном времену. Фактори који могу утицати на перформансе укључују брзину обраде, скалабилност, толеранцију грешака и коришћење ресурса система приликом процесирања низова података. Да би се извршила оптимизација перформанси, важно је узети у обзир структуру система, алгоритме који се користе за обраду и доступне хардверске ресурсе.

2.3. Cloud рачунарство

Cloud рачунарство може се превести као Рачунарство у облаку је концепт који пружа корисницима одређене предности у погледу ефикаснијег приступа и управљања рачунарским ресурсима. Карактерише га велика флексибилност у приступу ресурсима, корисничким апликацијама и подацима са било које локације где је могућ приступ Интернету. Предност је и скалабилност система, пошто је могућ одабир ресурса већег или мањег обима сходно потребама, као и исплативост пошто се плаћају трошкови само за оне ресурсе и услуге које се користе [93].

Свеобухватно посматрано услуге Cloud рачунарства се могу поделити у три класе у зависности нивоа апстракције:

- Инфраструктура као сервис (Infrastructure as a Service - IaaS) нуди корисницима виртуализоване ресурсе као што су процесорска снага, скаладишни простор и комуникациони капацитет. Често се од стране провајдера нуде Виртуалне машине које могу бити конфигурисане према потребама корисника. У тим варијантама корисници могу инсталирати додатни софтвер или додати виртуелни диск. Овај вид сервиса везан за инфраструктуру може се сматрати као базични слој код Cloud рачунарства.
- Платформа као сервис (Platform as a Service - PaaS) пружа корисницима окружење у коме могу да реализују своје апликације без потребе да управљају базичним ресурсима. На тај начин корисници не подешавају конфигурацију система као што су број процесора или меморијски простор. Као пример може се навести платформа која нуди окружење за развој и поставку веб апликација уз подршку одговарајућег програмског језика као што је Јава.
- Софтвер као сервис (Software as a Service - SaaS) омогућава приступ софтверским апликацијама преко веб портала. Тако да су корисници у

многим случајевима прелазили са својих локално инсталираних рачунарских програма на онлајн сервисе са истом функционалношћу. На тај начин се олакшава одржавање софтвера, а са друге стране једноставнији је његов развој и тестирање.

Cloud окружење треба да поседује одређене карактеристике које га чине у правом смислу концептом Cloud рачунарства. Корисници треба да буду у могућности да самостално остваре приступ, што би подразумевало слање захтева, прилагођавања, плаћање и коришћење сервиса без помоћи особе оператера.

Још једна од својствених карактеристика односи се на плаћање услуга према коришћењу ресурса. При томе је корисницима омогућено да сами бирају ресурсе који су им потребни и сходно њиховом коришћењу се врши плаћање.

Cloud рачунарство карактеришу високе перформансе и снажна рачунарска подршка, тако да наизглед делује да постоје неограничени ресурси иако у стварности постоје ограничења. Због тога је битна еластичност као својство Cloud рачунарства тако да се ресурси могу додатно проширити веома брзо сходно повећаном оптерећењу. Такође, прилагодљивост потребних ресурса је веома важно код Cloud рачунарства пошто корисници имају различите захтеве [94].

IoT концепт и Cloud рачунарство су међусобно повезани како би се пружила подршка за достизање пуног потенцијала у раду IoT уређаја. Примена великог броја IoT уређаја доводи до генерисања огромне количине података, које је потребно процесирати, сачувати и извршити додатну анализу. Cloud рачунарство управо обезбеђује потребну инфраструктуру и ресурсе за прихватање података. На тај начин IoT уређаји могу директно прослеђивати податке без потребе за обрадом и чувањем података што њихову имплементацију и функционисање чини ефикаснијим. Једна од одлика Cloud рачунарства је и изражена скалабилност која омогућава проширење система кроз увећање броја уређаја у оквиру IoT концепта. Због њихове снажне споне, Cloud рачунарство је постало веома важан чинилац у реализацији IoT апликација.

Одређени недостаци Cloud рачунарства могу се манифестовати кроз проблеме који се могу сагледати на перформансама система, пре свега обрада података у реалном времену може бити упитна при реализацији на Cloud окружењу. Традиционално Cloud рачунарство има ограничења, као што су потенцијални сигурносни ризици, изазови скалабилности и зависност од мрежне, односно Интернет конекције, као и могуће кашњење у испоруци резултата..

Управо, једна од могућих појава код Cloud рачунарства је постојање одређеног кашњења у преносу података између уређаја корисника и Cloud окружења. У том случају може постојати утицај на перформансе апликација и испоруку услуга у реалном времену, како би корисници добили резултат правовремено и на очекивани начин. Код одређених апликација важно је узети у обзир проблеме са кашњењем када се пројектују решења заснована на Cloud архитектури.

Ограничења у погледу пропусног опсега могу имати утицај на брзину и ефикасност преноса података између уређаја корисника и Cloud рачунарства. Ограничени пропусни опсег може довести до спорије реализације преноса података, посебно када се ради са великим количинама података.

Такође, на брзину преноса података може утицати удаљеност између корисника и Cloud сервера, јер подаци морају да путују преко одређене мрежне инфраструктуре Интернета. Затим утицај могу имати ограничења у случају загушења мреже и количине података које се преносе. Такође на брзину могу утицати врста генерисаних података

(нпр. велике датотеке, стримовање у реалном времену, итд.), као и протоколи који се користе за пренос.

Пошто се подаци прослеђују преко мреже на удаљене локације до Cloud рачунарства где се и складиште, сигурносни проблеми морају бити узети у обзир јер такви подаци могу потенцијално бити рањиви на спољне нападе.

2.4. Fog рачунарство

Fog рачунарство је децентрализована рачунарска инфраструктура представљена као нова парадигма намењена за коришћење заједно са Cloud рачунарством и проширује његове функционалности до крајева мреже, обично у близини извора где се подаци генеришу.

Применом оваквог приступа може се утицати на решавање изазова традиционалног Cloud рачунарства, као што су потенцијално веће кашњење, ограничења пропусног опсега и забринутост за приватност и безбедност података. Fog рачунарство је посебно корисно у сценаријима, када се захтева обрада података у реалном времену и ефикасна употреба мрежних ресурса, као што су IoT апликације, паметни градови или индустријска аутоматизација.

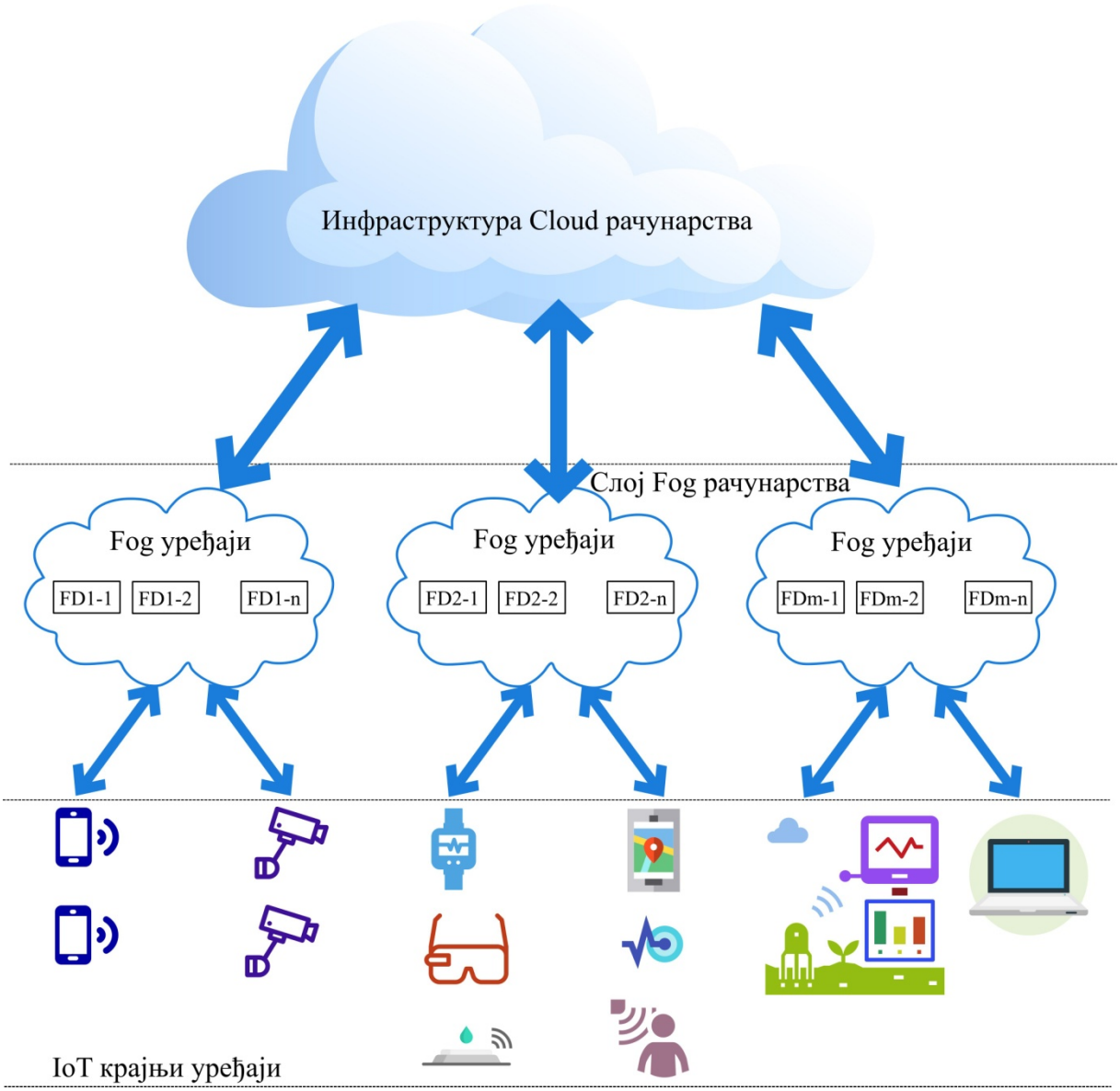
Слој Fog рачунарства се може сматрати додатном подршком између IoT уређаја и Cloud рачунарства и представља проширење Cloud рачунарства на ниво ближи IoT уређајима [95]. Као што је приказано на слици 1, ниво Fog рачунарства садржи функционалности Cloud рачунарства и обезбеђује рачунарске, мрежне и складишне ресурсе. Обично се налази близу извора IoT података и може спровести одређени обим обраде података, без њиховог комплетног слања на Cloud окружење. IoT крајњи уређаји су повезани директно на чворове који се налазе у нивоу Fog рачунарства. Крајњи уређаји могу бити мобилни телефони, камере, носиви уређаји, паметне наочаре, GPS уређаји, различити типови уређаја засновани на сензорима или друга комплетна IoT решења. Структура Fog рачунарства би могла успешно да одговори на претходно наведене захтеве апликације, омогућавајући мање кашњење, динамичку скалабилност и редуковану потрошњу у Cloud окружењу [96].

Fog рачунарство уводи хијерархијску структуру, где Fog чворови делују као посредници између Edge уређаја и Cloud рачунарства. Ова хијерархија пружа могућност да се задаци обраде пренесу са Cloud на Fog чворове, при чему се остварују сви бенефити услед извршавања обраде у близини извора података.

Fog рачунарство је децентрализована рачунарска инфраструктура у којој се подаци, рачунарска подршка, складиште и апликације налазе ближе извору података. Ово омогућава бржу обраду и смањено кашњење. Неке од главних карактеристика Fog рачунарства укључују његову способност да управља обрадом података у реалном времену његову способност да ради у комбинацији са Cloud рачунарством како би се распоредили задаци обраде података..

Процесирањем података ближе крајевима мреже, Fog рачунарство може значајно смањити кашњење, што доприноси бржем времену одзива за апликације у реалном времену. Преносом задатака обраде са Cloud платформе на локалне Fog ресурсе, може се допринети оптимизацији пропусног опсега мреже и смањењу оптерећења на централизованим серверима. На овај начин предности су исказане и кроз омогућену дистрибуцију рачунарских ресурса широм мреже, што олакшава скалабилност односно

повећање или смањење општег оптерећења на основу захтева, али без преоптерећења централизованих сервера.



Слика 1. Илустрација структуре Cloud–Fog рачунарства

Побољшана сигурност се може добити применом приступа Fog рачунарства, јер оно омогућава да се подаци обрађују и анализирају локално, смањујући ризик преноса осетљивих информација на велике удаљености и потенцијално рањивих мрежних веза.

IoT апликације и Fog рачунарство иду заједно као једна моћна комбинација, јер Fog рачунарство може у великој мери побољшати перформансе и ефикасност IoT система, где се очекују одређени захтеви као што је кратко време одзива. Приближавајући рачунарске ресурсе крајевима мреже, Fog рачунарство омогућава IoT уређајима да локално обрађују податке, смањујући кашњење и побољшавајући доношење одлука у реалном времену.

Fog рачунарство пружа одговарајуће рачунарске ресурсе IoT уређајима, смањујући кашњење и побољшавајући могућности обраде података у реалном времену.

Такође, нуди и погодности да се додатни рачунарски ресурси лако додају или уклањају по потреби, омогућавајући ефикасно руковање различитим радним оптерећењима. Fog рачунарство користи Edge интелигенцију како би се омогућила обрада и анализа података на крајевима мреже, смањујући потребу за преносом великих количина података на централизоване сервере.

Fog рачунарство као приступ може да утиче на побољшање поузданости IoT апликација расподелом задатака обраде на више Fog чворова. Ова редундантност може помоћи у ублажавању утицаја хардверских кварова или прекида мреже, обезбеђујући континуиран рад у случају критичних система. Пребацивањем задатака обраде на Fog чворове који се налазе на страни IoT уређаја, Fog рачунарство може помоћи у оптимизацији коришћеног пропусног опсега и смањењу загушења мреже. Наведена поставка може довести до побољшања укупних перформанси мреже [97].

Недостатак Fog рачунарства може да се сагледа кроз потенцијално повећање сложености у управљању дистрибуираном мрежом Fog чворова. При томе се могу јавити изазови у координацији и одржавању различитих компоненти рачунарске инфраструктуре за Fog рачунарство, како би се обезбедиле захтеване перформансе на нивоу више чворова и ефикасно управљање ресурсима за складиштење и обраду података. Поред тога, постављање и одржавање рачунарске инфраструктуре за Fog рачунарство може захтевати посебне ресурсе и стручне кадрове, што би могла бити препрека у неким реализацијама. Још један потенцијални недостатак је ризик од сигурносних рањивости на крајевима рачунарских мрежа где се подаци обрађују локално за разлику од централизованих сервера у Cloud рачунарству.

Ограничена процесорска снага и простор за складиштење могу бити недостаци, пошто Fog чворови обично имају лимитиране могућности обраде у поређењу са централизованим серверима у оквиру Cloud рачунарства. Ово може ограничити сложеност и обим апликација које се могу применити на крајевима мреже. Управљање великим бројем дистрибуираних Fog чворова може бити сложено и захтевати додатне ресурсе, како би се осигурао ефикасан рада, праћење и одржавање ресурса Fog рачунарства.

Кроз пажљиво планирање, имплементацију и управљање, могуће је на прави начих искористити предности Fog рачунарства, док се истовремено ублажавају потенцијални недостаци.

2.5. Оптимизација Fog рачунарства

Оптимизација Fog рачунарства подразумева постизање максималних вредности параметара који се односе на ефикасност и перформансе система који се налази у основи. Очекиване перформансе система у оквиру Fog рачунарства могу се постићи кроз различите стратегије, као што су оптимизација додељивања ресурса, побољшање алгоритама за обраду података, унапређење мрежне повезаности и имплементација интелигентних процеса у доношењу одлука. Са наведеним побољшањима могуће је обезбедити да инфраструктура Fog рачунарства ради на најбољи могући начин, пружајући веће брзине процесирања података и свеукупно оптималне перформансе. Поред тога, континуирано праћење и анализа одговарајућих метрика система може допринети да се открију поједини сегменти где је могуће додатно унапредити или оптимизовати систем. Оптимизација Fog рачунарства подразумева и примену одговарајућих техника како би се на најбољи начин искористила ефикасност Fog

рачунарства за многе захтеве корисника при чему би његова својства максимално долазила до изражаја [98].

Додељивање ресурса код Fog рачунарства подразумева ефикасну дистрибуцију рачунарских ресурса Fog уређаја, као што су процесорска снага, складишни простор и пропусни опсег. Да би се извршила ефикасна додела ресурса у оквиру Fog рачунарства, потребно је узети у обзир факторе као што су захтеви за оптерећењем, мрежни услови и захтеви апликација. Често коришћене стратегије за додељивање ресурса код Fog рачунарства подразумевају балансирање оптерећења (Load Balancing), одређивање приоритета критичних задатака или имплементација интелигентних алгоритама за управљање ресурсима. На тај начин са оптималном расподелом ресурса остварује се минимално кашњење и унапређују се свеукупне перформансе система.

Одржавање баланса оптерећења (Load Balancing) у оквиру Fog рачунарства је посебан изазов којим се доприноси ефикасности система. Под тим се подразумева дистрибуција долазног мрежног саобраћаја и рачунарских задатака на више Fog уређаја на уравнотежен начин, како би се осигурало да не дође до преоптерећења неког од уређаја, а да други остају недовољно искоришћени. Равномерном расподелом радног оптерећења, односно успостављањем баланса оптерећења, код Fog рачунарства доприноси се максималном искоришћењу ресурса. Постоје различити алгоритми и технике за остваривање баланса оптерећења које се могу користити у окружењима Fog рачунарства. Ови алгоритми доприносе и динамичком прилагођавању приликом дистрибуције задатака на основу фактора као што су капацитет уређаја, услови мреже и приоритети задатака. Ефикасним управљањем са радним оптерећењем у Fog уређајима, могуће је обезбедити оптимално коришћење ресурса, побољшати скалабилност и испуњење корисничких захтева за апликације које раде у оквиру концепта Fog рачунарства.

Давање приоритета одређеним задацима који припадају више критичним апликацијама је од великог значаја код Fog рачунарства. На тај начин се осигурава да значајне и временски осетљиве операције имају предност. Задавањем приоритета високог нивоа за критичне задатаке, може се осигурати да се они извршавају што је ефикасније могуће, чак и у окружењима са ограниченим ресурсима. Примена задатих приоритета је веома важна за добијање очекиваног нивоа услуге, одржавање перформанси система и испуњење захтева корисника. Додељивање приоритета одређеним задацима у оквиру Fog рачунарства може се спровести помоћу различитих стратегија. На пример могу се поставити различити нивои важности појединим задацима, задати рокови за одређене задатке, као и динамички вршити усклађивање приоритета задатака на основу захтева у реалном времену. Помоћу одговарајућих алгоритама системи у оквиру Fog рачунарства могу да узму у обзир критичне задатке и сходно томе ефикасно да изврше расподелу ресурса како би осигурали њихово правовремено извршавање.

Примена алгоритама чија је намена управљање ресурсима у Fog рачунарству је од велике важности за оптимално коришћење ресурса у континуитету, што свеукупно гледано доводи до бољих перформанси система и унапређује његову ефикасности. Помоћу ових алгоритама остварује се динамичко додељивање ресурса, као што су рачунарска снага, складишни простор и пропусни опсег мреже за доступне Fog уређаје. Оваква расподела може да се извршава на основу захтева који могу представљати очекивано радно оптерећење и потребно извршавање задатка у реалном времену. Системи који формирају слој Fog рачунарства могу уз коришћење алгоритама за управљање ресурсима да постигну уравнотежену дистрибуцију задатака, али и да се прилагоде новонасталим променама како би очували оптималне перформансе.

Моделе машинског учења могуће је пренети и директно извршавати на Fog уређајима, што доприноси процесу доношења одлука на крајевима мреже. На овај начин се директно могу узети у обзир подаци у реалном времену приликом доношења одлука, а при томе није неопходно преносити све податке и одржавати сталну комуникацију са сервером.

2.6. Процесирање слика

Слика се може представити као дводимензионална или тродимензионална репрезентација визуелних појава, призора или неких других феномена [99]. На пример слика може да подразумева фотографију као запис формиран на основу обрасца интензитета светлости на оптичком сензору. При томе слика се може замислити као континуална функција две променљиве, дефинисана на неком ограниченом простору који обично представља правоугаону област у равни.

У рачунарској визији слика обично представља запис сачуван у дигиталном формату, односно добијена је форма дигиталне слике. Једна од главних карактеристика дигиталне слике је просторна квантизација. Рачунари са својим ресурсима, који су последњих година изузетно напредовали али су ипак ограничени, не могу да представе у потпуности идеално континуиране функције. Због тога се функције узоркују и добијају се низови дискретних елемената који се називају пиксели (pixels) код дводимензионих слика или воксели (voxels) код тродимензионих слика. Друга карактеристика дигиталне слике односи се на квантизацију тих узорака. Наведени процес може одредити дискретне вредности за сваки пиксел омогућавајући тако њихово представљање кроз записе у виду целих бројева. Тако да се записи са једним битом по пикселу односе на бинарне слике. У случају да се користи запис дужине 8 бита, онда су то слике у нијансама сиве боје (greyscale images). А ако се користи 24 бита онда се ради о сликама у боји.

У суштини дигитална слика обично може бити исказана као дводимензиони (или вишедимензиони) низ бројева који представљају одређени објекат или приказ, при чему ови бројеви могу бити у целобројном запису. Када постоји учитана оваква форма слике, са њом се даље може управљати од стране рачунарских система узимајући у обзир вредности ових бројева којима се у ствари описује дигитална слика.

Према томе процесирање (обрада) слике подразумева увођење такве слике, представљене кроз низове бројних вредности, у операције помоћу којих је могуће добити наменски резултат. Под тим се може подразумевати побољшање слике, откривање одређених битних или критичних карактеристика слике, мерење објеката унутар слике или класификација слика у једну од одређених категорија [100].

Систем за обраду слика у реалном времену подразумева континуирано снимање и преузимање слика, анализу слика и коришћене добијених података како би се извршиле одређене активности. При томе се подразумева да се целокупна обрада спроведе у задатом временском интервалу, а често то може да буде временски интервал између снимања слика. Системи у реалном времену подразумевају реакцију система на одговарајући догађај у оквиру тачно задатог временског оквира или у супротном сматра се да систем није испунио своју функционалност [101]. Постоје бројни примери система за процесирање слике у реалном времену као што су надгледање и контрола процеса у машинству, планирање путања и контрола робота, навигација и избегавање судара код аутономне контроле возила.

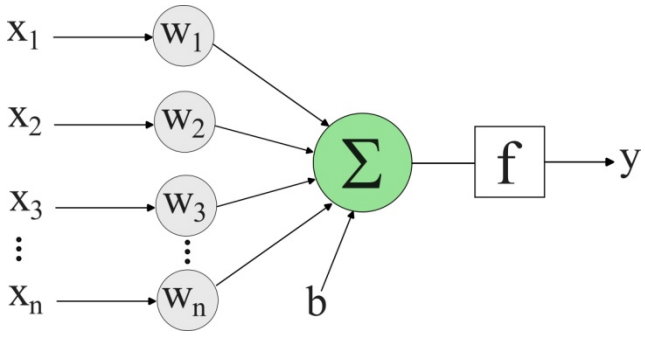
Системи реалног времена могу се разврстати у две категорије, hard системи где се резултати морају обезбедити у оквиру строго задатог времена, иначе се сматра да је дошло до отказа система и soft системи где не долази до отказа система у случају прекорачења задатог времена за одзив али долази до деградације перформанси [100].

2.7. Вештачке неуронске мреже

Вештачка интелигенција представља системе који могу да обављају задатке где се обично захтева људска интелигенција односно опонашање људске интелигенције. Примери таквих интелигентних понашања могу бити разумевање природног језика, препознавање говора или препознавање образаца. Машинско учење (Machine Learning - ML) подразумева развој алгоритама, који омогућавају рачунарима да уче и доносе предвиђања или одлуке на основу података, а да при томе нису експлицитно програмирани од стране човека да то раде.

Дубоко учење, с друге стране, је подскуп машинског учења који користи неуронске мреже са више слојева за учење сложених образаца над подацима. Посебно се може истаћи примена у задацима као што су препознавање слике и говора.

Вештачке неуронске мреже су рачунарски модели инспирисани структуром и начином функционисања људског мозга. Састоје се од међусобно повезаних чворова, или „неурона“, који обрађују и преносе информације. Сваки неурон прима улазне вредности, при чему сваком од тих улаза додељује одговарајућу тежину и врши математичку операцију на њима, а након тога прослеђује излазну вредност другим неуронима (Слика 2).



Слика 2. Пример вештачког неурона

Вредност на излазу једног неурона дефинише се према изразу (1).

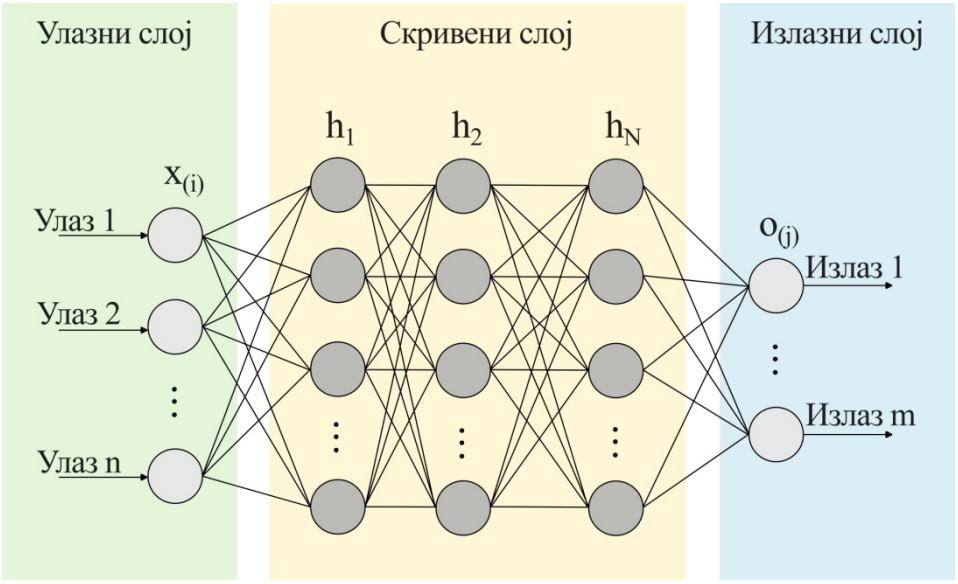
$$y = f\left(\sum_{i=1}^{n} w_i\, x_i + b\right) \tag{1}$$

x_i – улазни подаци;
w_i – параметар тежине;
b – bias.

2.7.1. Потпуно повезане вештачке неуронске мреже

Вештачке неуронске мреже састоје се од међусобно повезаних чворова, или вештачких неурона, који међусобно комуницирају како би учили из података. Потпуно повезани (Fully Connected - FC) слојеви представљају кључну компоненту код такозваних потпуно повезаних вештачких неуронских мрежа (Fully Connected Neural Network - FCNN), посебно у структури дубоког учења. За ове слојеве је везан термин потпуно повезани, пошто је сваки неурон једног слоја повезан са сваким неуроном у претходном слоју [102].

Директне или нерекурентне вештачке неуронске мреже (Feedforward Neural Network - FNN) – представаљају једну од првих и најједноставнијих неуронских мрежа. Састоје се од улазног, једног или више скривених слојева и излазног слоја, при чему дату структуру чине потпуно повезани неурони и информације теку од улаза, према напред, ка излазу без повратних петљи (Слика 3).



Слика 3. Директна или нерекурентна вештачка неуронска мрежа

Излаз неурона првог слоја на основу улазних података, у општем запису, дефинисан је према изразу (2).

$$h_{1j} = f\left(\sum_{i=1}^{N_i} w_{ji}\, x_i + b_{1j}\right) \tag{2}$$

i се креће у опсегу $1 \leq i \leq N_i$, gde N_i представља број улаза.

j се креће у опсегу $1 \leq j \leq N_1$, gde N_1 представља број неурона у првом слоју.

Излазне вредности било којег неурона у оквиру скривених слојева може се представити као у изразу (3).

$$h_{kj} = f\left(\sum_{i=1}^{N_{k-1}} w_{ji}\, h_{(k-1)i} + b_{kj}\right) \tag{3}$$

При чему се k креће у опсегу $1 \leq k \leq N$, где је N укупан број скривених слојева.

j се креће у опсегу $1 \leq j \leq N_k$, gde je N_k број неурона у k-tom слоју.

i се креће у опсету $1 \leq i \leq N_{k-1}$, gde je N_{k-1} број неурона у претходном слоју.

2.7.2. Активационе функције

Активациона функција представља математичку функцију на излазу сваког неурона. Омогућава унос нелинеарности у модел пошто би у супротном ANN мрежа била као класичан линеаран регресиони модел, без обзира на дубину мреже односно број неурона. На тај начин омогућено је мрежама да уче и представљају сложене обрасце у подацима. Уз помоћ активационе функције доноси се одлука да ли се одређени неурон активира прорачуном вредности његовог излаза за параметре тежина и улаза којима се додаје bias. Увођење ове нелинеарности код сваког неурона омогућава да модел може давати предикције и комплексне одлуке [102].

Сигмоидна функција (Sigmoid)

Sigmoid представља математичку функцију која трансформише континуиране вредности реалних бројева у опсег (0, 1), према једначини (4). При томе, за мале улазне вредности излаз је позитиван број близу нуле, док је за велике улазне вредности резултат функције близу један.

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{4}$$

Хиперболички тангенс (Tanh)

Tanh је активациона функција која трансформише улазне бројне вредности у опсег (-1, 1). Tanh се може схватити као проширење сигмоидне функције, имају сличан облик само је ова функција симетрична у односу на нулу, а дефинисана је према једначини (5).

$$tanh(x) = \frac{2}{1 + e^{-2x}} - 1 \tag{5}$$

Исправљена линеарна јединица (Rectified Linear Unit – ReLU)

ReLU активациона функција даје на излазу директно вредности улаза уколико су они позитивни, а у супротном излаз ће бити једнак нули.

$$f(x) = \max(0, x) \tag{6}$$

Запис ReLU функције се може представити на било који од два начина изразима (6) и (7).

$$f(x) = \begin{cases} x, & x \geq 0 \\ 0, & x < 0 \end{cases} \tag{7}$$

ReLU карактерише њена једноставност, пошто се слично као и обична линеарна функција лако имплементира и није рачунарски захтевна. ReLU може помоћи да се ублажи проблем нестајућег градијента код ANN мрежа, што даље поспешује сам процес учења током тренирања модела.

Softmax

Softmax активациона функција се обично користи у ANN мрежама за задатке класификације у више класа. Она претвара необрађене (ненормализоване) резултате из завршног слоја неуронске мреже у дистрибуцију вероватноћа, чинећи да излаз буде исказан као вероватноће класа, што је представљено у изразу (8).

$$s(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}}, \qquad i = 1, \ldots, n \tag{8}$$

x_i – представља излазне вредности завршног слоја ANN мреже који нису нормализовани;
n – број класа.

Излазне вредност функције се крећу и опсегу [0,1]. Излаз представља валидну дистрибуцију вероватноћа тако да је њихов збир једнак 1. Сваки излаз у ствари представља вероватноћу да улазни податак припада одређеној класи.

Softmax се користи у класификационим задацима са више класа, где обезбеђује да модел додели вероватноће свакој класи. Обично је класа са највећом вероватноћом предвиђена класа. Излаз из Softmax обично се упарује са функцијом губитка Categorical Cross-Entropy, која затим израчунава грешку на основу предвиђених вероватноћа.

2.7.3. Функција губитка (Loss function)

Функција губитка у вештачким неуронским мрежама квантификује разлику између предвиђених излаза у моделу и стварних вредности. Она има кључну улогу у тренирању модела тако што учествује у процесу оптимизације којим се прилагођавају параметари модела [102].

Често коришћене функције губитка се могу поделити на Функције губитка регресије и Функције губитка класификације.

Уколико се стварна вредност означи са y, а предвиђена вредност $\hat{y}$, онда се ове вредности могу користити у запису функција губитка. Две често коришћене функције губитка регресије могу се представити у наставку на следећи начин.

Средња квадратна грешка – MSE (Mean Squared Error), једначина (9).

$$MSE(y, \hat{y}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{9}$$

Средња апсолутна грешка – MAE (Mean Absolute Error), једначина (10).

$$MAE(y, \hat{y}) = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{10}$$

У случају модела који су намењени да врше предвиђање којој категорији, односно класи припадају улазни подаци, онда се могу користити Функције губитка класификације. У наставку су наведене три функције губитка које су веома заступљене, при чему је посебно Categorical Cross-Entropy коришћена као функција губитка у реализацији предложеног тестног модела.

Binary Cross-Entropy (BCE)

BCE има своју намену код модела класификације који садржи само два могућа излаза, односно две могуће класе, што је представљено у једначини (11).

$$BCE(y, \hat{y}) = -\frac{1}{n}\sum_{i=1}^{n}[y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)] \tag{11}$$

y_i – Тачна вредност за податак i ;
$\hat{y}_i$ – Вероватноћа предвиђене вредности за податак i ;
n – Број узорака.

Categorical Cross-Entropy (CCE)

CCE је функција губитка која се обично користи у случајевима када се тренирају модели за класификацију са више класа, где свака инстанца припада једној од већ дефинисаних класа. Она исказује дистрибуцију вероватноће сличности између предвиђених и стварних вредности, што је приказано према једначини (12).

$$CCE(y, \hat{y}) = -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{C} y_{ij} \log\left(\hat{y}_{ij}\right) \tag{12}$$

y_{ij} – Тачна вредност за податак i и класу j ;

$\hat{y}_{ij}$ – Вероватноћа предвиђене ведности за податак i и класу j, која представља излазну вредност из слоја Softmax функције;

n – Број узорака;

C – Број класа.

2.7.4. Стопа учења (Learning rate)

Стопа учења је кључни хиперпараметар у тренирању неуронских мрежа [102]. Одређује величину корака са којим ће се ажурирати параметри модела током процеса оптимизације у зависности од израчунате грешке. Стопа учења утиче на динамику, односно брзину којом ће модел да обави процес учења. Брзина конвергенције током тренирања и коначни квалитет тренираног модела зависиће од одабране стопе учења. Вредности стопе учења су обично мали позитивни бројеви као на пример 10^{-4}, 10^{-3} или 10^{-2} у зависности од примењеног алгоритма оптимизације.

У случају када је стопа учења премала, процес оптимизације напредује веома споро. Модел прави ситна ажурирања својих параметара у свакој итерацији, што доводи до споре конвергенције и потенцијално може да се заглави у локалним минимумима. С друге стране, када је стопа учења претерано велика може се десити да алгоритам оптимизације прескочи минимум, што доводи до дивергенције или осцилација које онемогућују погодну конвергенцију.

Током фазе тренирања могу се користити различите стратегије за дефинисање и управљање стопом учења. Може се користити фиксна стопа учења која се често одабира експериментисањем. Такође, стопа учења се може постепено смањивати како тренирање модела напредује, да би се омогућило фино подешавање. На пример, промене стопа учења се могу одвијати након фиксног броја епоха или се њена вредност може скалирати за одређени фактор након сваке епохе.

Код примене алгоритама као што су AdaGrad, RMSprop и Adam стопа учења се прилагођава динамички у зависности од величине градијента.

2.7.5. Пропагација уназад и поступак оптимизације

Приликом тренирања модела вештачке неуронске мреже пролази се прво унапред и рачунају се излазне вредности свих слојева, а на крају и коначни излаз модела, који би требало да представља предвиђену вредност. Нова предвиђена вредност обично се у већој или мањој мери разликује од стварне вредности и потребно је утврдити то одступање. Предвиђање или процене излаза приликом проласка кроз мрежу унапред врше се у односу на параметре модела, а то су тежине између неурона и bias вредности, који на почетку имају произвољно задате вредности. Да би на свом излазу модел предвидео тачну вредност потребно је да учи, тако што ће се наведени параметри модела ажурирати, док не добију своје оптималне вредности. Наведени процес се одвија кроз итеративне кораке, при чему је веома важна техника проласка уназад односно пропагације уназад.

Утврђивање разлике између предвиђене и стварне вредности врши се помоћу функције губитка. Ово се постиже пропагирањем грешке са излазног слоја назад кроз мрежу, да би се ажурирали параметри коришћењем алгоритма оптимизације.

У ANN мрежама, оптимизатор је кључна компонента процеса тренирања модела. Он прилагођава параметре тежине модела да би минимизирао функцију губитка итеративним ажурирањем параметара, на основу градијената израчунатих током повратне пропагације [102].

2.7.5.1. Алгоритам Градијентно опадање (Gradient Descent)

Градијент представља извод функције помоћу кога се одређује ефекат мале промене улазне величине на излаз функције.

Градијентно опадање (Gradient Descent - GD) је оптимизациони алгоритам који има за циљ да пронађе оптималне параметре неуронске мреже (тежине и bias), тако што врши минимизирање функције губитка. Градијентно опадање, као алгоритам оптимизације, има своју важну улогу код тренирања модела дубоког учења. Ради на том принципу да се кроз итеративне поступке врши прилагођавање параметара модела у правцу негативног градијента функције губитка, док се не достигне њен минимум (према изразу (13)).

Алгоритам израчунава градијенте који у ствари представљају парцијалне изводе функције губитка за поједини параметар. Израчунате вредности градијента, записане исто према изразу (14), користе се за ажурирање параметара, тако да се остварује њихова конвергенција ка оптималним вредностима, а који ће даље довести до најниже могуће функције губитка.

$$p_{i+1} = p_i - l \frac{\partial}{\partial p} f(p_i) \tag{13}$$

l – стопа учења;

$f(p_i)$ – функције губитка, при чему је p_i одговарајући параметар модела (тежински параметар или bias).

При томе се може користити следећи формални запис:

$$\frac{\partial}{\partial p} f(p_i) = \nabla_p f(p_i) \tag{14}$$

$\nabla f(p_i)$ – градијент функције губитка у односу на одређени параметар p.

Постоје три позната типа градијентног опадања који се разликују на основу тога колико података се користи за прорачун градијента функције губитка, а то су Batch, Stochastic and Mini-Batch GD. При томе се прави компромис између тачности ажурирања параметара са једне стране и брзине тог процеса са друге стране [103].

Batch Gradient Descent (BGD)

BGD користи цео скуп података за израчунавање градијената, пре него што се изврше ажурирања параметара, што је приказано у изразу (15). Као предност може се навести стабилно и тачно ажурирање параметара уз прикладну конвергенцију. Код великих скупова података, поступак конвергенције тече доста споро и време тренирања модела је дуже, али може довести до прецизнијег коначног модела. За веће скупове података BGD постаје рачунарски захтеван и потребно је више меморијског простора.

$$p_{i+1} = p_i - l \frac{1}{n} \sum_{\mathrm{j}=1}^{\mathrm{n}} \nabla_p f_{\mathrm{j}}(p_i) \tag{15}$$

n – број који означава скуп података за тренирање модела.

Stochastic Gradient Descent (SGD)

SGD користи један по један узорак података за израчунавање градијената и ажурирање параметара, приказано у изразу (16). Предности се огледају у бржем ажурирању параметара и мањој употреби меморије. Такође избегава локалне минимуме због своје стохастичке природе. При томе његове скоковите вредности, приликом ажурирања параметара, воде ка мање стабилној конвергенцији и захтевају пажљиво подешавање стопе учења.

$$p_{i+1} = p_i - l\,\nabla_p f(p_i) \tag{16}$$

Mini-Batch Gradient Descent (Mini-BGD)

Mini-BGD нуди компромис између BGD и стохастичког SGD користећи мање групе података за израчунавање градијента, што је представљено у изразу (17). Обично се у овим применама користе групе података које садрже 32, 64 или 128 узорака из целокупног скупа података за тренирање. Карактерише га ефикасна и стабилна конвергенција и може добро да ради са модерним хардверским решењима (GPU). Са друге стране захтева фино подешавање одабраних група података.

$$p_{i+1} = p_i - l\,\frac{1}{m}\sum_{\mathrm{j}=1}^{\mathrm{m}} \nabla_p f_{\mathrm{j}}(p_i) \tag{17}$$

m – представља мини групу података.

2.7.5.2. Оптимизациони алгоритми засновани на Градијентном опадању

Када се тренирају неуронске мреже дубоког учења, избор оптимизатора може имати значајан утицај на перформансе тренирања и брзину конвергенције. Одабир правог оптимизатора може да варира у зависности од специфичног скупа података и архитектуре модела. Постоји неколико варијанти алгоритма Градијентно опадање, које се разликују по начину на који се ажурирају параметри модела и како се бира величина стопе учења.

SGD са Моментумом (SGDM)

SGDM алгоритам побољшава SGD додавањем моментума за убрзање конвергенције и смањење осцилације на путу оптимизације. Моментум је варијанта SGD алгоритма, која укључује информације из претходних ажурирања параметара, како би помогао алгоритму да брже конвергира до оптималног решења. Ажурирања параметара су заснована на тренутном градијенту и претходним ажурирањима, што је представљено у изразима (18) и (19). Ово може помоћи у спречавању да се процес оптимизације заглави у локалним минимумима и брже достигне циљани глобални минимум [104].

$$v_i = \beta v_{i-1} + (1-\beta)\nabla_p f(p_i) \tag{18}$$

$$p_{i+1} = p_i - l\, v_i \tag{19}$$

Параметар β је коефицијент моментума, који може имати вредности у опсегу [0,1), при чему се често користе вредности између 0,5 и 0,9.

Adaptive Gradient (AdaGrad)

AdaGrad је адаптивни алгоритам за оптимизацију описан у раду [105] код кога се стопа учења прилагођава за сваки параметар на основу претходних градијената, што је приказано у изразима (20) и (21). Карактеришу га могућности дефинисања различитих вредности стопе учења у зависности од димензија неуронске мреже. Предности AdaGrad подразумевају да није потребно ручно подешавати стопу учења. Али постоји и одређена слабост пошто се акумулирају квадрати градијената у имениоцу разломка, што некада током тренирања може довести до вредности стопе учења која ће бити сувише мала.

$$g_i = g_{i-1} + (\nabla_p f(p_i))^2 \tag{20}$$

$$p_{i+1} = p_i - \frac{l}{\sqrt{g_i} + \varepsilon}\nabla_p f(p_i) \tag{21}$$

l – иницијална вредност стопе учења;
ε – константа мале вредности како би се спречило дељење са нулом.

Root Mean Square Propagation (RMSProp)

RMSProp је адаптивни алгоритам за оптимизацију који скалира вредност стопе учења у зависности од величина претходних градијената, што је дато изразима (22) и (23). За разлику од AdaGrad, где се врши акумулација свих претходних градијената, код RMSProp алгоритма се користи њихова покретна просечна вредност. На тај начин дат је већи значај последњим градијентима, омогућавајући тако доста ефикаснију адаптацију стопе учења. Према томе избегава се пребрзо смањење стопе учења до вредности која би могла да заустави процес тренирања у каснијим фазама [106].

$$g_i = \alpha g_{i-1} + (1-\alpha)(\nabla_p f(p_i))^2 \tag{22}$$

$$p_{i+1} = p_i - \frac{l}{\sqrt{g_i} + \varepsilon}\nabla_p f(p_i) \tag{23}$$

l – иницијална вредност стопе учења;
α – стопа опадања и обично има вредност око 0,9.

Adaptive Moment Estimation (Adam)

Adam је адаптивни алгоритам за оптимизацију стопе учења који комбинује предности Adagrad и RMSProp алгоритама. Он прилагођава стопу учења за сваки

параметар на основу првог и другог момента нагиба [107]. Adam алгоритам израчунава експоненцијални покретни пресек градијената (израз (24)) и квадрат градијената параметра модела (израз (25)). Adam има адаптивну стопу учења, која се прорачунава за сваки параметар, убрзавајући конвергенцији ка оптималним вредностима (израз (26)). Карактерише их мали захтеви за меморијом, тако да су погодни за велике скупове података и моделе. Такође, Adam алгоритам је мање осетљив на почетни одабир параметара, међу којима је и иницијална вредност стопе учења.

$$m_i = \beta_1 m_{i-1} + (1 - \beta_1)\, \nabla_p f(p_i) \tag{24}$$

$$v_i = \beta_2 v_{i-1} + (1 - \beta_2)\, (\nabla_p f(p_i))^2 \tag{25}$$

$$p_{i+1} = p_i - l \cdot \frac{\sqrt{1 - \beta_2}}{1 - \beta_1} \cdot \frac{m_i}{\sqrt{v_i} + \varepsilon} \tag{26}$$

l – иницијална вредност стопе учења;
m_i, v_i – прорачунати први и други момент;
β_1, β_2 – експоненцијалне стопе опадања;
ε – константа мале вредности како би се спречило дељење са нулом.

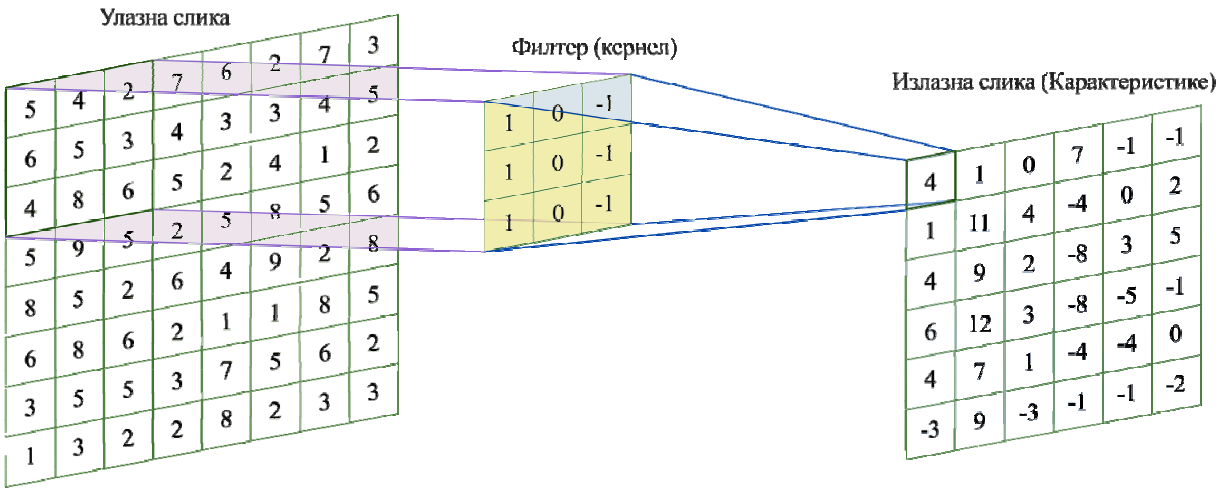
2.7.6. Конволуцијске неуронске мреже (CNN)

Конволуцијска неуронска мрежа (Convolutional Neural Network - CNN) представља једну од класа модела дубоког учења, која је превасходно пројектована за обраду и анализу визуелних података, као што су слике и видео снимци. CNN има широку примену у задацима рачунарске визије, као што су класификација слика, детекција објеката и сегментација слике [108].

CNN мреже раде тако што прогресивно издвајају карактеристике вишег нивоа из улазне слике кроз серију конволуцијских слојева, праћених потпуно повезаним слојевима који обављају задатак коначне класификације. Ова хијерархијска структура омогућава CNN мрежама да науче сложене обрасце и односе унутар података, што их чини веома ефикасним управо за различите задатке рачунарске визије.

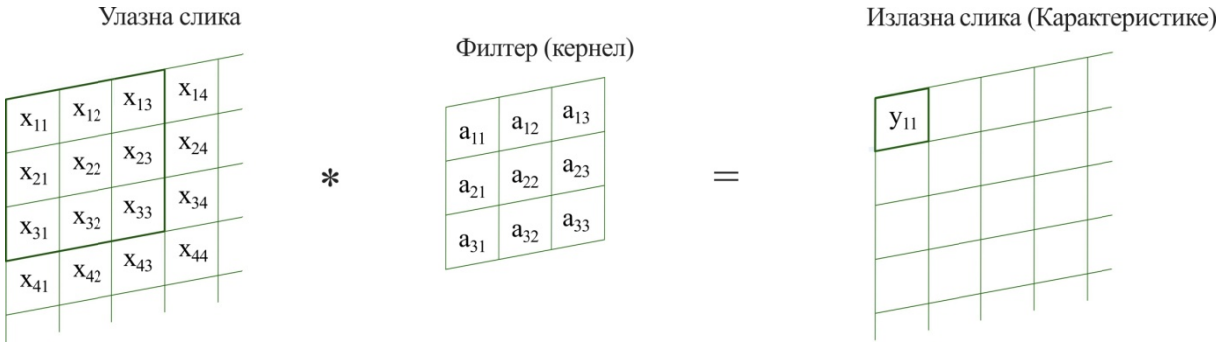
Основну структуру CNN мрежа сачињавају кључне следеће компоненте: Конволуцијски слојеви, функција активације, Обједињавање слојева (Pooling), Потпуно повезани слојеви и на крају излазни слој.

Конволуцијски слојеви примењују операције конволуције на улазне податке, користећи филтере са којима се скенирају слике и откривају различите карактеристике као што су ивице и текстуре. За појам филтер у овом случају користи се још и термин kernel. Филтери клизе, односно прелазе, преко улазне слике и производе мапе карактеристика у којима су садржана присуства специфичних обележја на различитим просторним локацијама унутар слике. Конволуцијски слој подржава неколико филтера, који у ствари представљају мале матрице димензија нпр. 3x3 или 5x5, са којима се скенира улазна слика. У ствари приликом преласка филтера преко улазне слике, изводи се по елементима множење и сумирање између преклапајућег региона улазне слике и филтера. Ово резултира једном вредношћу која исказује присуство одређене карактеристике на тој локацији, а која је одређена типом одабраног филтера (Слика 4).



Слика 4. Поступак конволуције над улазном сликом

Уместо бројних вредноси које су дате на претходној слици, подаци се могу представити са општим записом као на слици 5. Може се уочити како се добија резултујућа вредност која је део једне карактеристике на основу улазне слике и филтера.

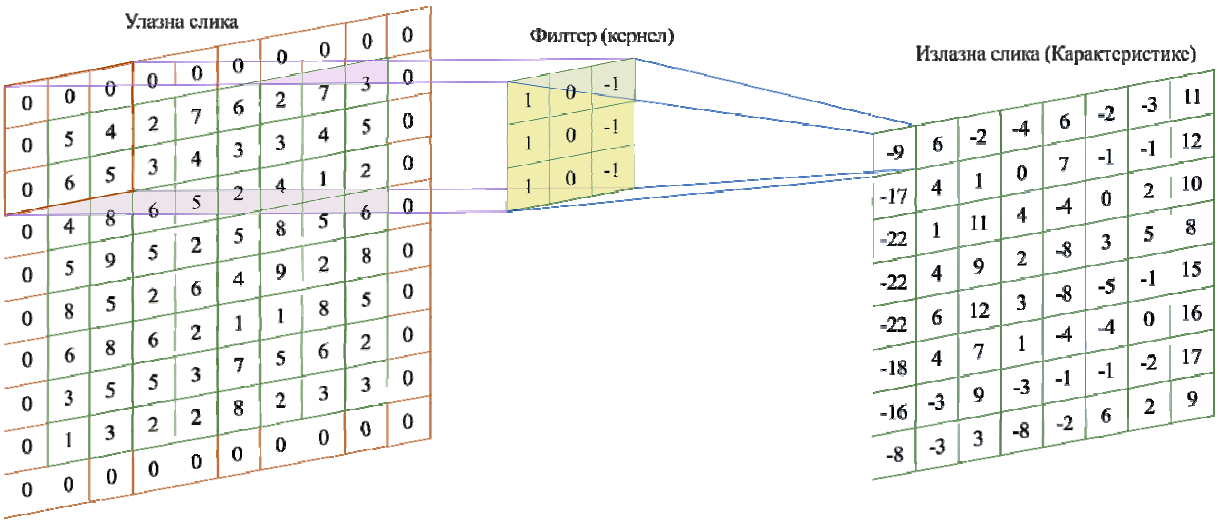


Слика 5. Општи подаци који указују како се добијају вредности нове карактеристике

У тој варијанти резултујућа вредност се добија на основу израза (27), док би се исти принцип понављао и за друге матрице издвојене клизним кретањем филтера преко улазне слике.

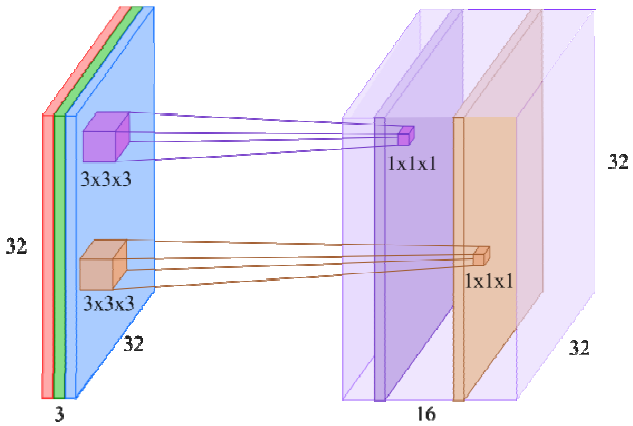
$$y_{11} = x_{11}a_{11} + x_{12}a_{12} + x_{13}a_{13} + x_{21}a_{21} + x_{22}a_{22} + x_{23}a_{23} + x_{31}a_{31} + x_{32}a_{32} + x_{33}a_{33} \quad (27)$$

У случају да је матрица која представља филтер димензија 3x3 и да корак проласка има вредност 1 приликом конволуције, добиће се излазна слика која је мањих димензија од улазне слике. Да би се избегло умањење димензије слика пролазећи кроз слојеве CNN мреже, улазна слика се допуњује са додатном редовима и колонама кроз технику познату као Padding (Слика 6). За наведену поставку параметара конволуције около слике се додају елементи који имају вредност нула и у том случају излазна слика је истих димензија као и првобитна улазна слика.



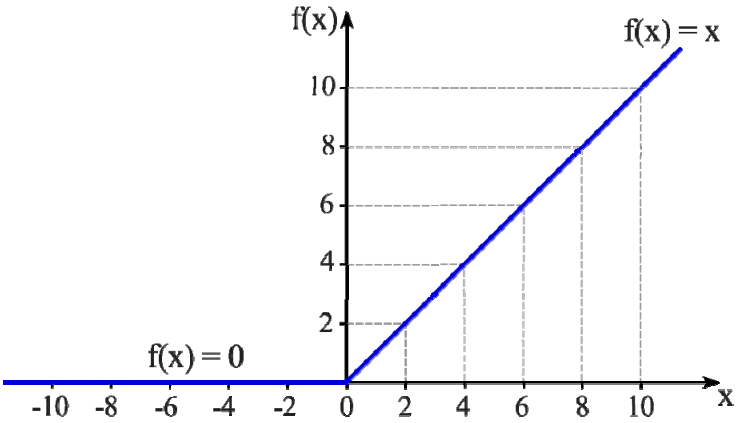
Слика 6. Поступак конволуције уз додатак Padding технике

Резултат примене сваког од појединих филтера на целој слици је мапа карактеристика, која наглашава присуство и локацију појединих карактеристика које су детектовали коришћени филтери у сваком од пролазака и примене операција конволуције (Слика 7).



Слика 7. Мапа карактеристика слике добијене применом конволуција

Активациона функција ReLU се примењује након конволуције за увођење нелинеарности у мрежу. Представља најчешће коришћену функцију активације у CNN моделима и дефинисана је према претходном поглављу, односно $f(x) = max(0,x)$. Укратко речено, ова операција поставља све негативне вредности на нулу, задржавајући и прослеђујући само позитивне вредности (Слика 8).

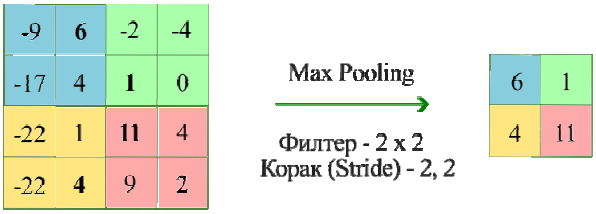


Слика 8. ReLU активациона функција

Слој CNN мреже за Обједињавање (Pooling) се користи како би се смањиле просторне димензије код мапа карактеристика, при чему се обично користе операције као што је обједињавање са максималним вредностима (MaxPooling) или обједињавање са просечним вредностима (AveragePooling).

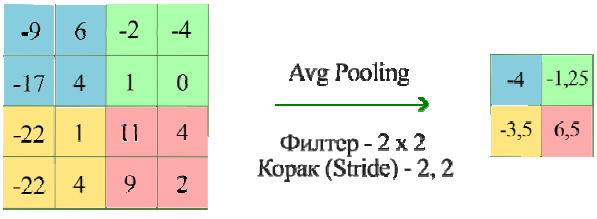
На овај начин, смањује се рачунарско оптерећење и број параметара, што даље доприноси ефикаснијој мрежи по питању рачунарских захтева и која је мање склона преоптерећењу. Током процеса обједињавања, редукују се просторне димензије код мапа карактеристика (ширина и висина), док се задржавају најважније информације.

Најчешћа операција обједињавања је максимално груписање, где филтер (димензије нпр. 2x2) прелази преко мапе карактеристика и бира се максимална вредност унутар селектованог региона који покрива филтер (Слика 9).



Слика 9. CNN слој за обједињавање користећи максималне вредности

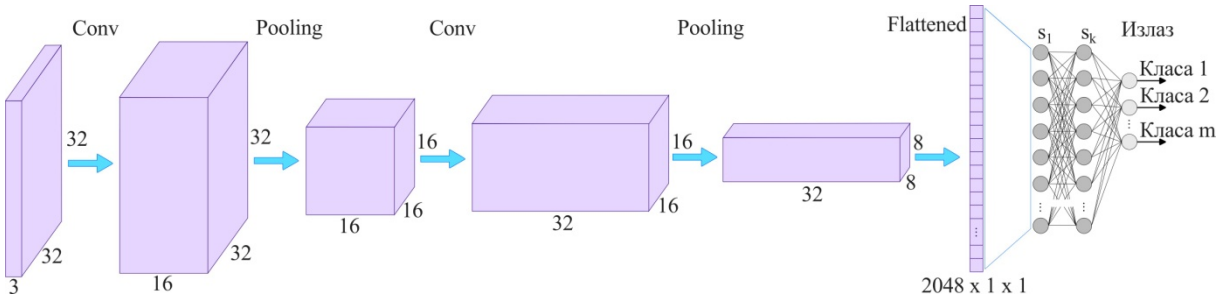
Сличне активност се одвијају и код просечног обједињавања, само у том случају се уместо максималне вредности израчунава просечна вредност унутар региона који прекрива филтер (Слика 10).



Слика 10. CNN слој за обједињавање користећи просечне вредности

Више конволуцијских слојева и слојева обједињавања се обично сукцесивно надовезују један на други, при чему сваки наредни слој извлачи све апстрактније и сложеније карактеристике из улазне слике.

Након одговарајућег броја конволуцијских слојева, процес резоновања високог нивоа се остварује преко потпуно повезаних слојева који се додају у наставку. Процес се спроводи тако што се на крају мапе карактеристика преводе у једнодимензиони вектор (Flattened) и на који се надовезују један или више потпуно повезаних слојева. На крају се добија коначни излаз, који даје предвиђену вредност на основу вероватноће појединих класа (Слика 11).



Слика 11. Шематски приказ CNN модела за класификацију

Последњи односно излазни слој обично користи Softmax активациону функцију за задатке класификације, производећи управо дистрибуцију вероватноће по могућим класама. За предвиђени резултат на излазу се бира класа са највећом вероватноћом.

2.7.7. Својства CNN мрежа

CNN мреже нуде неколико предности које их чине погодним за задатке који укључују обраду слика и видео записа [108].

- Издвајање карактеристика - CNN мреже су пројектоване тако да самостално, без надзора човека, издвајају одређене карактеристике из слика, као што су ивице, текстуре и облици. На овај начин одликује их способност да издвајају хијерархијске карактеристике, почевши у нижим слојевима од једноставних елемената, као што су ивице, до виших слојева и препознавања сложених образаца у сликама.
- Просторна непроменљивост - CNN мреже имају могућност да препознају постојеће обрасце на сликама без обзира где се они налазе унутар слике и која је њихова оријентација. Оваква независност од положаја обрасца на слици остварује се захваљујући конволуционим слојевима код којих се примена филтера спроводи на целој улазној слици.
- Дељење параметара - Код CNN мрежа користи се исти филтер на целу улазну слику, значајно смањујући на тај начин број параметара. Овако дељење параметара тежине доводи до мањег броја параметара мреже и мањег сувишног поклапања са тренираним подацима у односу на потпуно повезане слојеве у једноставним неуронским мрежама. На овај начин добијени мањи број параметара утиче на бољу ефикасност мреже и поспешује процес тренирања.
- Трансферно учење - Унапред тренирани CNN модели (као што су VGG или ResNet) могу се фино подесити за специфичне задатке које карактеришу мањи скупови података. На тај начин је омогућено да се искористе постојећи модели и значајно смање време тренирања и захтевани скупови података за нове задатке. Применом већ тренираних CNN модела кроз трансферно учење

могуће је остварити добре перформансе и у случајевима када постоје само ограничени скупови података.

Са друге стране проблеми који се могу јавити током тренирања дубоких неуронских мрежа односе се на вредност градијента, која може да нестане или насупрот томе да има превисоке вредности. Ови проблеми могу значајно успорити процес тренирања или чак у потпуности онемогућити мрежу да учи. Проблем нестајања градијента јавља се када градијенти постану премали током пропагације уназад. Као резултат тога, параметри тежине се не мењају значајно, а мрежа није у стању да открије основне обрасце у подацима. Вишеслојне дубоке неуронске мреже су посебно склоне овом проблему. Вредности градијента опадају, како се крећу уназад кроз слојеве, што отежава ефикасно ажурирање параметара тежина у првим слојевима.

Проблем превисоке вредности градијента, с друге стране, настаје када градијенти постану сувише велики током повратне пропагације. Када се то догоди, параметри тежине се ажурирају за велике износе, што може довести до дивергенције или осциловања мреже, што у многоме отежава приближавање оптималном решењу.

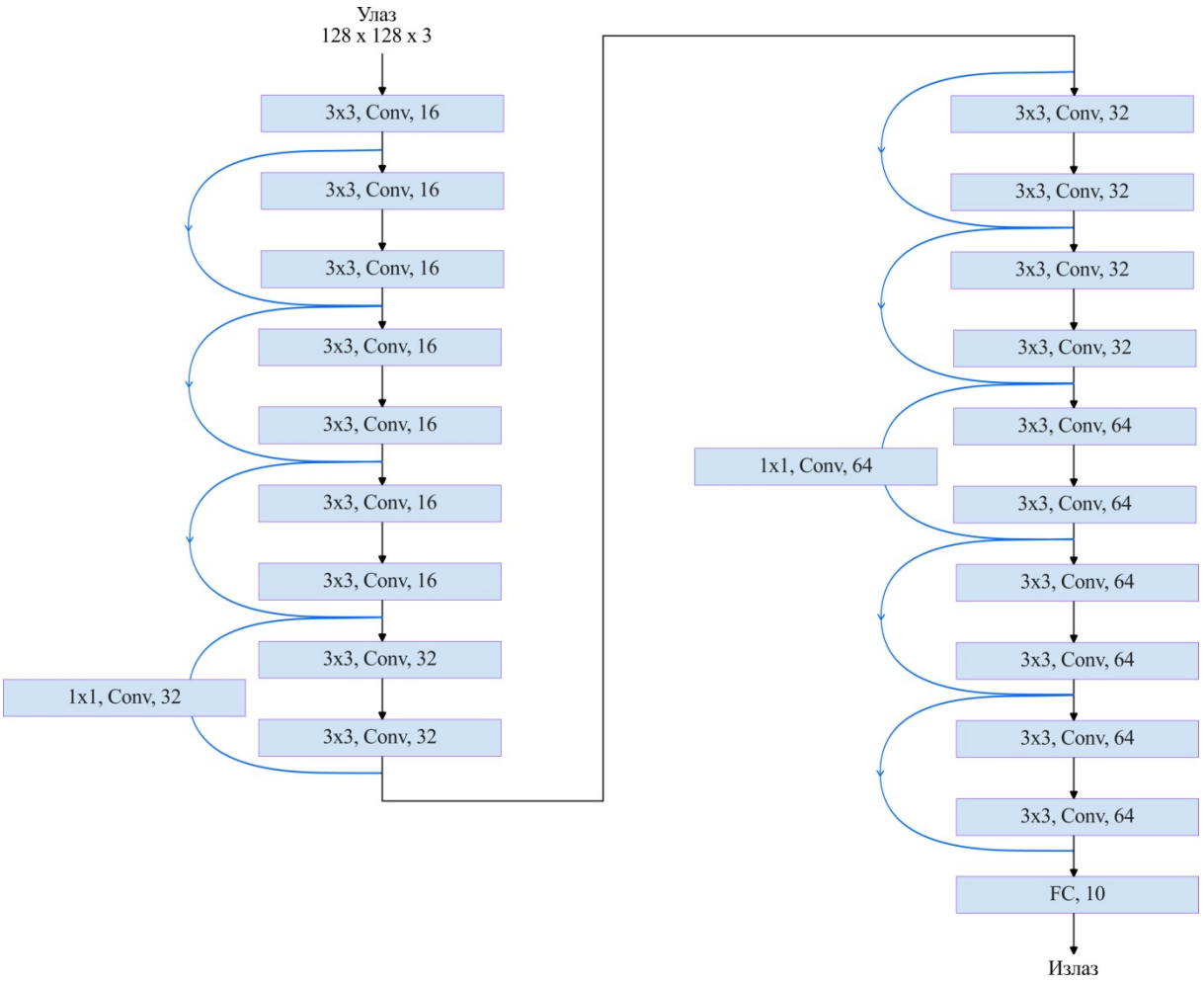
За решавање ових проблема могу се користити технике као што је регулисање иницијалних вредности параметара тежина, како би се покушало осигурати да остану у погодном опсегу. При томе употребом ReLU активационе функције може се ублажи проблем нестајања градијента. Техником одсецања градијената могу се ограничити максималне и минималне вредности градијента, што може онемогућити појаву превеликих и премалих вредности. Још један од начина који може помоћи код ових проблема јесте примена ResNet мрежа.

2.7.8. Резидуалне неуронске мреже (ResNet)

Резидуална конволуцијска неуронска мрежа, са ознаком ResNet, представља врсту дубоке неуронске мреже која уводи прескачуће везе или пречице, како би се одговорило на решавање проблема везаног за нестајања градијента. Ово омогућава мрежи да учи на ефикаснији начин, чак и када постаји већи број скривених слојева у дубини. ResNet мреже су први представили аутори Kaiming He et al. [109] са резултатима који су освојили награду за најбољи рад на Конференцији о рачунарској визији и препознавању узорака (CVPR) 2015. године. Код ResNet кључне компоненте чине резидуални блок и прескачуће везе (Skip connections).

Резидуални блок се састоји од неколико конволуцијских слојева, обично два или три, уз додатну пречицу или везу за прескакање, која заобилази ове слојеве. На тај начин се излаз датог резидуалног блока, односно излаз са краја тих конволуцијских слојева, додаје оригиналном улазу блока (веза идентитета). Прескачућа веза помаже да се ублажи проблем нестајања градијента, дозвољавајући градијентима да теку директно кроз мрежу током повратне пропагације, чиме се побољшава тренирање дубоких мрежа.

Пример ResNet мреже се може сагледати на слици 12. која садржи резудуалне блокове са по два конволуцијска слоја, уз додатак на крају слоја са потпуно повезаним неуронима (FC).



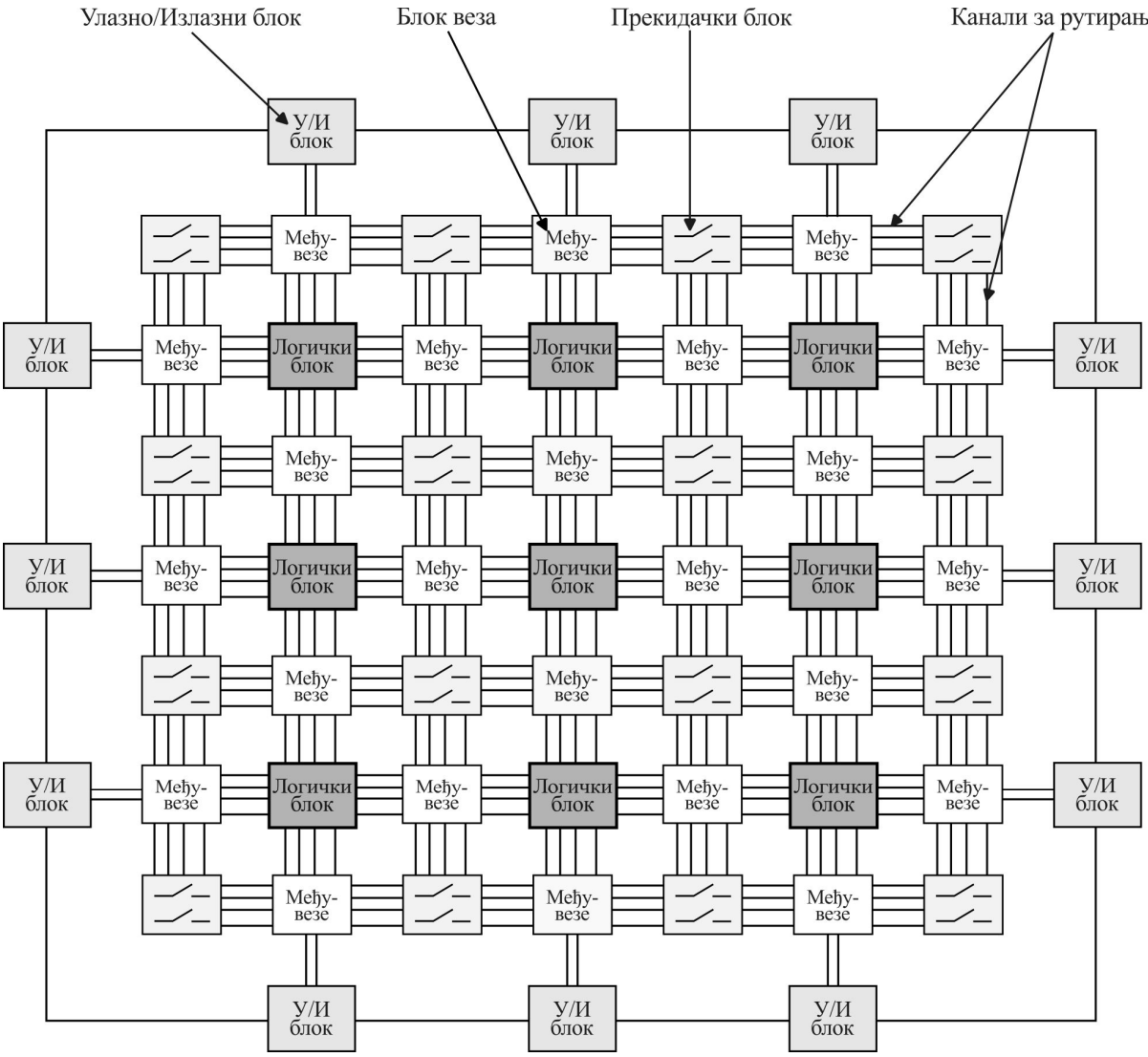
Слика 12. Пример ResNet мреже

Предности ResNet мреже се огледају у могућности тренирања веома дубоких мрежа, којима се могу добити квалитетни коначни модели са постигнутом већом прецизношћу. Прескачуће везе омогућавају да се ублажи проблем нестајања градијента током пропагације уназад, што чини ResNet моделе ефикаснијим за тренирање.

2.8. FPGA (Field-Programmable Gate Array) коло

FPGA (Field-Programmable Gate Array) је врста интегрисаног кола које се може програмирати или конфигурисати накнадно, након производње, тако да могу постати било која врста дигиталног кола или система. FPGA нуде висок ниво флексибилности и прилагођавања у поређењу са традиционалним интегрисаним колима фиксних функција. Могуће је специфицирати жељену функционалност хардверског пројекта, која се затим синтетише у конфигурациони фајл, помоћу кога се програмира FPGA.

FPGA се обично користе у широком спектру апликација, укључујући дигиталну обраду сигнала, телекомуникације, умрежавање, аутомобилску индустрију, ваздухопловство и сличне примене. Посебно су погодни за задатке који захтевају брзу обраду, паралелно рачунарство и перформансе у реалном времену.



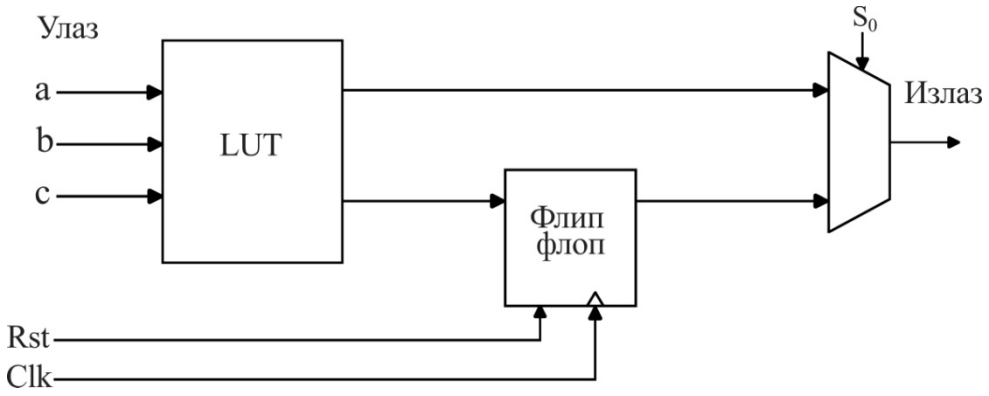
Слика 13. Илустрована основна структура FPGA

FPGA представља снажну платформу, која може да послужи за израду прототипа за нове пројекте, оптимизацију перформанси за специфичне апликације као и имплементацију прилагођених хардверских акцелератора. У реализацији ових хардверских решења њихова прилагодљивост и реконфигурација представљају главне предности. FPGA се састоје од неколико кључних елемената, који чине јединствену целину како би се омогућило програмирање и постизање жељених функционалности. Главне компоненте FPGA укључују програмабилне логичке блокове, међувезе, улазно/излазне блокове и конфигурациону меморију (Слика 13).

Програмабилни логички блокови (Programmable Logic Block - PLB) су основни градивни блокови унутар FPGA који су одговорни за имплементацију захтеваних логичких функција. PLB се састоје од комбинације LUT табела (LookUp Table - LUT), флип-флопова, мултиплексера и других логичких елемената који се могу програмирати да обављају специфичне функције на основу корисничких захтева који се односи на потребни пројекат.

PLB су обично организовани у облику мреже, помоћу које је могуће остварити имплементацију сложених логичких функција међусобним повезивањем више PLB. Сваки PLB садржи скуп улаза, излаза и интерних ресурса за рутирање који омогућавају

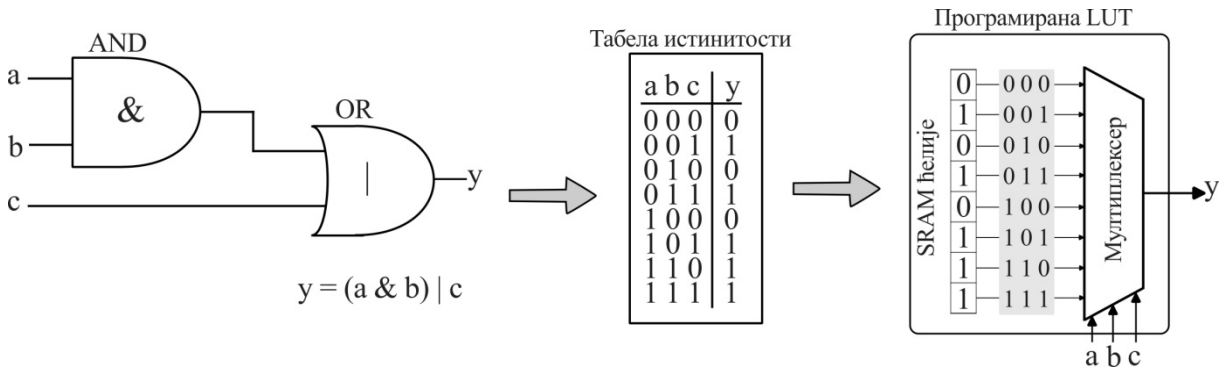
кориснику да дефинише жељено логичко понашање кроз језик за опис хардвера (Hardware Description Language - HDL). Програмабилна логика се састоји од скупа ситно грануларних блокова преко којих се имплементира захтевана функција. Логички блокови су обично засновани на архитектури табеле претраживања (LUT), на основу којих је могуће имплементирати било коју функцију улаза (Слика 14).



Слика 14. Програмабилни логички блок

Сваки LUT у FPGA колу састоји се од фиксног броја улаза (обично 3, 4, 5 или 6) и одговарајућег излаза. Величина табеле истинитости одређена је бројем улаза у LUT табели. На пример, LUT са 4 улаза може да ускладишти табелу истинитости са 16 уноса ($2^4 = 16$), док LUT са 6 улаза може да ускладишти табелу истинитости са 64 уноса ($2^6 = 64$). Ови улазни контакти примају улазне сигнале који се користе за израчунавање излаза на основу ускладиштене табеле истинитости.

Управо када треба имплементирати логичку функцију у FPGA колу, специфицира се жељена табела истинитости за ту функцију. Софтверски алати за програмирање FPGA затим мапирају ову табелу истинитости у одговарајуће LUT табеле унутар уређаја. Током рада, FPGA коло користи ускладиштене табеле истинитости у LUT да би израчунао излаз на основу тренутних улазних вредности (Слика 15).



Слика 15. Пример LUT табеле са три улаза

Табеле за претраживање (LUT табеле) су основне компоненте унутар FPGA кола које играју кључну улогу у имплементацији логичких функција. LUT табеле су у суштини мале, конфигурабилне меморијске јединице које чувају табеле истинитости за специфичне логичке функције. Код FPGA кола, LUT табеле се обично користе за

имплементацију комбинационих логичких функција, где излаз зависи искључиво од тренутних улазних вредности.

LUT табеле су кључни фактор за флексибилност и реконфигурацију FPGA кола, јер омогућавају имплементацију широког спектра логичких функција без потребе за физичким променама на хардверу. Ефикасним коришћењем LUT табела у FPGA дизајну, могуће је остварити високе перформансе, прилагођена решења за различите апликације, укључујући дигиталну обраду сигнала, комуникацију, умрежавање и друго. У оквиру FPGA кола LUT се могу реконфигурисати, што значи да садржај табеле истинитости може да програмира или конфигурише сам корисник за имплементацију специфичних логичких функција. Конфигурациона меморија чува вредности табеле истинитости које је дефинисао корисник. Према томе конфигурациона меморија чува конфигурационе податке који дефинишу функционалност FPGA. Ови подаци се користе за програмирање логичких блокова, међувеза и других компоненти FPGA како би се имплементирао жељени пројекат.

LUT табеле често садрже мултиплексере који се користе за избор одговарајуће излазне вредности из табеле истинитости, на основу тренутних улазних сигнала. Мултиплексери помажу у ефикасној имплементацији сложених логичких функција користећи комбинацију основних логичких функција.

Међувезе су путање који повезују различите логичке блокове унутар FPGA. Они омогућавају комуникацију између различитих делова FPGA и рутирање сигнала између логичких блокова.

Улазно/излазни блокови (Input/Output Blocks - IOB) омогућују интерфејс између FPGA и спољних уређаја. Они су одговорни за управљање улазним и излазним сигналима, као и за обезбеђивање неопходног баферовања и промене нивоа напона.

Целокупна структура и прекидачки блокови код FPGA су контролисане од стране конфигурационих битова. Конфигурациони подаци се налазе унутар конфигурационог фајла, који се користи за програмирање система. Процес конфигурације FPGA кола постиже се учитавањем конфигурационог фајла у интерну меморију. Код независних FPGA система конфигурациони фајлови се обично налазе на екстерној флеш меморији, подешени тако да се фајл аутоматски учитава приликом укључивања FPGA кола.

FPGA се може поново конфигурисати када год је то потребно, тако што се поново покрене рад када је учитана нова конфигурација. Приликом реконфигурације не остају информације које су биле везане за дотадашњи рад, већ се све информације стања морају сачувати ван чипа и накнадно учитати након реконфигурације [100].

Приликом примене FPGA важно је истаћи њену својственост, која се односи на динамичку и парцијалну реконфигурацију што је приказано у раду [110] кроз детаљан преглед многих аспеката у литератури. Предочена су претходна истраживања и указано је на изазове са чијим одговорима би се указало на шире усвајање парцијалне реконфигурације.

Када се пројектују системи засновани на FPGA колима, обично се користе језици за опис хардвера (Hardware Description Language - HDL) како би се дефинисала жељена функционалност система. HDL код се затим синтетише у конфигурациону датотеку која програмира FPGA. Пројектанти могу специфицирати логичке функције, међусобне везе и друге параметре FPGA пројекта користећи HDL код, омогућавајући висок степен прилагођавања и флексибилности.

Синхрони системи се обично имплементирају на RTL (Register Transfer Level) нивоу, користећи HDL језик. HDL представљају језике за описивање хардвера и две

највише заступљене варијанте су VHDL и Verilog језици, који су постали IEEE стандарди. Намена VHDL и Verilog језика првобитно је била за документовање и моделирање електронских система. Тако да се одговарајући елементи и модули лако могу написати, користећи било који од ова два језика ради тестирања функционалности електронског кола. На овај начин омогућено је тестирање пројекта и детаљна провера исправности свих елемената пре него што се изврши синтеза.

И VHDL и Verilog имају своје предности и мане, а избор између њих често зависи од специфичних захтева пројекта и познавања језика од стране пројектанта. Некада се можда пре одабира VHDL због његове робусности и могућности формалне верификације, док је у другим случајеви одлука за Verilog због његове једноставности и лакоће употребе.

VHDL и Verilog су хијерархијски структурни језици, што значи да могу описати хардвер у форми блокова логичких компоненти и њихових међувеза. Овакав начин омогућава да се сложени пројекат подели на модуле тако да се може изградити од једноставнијих компоненти.

FPGA кола поседује и одређени облик синхронизације такта за контролу коришћеног сигнала такта у односу на спољашњи извор. Мрежа за дистрибуцију такта омогућава такт сигнале за све делове пројекта и при томе ограничава одступања вредности такта између различитих делова.

У оквиру FPGA кола постоји и наменска контролна јединица за учитавање захтеваног пројекта или конфигурације у FPGA. Оваква наменска логика не представља директно део захтеваног корисничког пројекта, већ је део пратеће логике која омогућава управо програмирање FPGA кола.

У случају сложених функција њихова реализација се остварује каскадном поставком више LUT табела, које би преузимале излазе са претходних слојева као улазне вредности за следеће табеле у низу. Одабир величине LUT табеле представља посебан изазов и зависи од компромиса са којим се суочава произвођач. У случају великих LUT табела, значајан део логичке јединице остаје неискоришћен. Са друге стране код мањих LUT табеле боља би била искоришћеност ресурса, али у случају сложених функција било би потребно више каскадних логичких ћелија што повећава рутирање, као и пропагационо кашњење. У оквиру блока постоје директне везе тако да се смањи број сигналних путања за рутирање у оквиру матрице међуповезивања, при чему се такође редукује и време пропагационог кашњења.

Првобитне FPGA структуре су биле хомогене са само једним типом логичких ресурса. Каснијим развојем и посебним применама рачунарских ресурса дошло се до потребе за тачно одређеним функцијама чија имплементација преко основне, ситно грануларне, логике постаје доста скупа. На тај начин је архитектура постала у ствари хетерогена, тако што садржи више комплексне блокове формирајући крупније грануларне елементе. На пример такви блокови могу обухватати DSP (Digital Signal Processing) апликацију за процесирање дигиталног сигнала.

Потребна меморија код FPGA кола се највише реализује додавањем мањих блокова меморије, што се може остварити прилагођавањем структуре логичких ћелија, тако да се користе као меморија са непосредним приступом (RAM меморија). Већи елементи RAM-а се имплементирају додавањем наменских меморијских блокова статичког рама (BRAM). Меморијски блокови обично поседују два порта, што доста поједностављује конструкцију FIFO бафера и прилагођене кеш меморије.

Уколико је за одређене апликације потребан процесор за задатке са серијским низом инструкција, наведени процесор се такође може имплементирати преко FPGA логике. Али у том случају, кашњења повезана са програмабилном логиком и међувезама ће ограничити брзину такта таквог процесора. Тако да многе фамилије FPGA кола имплементирају на чипу и серијски процесор високих перформанси. У данашњим системима то је обично вишејезгарни ARM процесор. Наведена комбинација FPGA и CPU се често назива „систем на чипу“ (System On Chip - SoC), пошто даје могућност комплетног пројекта управо на само једном чипу. Употреба SoC-а при реализацији специјализованих система може утицати на смањење цене, величину система и потребну снагу.

Интерфејс између програмске логике FPGA кола и уграђеног ARM процесора у оквиру система на чипу (SoC) најчешће је остварен преко AXI магистрале. Она успоставља комуникацију између управљача и подређеног интерфејса (master - slave). При томе управљач даје адресу на адресном порту, као и податке за пренос на порту за податке. У случају више трансакција, које је потребно остварити управљач не мора да чека завршетак започетих трансакција да би покренуо следеће. Сваки трансфер има своју идентификациону вредност тако да трансакције могу бити спроведене независно [100].

За потребе пројекта на ниском нивоу могуће је инстацирати примитиве за одговарајуће логичке ћелије и друге основне градивне блокове који су доступни на FPGA колу. Дати начин представља погодну варијанту у реализацији библиотека са блоковима интелектуалне својине (IP). Након тога дефинисани блокови се могу инстанцирати више пута и примењивати у другим пројектима.

Пројекат на ниском ниво је више фокусиран директно на карактеристике хардвера, а мање на алгоритме који представљају захтеве у многим апликацијама. Да би се одговорило датом изазову, спроведена су значајна истраживања како би се синтеза хардвера могла извести из језика високог нивоа, као што је C језик. Наведено је постало могуће посредством софтверског алата High-level Synthesis (HLS).

HLS пружа подршку да се одговарајући софтверски код лако компајлира за хардверску имплементацију. Потребни алгоритми пројекта се реализују и тестирају користећи C језик, а након тога се конвертују у HDL језик, где је такође потребна верификација и дорада пројекта. Ипак овакав процес конверзије и припреме пројекта је у неким случајевима дуготрајан и склон грешкама. Уколико захтевани пројекат садржи временски критичне компоненте и компоненте које је лакше паралелизовати, онда се исте могу компајлирати у хардвер, док се више секвенцијалне компоненте компајлирају као софтвер. Наведена поставка је изузетно важна код пројектовања система на чипу (SoC) где се може користити FPGA како би се убрзали кључни делови пројекта. Такође, употреба софтверског језика за опис хардверског пројекта има своје предности пошто на тај начин може да буде доступан и софтверским инжењерима [100].

Тако да HLS често представља популарно решење за пројектовање система високих перформанси и очекиване енергетске ефикасности, који са друге стране нуде краће време за реализацију. HLS пружа подршку пројектантима да раде на високом нивоу апстракције тако што би користили софтверске алате за дефинисање функционалности хардвера [111].

Иако HLS пружа подршку за брз развој пројеката, он не представља замену за посебно детаљан пројекат хардвера где је потребно посебно обратити пажњу на многе карактеристике хардверских перформанси. Са друге стране HLS постаје све више погодан када се користи у одговарајућим случајевима, где својом брзином реализације

и други својствима могу постићи одређене предности у односу на класичан RTL пројекат [100].

FPGA је од своје појаве до савремене примене имао раст што се тиче перформанси са фактором пута 100, док су са друге стране трошкови и потрошња енергије по операцији смањени са истим фактором. Овакав напредак је настао захваљујући могућности скалирања процесне технологије, што је даље довело и до квалитативних промена у архитектури, алатима и реализованим апликацијама [112].

Потенцијал употребе FPGA као акцелератора у центрима података и Cloud рачунарству сумирана је у раду [113]. Акценат је био на истраживању архитектура, управљању ресурсима и примени FPGA апликација на Cloud архитектурама. Представљене су постојеће архитектуре и разматрана је скалабилност и апстракција, док је приказана и погодност датих архитектура за апликације као што су дубоко учење и аналитика података.

Захваљујући својим карактеристикама FPGA представља предмет разматрања и примене у области IoT апликација као што је процесирање слика [114]. CNN модели имају велику примену посебно у апликацијама за рачунарску визију при чему се у тим случајевима намећу захтеви интензивне рачунарске обраде. У тим случајевима обично се јављају потешкоће у одржавању нивоа перформанси применом стандардних процесора. Да би се одговорило на наведене захтеве у циљу побољшања перформанси могу се користити хардверски акцелератори засновани на GPU, ASIC и FPGA решењима. Са друге стране постоје захтеви за процесирање података у реалном времену и са што мањом потрошњом енергије. Наведени захтеви се односе на примене код уграђених система ограничених ресурса, што је довело до додатне употребе акцелератора. При томе је FPGA доста прихваћен за реализацију акцелератора код мрежа дубоког учења, захваљујући њиховој способности да увећају паралелизам у обради и енергетску ефикасност.

У раду [115] представљене су савремене технике помоћу којих је могуће остварити убрзање мрежа дубоког учења, као што су CNN мреже, помоћу FPGA кола. При томе је дат нагласак на кључне карактеристике коришћене од стране различитих техника за побољшање перформанси акцелератора.

Детаљно истраживање о сложености рачунања и меморијским захтевима за сваки CNN слој представљено је у раду [116]. Предложен је скалабилни радни оквир који садржи четири нивоа паралелизма у хардверском акцелератору. А затим је истакнута методологија истраживања да би се пронашла оптимална решења која би унапредила пропусност акцелератора и одговорила на ограничења код FPGA, која се могу односити на меморију на чипу, пропусни опсег екстерне меморије и рачунарске ресурсе. Наведена методологија оптимизације је примењена на три истренирана CNN модела који су били покренути на Xilinx VC709 платформи. Показано је да перформансе наведена три акцелератора, заснована на CNN моделима, надмашују примере са класичним CPU, као и претходне сличне примене.

Област истраживања која је посебно добила на значају у последње време односи се на примену модела машинског учења као подршке за доношење одлука на мобилним уређајима и уграђеним системима. У раду [117] дат је преглед оваквих система заснованих на FPGA са фокусом на могуће апликације, различите платформе и изазове са којима се суочавају. Представљена истраживања имају за циљ да заинтересују и подстакну развој нових практичних решења са FPGA подршком намењених за примену код уграђених система.

Често је изазовно применити стандардне вештачке неуронске мреже као што је CNN на уграђене уређаје због великог броја операција и параметара. Једна од савремених CNN мрежа MobileNet уводи дубински одвојиву конволуцију, која може заменити стандардну конволуцију. На тај начин се добија значајно мање параметара уз контролисане губитке у тачности. У раду [118] је предложен један такав CNN систем високих перформанси заснован на FPGA.

Значајан напредак током година имало је реконфигурабилно рачунарство, што је истакнуто у раду [119], где је дата сажета студија о FPGA као најшире заступљена структура у овој области рачунарства. Наведени су потенцијални примери за примену FPGA као што су ауто индустрија, комуникације, апликације дубоког учења или потрошачка електроника. Дати преглед може послужити корисницима и истраживачима да препознају апликације где могу искористити предности FPGA.

У раду [120] представљени су алати који се могу користити за реализацију токова преноса система од CNN модела до FPGA платформе (CNN – FPGA). Дата је упоредна анализа њихових карактеристика и разматране су њихове предности и слабости, као и технике мапирања.

У поређењу са ASIC (Application Specific Integrated Circuits) системима FPGA представља јефтиније решење у случају мањег и средњег обима производње пошто је нуди краће време за добијање уређаја за потребе тржишта.

Да би се добио први ASIC уређај неопходно је много више ресурса у виду времена и новца за реализацију. А за FPGA све је оствариво за врло кратко време, често мање чак од минута, да би се конфигурисао односно програмирао уређај. При томе су лако доступни многим корисницима и то по доста приступачној цени. Сходно захтевима, FPGA се може реконфигурисати на једном делу, док преостали део система и даље успешно ради. Ажурирање постојећег система може се једноставно спровести преносом записа битова нове апликације. Флексибилност FPGA се може сматрати њиховом великом предношћу, али са друге стране такође утиче на друге карактеристике FPGA, тако да су већих димензија, спорији су и имају већу потрошњу енергије у односу на ASIC уређаје. Наведени недостаци највише потичу као последица да међувезе код FPGA које се користе за програмирање сачињавају близу 90% укупне површине FPGA система. И поред тога FPGA представља прихватљиво решење за реализацију дигиталних система услед нижих трошкова и краћег времена потребног за реализацију уређаја [121].

2.8.1. Предности коришћења FPGA кола

Може се навести неколико разлога који су довели до тога да и у савременим системима FPGA има посебан значај код рачунарства високих перформанси. Особине као што су флексибилност, прилагодљивост, могућности паралелне обраде, мало кашњење, могућност реконфигурације, енергетска ефикасност и скалабилност FPGA чине вредним алатом за реализацију хардвера када се желе остварити високе перформансе и оптимизовани системи за различите апликације.

Код секвенцијалне обраде увођењем слојева апстракције, који скривају основни хардвер, отежано је проналажење правог облика паралелизма. Тако да постоје потешкоће са покушајем скалирања перформанси једноставним повећањем број језгара или процесора за извршавање апликације. FPGA нуде могућности паралелне обраде,

омогућавајући извршавање више задатака истовремено, што последично води ка побољшању укупних перформанси система.

Са развојем технологија повећана је густина транзистора што је допринело да се добију веома велике FPGA конфигурације у односу на прошло време. Постао је доступан огроман број логичких ћелија, тако да је омогућена реализација сложених рачунарских задатака. Пошто FPGA ради на нижим фреквенцијама не достижу се лимити густине снаге као код CPU варијанти.

Флексибилност FPGA омогућава реализацију скоро сваке рачунарске конфигурације, која се може замислити и користити било који облик паралелизма који одговара апликацији. Ова флексибилност пружа могућност стварања идеалних решења за специфичне апликације, које би у неком прошлом времену биле малог обима, али у савремено доба оне могу бити заиста велике, важне и прилагођене многим занимљивим апликацијама [122].

Предности коришћења FPGA кола могу се сагледати према следећем:

- Прилагодљивост код FPGA је карактеристична пошто омогућава имплементацију прилагођених хардверских акцелератора према потребама одређених апликација, пружајући оптимизоване перформансе и ефикасност.
- FPGA имају мало кашњење, омогућавајући обраду података у реалном времену и кратко време одзива када се захтева брзо доношење одлука.
- FPGA се могу реконфигурисати када год је то потребно, омогућавајући динамичке промене у дизајну хардвера без потребе за физичким модификацијама осталог хардвера.
- FPGA се могу оптимизовати за енергетску ефикасност селективним активирањем само неопходних логичких блокова, смањујући укупну потрошњу енергије у систему.
- FPGA се може лако скалирати да би се прилагодили променљивим захтевима, што их чини погодним за широк спектар апликација од малих уграђених система до великих центара података.

2.8.2. Уграђени системи

Уграђени систем (Embedded System) представља рачунарски систем посебне намене који је имплементиран у објекат или производ који се надзире или контролише. Пројектован је за извршавање једне или неколико повезаних функција и често је ограничен захтеваним прорачуном у реалном времену.

Уграђени системи такође представљају корисно решење код визије слике, тако да на пример паметне камере поред снимања слике остварују и њихову обраду како би се добиле нове информације. Овакве примене могу се сагледати кроз примере као што су „интелигентни“ системи за видео надзор, индустријско надгледање и контрола или визија слике у области роботике.

Код уграђених система обично постоје захтеви да буду малих димензија и мале тежине. У случају да раде на батерије врло важан захтев је да своју оперативност заснивају на малој потрошњи електричне енергије, односно да раде као уређаји мале снаге. Чак и они системи који се не заснивају на батеријском напајању имају дефинисано ограничење снаге. Такође сви уграђени системи имају ограничења по питању опсега радне температуре, пошто хлађење може бити изазовно у случају великог оптерећења приликом обраде података.

Применом IoT концепта многи сензори се могу дистрибуирати за надзор или праћење. Такође и са појавом јефтиних камера мале снаге омогућено је да се сензори за визију споје са IoT концептом.

Карактеристично за уграђене системе јесте да могу обрађивати податке, на пример слике, локално и преносити само изведене податке. У таквој форми процесирањем података на ивичним уређајима мреже смањују се трошкови преноса података и обраде на серверској страни. Такође као последица добија се боља ситуација која се тиче приватности и сигурности пошто се подаци не преносе на удаљене локације [100].

2.9. Алгоритам оптимизације (PSO)

У циљу проналажења оптималних вредности коришћен је алгоритам заснован на рoју честица (Particle Swarm Optimization - PSO). PSO је један од оптимизационих алгоритама, који је инспирисан појавама у природи који за циљ има тражење оптималног решења у посматраном простору. Овај алгоритам је постао занимљив међу многим истраживачима због своје једноставне употребе и мањег броја контролних параметара. Дефиниција алгоритма се може представити по аналогији са понашањем рojева у природи, као што су јата птица или јата риба. Чланови роја, попут птица, могу да комуницирају једни са другима, имају начин да поделе своја искуства и прате птицу која је најближа извору хране. Ово својство рojева у природи је препознато и уграђено у алгоритам за решавање проблема оптимизације. Птице, као елементи у јату, уводе се у алгоритам као честице у рoју, где свака честица представља једно од могућих решења. Дефинисани рој честица може се користити за тражење оптималног решења у одређеном простору. PSO алгоритам се имплементира са почетном величином популације на првом месту која представља број честица у рoју. Ако је коришћена величина популације премала, резултати оптимизације обично могу дати лоша решења. Када је величина популације довољно велика, могу се очекивати и добити бољи резултати евалуације. PSO је постао прихваћен и погодан алгоритам за оптимизацију због својих својстава, која се могу укратко истаћи на следећи начин [123]:

- PSO није компликован за коришћење и кодирање.
- PSO алгоритам садржи само три главна контролна параметра (тежина инерције, константа честице и константа роја). Са овим параметрима могуће је управљати перформансама алгоритма.
- PSO је довољно флексибилан да се комбинује са другим алгоритмом за оптимизацију.

$$v_i(\mathrm{t+1}) = \mathrm{w}v_i(\mathrm{t}) + c_1 r_1 \left(Pbest_i\ (\mathrm{t}) - x_i\ (\mathrm{t})\right) + c_2 r_2\ (Gbest\ (\mathrm{t}) - x_i\ (\mathrm{t})) \tag{28}$$

$$x_i\ (\mathrm{t+1}) = x_i\ (\mathrm{t}) + v_i\ (\mathrm{t+1}) \tag{29}$$

c1 – коефицијент когнитивног убрзања,

c2 – коефицијент социјалног убрзања,

r1 и r2 су две униформне случајне вредности из интервала [0, 1].

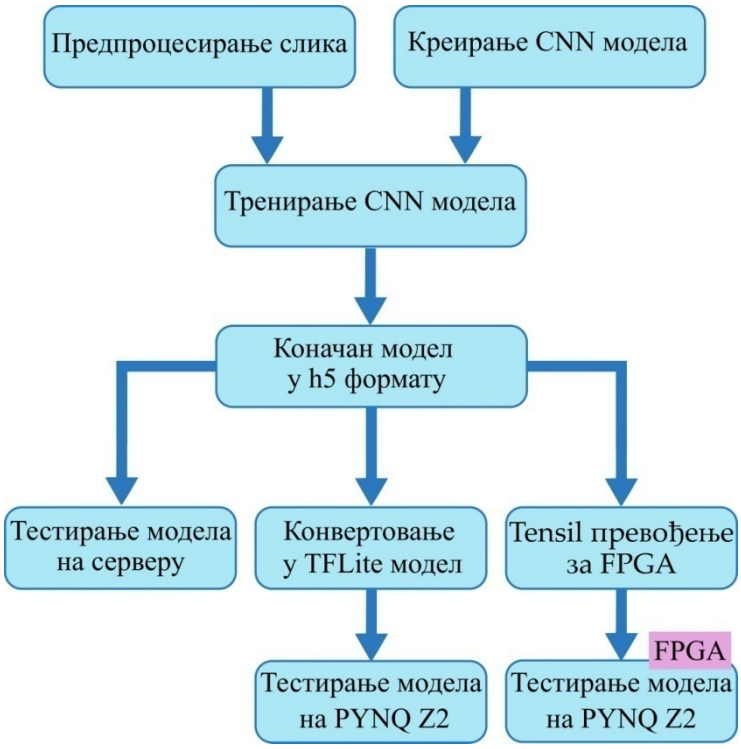
Први члан у изразу (28), wv(t), представља инерцију, а w је параметар тежине инерције. Ако је параметар w једнак `1`, честица иде у истом правцу и има брзину као претходно кретање. Ако w има вредност $0 \leq w < 1$, смањује се утицај претходне брзине и добија се могућност да честица уђе у други део домена претраживања. Други члан у изразу (28) представља лични утицај честице где је Pbest позиција ове честице за која има најбољу вредност добијену из коришћене функције. Трећи члан садржи Gbest и представља утицај глобално најбоље позиције узимајући у обзир све честице. Након израчунавања PSO брзине, нова вредност за локацију је дата у изразу (29) унутар области за претрагу.

3. КОРИШЋЕНИ РЕСУРСИ И ТЕХНИКЕ

У поглављу Коришћени ресурси и техинике представљене су ресурси, алати и технике који су коришћени као подршка у реализацији показних примера и спровођењу потребних тестова.

3.1. Поступак реализације модела за класификацију

Целокупно окружење задужено за креирање, тренирање, а на крају и тестирање модела класификације, постављено је на серверу. Добијени готови модели су затим превођени у оптимизоване моделе за рад на уређајима са ограниченим ресурсима. Цео процес припреме CNN модела приказан је на слици 16. Прво су слике, пре учитавања, обрађене како би одговарале форматима намењеним за оптимизоване моделе. Креиран је одређени тип CNN модела који одговара ResNet20 и дефинисани су сви потребни параметри како би се процес тренирања одвијао на ефикасан начин. Након постизања задовољавајуће тачности, дати CNN модел се чува у h5 формату као коначна верзија. Добијени модел се може тестирати на серверу уз помоћ тест података који нису коришћени приликом тренирања модела.



Слика 16. Блок шема поступка за реализацију CNN модела за класификацију

Добијени модел у h5 формату се учитава и врши се његова конверзија у TensorFlow Lite лите модел. Такође, исти модел је конвертован уз помоћ Tensil алата у формат модела који би био намењен за извршавање на FPGA. Ови нови облици модела, као и подаци о тестовима се преносе на PYNQ Z2, где се покрећу процеси класификације.

3.2. TensorFlow подршка

Креирање и тренирање модела вештачких неуронских мрежа може се обавити коришћењем различитих софтверских платформи или колекција које обезбеђују потребне алате и библиотеке. Могу се навести неколико примера софтверских колекције за тренирање вештачких неуронских мрежа:

1. TensorFlow је развијен од стране Google истраживача. TensorFlow је софтверска колекција отвореног кода и нуди флексибилну платформу за изградњу и тренирање вештачких неуронских мрежа, поготову за мреже дубоког учења. Пружа апликационе програмске интерфејсе (Application Programming Interfaces - APIs) високог нивоа за ефикасну изградњу модела. Такође нуди и APIs ниског нивоа који омогућавају креирање високо конфигурабилних модела са посебно прецизном контролом [124].

2. PyTorch представља софтверску колекцију отвореног кода за машинско учење засновану на Python програмском језику и Torch библиотеци. PyTorch је популарна варијанта софтверска платформе за дубоко учење, која нуди динамичко израчунавање графова и интерфејс више налик Python окружењу у поређењу са TensorFlow. Има широку примену у истраживањима у оквиру науке о подацима и брзој изради прототипова модела вештачких неуронских мрежа [125].

3. Keras је API за неуронске мреже високог нивоа написан у Python програмском језику који може да ради над плаформама као што су TensorFlow, Theano или Microsoft Cognitive Toolkit. Пружа подршку корисницима да реализују комплетне моделе дубоког учења веома једноставно са минималним бројем линија кода. Такође је погодан избор јер пружа корисницима велику флексибилност, задржавајући при томе API својства високог нивоа. Најшире прихваћено коришћење Keras библиотеке је са TensorFlow платформом у позадини где се код написан у Keras преводи у TensorFlow варијанту [126].

4. Caffe (Convolutional Architecture for Fast Feature Embedding) представља још једну варијанту платформе отвореног кода за дубоко учење који је развио Berkeley Vision and Learning Center. Caffe је погодан за истраживачку употребу због своје модуларности и раздвајања дефиниције мреже од стварне имплементације. Caffe је познат по својој брзини и ефикасности што омогућава једноставну реализацију конволуцијских неуронских мрежа. Обично се користи за задатке рачунарске визије због своје ефикасности у процесирању слика [127].

За реализацију модела и њихово тренирање коришћен је TensorFlow, уз Keras подршку високог нивоа, верзија 2.12.0. TensorFlow пружа свеобухватан екосистем за изградњу и примену модела машинског учења, посебно модела дубоког учења [124]. Флексибилност је једна од карактеристика TensorFlow 2.12.0 јер подржава APIs високог нивоа (као што је Keras 2.12.0) за брзо прављење прототипа, али и APIs ниског нивоа за остваривање додатних прилагођавања модела. Погодан је за мале моделе на персоналним уређајима као и за велике моделе на дистрибуираним рачунарским

системима. TensorFlow 2.12.0 нуди широк спектар унапред изграђених модела, алата и библиотека за различите задатке машинског учења. Има снажну подршку заједнице са обимном документацијом, упутствима и ресурсима [128].

Дубоко учење на Edge уређајима подразумева примену модела машинског учења директно на уређајима као што су паметни телефони, IoT уређаји и уграђени системи. Edge уређаји имају ограничења у меморији, процесорској снази и трајању батерије. Модели морају бити оптимизовани за очекивану величину и ефикасност.

Модели за уређаје са ограниченим ресурсима могу се припремити компресијом готових тренираних модела са техникама квантизације и ограничавања вредности, да би се добила мања величина модела. Квантизација након тренирања се користи за претварање параметара модела из 32-битног покретног зареза у нижу прецизност (нпр. 8-битне целе бројеве), смањујући величину модела и побољшавајући брзину закључивања. А у другој варијанти, током тренирања модела користе се квантизоване вредности тако да се добија модел мање величине уз боље очување тачности у односу на прву варијанту.

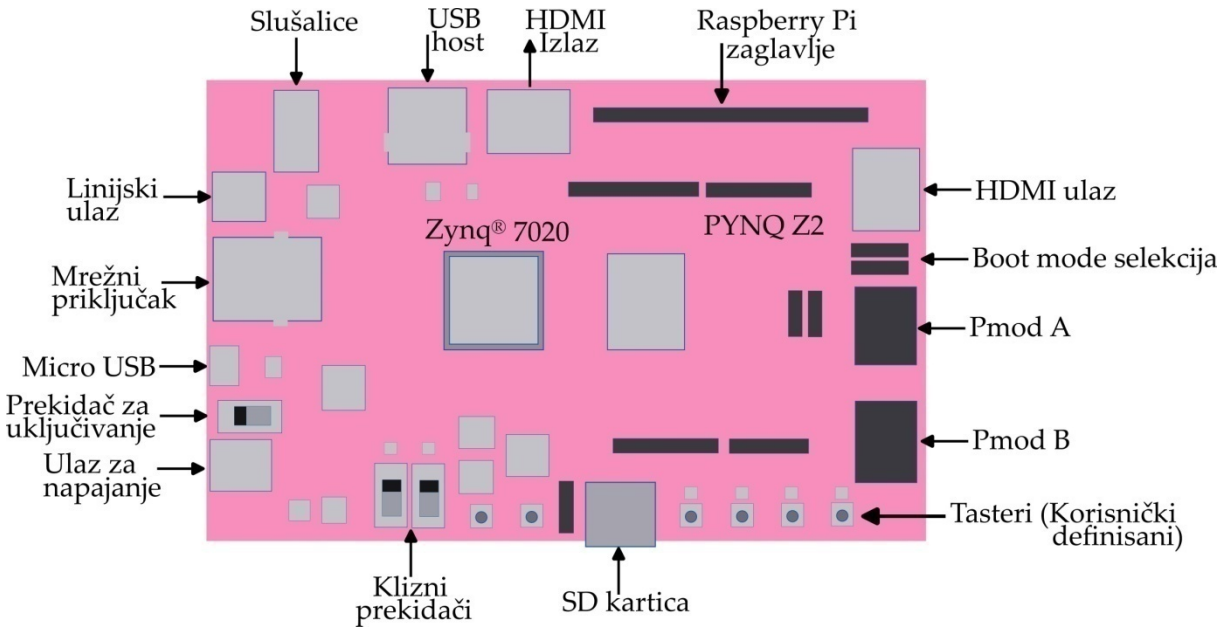
TensorFlow Lite представља скуп алата који се може користити за конверзију и оптимизацију DL модела, као што су TensorFlow модели, за мобилне и Edge уређаје. TensorFlow Lite управо омогућава извршавање тренираних модела машинског учења високог Cloud нивоа на мобилним уређајима, уграђеним системима и IoT уређајима. Након конверзије постојећег модела добија се оптимизована верзија модела у верзији фајла са .tflite екстензијом. Овакве верзије модела намењене су за ефикасно извршавање на уређајима са ограниченим ресурсима при чему се не захтева стална веза са сервером или конекција са Интернетом. Може да ради на многим платформама као што су Android, iOS и уграђени Linux уређаји [129].

3.3. Хардверска подршка – Систем на чипу

PYNQ Z2 је развојна плоча која комбинује снагу Xilinx Zynq-7000 SoC (System on Chip) са флексибилношћу Python програмирања. Пројектована је да омогући брзу израду прототипова, развој уграђених система и одређеног дигиталног пројекта. Плоча садржи Xilinx Zynq-7000 SoC (систем на чипу), који укључује двојезгарни ARM Cortex-A9 процесор и FPGA програмабилну логику. Ова комбинација омогућава и софтверску обраду високог нивоа и прилагодљиву хардверску подршку за убрзање обраде.

PYNQ Z2 плоча нуди низ карактеристика које је чине погодном за различите примене. Има 512MB DDR3 меморије, што даје довољно меморијског простора за покретање сложених алгоритама и складиштење података. Плоча такође укључује 16MB Quad-SPI флеш меморије за складиштење датотека за покретање и других битних података.

У погледу повезивања, PYNQ Z2 нуди низ опција. Има HDMI улазне и излазне портове, што омогућава лаку интеграцију са екранима. Такође укључује USB портове, Ethernet и Wi-Fi могућности конекције, омогућавајући комуникацију са другим уређајима и мрежама (Слика 17). Поред тога, плоча има Arduino и Raspberry Pi заглавља, обезбеђујући компатибилност са широким спектром модула за проширење [130].



Слика 17. PYNQ Z2 платформа

Једна од важних карактеристика PYNQ Z2 је његова подршка за PYNQ (Python Productivity for Zynq) пројекат. Овај пројекат омогућава програмерима да искористе моћ Python програмског језика и флексибилност FPGA како би убрзали своје апликације. Уз PYNQ, могуће је једноставније програмирати FPGA користећи Python библиотеке и искористити предности хардверског убрзања за рачунарски интензивније задатке. Његове карактеристике чине га погодним за широк спектар примена, од развоја уграђених система до израде прототипа дигиталног пројекта. ARM језгра и FPGA могу да комуницирају једни са другима преко интерконекције високог пропусног опсега, омогућавајући ефикасну размену података и сарадњу између софтверских и хардверских компоненти система [131]

Zynq-7000 SoC има двојезгарни ARM Cortex-A9 процесор, који пружа могућности обраде високих перформанси за покретање оперативних система, апликација и софтверских алгоритама. Cortex-A9 језгра су способна да раде на брзинама до 1 GHz, омогућавајући ефикасно извршавање сложених задатака.

Поред ARM језгара, Zynq-7000 SoC (систем на чипу) укључује програмабилну логику у облику FPGA. FPGA део омогућава имплементацију прилагођених хардверских акцелератора, пружајући велику брзину и паралелну обраду за рачунарски интензивне задатке.

Xilinx Zynq-7000 SoC нуди снажну комбинацију могућности обраде, програмабилне логике и периферних уређаја на чипу. Његове карактеристике га чине погодним за апликације које захтевају равнотежу између софтверске обраде високог нивоа и прилагодљивог хардверског убрзања, као што су уграђени системи, роботика, индустријска аутоматизација и рачунарство високих перформанси.

Програмабилна логика у PYNQ-Z2 омогућава имплементацију хардверских акцелератора прилагођених специфичним задацима Fog рачунарства. Ово може значајно побољшати перформансе и ефикасност обраде података, посебно за рачунарски интензивна оптерећења.

Пројектовање дела система који би се извршио на FPGA, односно у оквиру програмибилне логике код PYNQ-Z2, може се релизовати помоћу Xilinx Vivado.

Укратко, Xilinx Vivado представља свеобухватно развојно окружење и скуп алата посебно дизајниран за пројектовање и оптимизацију Xilinx FPGA. Xilinx Vivado користи RTL дизајн, што значи да подржава језике за опис хардвера као што су Verilog, VHDL i SistemVerilog за спецификацију дигиталних кола. Такође, синтеза високог нивоа (HLS) би се могла користити за претварање C/C++ кода у RTL за имплементацију FPGA. Xilinx Vivado има могућност покретања симулација које врше функционалну верификацију дизајна FPGA пре синтезе. Обезбеђује да дизајн испуњава временске захтеве и ограничења, и пружа могућности отклањања грешака за идентификацију и решавање проблема у дизајну FPGA. Омогућава пројектантима да прилагоде FPGA дизајн на ниском нивоу, оптимизујући перформансе за специфичне апликације и олакшава интеграцију у сложене системе кроз свеобухватни скуп алата и библиотеку IP компоненти.

Xilinx Vivado може се користити за дизајнирање прилагођених хардверских акцелератора за специфичне алгоритме или апликације, укључујући и закључивање дубоког учења. Xilinx Vivado садржи библиотеку унапред дизајнираних језгара интелектуалне својине (IP) за уобичајене функције, поједностављујући поновну употребу дизајна и убрзавајући развој [132].

3.4. Tensil AI

Tensil AI представља компајлер за моделе машинског учења и хардверски генератор посредством кога је могуће креирати прилагођени акцелератор за извођење закључака на основу претходно тренираног ML модела. Tensil се може схватити као скуп алата који пружају подршку за покретање модела машинског учења на прилагођеној архитектури акцелератора. Што значи да омогућава развој прилагођених акцелератора, компајлирање већ формираних ML модела и на крају пренос и покретање добијеног модела. Tensil садржи RTL генератор, компајлер модела и скупове драјвера.

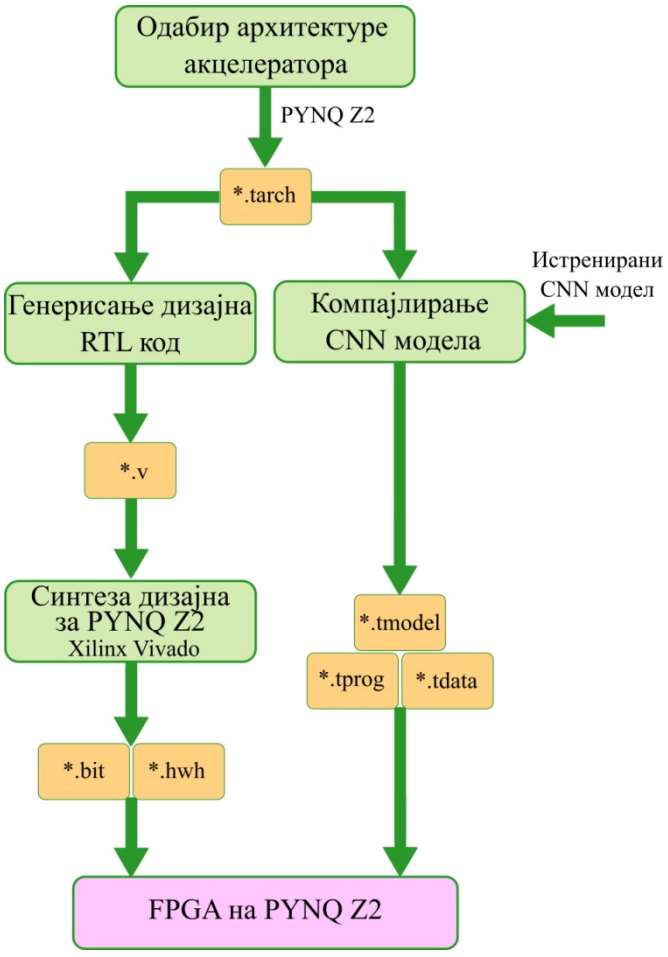
Tensil на одређени начин аутоматизује пројектовање хардверских акцелератора машинског учења. Помоћу Tensil алата свима је доступна могућност да убрзају своја радна окружења. Прва развојна усмерења су била на подршку за конволуцијске неуронске мреже и њихове примене на FPGA платформама у оквиру Edge рачунарства.

Када постоји потреба да се умањи оптерећење обраде података са постојећих рачунарских ресурса то се може остварити управо хардверским акцелератором добијеним уз помоћ Tensil подршке. Према томе Tensil се може одабрати као погодно средство када постоји готов CNN модел и потребно га је прилагодити за уређаје на крајевима мреже. То значи да се већ реализовани ML модел може компајлирати и покренути као хардверски акцелератор користећи Tensil, без додатних квантизација параметара и деградација. Постојећи ML модел који је истриниран у одговарајућем радном окружењу, потребно је прво претворити у ONNX формат. ONNX је скраћеница од Open Neural Network Exchange и дефинисан је као посебан формат за представљање модела машинског учења. У случају TensorFlow модела исти се може конвертовати у TensorFlow frozen graph верзију. Након тога врши се компајлирање модела узимајући у обзир одговарајућу архитектуру и добијају се три главна фајла који имају следеће екстензије: tmodel, tdata и tprog.

Први фајл је манифест са екстензијом tmodel који представља обичан текстуални JSON опис компајлираног модела и садржи информације које указују драјверу, приликом примене акцелератора, како да подеси улазе и излазе.

Затим Tensil програм са екстензијом tprog представља извршну датотеку и садржи инструкције које акцелератор узима у обзир и извршава. И фајл са подацима о тежинама модела, са екстензијом tdata, који су у бинарном формату и користе се током извршавања. Потребно је реализовати и RTL фајл акцелератора користећи неки од алата за пројектовање хардвера као што је Vivado Xilinx. Као крајњи резултат за пројектовани акцелератор генерише се бит фајл (*.bit) као и фајл са информацијама о хардверу (*.hwh). Када се добију сви потребни елементи може се покренути компајлирани модел на систему са припремљеним хардверским акцелератором (Слика 18).

Приликом примене акцелератора, драјвер се користи да преузме опис архитектуре и компајлирани модел како би започео интеракцију са компонентама система и пренео модел у хардвер. Драјвер се користи за постављање улаза и излаза као и управљање многим другим ресурсима важним за одабрану хардверску платформу [133].



Слика 18. Примена Tensil алата за добијање прилагођеног акцелератора

Tensil AI акцелератор за закључивање поседује квантификоване вредности у односу на првобитни тренирани модел, тако да се том операцијом може умањити прецизност коришћеног модела. Али са друге стране мањи број параметара омогућава њихову лакшу примену и складиштење. Примену Tensil AI акцелератора карактерише мало кашњење, велики пропусни опсег и обрада према којима могу бити право решење за развој апликација које захтевају високе перформансе. Наведена технологија се може

користити не само код апликација које се извршавају на Cloud окружењу већ и на ивичним (Edge) уређајима као што су IoT уређаји, паметни телефони или други крајњи уређаји. Tensil акцелератор као моћан алат отвореног кода може се ефикасно користити за убрзавање закључивања на основу модела машинског учења на платформама које се заснивају на FPGA [134].

FPGA концепт има своје место у реализацији система за убрзање рачунских операција у многим областима, захваљујући својим перформансама, енергетској ефикасности и флексибилности. Са друге стране FPGA можда није широко прихваћен код рачунарства високих перформанси сходно пажњи коју заслужује због сложености њиховог програмирања и проблемима у оптимизацији перформанси. Да би се превазишле наведене потешкоће може се користити Tensil акцелератор који је представљен у раду [134] где су показане перформансе ResNet20 модела преведеног и покренутог на FPGA колу. У наведеном раду представљен је унапређени пројекат хардвера, употребом компоненте Xilinx Ultra RAM. Док се применом напредних стратегија компајлера добијају боље перформансе закључивања акцелератора, при чему се издваја и побољшање у погледу енергетске ефикасности.

Још једна примена концепта конверзије CNN модела посредством Tensil алата за примену у оквиру Edge рачунарства, који карактеришу ограничени ресурси, приказана је у раду [135]. Реализовани су акцелератори на мобилним FPGA системима и фокус је био на оптимизацији RTL пројекта како би се побољшала брзина обраде и енергетска ефикасност. Реализација хардверског акцелератора за претренирани модел ResNet20 на CIFAR10 скупу података остварена је на PYNQ Z2 платформи, која је коришћена као FPGA систем на чипу. Показано је да се након оптимизације добијају још боље перформансе и потврђују FPGA као средство за обраду података у реалном времену.

У оквиру Edge рачунарства обично се подразумева окружење са ограниченим ресурсима, као што су мобилни уређаји или IoT уређаји. Примена акцелератора који се заснивају на CNN моделима представља све погоднije и привлачније решење због њихове скалабилности и високе тачности. Оптимизивони акцелератор на FPGA и његова примена представљен је у раду [136] при чему је коришћен Xilinx PYNQ-Z1 платформа за реализацију апликације базирана на CNN моделу за детекцију објеката. Такође, је коришћен Tensil скуп алата за компајлирање модела и генерисање RTL кода за пројекат основног хардвера који се односи на одабрану архитектуру. Након тога у представљеном примеру извршена је оптимизација хардвера како би се добио акцелератор са побољшаном брзином обраде и смањеном потрошњом енергије.

Примена система заснованих на хардверским акцелераторима који су развијени уз помоћ Tensil алата има своју значај и у области медицине. У раду [137] представљен је систем који указује на нови правац подршке радиолозима у процесу дијагностике рака дојке помоћу дигиталног мамографа. Пошто је примена алгоритама дубоког учења већ у истој намени показивала одговарајућу тачност, до проблема долази када се очекује њихова примена у реалном времену, где долазе до изражаја високи захтеви по питању рачунарских ресурса, коришћене снаге и брзине обраде. У ту сврху предложена је неуронска мрежа са смањеним бројем параметара, која је квантизована и након тога се убрзава тако што је извршена њена припрема за извршавање на FPGA. На тај начин добијен је модел којим се постиже већа брзина детекције и минимизује потрошња енергије уз очување тачности.

Примена FPGA у области биомедицинског рачунарства за постизање високих перформанси приказана је у раду [138]. Наведена студија обухвата архитектуре неуронских мрежа као што је CNN мрежа, која је примењена као средство за

побољшање анализе EKG сигнала преко FPGA. У наведеном истраживању показано је да се применом Tensil алата, уз одабир одговарајуће структуре хардвера може искористити FPGA на ефикасан начин како би се остварили перформансе на завидном нивоу и одлична тачност. Управо је представљен акцелератор са конволуцијским моделом машинског учења добијен помоћу Tensil алата намењен за PYNQ Z1 платформу како би се убрзало ML закључивање преко FPGA.

У раду [139] спроведен је пројекат система за сегментацију слика, који се користи за детекцију пожара у реалном времену, користећи FPGA и хардверски акцелератор машинског учења добијен посредством алата Tensil. Представљени пројекат је имплементиран на PYNQ Z2 платформи заснованој на Xilinx Zynq 7000 FPGA где је вршена његова процена. Између више пројектованих модела одабран је један са најбољим перформансама, са временом одзива од 0,63 s и потрошњом енергије од 2,2 W.

3.5. Подршка за моделовање Cloud–Fog структура (iFogSim)

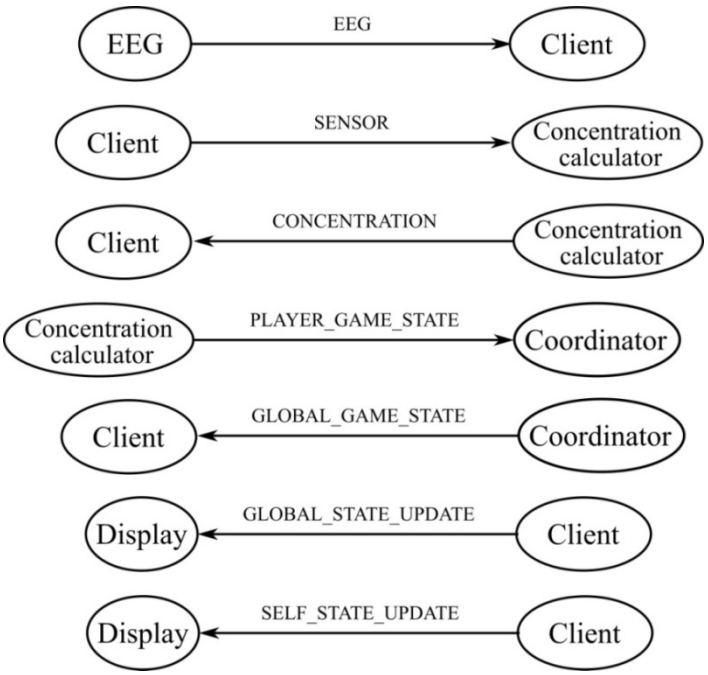
Структура Cloud–Fog рачунарства са придруженим апликацијама може се моделовати и симулирати коришћењем iFogSim програмског пакета. IFogSim представља проширење постојећег CloudSim [140] пројекта коришћеног за системе Cloud рачунарства и реализован је исто на програмском језику Java. При томе се користе основне компоненте CloudSim пројекта уз допунске пакете који могу помоћи да се дефинише окружења Fog рачунарства [141]. IFogSim се може користити као алат од стране истраживача да верификују своје концепте у оквиру Fog рачунарства за различите системске конфигурације и корисничке апликације. Оваква могућност моделовања система има посебан значај у случају када корисници у датом моменту не поседују целокупну жељену инфраструктуру.

Конфигурација система подразумева дефинисање свих уређаја, њихових међусобних веза као и свих пратећих параметара који су карактеристични за одабрану Cloud–Fog структуру. Након тога дефинишу се корисничке апликације које ће се извршавати над одабраном конфигурацијом. Апликација се може сагледати као скуп модула који врше обраду података, при чему се резултати из једног модула као излазни подаци могу проследити другом модулу, остварујући тако њихову међусобну повезаност. Апликације представљене на дати начин за поставку у Фог окружењу заснивају се на приступу дистрибуираног тока података (DDF - Distributed Dataflow) [142].

Овако издељени модули би се могли дистрибуирати у Cloud–Fog структури и извршавати на различитим нивоима. Неки апликациони модули могу бити намењени за крајње уређаје, други само за извршење на Cloud нивоу, док преостали имају могућност да буду постављени на Fog уређаје. Извршавање апликације за различите дистрибуције својих модула у Cloud–Fog структури може довести до различитих вредности кашњења резултата, оптерећења мреже и потрошње енергије.

Један од изазова је пронаћи оптимална решења у поставци апликационих модула како би се задовољили захтеви за одговарајућим перформансама апликације. Многи кориснички сценарији и модели оптимизације намењени за остваривање најповољније поставке модула, могу се ефикасно проверити уз помоћ iFogSim програма. Модели оптимизације имају за циљ проналажење погодног решења за припрему и извршавање корисничких апликација у дефинисаној Cloud–Fog архитектури. Прихватљива решења

могу се изразити кроз минималне вредности за кашњење, оптерећење мреже или потрошњу енергије. Конципирани модел Cloud–Fog структуре могао би да се користи у комбинацији са више апликација, како би се кроз симулацију и тестирање добили резултати понашања једног таквог система. У ту сврху и за валидацију предложеног модела коришћене су апликације које су већ део iFogSim пројекта, а које су сада означене са App-V1 и App-D0 ради лакшег референцирања. Ове две врсте апликација су одабране за евалуацију због својих познатих карактеристика и већ спроведених бројних истраживања где су коришћене у многим сценаријима.



Слика 19. Модули апликације App-V1

Апликација са ознаком App-V1 представља игру између људи засновану на обради EEG сигнала у реалном времеу и односи се на апликацију са критичним кашњењем [143]. При томе EEG (Electroencephalography) представља ознаку за електроенцефалографију, методу која мери електрично поље односно елетричну активност мозга. Учесници у игри користе носиве EEG слушалице са сензором које су повезане са паметним телефоном на коме се извршава део апликације. Резултати и остали учесници у игри се приказују на дисплеју апликације. Циљ игре је да сваки учесник привуче одговарајући предмет при чему би привлачна сила била пропорционална нивоу концентрације. Ова вредност би била процењена на основу карактеристика EEG сигнала који се очитава за сваког учесника у игри. На основу датог описа апликације App-V1 може се закључити да је изразито битан захтев добијање прорачуна у реалном времену, јер свако додатно кашњење утиче на исправност игре.

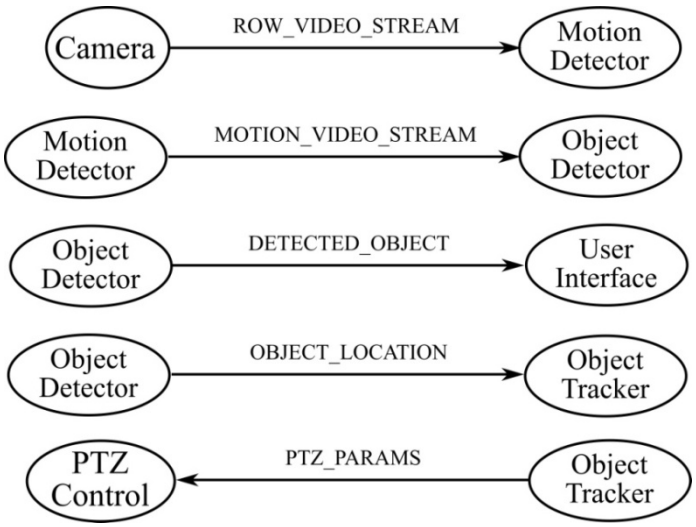
Апликација App-V1 се састоји од три модула који су наведени са првобитним називима као што су коришћени у примерима iFogSim пројекта, а то су: Client, Concentration Calculator и Coordinator. Улазни сигнали у апликацију се уводе преко сензора EEG, а резулат се приказује на екрану означеном као актуатор DISPLAY.

На слици 19 су представљени парови модула који су међусобно условљени подацима који један другоме преносе сходно дефинисаној апликацији. Наведена зависност је представљена линијом са усмереном стрелицом изнад којих је наведен дефинисани атрибут са којим се указује на податак који се преноси.

Модул Client је повезан са сензором и добија директно EEG сигнал. Након тога шаље вредност сигнала до модула Concentration Calculator како би се добио ниво концентрације из сигнала за датог учесника. Модул Concentration Calculator се користи да утврди стање учесника на основу добијених вредности сигнала, односно да процени ниво концентрације. Обавештава модул Client о новој вредности како би се стање учесника ажурирало на дисплеју. Према томе, када добије вредност концентрације врши приказ за корисника слањем вредности на DISPLAY. Модул Coordinator има своју улогу на глобалном нивоу, тако што континуирано шаље опште тренутно стање у игри до Client модула свих повезаних уређаја.

Апликација са ознаком App-D0 представља интелигентни надзорни систем са мрежом дистрибуираних камера. Анализа података генерисаних од стране камера на централизован начин није увек погодна због великог обима података, које је потребно слати до централних рачунарских платформи. У том случају би дошло до већег кашњења али и до великог заузимања доступног комуникационог пропусног опсега. Према томе децентрализовано процесирање видео сигнала би била много пожељнија метода анализе података код сличних надзорних система. Пример оваквог типа апликације који је дат у iFogSim пројекту реализован је на основу рада објављеном у [144]. Апликација App-D0 састоји се од пет главних модула који се баве процесирањем података, а то су: Motion Detector, Object Detector, Object Tracker, PTZ Control и User Interface.

Модули апликације App-D0 су представљени на слици 20 са њиховим међусобним везама означеним усмереним линијама као показатељима преноса података.



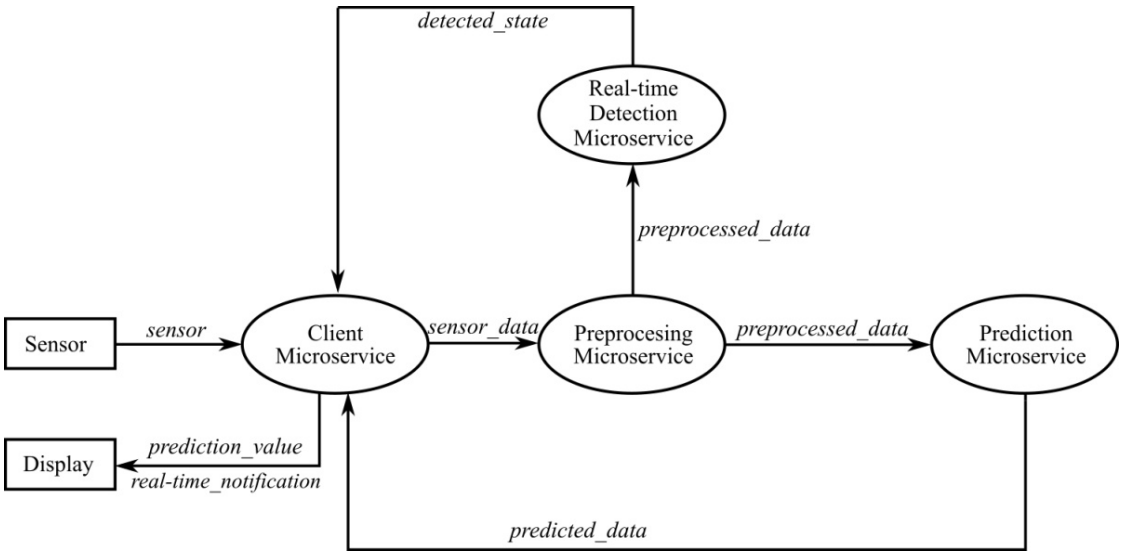
Слика 20. Модули апликације App-D0

Motion Detector представља модул који је уграђен у саму паметну камеру који очитава директно видео стрим добијен од камере како би открио кретање објеката. У случају детекције кретања видео стрим се прослеђује даље до модула Object Detector за даљу обраду. Модул Object Detector добија видео стрим у коме је откривено кретање, издваја покретни објекат и пореди га са претходно откривеним објектима који су активни. У случају да детектовани објекат није претходно откривен, активирано је праћење тог објекта. Такође врши нови прорачун координата за објекте. Модул Object Tracker прихвата последње прорачунате координате праћених објеката и порачунава оптималне конфигурационе параметре за камере како би се објекат могао уснимити на

најбољи могући начин. Ове информације се периодично прослеђују до модула PTZ Control. Модул PTZ Control се извршава на свакој паметној камери и врши подешавање камере на основу добијених информација од модула Object Tracker. User Interface модул је представљен тако да има део видео стримова који садрже сваки праћени објекат и који се прослеђује до корисничких уређаја.

Поред наведене две апликације приликом тестирања предложеног модела коришћена је и трећа апликација која је саставни део нове верзије iFogSim означене са iFogSim2 [145]. У новој верзији претходни пројекат је проширен са моделима који су пружили подршку за мобилност односно миграцију сервиса, дистрибуирано креирање кластера и управљање микросервисима апликације.

Представљена апликација са ознаком App-SF0 садржи микросервисну архитектуру и приказана је као DDF модел на слици 21. Ова апликација се састоји од четири микросервиса: Client Microservice, Preprocessing Microservice, Real-time Detection Microservice и Prediction Microservice. У овом случају, модули апликације могу се сматрати сличним микросервисима, где свако појединачно покретање дефинише обраду података независно од других. Линије на графу представљају зависности између микросервиса. Дакле, модел апликације представља микросервисну архитектуру која формира модуларну структуру и дели читаве задатке у засебне јединице за обраду података. У овој апликацији, одређени задаци треба да буду распоређени на ресурсе Fog уређаја и да обезбеде резултате у реалном времену, док се неки од микросервиса могу поставити у Cloud окружење и дати резултате толерантне на кашњење.



Слика 21. Модел апликације App-SF0

Микросервис клијент (Client) постављен је на паметним телефонима корисника и повезан је бежичним путем са сензором од кога стиже EKG сигнал. У оквиру микросервиса предпроцесирање (Preprocessing) врши се чишћење података и отклањање аномалија података. Микросервис за хитну дијагнозу (Real-time Detection) има значајну улогу у идентификацији унапред дефинисаних услова и слању сигнала упозорења назад клијентском микросервису (Client) да активира хитно обавештење. Овај апликативни модул треба да изврши анализу у реалном времену, а уколико резултати укажу на лоше услове, добијени сигнал се мора одмах послати корисницима. Микросервис предвиђања (Prediction) има за циљ да складишти и анализира

прикупљене податке, користећи технике машинског учења, да предвиди будућа стања или услове. Главни детаљи конфигурације се такође користе из iFogSim пројекта у сврхе тестирања, као што су карактеристике дефинисане у раду [145].

Први део апликације App-SF0 треба да се извршава у реалном времену у близини извора података, а други део је намењен за рад на Cloud окружењу за прикупљање података и извршавање модела неуронских мрежа или машинског учења. Разлика у односу на прве две апликације, поред микросервисне архитектуре сличне модулима, је и могућност коришћења кластера Fog уређаја.

4. РЕЗУЛТАТИ И ДИСКУСИЈА

У поглављу Резултати и дискусија представљени су сви видови излазних резултата током спровођења истраживања. Представљен је целокупни систем за тренирање и тестирање CNN модела помоћу кога се врши класификација слика. Затим представљена је конверзија модела у варијанте намењене за извршавање на нивоу Фог рачунарства, где спада извршавање на хардверу односно на FPGA колу. У другом делу представљено је моделовање Cloud–Fog структуре и процене стања у случају различитих конфигурација и промене оптерећења повећањем броја учесника у коришћењу апликација. При томе су представљени резултати тестова на спроведеним показним примерима.

4.1. Реализација апликације са моделима за класификацију слика у оквиру Cloud–Fog рачунарства

Апликације са својим главним делом који представља класификациони модел припремљени су се за покретање на серверу и на платформи у оквиру Fog рачунарства као што је PYNQ Z2. При томе су приликом њиховог извршавања праћење перформансе коришћеног система у погледу кашњења резултата обраде података, мрежног оптерећења и енергетске ефикасности.

4.1.1. Припрема података коришћених у предложеном моделу

Приказ предложеног модела и концепта дат је кроз три апликације у области паметне пољопривреде (Smart Farming). Ове три апликације односе се на класификацију слика за следеће скупове података:

- слике листова парадајза,
- слике инсеката,
- слике корова у кукурузу.

Све три апликације подразумевају анализу слике и заснивају се на наведена три скупа података који су коришћени у тренирању и тестирању DL модела. У питању су подаци преузети из Kaggle репозиторијума, а то су: слике болести листова парадајза [146], штетних инсеката [147] и корова у кукурузу [148]. Према томе ове апликације су одабране јер постоје адекватни скупови слика у свакој групи, а који су прикупљени и већ коришћени од стране других истраживача и могу се препознати као валидни скупови у припадајућој области.

Такође, наведене апликације су представљале занимљив приступ како би се препознало стање на терену у различитим ситуацијама и условима. На основу

препознавања тих услова корисници би могли да предузму одговарајуће акције у превенцији и заштити биљака. Добијање информација о тренутним условима у пољопривреди за ове примере може резултирати оптималном употребом пестицида или инсектицида. Исто тако дати скупови слика представљају погодно средство за доказивање концепта датог кроз модел за класификацију слика.

Слике листова парадајза су класификоване у 10 категорија у зависности од стања листова парадајза, које могу указивати на одређену болест парадајза или здраво стање [146]. Категорије у којима се може наћи лист парадајза су следеће: бактериозна пегавост листова, црна пегавост, пламењача, плеснивост листа парадајза, сива лисна пегавост, паукове гриње, лисне мрље, вирус жуте коврџавости листа, вирус мозаика парадајза и здраво стање. Све слике су претходно обрађене, а димензије слика које се користе у процесима тренирања и тестирања су промењене на 128 × 128 пиксела. Ово је прва апликација у предложеном систему и носи ознаку TL-01 (Tomato Leaf-01).

Друга апликација је дефинисана за обављање класификације на скупу слика штетних инсеката [147]. Коришћени скуп података садржи девет категорија слика: лисне ваши, јесењи црв, кукурузна совица, тврдокрилци, скакавци, комарци, гриње, лисне осе и мољци. Слике су претходно обрађене тако да су нове димензије слике 96 × 96 пиксела. Ова апликација садржи другу највећу величину улазних података односно слика и носи ознаку IP-02 (Insect Pests-02).

Трећа примена се односи на детекцију корова у кукурузу [148]. У овом случају, постоје само две категорије, један скуп слика који представља кукуруз без корова и други скуп слика који представља коров. Све слике су такође претходно обрађене тако да је нова димензија слика која се користи у процесу 64 × 64 пиксела. Трећа апликација која је коришћена носи ознаку CW-03 (Corn Weed-03). Од трећег скупа слика формиран је још један скуп, где су димензије слика 32 × 32 пиксела. То је апликација која има иста својства као претходна CW-03, али је формирана као четврта варијанта са умањеном величином слика. Њихова намена је била да се провери класификација слика на PINQ Z2 платформи и за тај најмањи формат. Апликација која користи скуп слика са наведеним редукованим форматом је додатна варијанта треће апликације и има ознаку CW-04.

4.1.2. Креирање и тренирање DL модела над одабраним скуповима слика

Одабрани скупови слика су прво пред-процесирани како би се уједначио формат свих слика односно да би се димензије слика промениле на планирану величину. Код првог скупа који се односи на слике листа парадајиза, било је потребно свести димензије слика са 256 x 256 пиксела на 128 x 128 пиксела. За то је коришћена PIL (Python Imaging Library) библиотека у оквиру Python окружења. PIL представља додатак Python окружењу отвореног кода који омогућава учитавање, модификовање и снимање слике у многим форматима. У наведеном случају свођење слика на мање димензије извршено је коришћењем LANCZOS филтера. На тај начин припремљене су слике које ће се користити за тренирање DL модела, а један део слика је одвојен како би се касније извршило тестирање коначног модела. Однос ових скупова је 80% за тренирање наспрам 20% за тестирање од укупног броја слика.

Читав процес припреме података, тренирања и тестирања модела спроведен је на серверу користећи Keras модуле над TensorFlow платформом. На почетку процеса тренирања DL модела учитане су слике из скупа за тренирање у низ слика. Односно

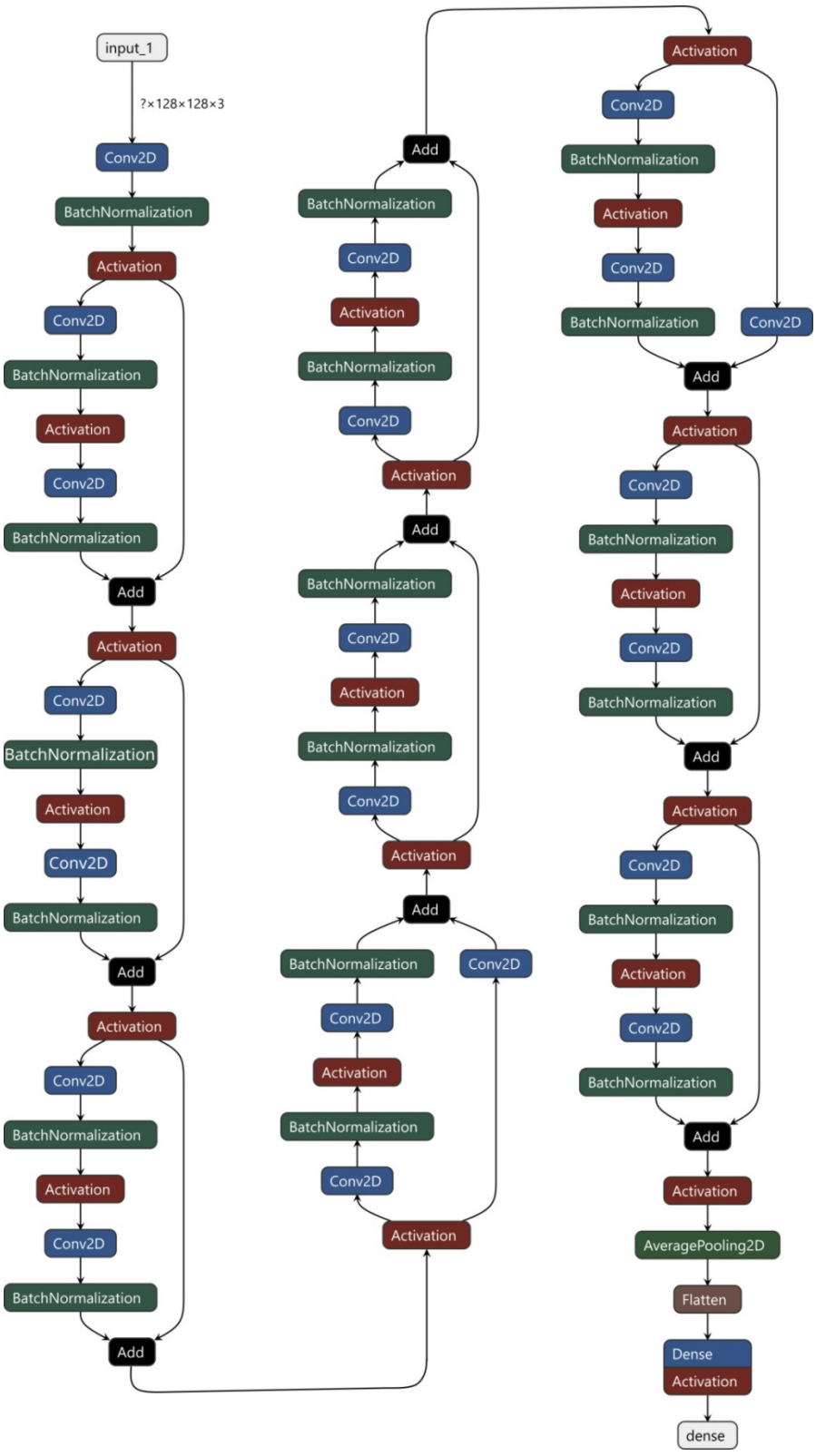
добија се вишедимензионални низ где сваки његов елемент у првој димензији представља једну учитану слику. Посматрано најдубље елементе низа они представљају вредности једног пиксела и записани су у форми три броја чије се целорбројне вредност крећу од 0 до 255. Након тога извршена је нормализација ових података тако што су све вредности подељене са 255. Овако добијене вредности низа крећу се у границама од 0 до 1 што смањује опсег и варирање података. Ово је веома важно јер помаже бољем тренирању DL модела и повећава се шанса за бржом ковергенцијом приликом тренирања. Такође се приликом учитавања слика формира други низ где се чувају ознаке за категорију којима слике припадају. Тако да се добија низ који представља стварни излаз система, што значи за сваку слику улазног низа исти елемент излазног низа садржи ознаку њене категорије.

Скуп слика предвиђен за тренирање, који је учитан у низ, као и низ категорија подељени су тако да се 20% од елемената оба низа користи за валидацију приликом тренирања DL модела.

Стопа учења је постављена на тај начин да се њена првобитна вредност умањује након одређеног броја епоха. Њена почетна вредност износи 0,001 која се након 80 епоха мења на 0,0001 док се на 120 епоха њена вредност мења на 0,00001. Оваква поставка параметара је са циљем да се у одмаклим деоницама тренирања користе мањи кораци како би се омогућило фино подешавање трениранога DL модела.

Креиран је DL модел који се ослања на конволуцијске слојеве али у форми резидуалне мреже, тако да је употребљена верзија модела ResNet20 који је описан у поглављу 2. При томе је одабран Adam оптимизациони алгоритам, а Тачност (Accuracy) модела је узета као метрика у процени трениранога модела. Такође, приликом дефинисања свих параметара потребних за тренирање модела одабрана је Categorical Cross-Entropy за функцију губитака која је такође описана у поглављу 2. Приликом тренирања модела на крају сваке епохе проверава се тачност модела и ако је она већа од претходно сачуване вредности онда се тренирани модел у том тренутку сматра најбољим и комплетно се снима у директоријуму у .h5 формату. У одмаклим фазама тестирања када је тачност модела дистигла високу и устаљену вредност, или на крају тестирања, одабира се последњи снимљени модел као коначна варијанта. Односно узима се модел који је имао најбољу тачност и након тога се спроводи тестирање модела над тестним скупом слика.

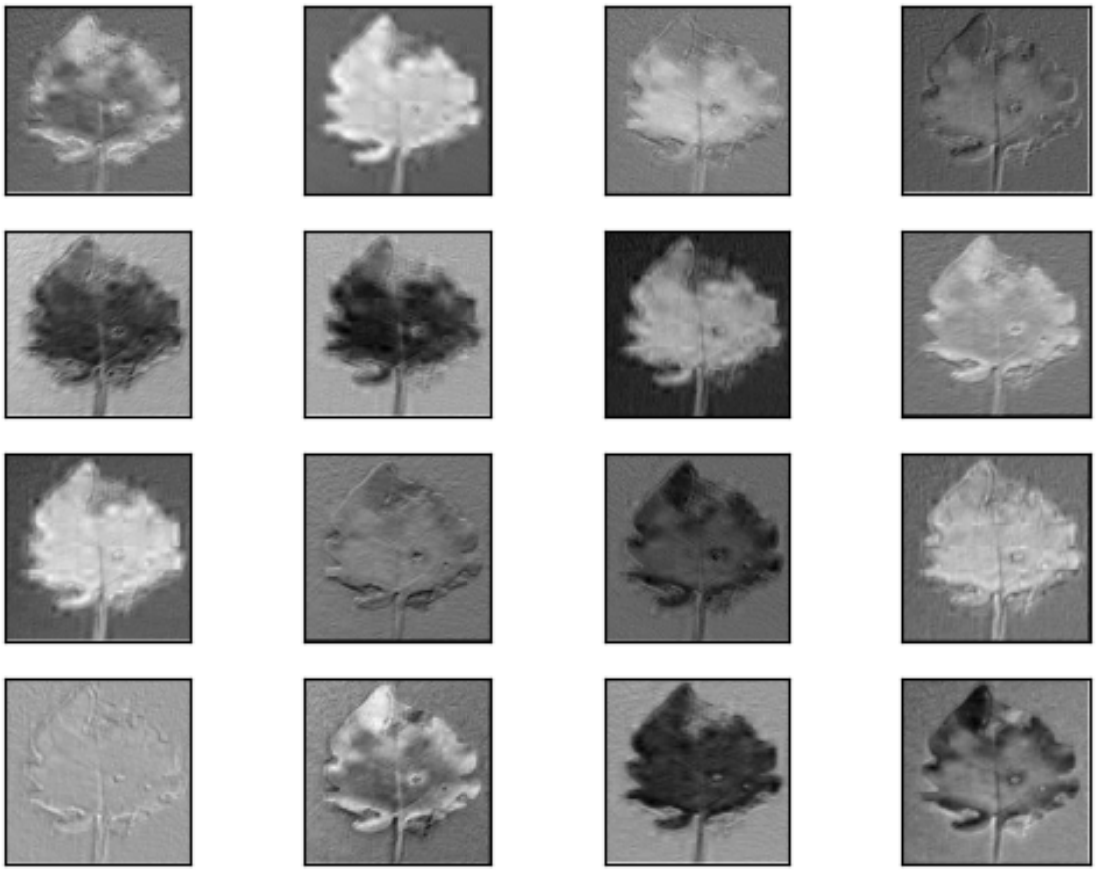
Графички приказ класификационог модела са његовим елементима и слојевима дат је на слици 22. Може се уочити величина улазних података, а то је овде слика као низ величине 128 x 128, где је основни елемент један пиксел. Представљени су резидуални блокови коришћеног ResNet модела и места где су дате активационе функције. У целој приказаној структури користи се ReLU активациона функција, осим на самом излазу где се користи Softmax активациона функција.



Слика 22. Блок дијаграм реализованог класификационог модела

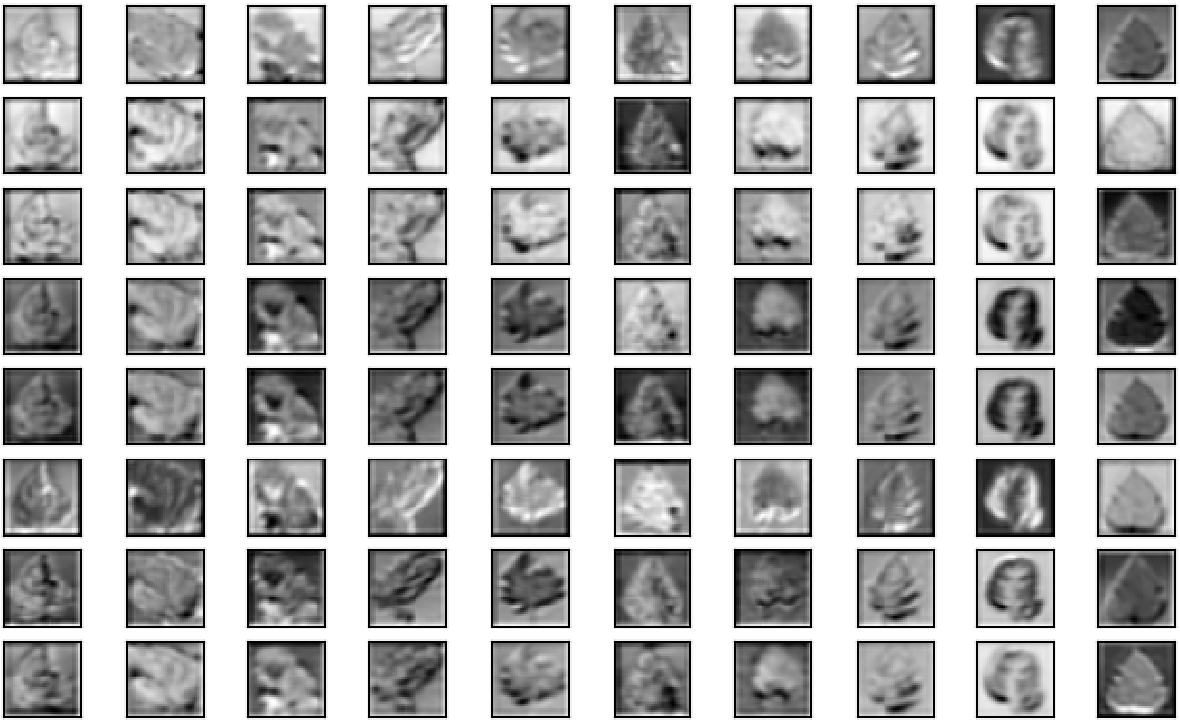
После конволуцијских слојева формирају се мапе карактеристика. На слици 23. може се видети изглед мапе карактеристика после првих слојева за лист парадајза у случају сиве лисне пегавости парадајза. Након проласка кроз све конволуцијске слојеве

добија се мапа карактеристика која се преводи у једнодимензонални низ и представља улаз у други део који чине потпуно повезани слојеви.



Слика 23. Приказ мапе карактеристика на излазу конволуцијског слоја код слике листа парадајза

Слично претходној слици, приказ само једног дела из мапе карактеристика у дубљим слојевима класификационог модела дат је на слици 24, при чему је узета у обзир само по једна слика из свих десет категорија. У питању су 64 карактеристике за сваку од извојених слика, али је приказано само њих осам ради могућности упоредног сагледавања. Дати приказ је само илустративног карактера да би се могле увидети разлике између ових слика, а које се у току класификације аутоматски спроводе до краја модела кроз слојеве који ће на крају дати одговарајућу категорију.



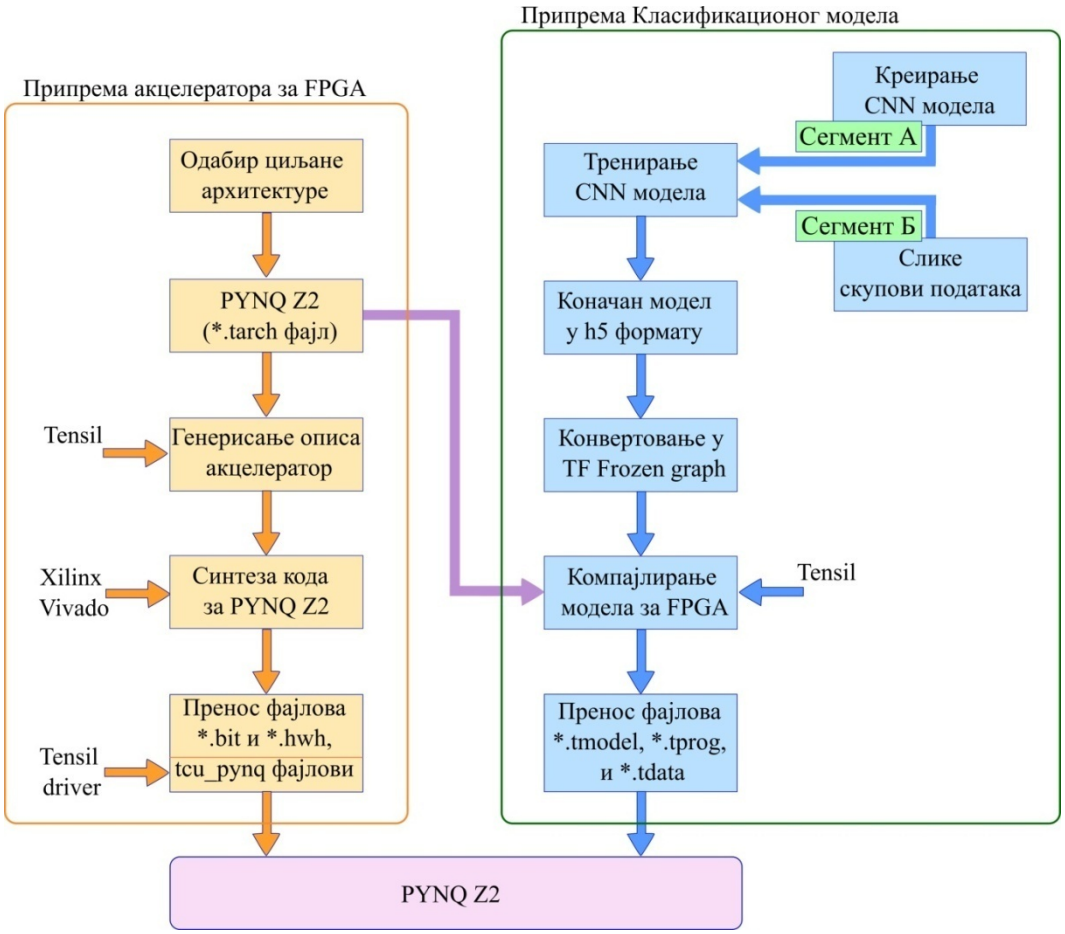
Слика 24. Издвојене мапе карактеристика за десет слика које припадају десет различитих категорија

На претходној слици је дато десет различитих слика. При томе, као што је описано у поглављу 2, за сваку слику на излазу се добија прво вредност вероватноће за сваку од постојећих категорија и затим се на крају одбира она са највећом вероватноћом.

4.1.3. Поступак припреме акцелератора

Цео процес припреме система за класификацију у виду акцелератора који се састоји из две гране приказан је на слици 25. У циљу поједностављења имплементације и примене модела, процес је подељен на поменуте две гране, чији се резултујући фајлови преносе на PYNQ Z2 платформу, где се на крају врши класификација слика. Прва грана подразумева избор архитектуре, која је у овом случају PYNQ Z2, затим генерисање дизајна акцелератора, његову припрему и синтезу уз помоћ Xilinx Vivado програма. Резултујући бит фајл и фајл драјвера се преносе и користе на PYNQ Z2. У случају избора наведене архитектуре, описане процедуре се спроводе једном и резултујуће датотеке се могу даље користити за различите DL моделе класификација, који се извршавају на истом хардверу.

Друга део није потпуно независан, али садржи све кораке за избор модела класификације и његову припрему за извршавање на FPGA. При томе се у другој грани врши избор модела и његово тренирања на серверу, а затим се врши његова конверзија и пренос на уређај са ограниченим ресурсима. Овако конвертован модел би имао намену да се изврши у виду акцелератора на FPGA. Зависност од првог дела огледа се у чињеници да се приликом компајлирања претходно тренираног модела, уз помоћ Tensil алата [133], узима у обзир одабрана архитектура, што се јасно види са слике 25.



Слика 25. Припрема акцелератора са моделом за класификацију слика за FPGA

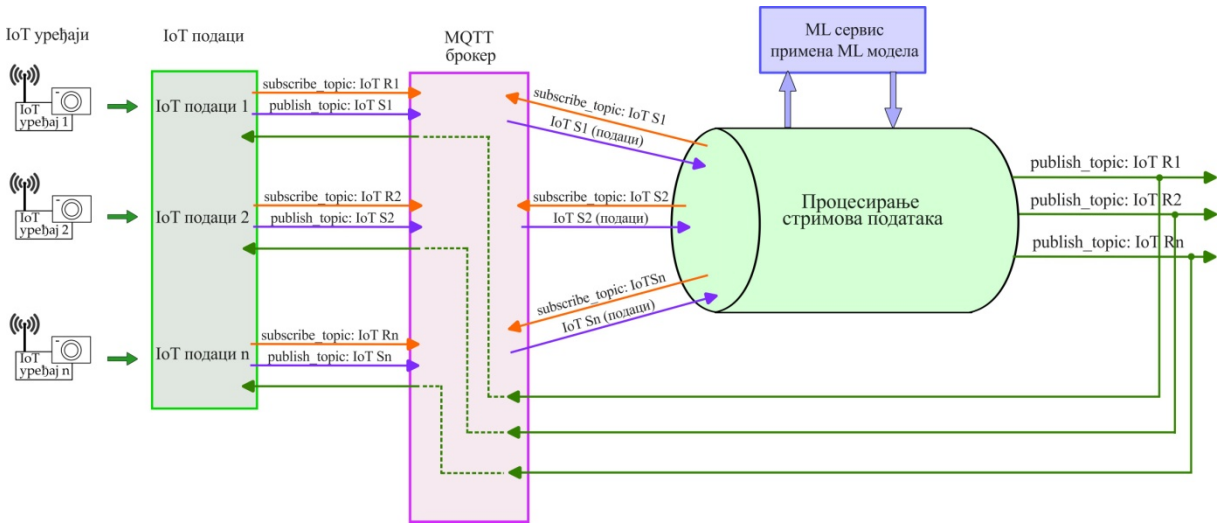
Кораци у другом делу се понављају сваки пут када се припрема нови модел за класификацију слика. Овако формиран приступ имплицира да је врло једноставно поновити процес за сваки нови модел. Једино што треба укључити је управо модел класификације (Сегмент А) и улазне података који би чинили одговарајући скупови слика (Сегмент Б). Сви остали процеси у наставку се понављају и на тај начин се ефективно добија одабрани и прилагођени модел класификације за FPGA без потребе да се у сваком кораку разматрају детаљи FPGA имплементације [149].

4.1.4. Тестни систем за проверу перформанси система

У циљу провере перформанси система, који се ослања на одабрану Cloud–Fog структуру, реализован је тестни систем који се заснива на апликацији за класификацију слика преко добијених тренираних модела. Оваква провера понашања система је значајна како би се узели у обзир различити утицаји места, односно нивоа, где се процесирају подаци и начини слања тих података.

Главни део тестног система чини апликација за класификацију података који пристижу у низу и састоји се од две главне компоненте. Први део чини компонента апликације која се имплементира код крајњих уређаја и задужена је за слање тестних података према задатој динамици. Други део апликације налази се на платформи где се врши процесирање података и може да покрене више инстанци програмских модула који позивају класификационе моделе.

Након реализације DL модела они постају део креиране IoT апликације у склопу тестног система за испитивање перформанси система у случају када се спроводи процес класификације слика. IoT апликација може да прихвати токове података односно слика и да за сваку од њих изврши препознавање којој класи слика припада. На крају добијени резултат класификације би био прослеђен до корисника како би био заокружен цео процес. Дата апликација са своје две компоненте између којих је MQTT брокер приказана је на слици 26.

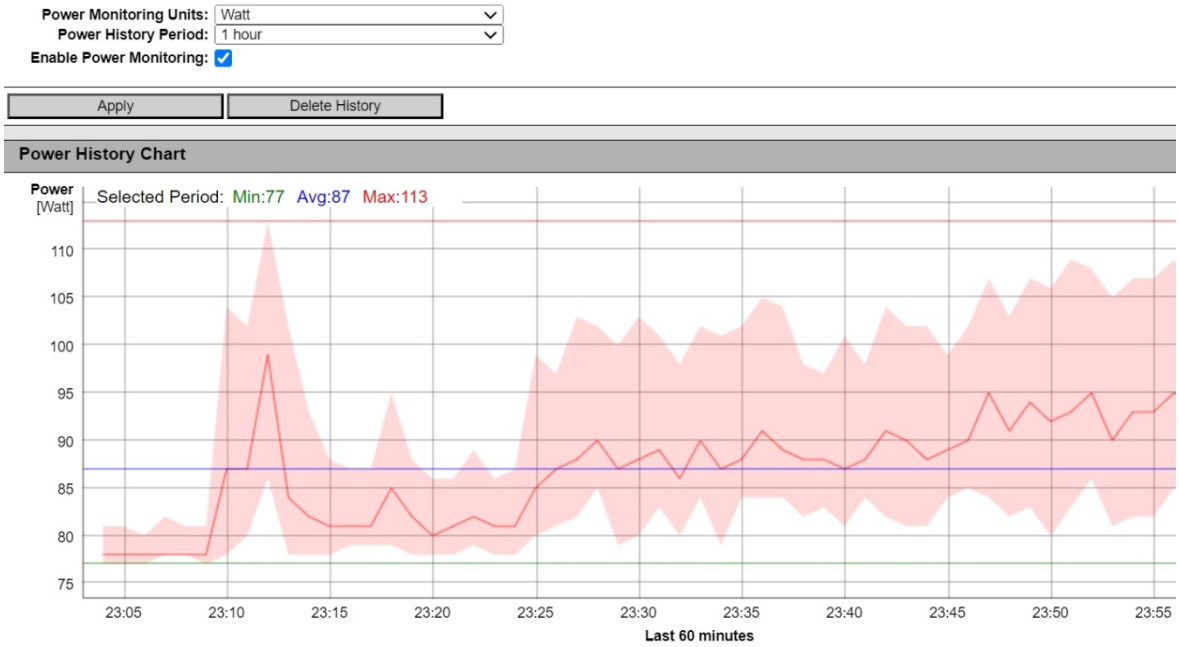


Слика 26. Апликација за прихватање и процесирање стримова података

Подаци који представљају излазе IoT уређаја у овом случају су слике које се даље преносе користећи MQTT протокол. Апликација је реализована у Python програмском језику и може подржати пренос података са више извора, при чему се могу користити различити DL модели. Излаз апликације је препозната класа која се прослеђује до корисника такође преко MQTT протокола.

Према томе други део апликације подразумева пријем података из више извора над којима би се вршила анализа. У прослеђиване тестне податке је укључена и временска одредница, која се након процеса класификације уграђује у одговор који се формира на крају. Тако да се из добијеног одговара може извући време слања слике и узимајући у обзир тренутно време може се одредити његово кашњење. На тај начин, преко дате апликације могуће је дефинисати динамику слања улазних података и број примењених класификационих модела чиме се може управљати оптерећењем постојеће Cloud–Fog структуре. Директно из евиденције о послатим сликама и добијеним резултатима може се сагледати величина прослеђених података и кашњење код апликације. При томе се надгледају и друге перформансе система праћењем потрошње енергије и мрежног оптерећења на платформи где је апликација за класификацију постављена. Овакав један тестни систем, који се ослања на апликацију за анализу стримова података, може имати своју намену и код провере других конфигурација.

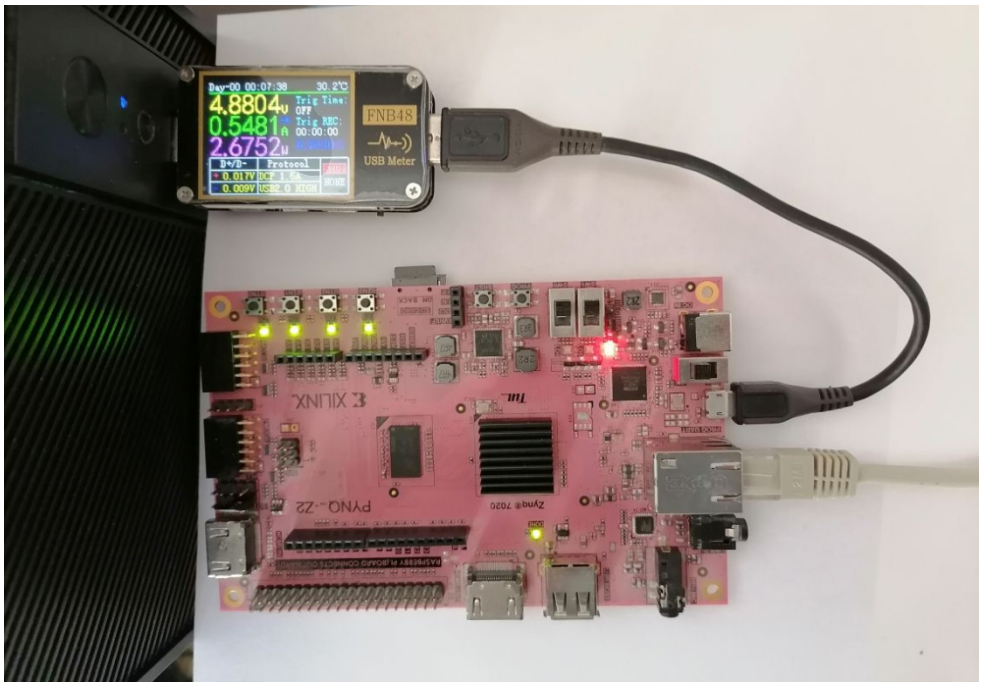
Сервер који је коришћен за спроведене примере је PRIMERGY TX2560 M1 са процесором Intel(R) Xeon(R) CPU E5-2620 v3 @ 2.40GHz и меморијом од 16GB DDR4. Помоћу iRMC S5 остварен је удаљени приступ серверу и омогућено је његово надгледање и контрола. Преко датог интерфејса вршено је праћење снаге односно енергетске потрошње у одабраном временском интервалу (Слика 27).



Слика 27. Надгледање енергетске потрошње на серверу

На претходној слици може се уочити дијаграм кретања вредности снаге у претходном сату, при чему је дата и њена просечна вредност. Преко параметара дефинисан је временски период трајања тестова према којем се врши праћење снаге, тако да се узима време и просечна снага како би се добила енергетска потрошња.

Надгледање мрежног саобраћаја односно праћење количине података која се прослеђује до сервера вршена је помоћу Wireshark алата [150]. Током истог временског периода, када се спроводе тестове, надгледа се и количина пристиглих података на серверу.

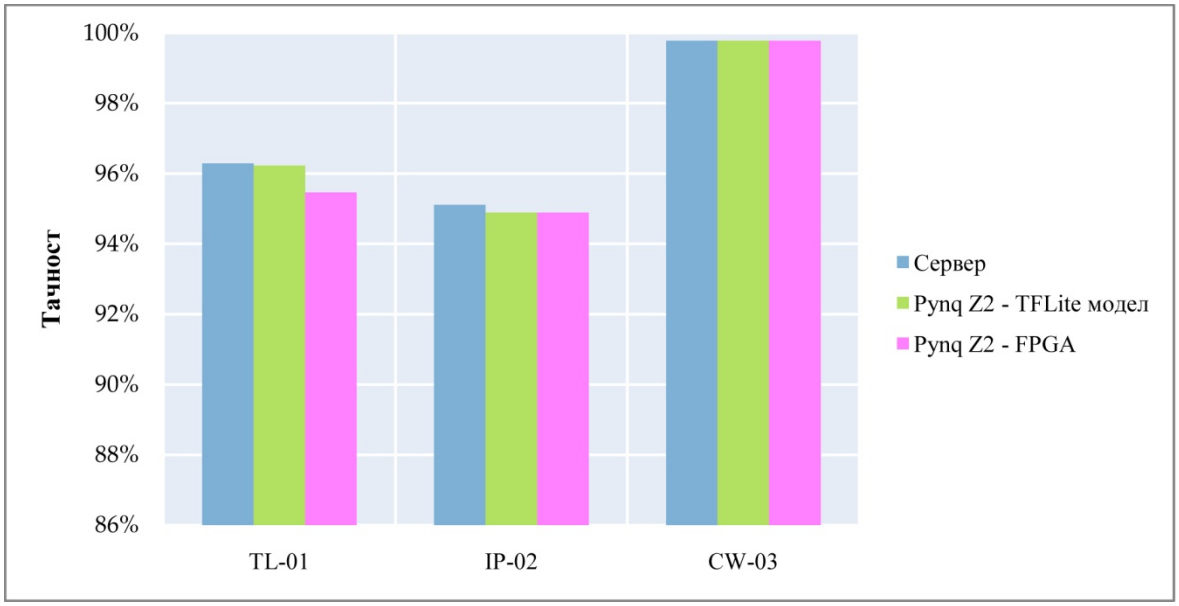


Слика 28. Праћење енергетске потрошње код PYNQ Z2 платформе

Што се тиче праћења потрошње енергије на PYNQ Z2 платформи коришћен је USB метар преко кога је извршено прикључење на напајање (Слика 28). На тај начин су могли да се сагледају тренутне вредности напона, јачине струје, као и снаге које платформа повлачи током посматраног времена. Такође, омогућено је чување вредности наведених параметара тако да се за одређени временски периода спровођења тестова на PYNQ Z2 могу добити вредности енергетске потрошње.

4.1.5. Тачност класификационог модела

Тестирање CNN модела, који се користе у три имплементиране апликације, обављено је на посебним скуповима података који нису коришћени током тренирања модела. За сваки пример су обављена три теста, један тест на серверу са оригиналним моделом и два теста на PYNQ Z2 са конвертованим CNN моделима за уређаје са ограниченим ресурсима. Тачности приликом спровођења тестова приказане су на графикону 1, где се може видети да није дошло до значајног смањења тачности након конверзије и коришћења модела на слоју Fog рачунарства [149].



Графикон 1. Тачност CNN модела за тестирања на серверу и PYNQ Z2 платформи

Постигнута тачност при коришћењу датог модела класификације има доста високе вредности, што се види са графикона 1, док је у трећој примени тачност достигла чак 99%. Овако висока вредност је постигнута јер у трећој апликацији (CW-03) постоје само две класе, што је допринело да се у коришћеној поставци веома успешно препозна једна од две могуће варијанте на датом скупу слика.

Након тестирања CNN модела, спроведена је провера перформанси система у зависности од тога где се подаци процесирају. Проверени CNN модели су коришћени за класификацију слика у примењеној апликацији тако што је из формираног тестног скупа података односно слика прослеђивана једна по једна слика. При томе се поред категорије слике добија и време слања тако да се на крају може израчунати време које протекне од тренутка слања слике до тренутка када се добије резултат класификације.

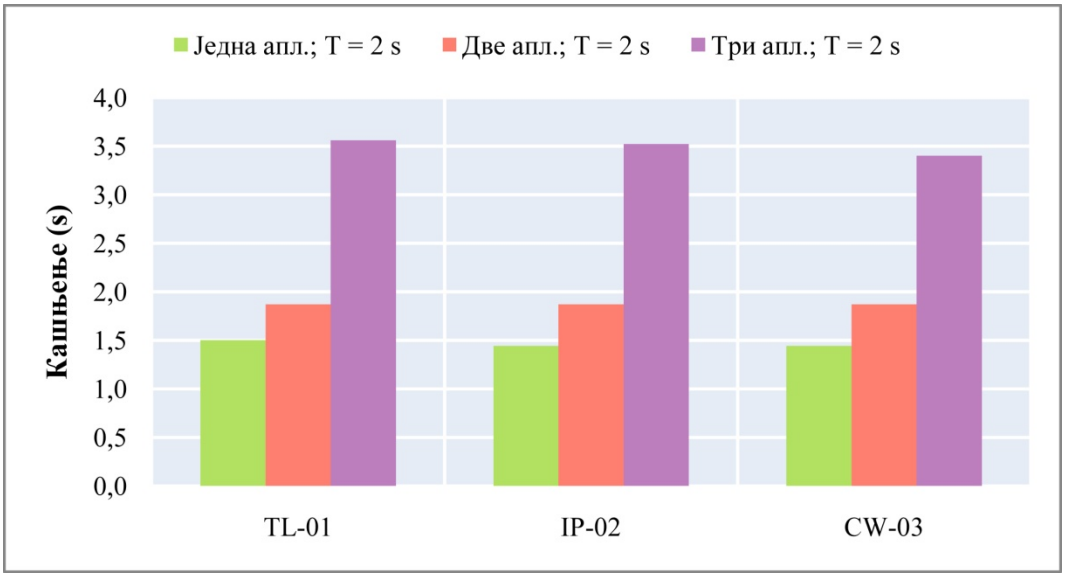
4.1.6. Перформансе система са класификационим моделом

Пероформансе система са примењеним класификационим моделом представљене су у наставку за различите динамике слања података и различита места извршавања процеса класификације. При томе су разматрана кашњења у испоруци резултата, заузетост мрежних ресурса као и енергетска потрошња.

4.1.6.1. Кашњења у испоруци резултата приликом примене апликација за класификацију

Постојале су три различите главне варијанте у погледу места где се спроводи процес класификације. У првој варијанти слике су слате на сервер преко MQTT протокола, где је пријемни део апликације преузимао слике и вршио класификацију уз помоћ постојећег, унапред тренираног CNN модела. Након тога, користећи MQTT протокол, одговор би био враћен уређају који је послао захтев. Протекло време или кашњење у пријему одговора од сервера у случају све три апликације, као и за њихове комбинације, приказано је на графикону 2.

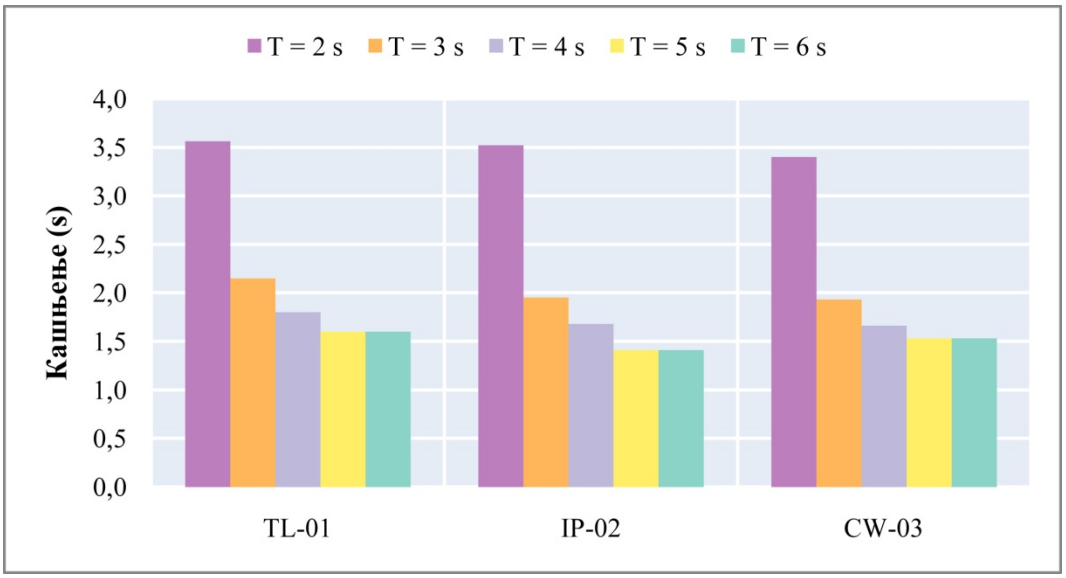
Када је покренута само једна апликација, може се видети да је кашњење око 1,5 s. У тестним примерима одабрано је да први период слања слика буде 2 s (T = 2 s). Ако би се слике континуирано слале у временском интервалу мањем од 1,5 s у овој поставци, време потребно за процес класификације би се акумулирало и добила би се већа времена кашњења, што би спречило очекивану функционалност система. Када су покренуте две апликације, добија се нешто већа вредност кашњења, али још увек испод периода слања од 2 s. Ако су све три апликације на серверу покренуте и периодично примају слике са Fog уређаја, онда се њихово кашњење повећава на вредност која износи око 3,5 s [149]. Даље повећање преноса података или додатна мрежна активност на серверу може проузроковати још већу вредност кашњења, што доводи до ситуације неочекиваног понашња система. Овакво стање може угрозити перформансе апликације по питању захтева за испоруком резултата обраде података у задатом временском интервалу.



Графикон 2. Кашњење у класификацији слика на серверу за различите примере апликација

Ако се у варијанти са све три апликације повећа период слања података, односно интервал између слања, онда се смањује оптерећење сервера, а тиме и вредност кашњења (Графикон 3).

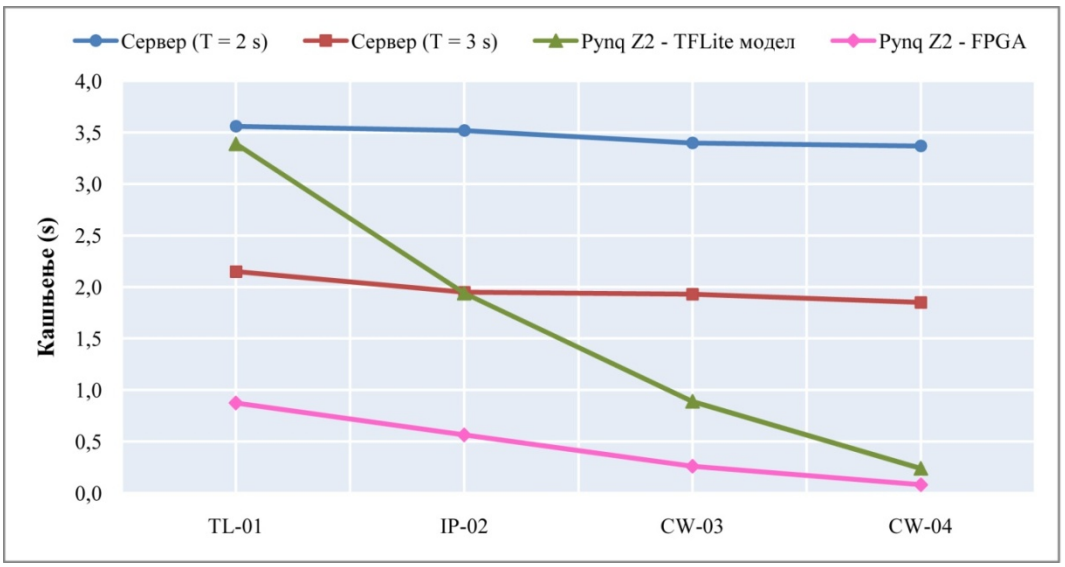
Када се период слања слике повећа на 5 s, као и на 6 s, у датом подешавању система, кашњење се смањује на близу 1,5 s, добијајући вредност коју је имала у случају појединачних апликација.



Графикон 3. Кашњење у класификацији слика на серверу када се извршавају све три апликације

У опису предложеног система демонстрирана је идеја о преношењу модела класификације на ниво Fog рачунарства како би се, између осталог, побољшала поузданост апликације. Уколико би се модел извршавао на серверу, количина података би утицала на време одзива, тако да би у случају већег броја покренутих апликација или због повећања учесталости слања података време кашњења било значајно повећано. У таквим условима кашњење би било велико, тако да би угрозило нормално функционисање система. Ако би се модел класификације извршио на Fog уређају близу извора података, одговор би се добио у реалном времену и поузданост система не би зависила од сталне везе са Cloud окружењем и мрежног оптерећења.

Поред класификације слика на серверу, друге две варијанте су везане за обраду података на PYNQ Z2 платформи. Тако да су слике прослеђиване према TensorFlow Lite моделу, односно према FPGA делу уређаја за извршење процеса класификације слика. Време потребно за класификацију слике (латенција) у случају извршења на серверу као и на нивоу Fog рачунарства преко PYNQ Z2 платформе приказано је на графикону 4.



Графикон 4. Кашњење у класификацији слика за различите платформе

На овом графикону се могу видети исти подаци о кашњењу сервера за периоде T = 2 s и T = 3 s, као на претходном графикону, али сада у односу на вредности кашњења код PYNQ Z2 платформе. Као што се и очекивало, кашњење је ниже на слоју Fog рачунарства, али вредности нису исте за све три варијанте апликација. Кашњења приказани на графикону 4 дата су и у виду бројних вредности у табели 1 због лакшег сагледавања малих вредности.

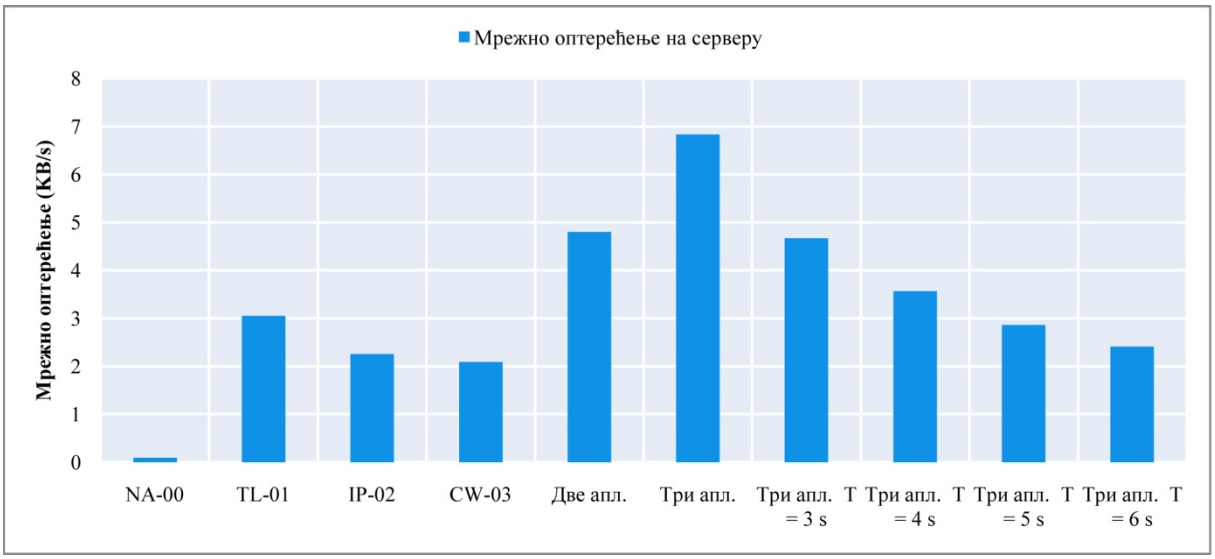
Табела 1. Кашњење у добијању резултата код класификације слика на различитим платформама

Апликација	Кашњење (s) код класификације слика			
	Сервер (T = 2 s)	Сервер (T = 3 s)	Pynq Z2 – TFLite модел	Pynq Z2 – FPGA
TL-01	3,56	2,15	3,39	0,875
IP-02	3,52	1,95	1,94	0,565
CW-03	3,40	1,93	0,89	0,260
CW-04	3,37	1,85	0,24	0,082

За прву апликацију, TL-01, класификација се врши на сликама које имају највеће димензије од 128 × 128 пиксела, па је стога током тестирања настало највеће кашњење. У другој апликацији, IP-02, коришћене су слике димензије 96 × 96 пиксела и сходно томе добијено је мање кашњење. Трећа апликација, CW-03, прихвата и класификује слике димензије 64 × 64 пиксела где постоји кашњење од 0,89 s за извршавање на CPU и 0,26 s за покретање на FPGA делу Pynq Z2 платформе. Узимајући ове вредности кашњења, може се одредити број слика које се могу класификовати у секунди, што за апликацију CW-03 износи 3,85 слика у секунди. На графикону 4 такође је приказана апликација са ознаком CW-04, која има слична својства као CW-03, само са различитим улазом, пошто користи слике димензије 32 × 32 пиксела. У овој варијанти се добија још мање времена кашњења и самим тим се постиже значајно убрзање у обради слике преко FPGA кола [149].

4.1.6.2. Мрежно оптерећење и енергетска потрошња приликом примене класификационог модела

Обрада података на слоју Fog рачунарства и добијање брзих и поузданих одговора је само по себи корисно, али такође штеди ресурсе на страни сервера. Слике са крајњих уређаја се не шаљу на удаљени сервер и самим тим је пренос података преко мреже мањи, посебно ако постоји велики број уређаја. У случају представљеног система, оптерећење мреже за различите варијанте три одабране апликације (TL-01, IP-02 и CW-03) приказано је на графикону 5.

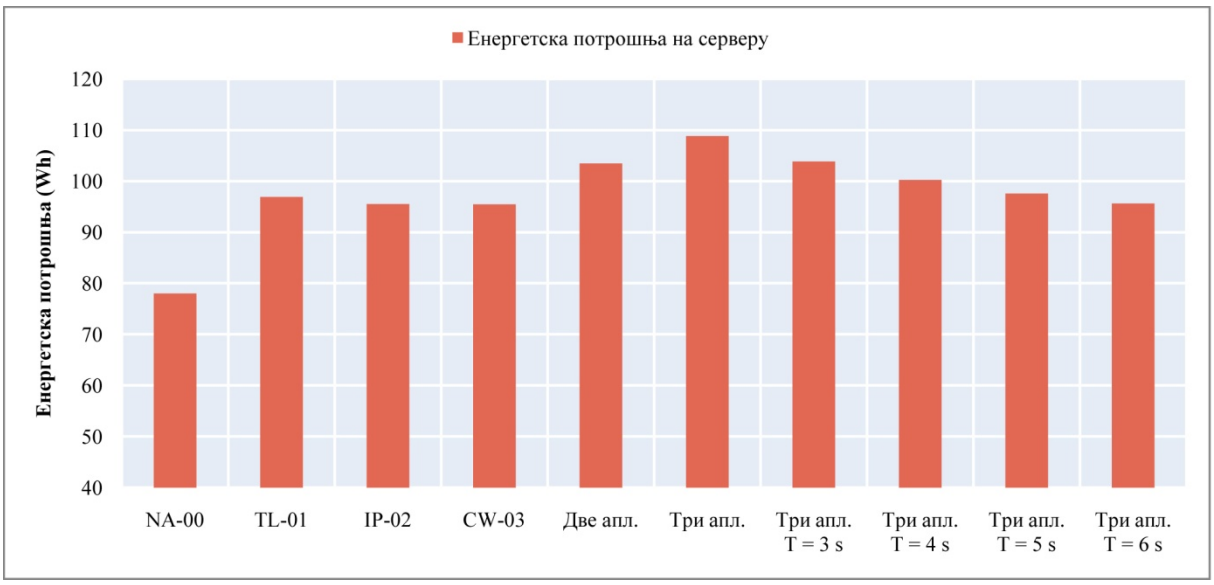


Графикон 5. Мрежно оптерећење на серверу за различите варијанте апликација

У оквиру дефинисаних системских подешавања праћен је утицај покренутих апликација на мрежно оптерећење код сервера. Током тестирања, подаци о протоку података на мрежи су снимани и њихове просечне вредности су приказане на графикону 5, где NA-00 представља оптерећење мреже када није покренута ниједна апликација. Следеће три вредности представљају количине мрежних података у случају појединачних апликација, односно величина података која је пренета за сваку апликацију посебно.

Већа вредност се добија када су покренуте прве две апликације, а највећа вредност мрежног саобраћаја је за све три апликације. Описани тестови су спроведени за период слања слика на сервер од 2 s (T = 2 s), а уколико би се повећао период (3, 4, 5 или 6 s), учесталост слања слика је мања и сходно томе просечни мрежни пренос података на сервер би се смањио [149].

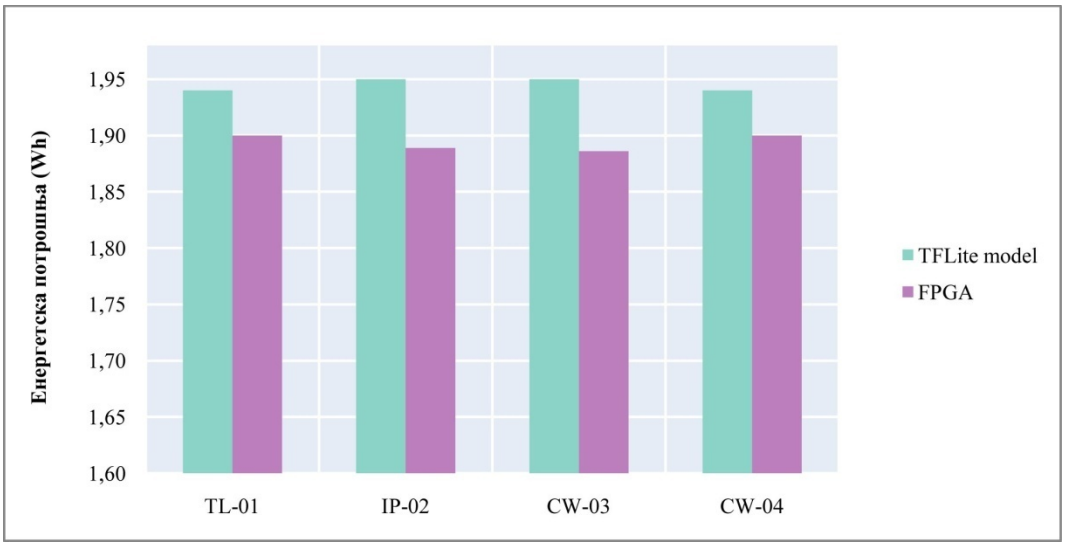
У случају када се већина обраде токова података пренесе на слој Fog рачунарства, штеди се и потрошња енергије на серверу. Вредности потрошње енергије на серверу, током тестирања, посматране у просеку за један сат, приказане су на графикону 6.



Графикон 6. Потрошња енергије на серверу током тестирања апликација

Вредност означена са NA-00 представља вредност потрошње када ниједна од тест апликација није покренута; сервер има активне само остале услуге, конфигурисане према претходно дефинисаним системским подешавањима. Слично као на претходном графикону, највећа потрошња је у случају када су покренуте све три апликације. Ако се период слања података на серверу повећа, учесталост податка је мања, а са тиме се смањује вредност енергије потребне за извршавање апликација у посматраном периоду.

Енергетска потрошња на PYNQ-Z2 платформи приказана је на графикону 7, при чему су узете у обзир две солуције, примена TensorFlow Lite модела и FPGA акцелератора. Пошто су оба класификациона модела прилагођена, односно оптимизована за извршавање на PYNQ-Z2 платформи, није велика разлика у њиховој енергетској потрошњи, уз наравно мало мању вредност код FPGA кола. Док се права предност примене акцелератора огледа у већим брзинама обраде резултата што је приказано у табели 1.



Графикон 7. Потрошња енергије на PYNQ-Z2 платформи

Приказани резултати тестног система показују односе између важних параметара перформанси система у случају извршавања апликација. При томе коришћени CNN модели су задржали адекватне вредности тачности и након оптимизације за ниво Fog рачунарства.

4.1.7. Разматрање тачност предложеног решења у препознавању слика са болестима листа парадајза

Један од скупова података који се користио за демонстрацију представљеног система класификације слика је скуп слика болести листова парадајза, које одговарају сликама из скупа података Plant Village [151]. Многи истраживачи су користили скуп података који се посебно односи на наведене слике листова парадајза у експериментима за класификацију болести парадајза. Коришћени су различити алгоритми, а резултати примењених модела изражени кроз тачност (Accuracy) модела приказани су у табели 2. Табела садржи неколико студија које представљају основу за поређење у погледу процене погодности коришћеног модела [33]. Сваки ред у табели садржи референцу истраживања, коришћени алгоритам односно модел и тачност тестирања, при чему је коришћени скуп слика парадајза. Наведена истраживања се перма томе могу упоредити са последња два реда у табели, која садржи опис представљеног модела.

Табела 2. Поређење тачности модела за класификацију болести листова парадајза

Ред. бр.	Истраживачки рад	Алгоритам/Модел	Тачност
I	Истраживање 1 [152]	DenseNet121	97,11%
II	Истраживање 2 [153]	Multi-Axis Vision Transformer	93,00%
III	Истраживање 3 [154]	VGG16	91,20%
IV	Истраживање 4 [33]	Vision Transformer	90,99%
V	Предложени модел – на среверу	ResNet20	96,29%
VI	Предложени модел – на FPGA акцелератору	ResNet20	95,46%

Иако представљени модел има своју компактну величину са циљем да се постави на уређаје са ограниченим ресурсима, његова тачност и даље има високу вредност. Након тренирања модела на серверу, тестиран је са неискоришћеним подацима и постигнута је тачност тестирања од 96,29%. Након што је модел оптимизован и преведен у верзију модела која се може покренути на FPGA, извршено је тестирање са истим новим подацима и добијена је тачност теста од 95,46% [149]. Иако је модел прилагођен и компајлиран за FPGA, дата прецизност тестирања је веома мало умањена

у поређењу са оригиналним тренираним моделом на серверу. У поређењу са резултатима из других истраживања, може се видети да се тачност предложеног модела за класификацију налази у опсегу са најбољим вредностима. Посебно је важно истаћи да се узимају вредности тачности за поређење и у случају примене конвертоване верзије модела намењене за FPGA кола.

4.1.8. Разматрање величине предложеног модела и његових перформанси приликом извршавања на хардверској платформи

Да би се испунили захтеви концепта Edge или Fog рачунарства разматрана је примена система за класификацију у случајевима где су уређаји са ограниченим ресурсима. Управо зато у предложеном моделу одабран је одговарајући алгоритам за класификацију слика и други погодни параметри како би се добио модел мање и компактније величине. У табели 3 приказани су различити модели дубоког учења [32], са бројем параметара модела, величином модела и бројем слика односно фрејмова у секунди (FPS) на ARM Cortex процесору.

Табела 3. Поређење параметара предложеног класификационог модела и других DL модела

Модел	Број параметара (10^6)	Величина модела (MB)	FPS (ARM Cortex процесор)
VGG19 [155]	200,25	229,0	0,47
VGG16 [155]	147,15	168,0	0,62
EfficientNetB0 [156]	4,05	46,9	2,69
MobileNetV1 [157]	3,23	37,1	8,23
MobileNetV3Small [158]	1,53	18,0	7,43
Vit Base [29]	86,00	345,0	0,21
GreenViT [32]	21,65	247,0	0,34
Предложени модел – апликација (TL-01)	0,29	3,8	0,30
Предложени модел – апликација (CW-04)			4,17

Према подацима у табели, може се видети да предложени модел за класификацију има далеко мањи број параметара од осталих модела, а такође и мању величину. Други параметар који показује перформансе модела је време извршења. У овом случају то је број слика у секунди на уређају са ARM Cortex процесором и то је представљено у последњој колони. За наш модел у последња два реда представљамо две варијанте у зависности од величине улазних података. У случају класификације слика димензија 128 × 128 пиксела, обрада од 0,3 слике у секунди се постиже на крајњем уређају са ограниченим ресурсима. Ако бисмо користили скуп мањих слика, 32 × 32 пиксела, онда би вредност обраде слике била 4,17 слика у секунди [149]. Наведене вредности у поређењу са осталим моделим уклапају се у њихов опсег, при чему је величина предложеног модел знатно мања.

Табела 4 приказује хардверске платформе које могу да чине Edge уређаје на којима се покрећу модели класификације. Одређене платформе су одабране на основу својстава и компарација истакнутих у раду [159], при чему је платформа коришћена у овом истраживању као FPGA акцелератор унета у последњи ред у табели.

Табела 4. Хардверске платформе као Edge и Fog уређаји

Хардверска платформа	Процесорска јединица	AI акцелератор	Меморија	FPS
Tinker Edge R	ARM Dual-core Cortex A72, Quad-core cortex A53	NPU	4 GB LPDDR4, 2GB LPDDR3	6,5
Raspberry Pi 4	ARM Quad-core Cortex A72	-	4 GB LPDDR4	4,8
Google Coral	ARM Dual-core Cortex A53	TPU	1 GB LPDDR4	3,6
NVIDIA Jetson Nano	ARM Cortex A57	128-core GPU	4 GB LPDDR4	7,2
Предложени модел на PYNQ Z2 – апликација (CW-04)	ARM Dual-core Cortex A9	Programmable Logic (FPGA)	512 MB DDR3	12,2

Нарочито када се узму у обзир вредности извршења истог модела компајлираног за FPGA, могу се уочити побољшања. Тестирањем модела на FPGA акцелератору добијамо резултате (Графикон 4) да је за прву TL-01 апликацију могуће процесирати 1,14 слика у секунди, док је у другој варијанти за апликацију CW-04 остварио процесирање 12,2 слика у секунди [149].

Прама наведеном могу се реализовати корисничке апликације тако да се резултати класификације добијају у реалном времену. Овакве погодности омогућавају даљи развој апликација у области паметне пољопривреде пружајући могућност оптимизације CNN модела и њихово извођења на уређајима са ограниченим ресурсима у близини локације извора података.

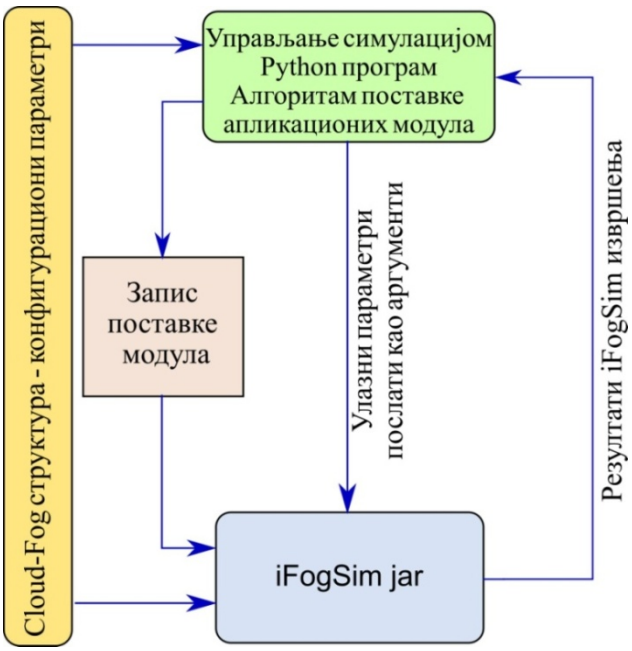
Цео систем за реализацију DL модела и проверу параметара перформанси добијених током тестирања апликација може се користити за разматрање предности коришћења Cloud–Fog рачунарства. Такође, постојећа структура тест система може се пренети у друго окружење, које има потпуно другачију конфигурацију и ресурсе. У новој ситуацији, исти показатељи могу се ефикасно добити на сличан начин, вршећи тестове корисничких апликација и обезбеђујући тако индикаторе за процену новог система Fog рачунарства.

4.2. Моделовање и симулација система Cloud–Fog рачунарства

Претходно спроведени тестови и добијени резулати су остварени на постојећој рачунарској инфраструктури и доступним SoC уређајима. Посматрање и процена Cloud–Fog структуре ширег обима, као и перформанси корисничких апликација може се спровести моделовањем и симулацијом једне такве стуктуре. Као што је објашњено у поглављу 3. коришћен је iFogSim програм за моделовање и симулацију како би се

представили системи засновани на Cloud–Fog рачунарству. Помоћу iFogSim пројекта могу се реализовати нови модели или модификовати модели постојећих система преко одабира одговарајућих конфигурација на којима би се покретале корисничке апликације.

Идеја је била да се стратегија за поставку модула одвоји од главног iFogSim пројекта тако да се оптимизација поставке модула спроводе пре покретања симулације. На тај начин би сви потребни параметри били дефинисани независно од iFogSim програма, који би могли бити приликом покретања датог програма и даље коришћени за моделовање Cloud–Fog структуре. Свеобухватни процес је на једноставан начин приказан на слици 29.



Слика 29. Блок шема симулације Cloud–Fog структуре

Независно од iFogSim пројекта налазе се конфигурациони параметри који садрже податке о свим коришћеним уређајима на свим нивоима и корисничким апликацијама. Такође посебно се чува и запис о поставци апликационих модула. Све припреме и симулација се покрећу из реализованог Python програмског решења. Python програмски језик је изабран и због његове подршке у имплементацији алгоритама оптимизације, као што је и случај поставке апликационих модула. Ова подршка је изражена кроз доступност примера и библиотека оријентисаних на машинско учење и алгоритме за оптимизацију. Могу се на погодан начин користити за проналажење оптималних решења за поставку апликативних модула и у другим сличним захтевима. На крају симулације резултати добијени као излаз iFogSim програма враћају се у Python програмску јединицу где се врши њихова додатна анализа.

4.2.1. Конфигурација изабраног модела Cloud–Fog рачунарства

Главни параметри моделованог система у оквиру конфигурационог записа односе се на његове саставне елементе, који поред Cloud окружења укључују одговарајући број крајњих уређаја, Fog уређаје и прокси уређаје.

У новим конфигурацијама система, којима се дефинише предложени модел Cloud–Fog структуре, користе се Fog уређаји са истим параметрима као у iFogSim пројекту, да би се постигла лакша и ефикаснија провера предложеног модела [141].

```
<device id="d-fd0">
        <num_of_devices>
                3
        </num_of_devices>
        <appId>
                0
        </appId>
        <name>
                fog_device
        </name>
        <mips>
                2800
        </mips>
        <ram>
                4000
        </ram>
        <upBw>
                10000
        </upBw>
        <downBw>
                10000
        </downBw>
        <level>
                2
        </level>
        <ratePerMips>
                0.0
        </ratePerMips>
        <busyPower>
                107.339
        </busyPower>
        <idlePower>
                83.4333
        </idlePower>
        <uplinkLatency>
                2.0
        </uplinkLatency>
        <sensorPrefix>
                0
        </sensorPrefix>
        <tupleType>
                0
        </tupleType>
        <transmitDistribution>
                0
        </transmitDistribution>
        <actuatorPrefix>
                0
        </actuatorPrefix>
        <actuatorType>
                0
        </actuatorType>
        <sensorLetancy>
                0
        </sensorLetancy>
        <actuatorLetancy>
                0
        </actuatorLetancy>
</device>

...
```

Слика 30. Део XML фајла који садржи податке о доступном Fog уређају

Да би се независно дефинисала Cloud–Fog структура и корисничке апликације, сви њихови параметри су издвојени изван iFogSim пројекта у конфигурациони XML (Extensible Markup Language) фајл. Према томе сви подаци који одређују Fog уређаје и доступне апликације се чувају у XML фајлу, стога се могу читати и записивати унутар али и изван iFogSim програма. Неопходни параметри се учитавају у програм iFogSim

приликом покретања симулације модела Fog рачунарства. Због тога се подаци о уређајима који се користе у моделу и подаци о апликацијама могу подесити током дефинисања Cloud–Fog структуре, пре и независно од фазе извршења симулације.

Конфигурација дата кроз ове елементе може се учитавати, ажурирати и уписивати из других програма, као што су Python програми. Према томе у сваком тренутку могу да се добију информације о постојећим елементима или да се додају нови елементи који би се користили у одабраном експерименталном сценарију.

Један део записа из XML фајла који се односи на Fog уређај приказан је као пример на слици 30. На основу датог садржаја може се уочити ID ознака уређаја, његов назив, ниво којем припада, захтевана рачунарска подршка изражена кроз CPU MIPS вредност, захтевана RAM меморија, кашњење линка према уређају на вишем нивоу, као и дефинисани пропусни опсег. Сви наведени подаци карактеришу један тип Fog уређаја у симулацији и утичу на понашање целокупног система.

```
<applications>

        <application id="appIdD0">

                <appmodule>
                        <moduleName>
                                motion_detector
                        </moduleName>
                        <module_ram>
                                10
                        </module_ram>
                        <moduleType>
                                client
                        </moduleType>
                </appmodule>

                <appmodule>
                        <moduleName>
                                object_detector
                        </moduleName>
                        <module_ram>
                                10
                        </module_ram>
                        <moduleType>
                                any
                        </moduleType>
                </appmodule>

                <appmodule>
                        <moduleName>
                                object_tracker
                        </moduleName>
                        <module_ram>
                                10
                        </module_ram>
                        <moduleType>
                                any
                        </moduleType>
                </appmodule>

                <appmodule>
                        <moduleName>
                                user_interface
                        </moduleName>
                        <module_ram>
                                10
                        </module_ram>
                        <moduleType>
                                cloud
                        </moduleType>
                </appmodule>

                ...
```

Слика 31. Део XML фајла који садржи запис о модулима апликације App-D0

Такође, у главној конфигурацији укључене су и одабране апликације које ће се извршавати приликом симулације система. Пример података о модулима који припадају апликацији са ознаком App-D0 приказан је на слици 31. Модули апликација имају записане параметре као што су ID апликације којој припада, назив модула, као и тип модула на основу кога се врши одређивање на којем нивоу дати модул може бити постављен.

4.2.2. Структура поставки апликационих модула

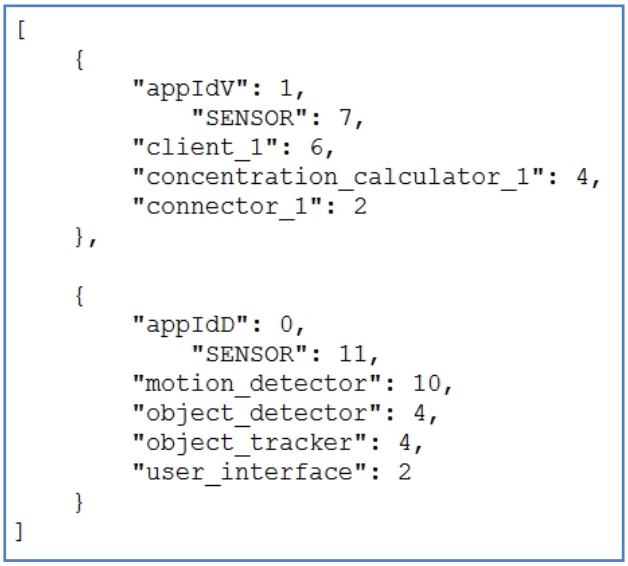
Једна од главних фаза је одређивање поставки модула којима се врши мапирање апликационих модула на претходно одабране уређаје. Нове стратегије за поставку апликационих модула могу се дефинисати у оквиру iFogSim пројекта у пакету Placement креирањем додатних класа. Ове класе могу садржати на пример расподелу модула на Fog уређаје засновану на заузетости њихових ресурса или имплементацију одабраног алгоритама за оптимизацију. Друга могућност је приказана у наставку и подразумева да се прво примени алгоритам, а да се онда резултујућа поставка модула учита у програм iFogSim. Таква једна готова поставка модула би се користила за мапирање апликационих модула пре покретања iFogSim симулације.

Прво се формира листа свих уређаја на основу конфигурационих записа тако што се поштују нивои којима уређаји припадају. Листа почиње од најнижег нивоа где су крајњи уређаји (End device), преко Fog уређаја, па до нивоа Cloud. Крајњи уређаји су такође сортирани према одређеном приоритету корисничких апликација. После њих се додају Fog уређаји при чему се узима у обзир одређено својство Fog уређаја као што је њихова близина или расположивост ресурса (CPU MIPS, RAM или мрежни пропусни опсег).

Такође, формирана је и листа свих модула пратећи својство апликација исказано кроз задати приоритет. Модули апликације са већим приоритетом позиционирани су на вишим позицијама листе и први се разматрају за поставку на одговарајућим Fog уређајима. Према томе, модули свих апликација су сортирани тако да апликација са највишим приоритетом има своје модуле на врху листе. Пратећи ову логику, следећи чланови у целокупној листи модула припадају другој приоритетној апликацији и овај принцип се примењује на све преостале апликације.

Када су формиране ове две одвојене листе од сортираних уређаја и апликационих модула, као што је претходно описано, онда се оне користе за формирање парова модул-уређај. Овакви парови у ствари представљају мапирање модула на одговарајуће уређаје и записани су као садржај фајла поставки.

Структура стратегије за поставку модула се чува у посебном JSON фајлу, који садржи записе о мапирању модула на одговарајуће ресурсе где ће се извршавати, узимајући у обзир сваког крајњег уређаја као учесника. Пример фајла поставки у JSON формату је приказан на слици 32, која садржи две апликације са по једним крајњим уређајем за сваку од њих (Ed-V1, Ed-D0) и укључује један Fog уређај (Fd0). Овај сценарио је изабран као показни пример због једноставног приказа и описа.



```
[
    {
        "appIdV": 1,
            "SENSOR": 7,
        "client_1": 6,
        "concentration_calculator_1": 4,
        "connector_1": 2
    },

    {
        "appIdD": 0,
            "SENSOR": 11,
        "motion_detector": 10,
        "object_detector": 4,
        "object_tracker": 4,
        "user_interface": 2
    }
]
```

Слика 32. Пример JSON документа са поставком апликационих модула

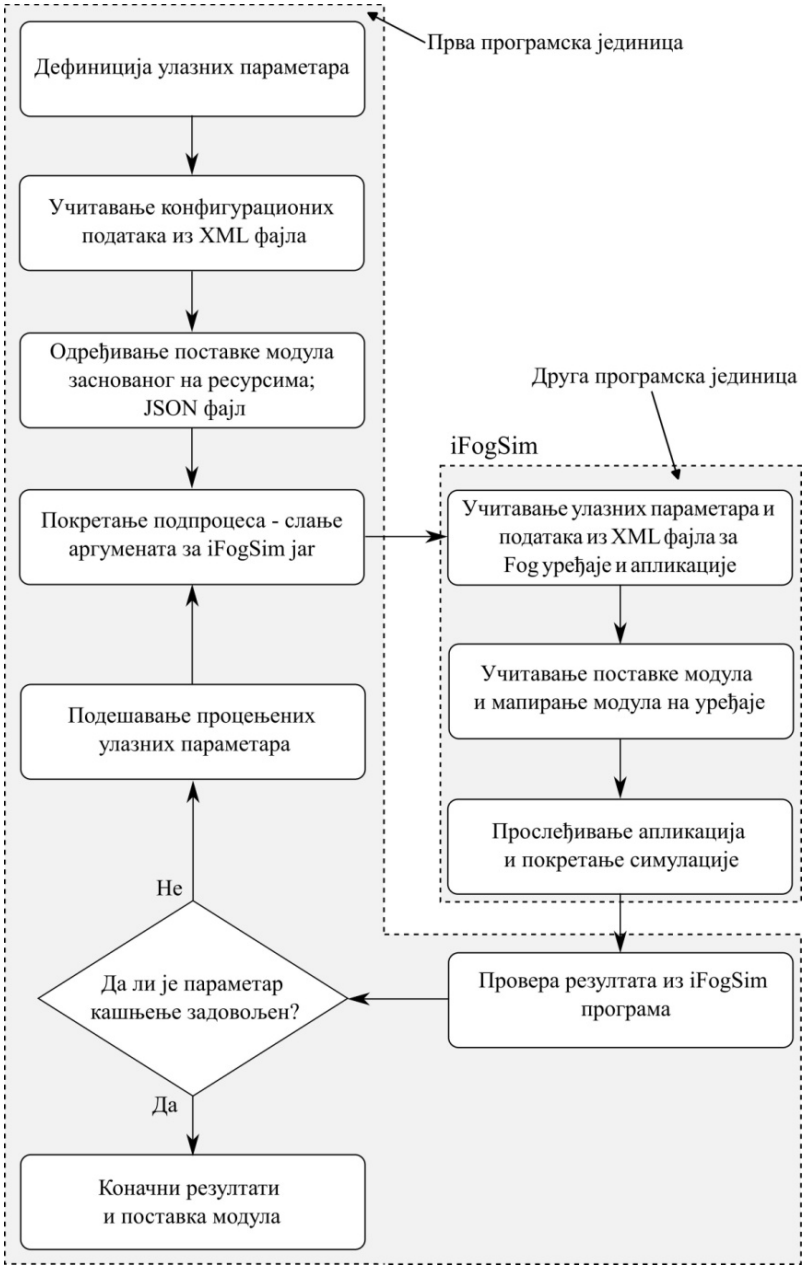
Број JSON елемента у структури зависи од броја модула свих апликација које се користе у дефинисаним сценарију. Први елемент у структури приказаног JSON фајла представља пар који чине ознака апликације и нумеричку вредност који заједно формирају ID апликације. Тако да први елемент указује на мапирање модула за прву апликацију са ID ознаком appIdV1. Следећи пар садржи назив сензора и ID припадајућег уређаја, који се генеришу током креирања свих Fog уређаја у моделу. Остали парови представљају мапирања модула, а састоје се од назива модула и ID ознаке одговарајућег уређаја на коме је намењено извршавање тог модула.

Ако је тип модула био означен са `Client`, тада је име модула упарено са ID ознаком доступног крајњег уређаја са листе. Ако је тип модула `Cloud`, овај модул би био постављен у Cloud чији ID износи 2, тако да је модул упарен са наведеном ID ознаком 2. Ако је тип модула `any` (било који), онда то значи да се може поставити на одговарајући уређај са листе одабран према алгоритму који проверава расположиве ресурсе или неке друге параметре уређаја. У предложеном случају, алгоритам проверава ресурсе Fog уређаја почевши од уређаја који је најближи корисничким уређајима, а даље наставља на следеће са листе на том истом нивоу. На тај начин сви преостали модули се мапирају на одговарајуће Fog уређаје према алгоритму који се користи за дефиницију поставки модула.

Положај модула у структури поставки одређен је према коришћеном методу или алгоритму. У овом случају, значи да су модули означени као „any“ постављени на први Fog уређај са листе који је имао довољно расположивих ресурса. Исто тако запис поставки модула могуће је променити у процесу оптимизације да би се добило најповољније решење за мапирање модула. Додатни крајњи уређаји би били повезани са првим Fog уређајем на листи који има расположиве ресурсе да испуни захтеве апликационих модула које доносе ти крајњи уређаји.

4.2.3. Управљање процесом моделовања и симулације Cloud–Fog структуре

Моделовање предложене Cloud–Fog структуре, од одређивања почетних подешавања до коначне верификације, може се посмарати у оквиру две програмске јединице. Прва програмска јединица је реализована у виду Python програма којим се управља симулацијом и читавим процесом. У овом делу се одвија конфигурација, покретање симулације модела и анализа резултата. Друга програмска јединица је одговорна за покретање и верификацију модела Cloud–Fog структуре од стране iFogSim програма и периодично се позива из прве програмске јединице.



Слика 33. Блок шема проналажења максималног броја крајњих уређаја за очекивано кашњење

У првој програмској јединици дефинише се модел Cloud–Fog структуре са свим потребним улазним параметрима и одређује се садржај фајла за поставку модула. Детаљи о свим коришћеним елементима модела налазе се у XML конфигурационом фајлу. Главни улазни параметри се постављају на почетку процеса од стране корисника, а запис у фајлу поставки модула се добија на основу података увезених из конфигурационог фајла и коришћеног алгоритма. Тако да је један излаз из Python програма фајл поставки, која уједно представља, на даље, улаз у моделовани Cloud–Fog систем. Обе ове програмске јединице могу да деле, односно да читају JSON фајл поставки модула и XML фајл са свим конфигурационим параметрима за уређаје и апликације.

Целокупни дијаграм тока за проналажење прихватљивог броја крајњих уређаја потенцијално повезаних са Cloud–Fog структуром приказан је на слици 33. Након покретања подпроцеса из Python програмске јединице, резултати би након моделовања и симулације били враћени из iFogSim програма назад и могли би се анализирати у сврху предузимања нове активности. У оквиру програма iFogSim такође су креирани додатни сегменти кода. Ови делови кода су одговорни за преузимање улазних параметара, али и учитавање свих потребних елемената из конфигурационог фајла. Такође постоји метода која чита структуру из JSON фајла поставки и преводи вредности парова у нову дефиницију мапираних модула. На овај начин добијено мапирање модула се користи за сваку апликацију пријављену током iFogSim извршења.

У варијанти предложеног Cloud–Fog модела одабрана је једна врста излазног резултата која се користи за проверу подобности система, а то је кашњење апликационе петље. Циљ је био да се задржи вредност кашњења испод одређених граничних вредности и да се тако испуне очекивани захтеви корисника. У овом случају потребно је поставити све модуле апликације на тај начин да се постигне минимално кашњење апликације, које не би прелазило граничне вредности. Према овом принципу, алгоритам почиње да резервише ресурсе на Fog уређајима који су најближи крајњим уређајима. Након формирања структуре поставки модула, следећи корак је извршавање методе која позива подпроцес који садржи iFogSim програм. Задатак прве програмске јединице јесте да након симулације модела прикупи информације са излаза iFogSim програма, међу којима су и кашњења апликационе петље. У случају да параметар није задовољавајуће вредности поново се врши корекције задатих односно процењених вредности и понавља се цео постаупак. Ако параметар кашњење има вредност унутар захтеваних граница, онда се постојећи систем може узети у обзир као оптимално решење.

4.2.4. Покретање iFogSim програма

Након подешавања параметара система и одређивање поставки модула, прелази се на другу програмску јединицу позивањем iFogSim програма, као подпроцеса. Позвани подпроцес прихвата улазне аргументе, креира и симулира Cloud–Fog структуру и на крају након симулације модела враћа резултате. Подпроцес се користи за покретање JAR пакета који садржи iFogSim програм, а који интегрише додатне функције за прихватање прослеђених главних параметара као аргумента од стране корисника. Такође iFogSim програм на почетку учитава све потребне податке о наведеној Cloud–Fog структури из XML фајла. Исто тако учитава и све записе о постави модула из постојећег JSON фајла. За покретање подпроцеса може се извршити линија у Python коду записана у изразу (30), са свим потребним аргументима намењеним за iFogSim програм.

args=[`RuniFogSim.jar`,`1`,`1`,`2`,`11`,`0`,`001`, `0`, `0`, `4`] (30)

Где су аргументи следећи:

args[0] = `RuniFogSim.jap` – Назив JAR фајла који садржи Cloud–Fog модел за симулацију дефинисаних апликација преко iFogSim програма;
args[1] = `1`– Означава Cloud окружење у систему;
args[2] = `1` – Број прокси уређаја;
args[3] = `2` – Број Fog уређаја;
args[4] = `11` – Број који се користи за нумерисање фајла који садржи запис поставке апликационих модула;
args[5] = `0` – Назнака да ли су додатне апликације укључене у процес, да (`1`), не (`0`);
args[6] = `001` – Назнака која су апликације са листе укључене у процес извршења (укључена је трећа апликација са листе) ;
args[7] = `0` - Број крајњих уређаја за прву апликацију на листи;
args[8] = `0` - Број крајњих уређаја за другу апликацију на листи;
args[9] = `4` - Број крајњих уређаја за трећу апликацију на листи.

4.2.5. Процена прихватљивог броја крајњих уређаја

На основу добијених резултата о кашњењима и другим параметрима од стране iFogSim програма, може се проценити да ли предложена Cloud–Fog структура задовољава захтеве корисника. Након употребе алгоритма за поставку модула, на основу утврђеног приоритета апликација, моделиран систем се користи у процесу процене. Циљ је проналазак оптималног број крајњих уређаја који би могли да се придруже постојећој Cloud–Fog структури, при чему се тражени оптимум у овом случају односи на прихватљиво кашњење. Потребно је да параметар апликационог кашњења буде испод утврђене границе да би се задовољили захтеви корисника. Поставља се питање колико се нових учесника као корисника апликације може додати без кршења наведеног критеријума. За процену прихватљивог броја нових, односно додатних, крајњих уређаја коришћен је алгоритам оптимизације заснован на рoју честица (Particle Swarm Optimization - PSO). При томе се структура поставки модула, као што је претходно наведено, дефинише према доступним ресурсима на Fog уређајима.

Применом PSO алгоритма израчунава се нова вредност локације у области оптимизације, а то је процењени број крајњих уређаја. Овај број крајњих уређаја се затим замењује у функцији погодности (Fitness function) PSO алгоритма. У нашем случају то је функција која израчунава расположиве ресурсе у Fog уређајима. Поново се врши поставка модула, а наша функција разматра нове вредности параметара и даје просечну вредност преосталих ресурса као излаз. Ови кораци PSO алгоритма се понављају све док капацитет расположивих ресурса у Fog уређајима не достигне минималну вредност. Алгоритам за поставку модула који се користи током примене PSO алгоритма је представљен у облику псеудокода у Алгоритму 1, са називом modulePlacementFd. Његова улога је да прорачунава просечну количину расположивих ресурса Fog уређаја за дати скуп крајњих уређаја. Затим се тај прорачун користи да би се утврдило да ли је тренутна поставка модула могућа на нивоу Fog уређаја. У случају када је поставка модула могућа, враћа се резултат који указује на нумеричку вредност расположивих ресурса.

Алгоритам 1: modulePlacementFd метода која врши проверу доступних ресурса Fog уређаја

```
Input: apps, numOfEndDevices, numOfFD, Fog_devs
Output: availableLoad
createListOfAppModules_1      // Креирање листе модула апликације 1
createListOfAppModules_2      // Креирање листе модула апликације 2
createListOfAppModules_N      // Креирање листе модула апликације N
createListOfEndDevices      //Креирање листе крајњих уређаја
placement[][]      // Матрица поставки модула која се пресликава у фајл поставки
f = 0      // Итератор за Fog уређаје
FdLoad[]      // Иницијална заузетост ресурса Fog уређаја
for i=0 to N-1 do
    m=0
    while(m<numOfAppModule[i])
        n=0
        while(n<numOfEndDevices[i])
            checkListOfFogDevices      //Проверава се листа доступних Fog уређаја
            if (FogDeviceCouldAcceptModule[f])      //Ако могу да прихвате модул
                placement[m+n][f] = 1      //Врши се поставка модула у матрици
                //Ажурира се вредност заузетости ресурса за Fog уређај - f
                FdLoad[f] = FdLoad[f] + appModuleMIPS
            else
                appModuleMoveToProxyOrCloud
            update (f)
            n=n+1
        m=m+1
availableCapacity = 0
sumAvailableCapacity = 0
if(fdLoad[idProxy]>0)
    availableCapacity = -1
else
    f =0
    while(f<numOfFogDevices)
        sumAvailableCapacity = sumAvailableCapacity + (FdMIPS – FdLoad[f])
        f = f + 1
    //Израчунавају се просечни доступни ресурси
    availableResourse = sumAvailableResourse / f
return availableResourse
```

Псеудокод дат у Алгоритму 2 представља класичну имплементацију PSO алгоритма, где се метода modulePlacementFd користи као функција погодности односно да ли се посматрана честица уклапа као прихватљиво решење. Пошто је прва фаза у дефиницији предложеног модела написан у Python програму, провера CPU ресурса и примена PSO алгоритма може се ефикасно имплементирати у оквиру постојећег програмског решења као интегрални део. Добијени број крајњих уређаја на овај начин још увек не значи да је то најбоље решење јер је потребно проверити и друге перформансе система, као што је кашњење.

Када добијемо потребан број крајњих уређаја на основу расположивих CPU ресурса, следећи корак је да проверимо изабрану конфигурацију преко iFogSim програма, да би се утврдиле перформансе датог система. Уколико изабрани параметри који дефинишу Cloud–Fog систем не доведу до задовољења критеријума, потребно је променити вредност задатих параметара система.

Алгоритам 2: PSO алгоритам за проналажење прихватљивог броја крајњих уређаја сходно доступним ресурсима Fog уређаја

```
Input: activeApps[], numOfFd, numOfEd2
Output: numOfEd1
Load configuration data:
    loadAppsData
    loadEdData
    loadFdData
Initializes PSO variables:
    n_particles
    particles_pos[]
    velocities[]
    p_best[]
    p_best_f_value[]
    g_best
    g_best_f_value
i=0
while i < n_particles:
    x = particles_pos[i]
    v = velocities[i]
    particles_pos[i] = update_position(x, velocities[i])
    numOfEndDevices = particles_pos[i]
    result_fit = modulePlacementFd(apps, numOfEndDevices, numOfFd,
                                   Fog_devs)
    if (result_fit < p_best_f_value [i])
        p_best[i] = particles_pos[i]
        p_best_f_value[i] = result_fit
        if (p_best_f_value[i] < g_best_f_value)
            g_best_f_value = p_best_f_value[i]
            g_best = p_best[i]
    i = i + 1
numOfEd1 = g_best
return numOfEd1
```

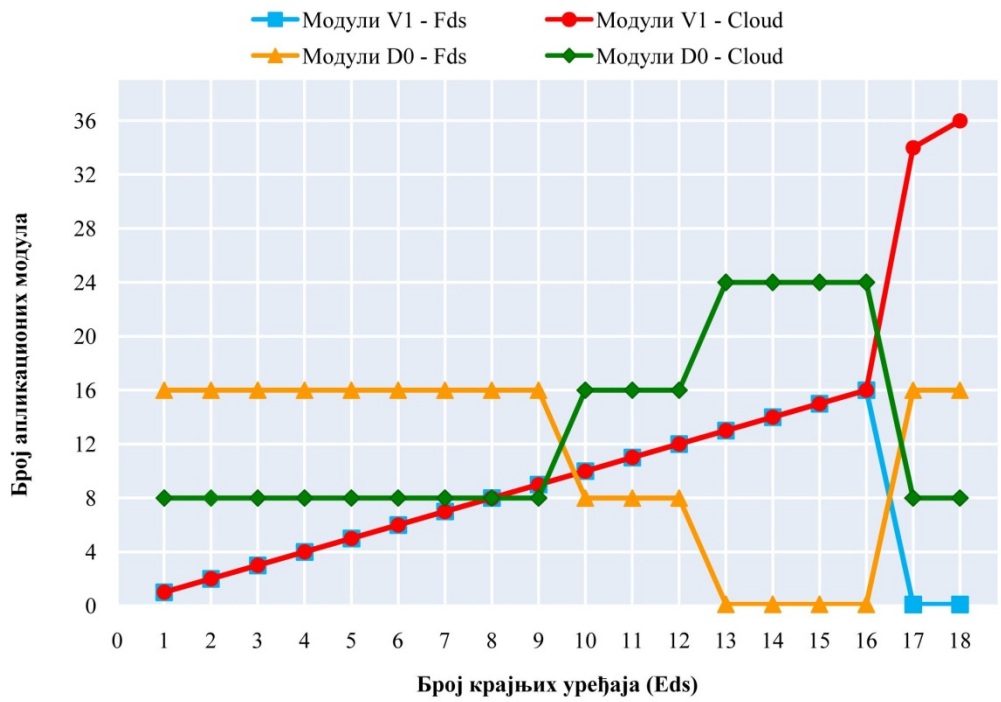
У предложеној варијанти система потребно је мењати број крајњих уређаја док се не добије најбоље решење, које постаје прихватљиво уколико су задовољени и захтеви за предефинисано кашњење. Са наведеним алгоритмима, процењује се број крајњих уређаја, односно оптерећење, које предложена Cloud–Fog структура може да прихвати на тај начин да је испуњен кориснички захтев, односно прихватљиво апликационо кашњење. У делу где се врши управљање симулацијом и целим процесом, анализирају се добијени параметари од стране iFogSim програма. У случају да прво пронађено решење није задовољавајуће може се такође користити PSO алгоритам како би се пронашло оптимално решење. Што значи након проналажења првих параметара сходно доступним ресурсима, врши се додатна примена PSO алгоритма кроз покретање симулације система како би се утврдило коначно оптимално решење у односу на апликацино кашњење.

Оваква додатна примена оптимизационог алгоритма у односу на апликационо кашњење представља главни метод за апликацију App-SF0. У овом случају није примењиван посебан алгоритам за поставку модула већ се користио постојећи алгоритам имплементације из iFogSim пројекта. На тај начин нису прво разматрани доступни ресурси у случају повећања оптерећења са додатком нових крајњих уређаја. У овој варијанти процена се вршила без претходног свођења на вредности близу оптималних и зависила је искључиво од излаза из iFogSim програма. Тако да је у овом случају PSO алгоритам коришћен са функцијом погодности (Fitness function) која је

позивала iFogSim програм и анализирала резултат кашњења. Број iFogSim позива је већи него у претходном случају пошто нема првобитне провере доступних ресурса, а позив функције би се понављао док се не пронађе прихватљив број крајњих уређаја.

4.2.6. Покретање целокупног процеса и резултати тестирања

Приликом тестирања предложеног модела прво су коришћене две апликације, App-D0 и App-V1, при чему су вршена покретања тестног оркужења са различитим бројем крајњих уређаја који припадају овим апликацијама. Пошто кашњење апликације App-V1 мора бити испод унапред дефинисане прихватљиве вредности, модули ове апликација треба да се извршавају на најближим Fog уређајима. Што значи ова апликација треба прва да заузме њихове ресурсе и стога има највиши приоритет. Други тип апликације која се користи, App-D0, може прихватити нешто веће вредности кашњења, тако да има нижи приоритет. Дакле, њени модули имају могућност извршавања на Fog уређајима или на вишим нивоима у Cloud–Fog структури. Сходно томе, одабране апликације су класификоване у два нивоа приоритета. App-V1 има ознаку највишег приоритета „1“, а App-D0 има ознаку нижег приоритета „2“.

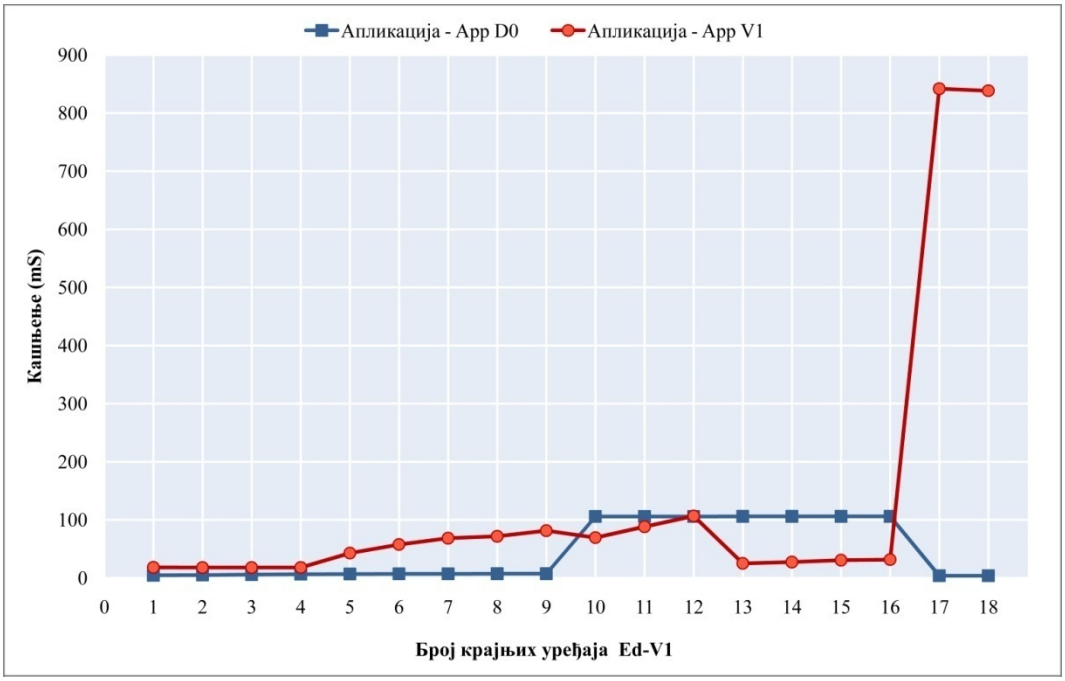


Графикон 8. Број апликационих модула са увођењем приоритетне апликације

Тестирање педложеног модела се спроводи према претходно описном поступку, а конфигурације модела су изабране према истраживању приказаном у раду [160] који је коришћен као полазна тачка. Дефинисан је фиксни број Fog уређаја који у овом случају износи четири, као и почетна вредност од осам крајњих уређаја Ed-D0. Затим су један по један додавани крајњи уређаји Ed-V1 који припадају апликацији App-V1 највишег приоритета и посматрани су резултати симулације. За ову поставку, број апликационих модула и место њиховог извршавања приказани су на графикону 8. Назив Modul-D0 односи се на модуле који припада апликацији App-D0, док Modul-V1 је везан за модуле који су део апликације App-V1. На датом графикону, у легенди, је наведено где се ови

модули извршавају тако што је у наставку дато Fds (Fog devices) ако се модули извршавају на Fog уређајима или Cloud ако се модули извршавају управо у Cloud окружењу.

У почетку се крајњи уређаји Ed-V1 додају како се повећава број учесника, тако да се модули из обе апликације могу извршавати на Fog уређајима. Када број Ed-V1 достигне 10 учесника, да би се ослободио простор за апликацију са вишим приоритетом, део модула апликације App-D0 (Модули D0) се преноси у Cloud окружење. Даље када се број Ed-V1 уређаја повећава и када достигне број 17 за њих више нема места на постојећим Fog уређајима, према томе модуле Modul-V1 је потребно пренети у Cloud. Ова промена ослобађа ресурсе Fog уређаја тако да се Модули D0 могу вратити на ниво Fog уређаја. Такође, у назначеним тренуцима долази до промене кашњења апликација, што се може видети на графикону 9. Са додатим 17. учесником, долази до прекретнице када апликација App-V1 више не може да се извршава са прихватљивим закашњењем. На основу тога поставља се питање колико је ресурса на нивоу Fog рачунарства потребно да би се подржао очекивани број учесника? Или, колико крајњих уређаја приоритетне апликације може бити прихваћено у систему са постојећим ресурсима Fog уређаја? Да би се пронашли потребни ресурси за различите апликације и радна оптерећења, предложени модел који прати концепт Fog рачунарства може послужити да пружи одговоре на претходна питања.



Графикон 9. Апликационо кашњење код повећања број Ed-V1 уређаја

Cloud–Fog структура у предложеном примеру је моделована тако да се на основу ње процени прихватљив број крајњих уређаја за дати број Fog уређаја. Ова евалуација се врши у складу са захтевима код одабраних апликација који треба да буду задовољени, као што су дозвољено кашњење апликације и приоритет извршења. Дати захтев за апликацију са највећим приоритетом подразумевао је њено извршење са минималним кашњењем. Што значи да се модули ове апликације не могу у потпуности пренети на Cloud окружење. Према томе размотрена су упоредо два сценарија, где сценарио 1 подразумева да се модули из обе апликације постављају на Fog уређаје (исти приоритет). Док сценарио 2 омогућава да се за постављање модула користи

могућност преноса на Cloud окружење оних модула који припадају апликацији нижег приоритета. Ова комбинација подразумева да у случају када нема довољно ресурса на нивоу Fog уређаја за све апликације, онда би се приступило пребацивању модула апликације нижег приоритета на Cloud.

Процена се вршила на неколико примера покретањем конфигурационих скупова који дефинишу системске ресурсе и садрже 8, 13, 21, 34 или 55 Fog уређаја (Fds), као што је приказано у табели 5. Изабран је фиксни број крајњих уређаја (Ed-D0) који су део апликације ниског приоритета, а циљ је био пронаћи максимално прихватљив број крајњих уређаја (Ed-V1) који припадају апликацији са највишим приоритетом. Прихватљиви број крајњих уређаја значи да ће кашњење њихове апликације у оквиру целокупног система бити испод захтеваног ограничења. У ту сврху је формирано пет конфигурација система са уређајима за оба сценарија и према тим поставкама тражен је прихватљив број крајњих уређаја за апликацију високог приоритета (Ed-V1). И да би се испунио исти услов да кашњење App-V1 буде испод дефинисане граничне вредности.

Табела 5. Конфигурације коришћених Cloud–Fog структура

Конфигурација	Назив конфигурације	Број Fog уређаја (Fd)	Број крајњих уређаја (Ed-D0)	Број крајњих уређаја (Ed-V1)
1	Config 1	8	19	X1
2	Config 2	13	30	X2
3	Config 3	21	49	X3
4	Config 4	34	79	X4
5	Config 5	55	128	X5

Променљиве X1, X2, X3, X4 и X5 представљају вредности које треба израчунати покретањем предложеног модела Cloud–Fog структуре како би се задовољила дефинисана граница апликационог кашњења. Због тога је циљ експеримента био да се добије максимални број крајњих уређаја који би могли да се повежу на изабрану конфигурацију система и одрже очекивано време испоруке одговора у оквиру жељеног ограничења. Конфигурациони скупови коришћени у експерименту (Табела 5) са резултатом процењених крајњих уређаја (Ed-V1) приказани су на следећим сликама.

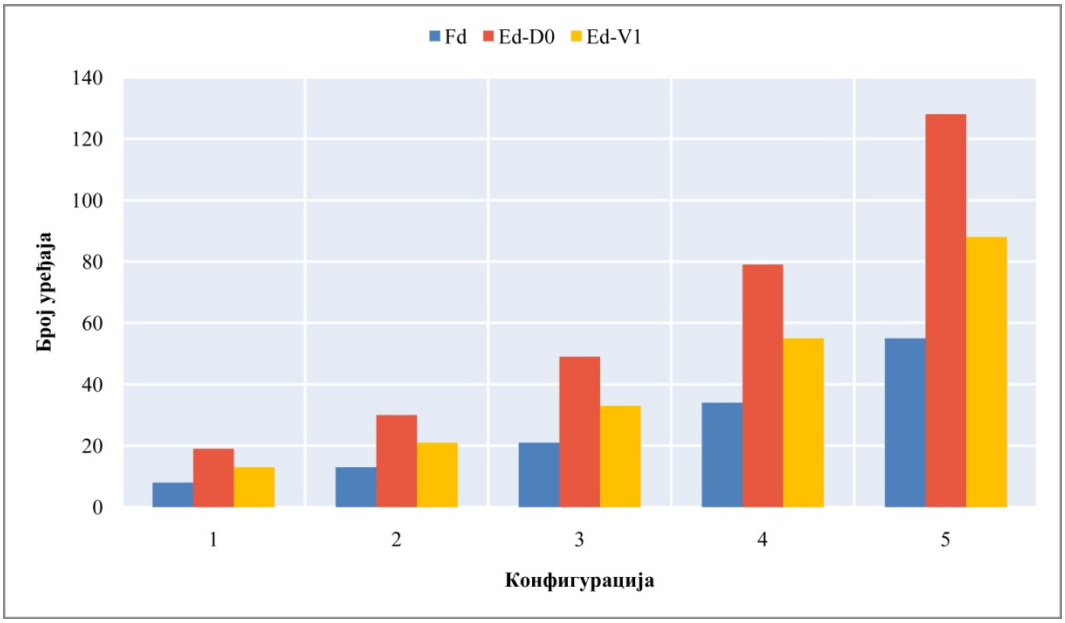
Табела 6. Карактеристике мрежних веза за моделовану Cloud–Fog структуру

Изворишни уређај	Одредишни уређај	Кашњење (ms)
Сензор	Крајњи уређај	6
Крајњи уређај	Fog уређај	2
Fog уређај	Прокси сервер	4
Прокси сервер	Cloud	100

За дефинисану Cloud–Fog структуру, постоје кашњења између различитих типова уређаја, као што је приказано у табели 6. Коришћена вредност за кашњење између Fog уређаја и Cloud окружења је 100 ms, где се могу сагледати услови испоруке резултата из Cloud окружења. У том случају поред очекиваног кашњења, може постојати потенцијални утицај загушења мреже или других неправилности на испоруке одговора. Наведено кашњење може бити посебно неповољно у случајевима где се очекује рад у реалном времену. Да бисмо осигурали да је извршавање обраде података постављено на

ниво Fog уређаја, одабрана је горња граница кашњења за приоритетну апликацију од 100 ms.

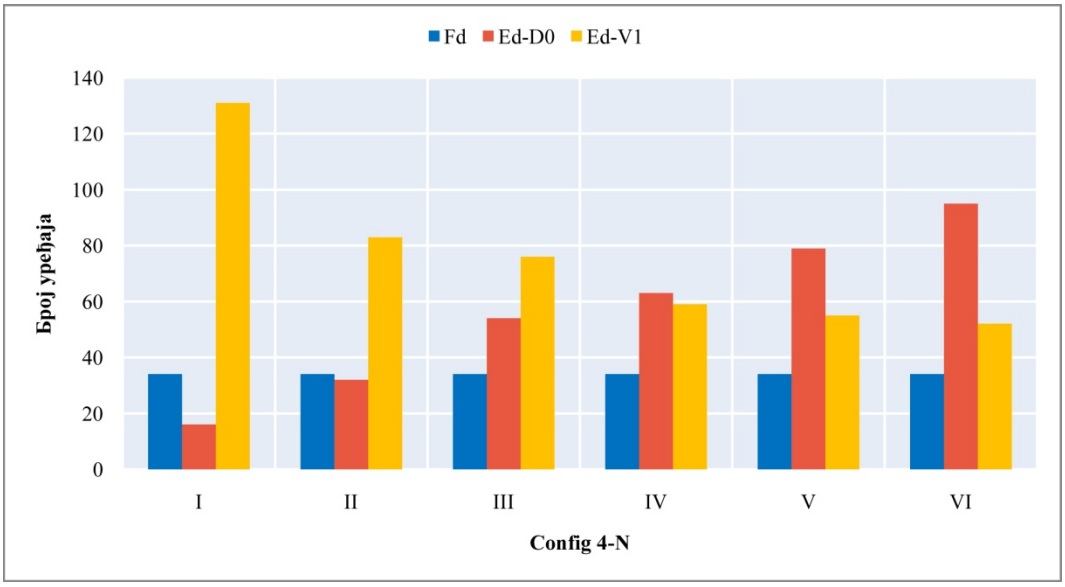
За сценарио 1, када се модули из обе апликације извршавају на Fog уређајима без преноса у Cloud окружење, број Fog уређаја и крајњих уређаја за обе апликације је приказан на графикону 10.



Графикон 10. Број уређаја за различите конфигурације у првом сценарију

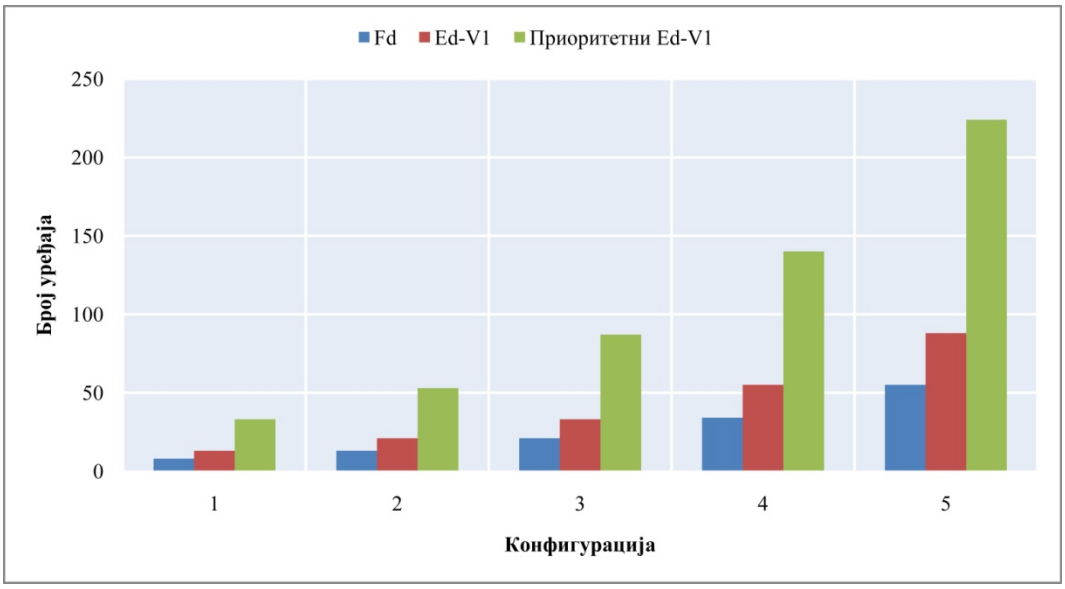
Ако је број Fog уређаја довољно велики, доступно је више ресуса, односно постоји већи простор за прихватање додатних учесника као крајњих уређаја и долази се до потпунијег искоришћење концепта Fog рачунарства. Да би се добијени резултати сагледали и из другачије перспективе на графикону 11. су приказани фиксни број Fog уређаја, постојећи бројеви крајњих уређаја Ed-D0 и прорачунати број прихватљивих уређаја Ed-V1.

Изабрани број Fog уређаја је из конфигурационе поставке Config 4, а то је 34 Fog уређаја, где се број Ed-D0 постепено повећава и укључено је шест различитих вредности. Овако модификоване конфигурације са шест варијанти је групно означена са Config 4-N. При првој изабраној вредности за Ed-D0, која је најнижа, добија се наравно највећи је број крајњих уређаја Ed-V1 које систем може да прихвати. Наведени односи се могу сагледати на графикону 11, тако да ако постоји већа употреба ресурса Fog уређаја, број добијених Ed-V1 се смањују. Израчунате вредности показују у којој мери се систем може оптеретити новим учесницима, који користе апликацију највишег приоритета без нарушавања критеријума очекиваног кашњења.



Графикон 11. Број крајњих уређаја за први сценарио са константним бројем Fog уређаја.

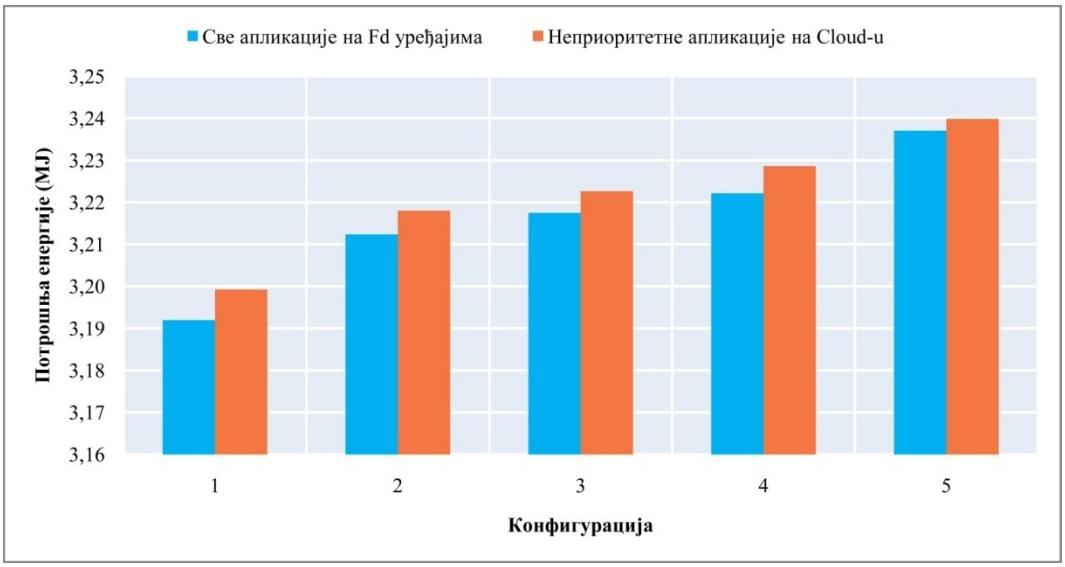
Сценарио 2 подразумева да су App-V1 модули намењени за извршавање на Fog уређајима, при чему се App-D0 модули могу преместити у Cloud окружење. Број Fog уређаја са процењеним бројем крајњих уређења који припадају апликацији App-V1 за оба сценарија представљен је на графикону 12.



Графикон 12. Број крајњих уређаја Ed-V1 у случају оба сценарија

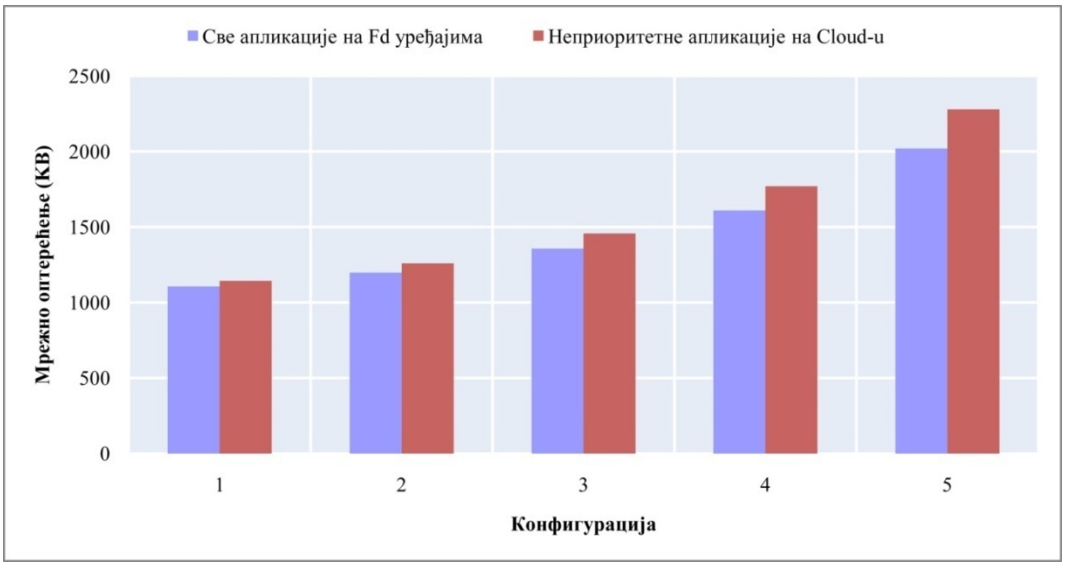
Приказани су бројеви Ed-V1 уређаја када се обе апликације покрећу на нивоу Fog рачунарства и друга варијанта када се само уређаји приоритетне апликације извршавају на нивоу Fog уређаја, који су означени са Ed-V1 приоритетни. Приказани односи указују да се број прихватљивих уређаја значајно повећава за App-V1 ако је могуће преместити апликацију нижег приоритета у Cloud, као што је случај са App-D0 у овом примеру. Након процене, односно одређивања, максималног оптерећења Cloud–Fog структуре које неће нарушити очекиване перформансе система у погледу кашњења, друге карактеристике Cloud–Fog модела се такође могу узети у обзир за доношење

коначне одлуке. Параметри добијени од iFogSim за сваку конфигурацију могу садржати показатеље потрошње енергије у Cloud окружењу и коришћење мрежних ресурса.



Графикон 13. Потрошња енергије у Cloud окружењу

На графикону 13. приказана је потрошња енергије у Cloud окружењу за оба сценарија. Када се повећа број крајњих уређаја, повећава се и број апликативних модула у Cloud окружењу, самим тим и потрошња је већа. Такође, ако се модули апликације, са нижим приоритетом, премештају у Cloud, потрошња је већа у поређењу са сценаријем где се обе апликације извршавају на Fog уређајима.



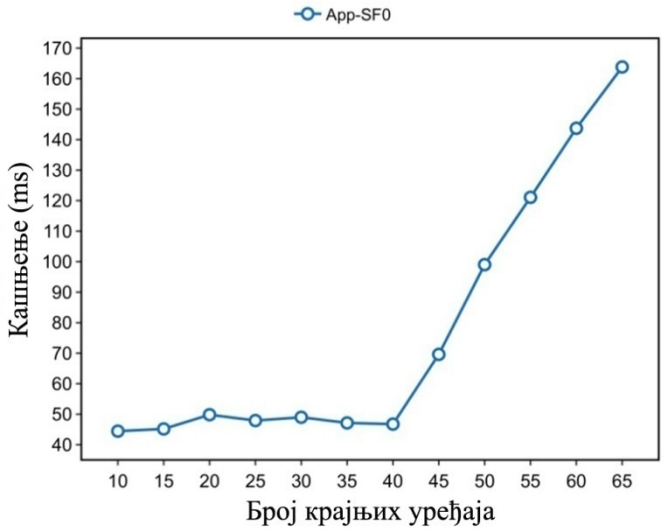
Графикон 14. Мрежно оптерећење дефинисаног Cloud–Fog модела

Слично, интензитет комуникације преко мреже се повећава како се додају нови уређаји и како се њихови апликативни модули постављају у комуникациону мрежу даље од крајњих уређаја (Графикон 14). Приказани резултати показују вредности оптерећења мреже у ситуацији коришћења концепта Fog рачунарства када постоји довољно ресурса и у другој варијанти, када је потребно пренети модуле у Cloud окружење.

Предложени модел Cloud–Fog структуре је прво намењен за процену прихватљивог броја крајњих уређаја, али се добијени резултати могу посматрати и кроз компромис. С једне стране, потребно је узети у обзир приоритет апликација и очекивано кашњење одабраних апликација, а такође је значајно узети у обзир и потрошњу енергије, коришћење мреже или друге параметре Cloud–Fog структуре.

Процена прихватљивог оптерећења са додатним крајњим уређајима је такође спроведена за апликацију са ID ознаком App-SF0 представљеном у поглављу 3 која је исто део iFogSim пројекта. Наведена апликација се састоји од микросервиса, где једна петља апликације треба да се изврши у реалном времену, док микросервис одговоран за технике предвиђања може бити распоређен у Cloud окружење. У овом случају употребе, Cloud–Fog структура је дефинисана тако да оближњи Fog уређаји формирају кластер. То значи да модули апликације или микросервиси већ имају погодност да буду постављени прво на блиске Fog уређаје у кластеру [145]. Сензор шаље сигнал крајњем уређају, који даље тај сигнал прослеђује као ток података до Fog уређаја. Део апликације, у виду микросервиса, постављен на Fog уређаје, обављао би процесирање података и вршио детекцију стања. Модел система за одабрану апликацију састоји се од 12 Fog уређаја који чине кластер и повезани су на један Прокси сервер, који је даље повезан са Cloud окружењем. Вршена је промена броја крајњих уређаја и праћено је понашање система у смислу кашњења и коришћења мреже.

Приликом тестирања параметри за конфигурацију система су коришћени из iFogSim пројекта, који садржи вредности већ проверене у другим истраживањима. При томе је циљ био праћење кашњења у пружању одговора од стране посматране апликације за различите услове. Да би се пронашло оптимално решење, тако да вредност кашњења не пређе горњу дефинисану границу, односно са кашњењем мањим од 100 ms, коришћен је алгоритам оптимизације роја честица (PSO).



Графикон 15. Апликационо кашњење за App-SF0 код повећања броја крајњих уређаја

Кашњења у петљи апликације App-SF0 за различит број крајњих уређаја учесника приказана су на графикону 15. Са PSO алгоритмом добија се да кашњење достиже 100 ms када се у систем укључи до 50 крајњих уређаја. Као што се може сагледати вредност кашњења почиње да расте након 40 крајњих уређаја и то су вредности које би требало узети у обзир приликом евалуације система. Процес би се могао поновити и за друге локације односно кластере како би се утврдило постојање уског грла на другим

местима. На тај начин би корисници имали прилику да делују превентивно и прошире ресурсе Fog уређаја или ставе под контролу ове спорне тачке док се не обезбеде ресурси.

Представљени Cloud–Fog модел може бити испробан и у многим другим примерима и апликацијама, а идеја је била да се кроз наведене примере приближи корисницима. Фокус је био на моделу који би могао да дефинише Cloud–Fog систем директним уношењем параметара у класичан XML фајл, али и проверу тог система у смислу испуњавања очекиваних захтева приликом изршавања апликација покетањем симулација. Добијене индикације о стању расположивих ресурса и подржаном оптерећењу имале би значајну улогу у одржавању захтеваних перформанси Cloud–Fog система.

5. ЗАКЉУЧАК

Приказана истраживања настала су као последица разматрања обраде великих количина података који се свакодневно размењују у савременим информационим системима. При томе посебну пажњу привлачи концепт Интернет ствари (IoT) код кога се подаци генеришу континуирано стварајући тако токове или низове података (Data Streams). IoT уређаје обично карактеришу ограничени рачунарски ресурси тако да је локална обрада података често тешко изводљива и као погодно решење појавио се принцип преноса података до Cloud окружења. Управо изузетна рачунарска подршка код Cloud окружења омогућила је њихову сврсиходну комбинацију која је постала изузетно популарна последњих година. Могао се уочити тренд да се иде у смеру централизованог прикупљања и процесирања података. Управо је то била мотивација да се размотре и другачија решења и пронађу начини да се укаже на више дистрибуирани приступ обраде. Идеја је била разматрање услова под којим би се процесирање података приближило извору података уместо да се сви ти подаци стално шаљу на Cloud окружење. Као прикладна парадигма у овој варијанти одговара Fog рачунарство, које као проширење Cloud рачунарства пружа подршку за пренос обраде података под одређеним условима. У данашње време анализа података све више захтева примену вештачких неуронских мрежа (ANN) чије тренирање и примена обично подразумева серверско окружење у оквиру Cloud рачунарства. Пошто је разматран случај уређаја са ограниченим ресурсима, који су близу извору података, онда такви уређаји углавном и формирају слој Fog рачунарства. Њихова својство се обично односе на недовољну рачунарску подршку за све могуће задатаке процесирања података са Cloud окружења. У конкретним примерима разматрана је област паметне пољопривреде (Smart farming) пошто се у тим применама могу очекивати уређаји са ограниченим ресурсима, јер се обично ради о уређајима који раде у ширим руралним срединама или удаљеним пољима. Тако да са једне старне постоје захтеви за реализацију апликације са применом ANN модела, док са друге стране се налазе захтеви за извршавање апликације под одређеним условима ограничених ресурса. Циљ у овим истраживањима је био испитати на који начин и под којим условима је могуће пренети одређене апликације за процесирање података до Fog уређаја, односно близу места где се подаци генеришу.

IoT апликације које се користе у многим областима захтевају процесирање података у реалном времену како би испунили очекивану тачност. У области паметне пољопривреде може зависити од брзине испоруке резултата да ли ће реакције система бити правовремене, односно да ли ће се применити оптималне вредности и избећи додатни губици. Из тог разлога је потребно обезбедити довољно ресурса за имплементацију апликација и у случају Fog уређаја са ограниченим ресурсима.

Пример апликација које су коришћене у испитивањима подразумевао је анализу слика, односно класификацију слика применом ANN. Примена оваквог процесирања слика подразумева реализацију ANN модела и њихово извођење на Fog уређајима може да буде изазовно. Такође, анализа слике у области пољопривреде омогућава аутоматско

препознавање различитих стања, што може допринети оптимизацији, многим уштедама ресурса и правовременој реакцији корисника.

У овом истраживању представљена је свеобухватна радна поставка за креирање, тренирање и проверу модела за класификацију слика. При томе је коришћен ResNet модел као тип CNN модела компактне величине, који је показао добре карактеристике током тренирања и тестирања модела. Реализована су три CNN модела над три различита скупа података, које је такође карактерисала и различита величина слика. У питању су били примери из области пољопривреде, а класификација у појединачним случајевима је подразумевала следеће: болести листова парадајиза, врсте штетних инсеката и детекцију корова у кукурузу. Након креирања и тестирања модела врши се његово превођење у варијанту која би могла да се изврши на Fog уређајима. Једна конверзија модела је подразумевала добијање TensroFlow Lite модела који би могао да се изврши на процесорима опште намена код уређаја са ограниченим ресурсима. И реализован је још један оптимизовани модел уз помоћ Tensil алата који може да се изврши на FPGA колу. Својство радног окружења, постављеног на овај начин, јесте да се нови модели реализују на ефикасан начин, пошто је потребно унети само два елемента. Један елемент је модел који се жели креирати и тренирати, а други елемент би били скупови података односно слика, док је цео остали низ корака потребно поновити на исти начин.

Примена модела класификације на различитим нивоима од Cloud до Fog рачунарства утиче на различитости у перформансама система. Да би се проверили различити утицаји места процесирања података реализиваон је тестни систем који се заснива на апликацији за класификацију која користи трениране моделе. Апликација се састоји од две компоненте, при чему први део чини део апликације који се имплементира код крајњих уређаја и задужен је за слање тестних података према задатој динамици. Други део апликације подразумева пријем података из више извора над којима би се вршила анализа и резултат би се враћао до стране које је слала податке. У прослеђене податке је укључивана и временска одредница тако да се из добијеног одговара поред резултата може одредити и његово кашњење. На тај начин, преко дате апликације могуће је оптеретити постојећу Cloud–Fog структуру и тако проверити њене перформансе. При томе се поред кашњења надгледају и друге перформансе система праћењем потрошње енергије и мрежног оптерећења на платформи где је апликација за класификацију постављена. Овакав један тестни систем може имати своју намену и код провере других конфигурација.

Коришњени модел класификације одабран је као CNN модел компактне величине који је погодан након оптимизоване конверзије за извршавање на уређајима са ограничени ресурсима. Такође, његова тачност након оптимизације задржала је своју високу вредност. Код класификације листова прадајза тачност је умањена са 96,3% само на 96,2% код TensroFlow Lite модела и на 95,4% код Tensil конвертоватног модела извршеног на FPGA колу.

Апликција за класификакацију има три своје инстанце сходно три скупа различитих слика и њима припадајућих модела. Постојећи серверски систем, који је био доступан за спровођење тестова, омогућавао је истовремно извршавање две апликације са кашњењем мањим до 2s. У случају покретања и треће апликације постојећа конфигурација система даје веће кашњење које износи око 3,5s. Покретањем тестова на PYNQ-Z2 платорми преко TensroFlow Lite модела добија се кашњење од 3,39s код TL-01 што је значајно веће од 0,24 код апликације CW-04, услед неколико пута већих димензија слика.

Применом акцелератора покретањем тестова на PYNQ-Z2 платорми, на FPGA делу са истим скуповима слика постиже се кашњење 0,87s код TL-01. У том случају могуће је процесирати 1,14 слика у секунди. Док у случају модела који врши закључивање код CW-04 кашњење износи 0,08, што омогућава процесирање 12,2 слика у секунди.

Имплементацијом акцелератора код уређаја са ограниченим ресурисма омогућава се процесирање слика у реалном времену, што један од важних захтева код IoT апликација, како би се одржала функционалност система. При томе се остварују други бенефити јер се не преносе континуирано сви подаци на сервер, тако да је мрежно оптерећење редуковано, као и енергетска потрошња на серверу.

Тестни део реализованог решења се може користити за проверу перформанси нових конфигурација или нових решења. Покретањем тестова на другим конфигурацијама било би могуће утврдити колика би била кашњења и оптерећење мреже у зависности од броја инстанци апликације и учесталости слања података. Такође, дата тестна поставка би се могла користити за одређивање потенцијала уређаја на Fog рачунарском слоју за извршавање одабраних IoT апликација.

Будући правци истраживања могли би да се односе на примену техника трансферног учења за решавање изазова ограничених скупова података у пољопривреди. Значајан увид би био да се испита ефикасност примене унапред тренираних модела за класификацију података и трансфер учења у случају малих скупова података. Коришћени модели класификације односили су се на апликације које се користе за препознавање корова, штетних инсеката и болести листова парадајза, при чему могу допринети поспешивању правовремене и прецизне реакције у заштити биљака.

У првом делу истраживања је коришћена доступна инфраструктура за спровођење тестова и показних примера. У општем случају за потребе процене систем Fog рачунарства представљен је концепт моделовања и симулације Cloud–Fog структура. Како би се могле проверити нове варијанте Cloud–Fog структура коришћен је iFogSim пројекат који је представљао средство многих истраживача за симулацију система Fog рачунарства.

Разматарне су перформансе задате конфигурације система од којих је посматрано време кашњења апликација за различита оптерећења. При томе су коришћена својства уређаја и апликација које већ постоје у iFogSim пројекту. Циљ је био да се предложени модел формира са већ тестираним основним елементима, а да се разматрају промене оптерећења и под којим условима би они утицали на перформансе система.

Поред програмске компоненте коју чини iFogSim пројекат развијена је одвојена програмска компоненета у Python окружењу. Ова програмска јединица омогућава да се припреме сви конфигурациони параметри, изврше поставке апликационих модула на одговарјуће уређаје и управља целокупним процесом симулације. Погодност коју нуди дато решење односи се на издвајање свих конфигурационих параметара којим се моделују уређаји и апликације у посебну XML датотеку. Што значи да се сви параметри захтеване Cloud–Fog структуре могу дефинисати независно и пре покретања iFogSim програма. Такође, поставка апликационих модула на одговарајуће уређаје чува се у посебној JSON датотеци. Одговарајуће мапирање модула може се извршити у Python окружењу, пре покретања симулације, што даје велике могућности употребе различитих алгоритама за које Python има велику подршку.

Представљени модел Cloud–Fog структуре омогућава задавање приоритета одређеним апликација које захтевају добијање резултата обраде података у реалном времену. За покренуте апликације вршена је процена модела у случају повећања броја крајњих уређаја односно учесника у коришћењеу апликације, што доводи до повећаног оптређења. У спроведеном примеру користи се алгоритам заснован на доступним ресурсима Fog уређаја да би се извршила поставка апликационих модула на уређаје Cloud–Fog структуре. Затим се користио PSO алгоритам за процену прихватљивог оптерећења израженог кроз број крајњих уређаја корисника за апликације са приоритетним кашњењем.

Предожена моделована Cloud–Fog структура може се користити за проналажење максималног броја крајњих уређаја прихватљивих за система без негативног утицаја на перформансе сервиса као што је кашњење одговора. Уз дефинисање приоритне апликације која има предност за извршавање на Fog уређајима може се утицати да се код ње повећа прихватљив броја учесника.

Представљени модел може бити основа за даља истраживања за креирање нових сценарија IoT апликације у Python окружењу и коришћење одговарајућих алгоритама како би се остваривала потребна оптимизација.

ЛИТЕРАТУРА

[1] E. Cavalcante, J. Pereira, M. P. Alves, P. Maia, R. Moura, T. Batista, F. C. Delicato, and P. F. Pires, "On the interplay of Internet of Things and Cloud Computing: A systematic mapping study," *Computer Communications*, vol. 89–90, pp. 17–33, 2016, doi: 10.1016/j.comcom.2016.03.012.

[2] M. Goudarzi, S. Ilager, and R. Buyya, "Cloud Computing and Internet of Things: Recent Trends and Directions," in *New Frontiers in Cloud Computing and Internet of Things*, R. Buyya, L. Garg, G. Fortino, and S. Misra, Eds., Cham: Springer International Publishing, 2022, pp. 3–29. doi: 10.1007/978-3-031-05528-7_1.

[3] P. Singh, Z. Elmi, V. Krishna Meriga, J. Pasha, and M. A. Dulebenets, "Internet of Things for sustainable railway transportation: Past, present, and future," *Cleaner Logistics and Supply Chain*, vol. 4, p. 100065, 2022, doi: 10.1016/j.clscn.2022.100065.

[4] "Statista - The Statistics Portal," Statista. Accessed: Nov. 06, 2024. [Online]. Available: https://www.statista.com/

[5] G. Mohyuddin, M. A. Khan, A. Haseeb, S. Mahpara, M. Waseem, and A. M. Saleh, "Evaluation of Machine Learning Approaches for Precision Farming in Smart Agriculture System: A Comprehensive Review," *IEEE Access*, vol. 12, pp. 60155–60184, 2024, doi: 10.1109/ACCESS.2024.3390581.

[6] E. Said Mohamed, AA. Belal, S. Kotb Abd-Elmabod, M. A. El-Shirbeny, A. Gad, and M. B. Zahran, "Smart farming for improving agricultural management," *The Egyptian Journal of Remote Sensing and Space Science*, vol. 24, no. 3, Part 2, pp. 971–981, 2021, doi: 10.1016/j.ejrs.2021.08.007.

[7] I. Charania and X. Li, "Smart farming: Agriculture's shift from a labor intensive to technology native industry," *Internet of Things*, vol. 9, p. 100142, 2020, doi: 10.1016/j.iot.2019.100142.

[8] V. Moysiadis, P. Sarigiannidis, V. Vitsas, and A. Khelifi, "Smart Farming in Europe," *Computer Science Review*, vol. 39, p. 100345, 2021, doi: 10.1016/j.cosrev.2020.100345.

[9] B. B. Sinha and R. Dhanalakshmi, "Recent advancements and challenges of Internet of Things in smart agriculture: A survey," *Future Generation Computer Systems*, vol. 126, pp. 169–184, 2022, doi: 10.1016/j.future.2021.08.006.

[10] D. Marković, D. Vujičić, S. Tanasković, B. Đorđević, S. Randjić, and Z. Stamenković, "Prediction of Pest Insect Appearance Using Sensors and Machine Learning," *Sensors*, vol. 21, no. 14, p. 4846, 2021, doi: 10.3390/s21144846.

[11] D. Marković, U. Pešović, D. Tomić, and V. Stević, "Crop weeds detection using neural network models," presented at the 1st International Symposium On Biotechnology, 2023, pp. 93–98. doi: https://doi.org/10.46793/SBT28.093M.

[12] D. Marković, R. Koprivica, B. Veljković, D. Vujičić, D. Stojić, and U. Pešović, "Primena mašinskog učenja za identifikaciju stanja kotrljajnih ležajeva kod

poljoprivredne mehanizacije na osnovu vibracija," *Poljoprivredna tehnika*, vol. 48, no. 4, pp. 100–110, 2023, doi: 10.5937/PoljTeh2304100M.

[13] D. Marković, R. Koprivica, B. Veljković, M. Gavrilović, D. Vujičić, U. Pešović, and S. Ranđić, "Detekcija gubitaka semena uljane repice pre berbe analizom slike u okviru Fog računarstva," *Poljoprivredna tehnika*, vol. 47, no. 4, pp. 28–37, 2022, doi: 10.5937/PoljTeh2204028M.

[14] U. Pešović, D. Marković, S. Đurašević, and S. Ranđić, "Remote monitoring of beehive activity," *Acta agriculturae Serbica*, vol. 24, no. 48, pp. 157–165, 2019, doi: 10.5937/AASer1948157P.

[15] D. Marković, S. Ranđić, S. Tanasković, and S. Gvozdenac, "Possibility of monitoring D.v. virgifera flight by processing image of phero-traps using Raspberry Pi based devices," *Acta agriculturae Serbica*, vol. 22, no. 44, pp. 207–217, 2017, doi: 10.5937/AASer1744207M.

[16] J. Gu, Z. Wang, J. Kuen, L. Ma, A. Shahroudy, B. Shuai, T. Liu, X. Wang, G. Wang, J. Cai, and T. Chen, "Recent advances in convolutional neural networks," *Pattern Recognition*, vol. 77, pp. 354–377, May 2018, doi: 10.1016/j.patcog.2017.10.013.

[17] Z. W. Awan, S. Khalid, and S. Gul, "A Theoretical CNN Compression Framework for Resource-Restricted Environments," in *2022 2nd International Conference on Digital Futures and Transformative Technologies (ICoDT2)*, IEEE, May 2022, pp. 1–8. doi: 10.1109/ICoDT255437.2022.9787431.

[18] T. Choudhary, V. Mishra, A. Goswami, and J. Sarangapani, "A comprehensive survey on model compression and acceleration," *Artif Intell Rev*, vol. 53, no. 7, pp. 5113–5155, Oct. 2020, doi: 10.1007/s10462-020-09816-7.

[19] M. P. Véstias, R. P. Duarte, J. T. de Sousa, and H. C. Neto, "Moving Deep Learning to the Edge," *Algorithms*, vol. 13, no. 5, Art. no. 5, May 2020, doi: 10.3390/a13050125.

[20] S. Mittal, "A survey of FPGA-based accelerators for convolutional neural networks," *Neural Comput & Applic*, vol. 32, no. 4, pp. 1109–1139, Feb. 2020, doi: 10.1007/s00521-018-3761-1.

[21] P. Xiyuan, Y. Jinxiang, Y. Bowen, L. Liansheng, and P. Yu, "A Review of FPGA-Based Custom Computing Architecture for Convolutional Neural Network Inference," *Chinese Journal of Electronics*, vol. 30, no. 1, pp. 1–17, 2021, doi: 10.1049/cje.2020.11.002.

[22] S. I. Venieris, A. Kouris, and C.-S. Bouganis, "Toolflows for Mapping Convolutional Neural Networks on FPGAs: A Survey and Future Directions," *ACM Comput. Surv.*, vol. 51, no. 3, p. 56:1-56:39, Jun. 2018, doi: 10.1145/3186332.

[23] A. Sateesan, S. Sinha, S. K. G., and A. P. Vinod, "A Survey of Algorithmic and Hardware Optimization Techniques for Vision Convolutional Neural Networks on FPGAs," *Neural Process Lett*, vol. 53, no. 3, pp. 2331–2377, Jun. 2021, doi: 10.1007/s11063-021-10458-1.

[24] Md. M. H. Shuvo, S. K. Islam, J. Cheng, and B. I. Morshed, "Efficient Acceleration of Deep Learning Inference on Resource-Constrained Edge Devices: A Review," *Proceedings of the IEEE*, vol. 111, no. 1, pp. 42–91, Jan. 2023, doi: 10.1109/JPROC.2022.3226481.

[25] Y. Liu, J. Xue, D. Li, W. Zhang, T. K. Chiew, and Z. Xu, "Image recognition based on lightweight convolutional neural network: Recent advances," *Image and Vision Computing*, vol. 146, p. 105037, Jun. 2024, doi: 10.1016/j.imavis.2024.105037.

[26] C. Wang and Z. Luo, "A Review of the Optimal Design of Neural Networks Based on FPGA," *Applied Sciences*, vol. 12, no. 21, p. 10771, 2022, doi: 10.3390/app122110771.

[27] M. Xia, Z. Huang, L. Tian, H. Wang, V. Chang, Y. Zhu, and S. Feng, "SparkNoC: An energy-efficiency FPGA-based accelerator using optimized lightweight CNN for edge computing," *Journal of Systems Architecture*, vol. 115, p. 101991, 2021, doi: 10.1016/j.sysarc.2021.101991.

[28] X. Feng, Y. Jiang, X. Yang, M. Du, and X. Li, "Computer vision algorithms and hardware implementations: A survey," *Integration*, vol. 69, pp. 309–320, Nov. 2019, doi: 10.1016/j.vlsi.2019.07.005.

[29] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale," Jun. 2021, doi: 10.48550/arXiv.2010.11929.

[30] S. Tummala, S. Kadry, S. A. C. Bukhari, and H. T. Rauf, "Classification of Brain Tumor from Magnetic Resonance Imaging Using Vision Transformers Ensembling," *Current Oncology*, vol. 29, no. 10, Oct. 2022, doi: 10.3390/curroncol29100590.

[31] S. Ercolino, A. Devoto, L. Monorchio, M. Santini, S. Mazzaro, and S. Scardapane, "On the robustness of vision transformers for in-flight monocular depth estimation," *Industrial Artificial Intelligence*, vol. 1, no. 1, pp. 1–14, Dec. 2023, doi: 10.1007/s44244-023-00005-3.

[32] S. Parez, N. Dilshad, N. S. Alghamdi, T. M. Alanazi, and J. W. Lee, "Visual Intelligence in Precision Agriculture: Exploring Plant Disease Detection via Efficient Vision Transformers," *Sensors*, vol. 23, no. 15, p. 6949, 2023, doi: 10.3390/s23156949.

[33] U. Barman, P. Sarma, M. Rahman, V. Deka, S. Lahkar, V. Sharma, and M. J. Saikia, "ViT-SmartAgri: Vision Transformer and Smartphone-Based Plant Disease Detection for Smart Agriculture," *Agronomy*, vol. 14, no. 2, p. 327, 2024, doi: 10.3390/agronomy14020327.

[34] H. Li, S. Li, J. Yu, Y. Han, and A. Dong, "Plant disease and insect pest identification based on vision transformer," in *International Conference on Internet of Things and Machine Learning (IoTML 2021)*, SPIE, Apr. 2022, pp. 194–201. doi: 10.1117/12.2628467.

[35] J. Maurício, I. Domingues, and J. Bernardino, "Comparing Vision Transformers and Convolutional Neural Networks for Image Classification: A Literature Review," *Applied Sciences*, vol. 13, no. 9, Jan. 2023, doi: 10.3390/app13095521.

[36] S. D. Alwis, Z. Hou, Y. Zhang, M. H. Na, B. Ofoghi, and A. Sajjanhar, "A survey on smart farming data, applications and techniques," *Computers in Industry*, vol. 138, p. 103624, 2022, doi: 10.1016/j.compind.2022.103624.

[37] G. Idoje, T. Dagiuklas, and M. Iqbal, "Survey for smart farming technologies: Challenges and issues," *Computers & Electrical Engineering*, vol. 92, p. 107104, 2021, doi: 10.1016/j.compeleceng.2021.107104.

[38] V. Sharma, A. K. Tripathi, and H. Mittal, "Technological revolutions in smart farming: Current trends, challenges & future directions," *Computers and Electronics in Agriculture*, vol. 201, p. 107217, Oct. 2022, doi: 10.1016/j.compag.2022.107217.

[39] T. Ayoub Shaikh, T. Rasool, and F. Rasheed Lone, "Towards leveraging the role of machine learning and artificial intelligence in precision agriculture and smart farming," *Computers and Electronics in Agriculture*, vol. 198, p. 107119, Jul. 2022, doi: 10.1016/j.compag.2022.107119.

[40] S. Ghazal, A. Munir, and W. S. Qureshi, "Computer vision in smart agriculture and precision farming: Techniques and applications," *Artificial Intelligence in Agriculture*, vol. 13, pp. 64–83, Sep. 2024, doi: 10.1016/j.aiia.2024.06.004.

[41] B. Darwin, P. Dharmaraj, S. Prince, D. E. Popescu, and D. J. Hemanth, "Recognition of Bloom/Yield in Crop Images Using Deep Learning Models for Smart Agriculture: A Review," *Agronomy*, vol. 11, no. 4, Apr. 2021, doi: 10.3390/agronomy11040646.

[42] J. Lu, L. Tan, and H. Jiang, "Review on Convolutional Neural Network (CNN) Applied to Plant Leaf Disease Classification," *Agriculture*, vol. 11, no. 8, Aug. 2021, doi: 10.3390/agriculture11080707.

[43] K. Ramana, R. Aluvala, M. R. Kumar, G. Nagaraja, A. V. Krishna, and P. Nagendra, "Leaf Disease Classification in Smart Agriculture Using Deep Neural Network Architecture and IoT," *J CIRCUIT SYST COMP*, vol. 31, no. 15, p. 2240004, Oct. 2022, doi: 10.1142/S0218126622400047.

[44] N. K. Trivedi, V. Gautam, A. Anand, H. M. Aljahdali, S. G. Villar, D. Anand, N. Goyal, and S. Kadry, "Early Detection and Classification of Tomato Leaf Disease Using High-Performance Deep Neural Network," *Sensors*, vol. 21, no. 23, Jan. 2021, doi: 10.3390/s21237987.

[45] N. Aishwarya, N. G. Praveena, S. Priyanka, and J. Pramod, "Smart farming for detection and identification of tomato plant diseases using light weight deep neural network," *Multimed Tools Appl*, vol. 82, no. 12, pp. 18799–18810, May 2023, doi: 10.1007/s11042-022-14272-2.

[46] A. S. Paymode and V. B. Malode, "Transfer Learning for Multi-Crop Leaf Disease Image Classification using Convolutional Neural Network VGG," *Artificial Intelligence in Agriculture*, vol. 6, pp. 23–33, Jan. 2022, doi: 10.1016/j.aiia.2021.12.002.

[47] E. Özbılge, M. K. Ulukök, Ö. Toygar, and E. Ozbılge, "Tomato Disease Recognition Using a Compact Convolutional Neural Network," *IEEE Access*, vol. 10, pp. 77213–77224, 2022, doi: 10.1109/ACCESS.2022.3192428.

[48] H. Pang, Z. Zheng, T. Zhen, and A. Sharma, "Smart Farming: An Approach for Disease Detection Implementing IoT and Image Processing," *IJAEIS*, vol. 12, no. 1, pp. 55–67, Jan. 2021, doi: 10.4018/IJAEIS.20210101.oa4.

[49] W. Li, T. Zheng, Z. Yang, M. Li, C. Sun, and X. Yang, "Classification and detection of insects from field images using deep learning for smart pest management: A systematic review," *Ecological Informatics*, vol. 66, p. 101460, Dec. 2021, doi: 10.1016/j.ecoinf.2021.101460.

[50] D. J. A. Rustia, J. J. Chao, L. Y. Chiu, Y. F. Wu, J. Y. Chung, J. C. Hsu, and T. T. Lin, "Automatic greenhouse insect pest detection and recognition based on a cascaded deep learning classification method," *Journal of Applied Entomology*, vol. 145, no. 3, pp. 206–222, 2021, doi: 10.1111/jen.12834.

[51] T. Kasinathan, D. Singaraju, and S. R. Uyyala, "Insect classification and detection in field crops using modern machine learning techniques," *Information Processing in Agriculture*, vol. 8, no. 3, pp. 446–457, Sep. 2021, doi: 10.1016/j.inpa.2020.09.006.

[52] C. Li, T. Zhen, and Z. Li, "Image Classification of Pests with Residual Neural Network Based on Transfer Learning," *Applied Sciences*, vol. 12, no. 9, Jan. 2022, doi: 10.3390/app12094356.

[53] K. Rimal, K. B. Shah, and A. K. Jha, "Advanced multi-class deep learning convolution neural network approach for insect pest classification using TensorFlow," *Int. J.*

Environ. Sci. Technol., vol. 20, no. 4, pp. 4003–4016, Apr. 2023, doi: 10.1007/s13762-022-04277-7.

[54] Z. Wu, Y. Chen, B. Zhao, X. Kang, and Y. Ding, "Review of Weed Detection Methods Based on Computer Vision," *Sensors*, vol. 21, no. 11, Jan. 2021, doi: 10.3390/s21113647.

[55] A. Wang, W. Zhang, and X. Wei, "A review on weed detection using ground-based machine vision and image processing techniques," *Computers and Electronics in Agriculture*, vol. 158, pp. 226–240, Mar. 2019, doi: 10.1016/j.compag.2019.02.005.

[56] A. S. M. M. Hasan, F. Sohel, D. Diepeveen, H. Laga, and M. G. K. Jones, "A survey of deep learning techniques for weed detection from images," *Computers and Electronics in Agriculture*, vol. 184, p. 106067, May 2021, doi: 10.1016/j.compag.2021.106067.

[57] N. Razfar, J. True, R. Bassiouny, V. Venkatesh, and R. Kashef, "Weed detection in soybean crops using custom lightweight deep learning models," *Journal of Agriculture and Food Research*, vol. 8, p. 100308, Jun. 2022, doi: 10.1016/j.jafr.2022.100308.

[58] H. Jiang, C. Zhang, Y. Qiao, Z. Zhang, W. Zhang, and C. Song, "CNN feature based graph convolutional network for weed and crop recognition in smart farming," *Computers and Electronics in Agriculture*, vol. 174, p. 105450, Jul. 2020, doi: 10.1016/j.compag.2020.105450.

[59] M. Taneja and A. Davy, "Resource aware placement of IoT application modules in Fog-Cloud Computing Paradigm," in *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, May 2017, pp. 1222–1228. doi: 10.23919/INM.2017.7987464.

[60] A. Yousefpour, G. Ishigaki, and J. P. Jue, "Fog Computing: Towards Minimizing Delay in the Internet of Things," in *2017 IEEE International Conference on Edge Computing (EDGE)*, Jun. 2017, pp. 17–24. doi: 10.1109/IEEE.EDGE.2017.12.

[61] Z. Rezazadeh, M. Rezaei, and M. Nickray, "LAMP: A Hybrid Fog-Cloud Latency-Aware Module Placement Algorithm for IoT Applications," in *2019 5th Conference on Knowledge Based Engineering and Innovation (KBEI)*, Feb. 2019, pp. 845–850. doi: 10.1109/KBEI.2019.8734958.

[62] A. R. Benamer, H. Teyeb, and N. Ben Hadj-Alouane, "Latency-Aware Placement Heuristic in Fog Computing Environment," H. Panetto, C. Debruyne, H. A. Proper, C. A. Ardagna, D. Roman, and R. Meersman, Eds., in Lecture Notes in Computer Science, vol. 11230. Cham: Springer International Publishing, 2018, pp. 241–257. doi: 10.1007/978-3-030-02671-4_14.

[63] F. Hosseinpour, A. Naebi, S. Virtanen, T. Pahikkala, H. Tenhunen, and J. Plosila, "A Resource Management Model for Distributed Multi-Task Applications in Fog Computing Networks," *IEEE Access*, vol. 9, pp. 152792–152802, 2021, doi: 10.1109/ACCESS.2021.3127355.

[64] M. Dadashi Gavaber and A. Rajabzadeh, "BADEP: Bandwidth and delay efficient application placement in fog-based IoT systems," *Transactions on Emerging Telecommunications Technologies*, vol. 32, no. 8, p. e4136, 2021, doi: 10.1002/ett.4136.

[65] J. U. Arshed and M. Ahmed, "RACE: Resource Aware Cost-Efficient Scheduler for Cloud Fog Environment," *IEEE Access*, vol. 9, pp. 65688–65701, 2021, doi: 10.1109/ACCESS.2021.3068817.

[66] R. K. Naha, S. Garg, A. Chan, and S. K. Battula, "Deadline-based dynamic resource allocation and provisioning algorithms in Fog-Cloud environment," *Future Generation Computer Systems*, vol. 104, pp. 131–141, 2020, doi: 10.1016/j.future.2019.10.018.

[67] S. Subbaraj and R. Thiyagarajan, "Performance oriented task-resource mapping and scheduling in fog computing environment," *Cognitive Systems Research*, vol. 70, pp. 40–50, 2021, doi: 10.1016/j.cogsys.2021.07.004.

[68] X. Fei, N. Shah, N. Verba, K.-M. Chao, V. Sanchez-Anguix, J. Lewandowski, A. James, and Z. Usman, "CPS data streams analytics based on machine learning for Cloud and Fog Computing: A survey," *Future Generation Computer Systems*, vol. 90, pp. 435–450, 2019, doi: 10.1016/j.future.2018.06.042.

[69] F. Gürcan and M. Berigel, "Real-Time Processing of Big Data Streams: Lifecycle, Tools, Tasks, and Challenges," in *2018 2nd International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT)*, Oct. 2018, pp. 1–6. doi: 10.1109/ISMSIT.2018.8567061.

[70] I. Flouris, N. Giatrakos, A. Deligiannakis, M. Garofalakis, M. Kamp, and M. Mock, "Issues in complex event processing: Status and prospects in the Big Data era," *Journal of Systems and Software*, vol. 127, pp. 217–236, 2017, doi: 10.1016/j.jss.2016.06.011.

[71] H. Nasiri, S. Nasehi, and M. Goudarzi, "Evaluation of distributed stream processing frameworks for IoT applications in Smart Cities," *J Big Data*, vol. 6, no. 1, p. 52, 2019, doi: 10.1186/s40537-019-0215-2.

[72] T. P. Raptis and A. Passarella, "A Survey on Networked Data Streaming With Apache Kafka," *IEEE Access*, vol. 11, pp. 85333–85350, 2023, doi: 10.1109/ACCESS.2023.3303810.

[73] T. Rabl, J. Traub, A. Katsifodimos, and V. Markl, "Apache Flink in current research," *it - Information Technology*, vol. 58, no. 4, pp. 157–165, 2016, doi: 10.1515/itit-2016-0005.

[74] M. Petrov, N. Butakov, D. Nasonov, and M. Melnik, "Adaptive performance model for dynamic scaling Apache Spark Streaming," *Procedia Computer Science*, vol. 136, pp. 109–117, 2018, doi: 10.1016/j.procs.2018.08.243.

[75] K. Dineva and T. Atanasova, "Design of Scalable IoT Architecture Based on AWS for Smart Livestock," *Animals*, vol. 11, no. 9, p. 2697, 2021, doi: 10.3390/ani11092697.

[76] C. S. Ranganathan, R. Raman, V. K. Pandey, S. Kavitha, M. Muthulekshmi, and K. Gopalakrishnan, "Ingestion of Google Cloud Platform Data using Dataflow," in *2023 7th International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*, 2023, pp. 338–342. doi: 10.1109/I-SMAC58438.2023.10290190.

[77] Y. Liu, K. Akram Hassan, M. Karlsson, Z. Pang, and S. Gong, "A Data-Centric Internet of Things Framework Based on Azure Cloud," *IEEE Access*, vol. 7, pp. 53839–53858, 2019, doi: 10.1109/ACCESS.2019.2913224.

[78] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1645–1660, 2013, doi: 10.1016/j.future.2013.01.010.

[79] W. Choi, J. Kim, S. Lee, and E. Park, "Smart home and internet of things: A bibliometric study," *Journal of Cleaner Production*, vol. 301, p. 126908, 2021, doi: 10.1016/j.jclepro.2021.126908.

[80] L. Babangida, T. Perumal, N. Mustapha, and R. Yaakob, "Internet of Things (IoT) Based Activity Recognition Strategies in Smart Homes: A Review," *IEEE Sensors Journal*, vol. 22, no. 9, pp. 8327–8336, 2022, doi: 10.1109/JSEN.2022.3161797.

[81] S. B. Atitallah, M. Driss, W. Boulila, and H. B. Ghézala, "Leveraging Deep Learning and IoT big data analytics to support the smart cities development: Review and future

directions," *Computer Science Review*, vol. 38, p. 100303, 2020, doi: 10.1016/j.cosrev.2020.100303.

[82] J. J. Peralta Abadía, C. Walther, A. Osman, and K. Smarsly, "A systematic survey of Internet of Things frameworks for smart city applications," *Sustainable Cities and Society*, vol. 83, p. 103949, 2022, doi: 10.1016/j.scs.2022.103949.

[83] R. Kumar, S. Rani, and M. A. Awadh, "Exploring the Application Sphere of the Internet of Things in Industry 4.0: A Review, Bibliometric and Content Analysis," *Sensors*, vol. 22, no. 11, p. 4276, 2022, doi: 10.3390/s22114276.

[84] S. Munirathinam, "Chapter Six - Industry 4.0: Industrial Internet of Things (IIOT)," in *Advances in Computers*, vol. 117, 1 vols., P. Raj and P. Evangeline, Eds., in The Digital Twin Paradigm for Smarter Systems and Environments: The Industry Use Cases, vol. 117. , Elsevier, 2020, pp. 129–164. doi: 10.1016/bs.adcom.2019.10.010.

[85] M. Haghi Kashani, M. Madanipour, M. Nikravan, P. Asghari, and E. Mahdipour, "A systematic review of IoT in healthcare: Applications, techniques, and trends," *Journal of Network and Computer Applications*, vol. 192, p. 103164, 2021, doi: 10.1016/j.jnca.2021.103164.

[86] M. R. Bashir, A. Q. Gill, G. Beydoun, and B. Mccusker, "Big Data Management and Analytics Metamodel for IoT-Enabled Smart Buildings," *IEEE Access*, vol. 8, pp. 169740–169758, 2020, doi: 10.1109/ACCESS.2020.3024066.

[87] D. A. Gzar, A. M. Mahmood, and M. K. A. Al-Adilee, "Recent trends of smart agricultural systems based on Internet of Things technology: A survey," *Computers and Electrical Engineering*, vol. 104, p. 108453, 2022, doi: 10.1016/j.compeleceng.2022.108453.

[88] S. Qazi, B. A. Khawaja, and Q. U. Farooq, "IoT-Equipped and AI-Enabled Next Generation Smart Agriculture: A Critical Review, Current Challenges and Future Trends," *IEEE Access*, vol. 10, pp. 21219–21235, 2022, doi: 10.1109/ACCESS.2022.3152544.

[89] M. U. Saleem, M. R. Usman, and M. Shakir, "Design, Implementation, and Deployment of an IoT Based Smart Energy Management System," *IEEE Access*, vol. 9, pp. 59649–59664, 2021, doi: 10.1109/ACCESS.2021.3070960.

[90] A. Rey, E. Panetti, R. Maglio, and M. Ferretti, "Determinants in adopting the Internet of Things in the transport and logistics industry," *Journal of Business Research*, vol. 131, pp. 584–590, 2021, doi: 10.1016/j.jbusres.2020.12.049.

[91] S. L. Ullo and G. R. Sinha, "Advances in Smart Environment Monitoring Systems Using IoT and Sensors," *Sensors*, vol. 20, no. 11, p. 3113, 2020, doi: 10.3390/s20113113.

[92] R. Krishnamurthi, A. Kumar, D. Gopinathan, A. Nayyar, and B. Qureshi, "An Overview of IoT Sensor Data Processing, Fusion, and Analysis Techniques," *Sensors*, vol. 20, no. 21, p. 6076, 2020, doi: 10.3390/s20216076.

[93] F. Durao, J. F. S. Carvalho, A. Fonseka, and V. C. Garcia, "A systematic review on cloud computing," *J Supercomput*, vol. 68, no. 3, pp. 1321–1346, 2014, doi: 10.1007/s11227-014-1089-x.

[94] R. Buyya, J. Broberg, and A. M. Goscinski, *Cloud Computing: Principles and Paradigms*. John Wiley & Sons, 2010.

[95] R. Mahmud, R. Kotagiri, and R. Buyya, "Fog Computing: A Taxonomy, Survey and Future Directions," in *Internet of Everything: Algorithms, Methodologies, Technologies and Perspectives*, B. Di Martino, K.-C. Li, L. T. Yang, and A. Esposito, Eds., in

Internet of Things. , Singapore: Springer, 2018, pp. 103–130. doi: 10.1007/978-981-10-5861-5_5.

[96] R. M. Abdelmoneem, A. Benslimane, and E. Shaaban, "Mobility-aware task scheduling in cloud-Fog IoT-based healthcare architectures," *Computer Networks*, vol. 179, p. 107348, 2020, doi: 10.1016/j.comnet.2020.107348.

[97] M. Goudarzi, M. Palaniswami, and R. Buyya, "Scheduling IoT Applications in Edge and Fog Computing Environments: A Taxonomy and Future Directions," *ACM Comput. Surv.*, vol. 55, no. 7, p. 152:1-152:41, 2022, doi: 10.1145/3544836.

[98] S. O. Ogundoyin and I. A. Kamil, "Optimization techniques and applications in fog computing: An exhaustive survey," *Swarm and Evolutionary Computation*, vol. 66, p. 100937, 2021, doi: 10.1016/j.swevo.2021.100937.

[99] R. M. Haralick and L. G. Shapiro, "Glossary of computer vision terms," *Pattern Recognition*, vol. 24, no. 1, pp. 69–93, 1991, doi: 10.1016/0031-3203(91)90117-N.

[100] D. G. Bailey, *Design for Embedded Image Processing on FPGAs*. John Wiley & Sons, 2023.

[101] P. Laplante and S. Ovaska, "Fundamentals of Real-Time Systems," in *Real-Time Systems Design and Analysis*, John Wiley & Sons, Ltd, 2011, pp. 1–25. doi: 10.1002/9781118136607.ch1.

[102] M. M. Hammad, "Artificial Neural Network and Deep Learning: Fundamentals and Theory," Aug. 12, 2024, *arXiv*: arXiv:2408.16002. doi: 10.48550/arXiv.2408.16002.

[103] S. Ruder, "An overview of gradient descent optimization algorithms," Jun. 15, 2017, *arXiv*: arXiv:1609.04747. doi: 10.48550/arXiv.1609.04747.

[104] Y. Liu, Y. Gao, and W. Yin, "An improved analysis of stochastic gradient descent with momentum," in *Proceedings of the 34th International Conference on Neural Information Processing Systems*, in NIPS '20. Red Hook, NY, USA: Curran Associates Inc., Dec. 2020, pp. 18261–18271.

[105] J. Duchi, E. Hazan, and Y. Singer, "Adaptive Subgradient Methods for Online Learning and Stochastic Optimization," *J. Mach. Learn. Res.*, vol. 12, pp. 2121–2159, 2011.

[106] T. Kurbiel and S. Khaleghian, "Training of Deep Neural Networks based on Distance Measures using RMSProp," 2017, *arXiv*: arXiv:1708.01911. doi: 10.48550/arXiv.1708.01911.

[107] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," Jan. 30, 2017, *arXiv*: arXiv:1412.6980. doi: 10.48550/arXiv.1412.6980.

[108] Z. Li, F. Liu, W. Yang, S. Peng, and J. Zhou, "A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 12, pp. 6999–7019, 2022, doi: 10.1109/TNNLS.2021.3084827.

[109] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016, pp. 770–778. doi: 10.1109/CVPR.2016.90.

[110] K. Vipin and S. A. Fahmy, "FPGA Dynamic and Partial Reconfiguration: A Survey of Architectures, Methods, and Applications," *ACM Comput. Surv.*, vol. 51, no. 4, p. 72:1-72:39, 2018, doi: 10.1145/3193827.

[111] R. Nane, V.-M. Sima, C. Pilato, J. Choi, B. Fort, A. Canis, Y. T. Chen, H. Hsiao, S. Brown, F. Ferrandi, J. Anderson, and K. Bertels, "A Survey and Evaluation of FPGA High-Level Synthesis Tools," *IEEE Transactions on Computer-Aided Design of*

Integrated Circuits and Systems, vol. 35, no. 10, pp. 1591–1604, 2016, doi: 10.1109/TCAD.2015.2513673.

[112] S. M. Trimberger, "Three Ages of FPGAs: A Retrospective on the First Thirty Years of FPGA Technology," *Proceedings of the IEEE*, vol. 103, no. 3, pp. 318–331, 2015, doi: 10.1109/JPROC.2015.2392104.

[113] C. Bobda, J. M. Mbongue, P. Chow, M. Ewais, N. Tarafdar, J. C. Vega, K. Eguro, D. Koch, S. Handagala, M. Leeser, M. Herbordt, H. Shahzad, P. Hofste, B. Ringlein, J. Szefer, A. Sanaullah, and R. Tessier, "The Future of FPGA Acceleration in Datacenters and the Cloud," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 15, no. 3, p. 34:1-34:42, 2022, doi: 10.1145/3506713.

[114] M. Elnawawy, A. Farhan, A. A. Nabulsi, A. R. Al-Ali, and A. Sagahyroon, "Role of FPGA in Internet of Things Applications," in *2019 IEEE International Symposium on Signal Processing and Information Technology (ISSPIT)*, Dec. 2019, pp. 1–6. doi: 10.1109/ISSPIT47144.2019.9001747.

[115] A. Shawahna, S. M. Sait, and A. El-Maleh, "FPGA-Based Accelerators of Deep Learning Networks for Learning and Classification: A Review," *IEEE Access*, vol. 7, pp. 7823–7859, 2019, doi: 10.1109/ACCESS.2018.2890150.

[116] Z. Liu, Y. Dou, J. Jiang, J. Xu, S. Li, Y. Zhou, and Y. Xu, "Throughput-Optimized FPGA Accelerator for Deep Convolutional Neural Networks," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 10, no. 3, p. 17:1-17:23, 2017, doi: 10.1145/3079758.

[117] K. P. Seng, P. J. Lee, and L. M. Ang, "Embedded Intelligence on FPGA: Survey, Applications and Challenges," *Electronics*, vol. 10, no. 8, p. 895, 2021, doi: 10.3390/electronics10080895.

[118] D. Wu, Y. Zhang, X. Jia, L. Tian, T. Li, L. Sui, D. Xie, and Y. Shan, "A High-Performance CNN Processor Based on FPGA for MobileNets," in *2019 29th International Conference on Field Programmable Logic and Applications (FPL)*, Sep. 2019, pp. 136–143. doi: 10.1109/FPL.2019.00030.

[119] P. Babu and E. Parthasarathy, "Reconfigurable FPGA Architectures: A Survey and Applications," *J. Inst. Eng. India Ser. B*, vol. 102, no. 1, pp. 143–156, 2021, doi: 10.1007/s40031-020-00508-y.

[120] S. I. Venieris, A. Kouris, and C.-S. Bouganis, "Toolflows for Mapping Convolutional Neural Networks on FPGAs: A Survey and Future Directions," *ACM Comput. Surv.*, vol. 51, no. 3, p. 56:1-56:39, 2018, doi: 10.1145/3186332.

[121] U. Farooq, Z. Marrakchi, and H. Mehrez, "FPGA Architectures: An Overview," in *Tree-based Heterogeneous FPGA Architectures: Application Specific Exploration and Optimization*, U. Farooq, Z. Marrakchi, and H. Mehrez, Eds., New York, NY: Springer, 2012, pp. 7–48. doi: 10.1007/978-1-4614-3594-5_2.

[122] W. Vanderbauwhede and K. Benkrid, Eds., *High-Performance Computing Using FPGAs*. New York, NY: Springer, 2013. doi: 10.1007/978-1-4614-1791-0.

[123] T. M. Shami, A. A. El-Saleh, M. Alswaitti, Q. Al-Tashi, M. A. Summakieh, and S. Mirjalili, "Particle Swarm Optimization: A Comprehensive Survey," *IEEE Access*, vol. 10, pp. 10031–10061, 2022, doi: 10.1109/ACCESS.2022.3142859.

[124] B. Pang, E. Nijkamp, and Y. N. Wu, "Deep Learning With TensorFlow: A Review," *Journal of Educational and Behavioral Statistics*, vol. 45, no. 2, pp. 227–248, 2020, doi: 10.3102/1076998619872761.

[125] "PyTorch," PyTorch. Accessed: Dec. 12, 2024. [Online]. Available: https://pytorch.org/

[126] J. Moolayil, "An Introduction to Deep Learning and Keras," in *Learn Keras for Deep Neural Networks: A Fast-Track Approach to Modern Deep Learning with Python*, J. Moolayil, Ed., Berkeley, CA: Apress, 2019, pp. 1–16. doi: 10.1007/978-1-4842-4240-7_1.

[127] E. Cengil, A. Çınar, and E. Özbay, "Image classification with caffe deep learning framework," in *2017 International Conference on Computer Science and Engineering (UBMK)*, Oct. 2017, pp. 440–444. doi: 10.1109/UBMK.2017.8093433.

[128] "TensorFlow." Accessed: Jun. 10, 2024. [Online]. Available: https://www.tensorflow.org/

[129] "TensorFlow Lite | ML for Mobile and Edge Devices," TensorFlow. Accessed: Aug. 10, 2024. [Online]. Available: https://www.tensorflow.org/lite

[130] "PYNQ Board Overview - TU/e PYNQ Pages." Accessed: Jun. 09, 2024. [Online]. Available: https://pynq.tue.nl/general/pynq/

[131] S. Sun, J. Zou, Z. Zou, and S. Wang, Eds., *Experience of PYNQ: Tutorials for PYNQ-Z2*. in SpringerBriefs in Applied Sciences and Technology. Singapore: Springer Nature, 2023. doi: 10.1007/978-981-19-9072-4.

[132] "Vivado Overview," AMD. Accessed: May 10, 2024. [Online]. Available: https://www.xilinx.com/products/design-tools/vivado.html

[133] "Tensil," Tensil - Open source ML accelerators for the edge. Accessed: Apr. 19, 2024. [Online]. Available: https://www.tensil.ai/

[134] M. Isik, K. Inadagbo, and H. Aktas, "Design Optimization for High-Performance Computing Using FPGA," in *Information Management and Big Data*, J. A. Lossio-Ventura, E. Ceh-Varela, G. Vargas-Solar, R. Marcacini, C. Tadonki, H. Calvo, and H. Alatrista-Salas, Eds., Cham: Springer Nature Switzerland, 2024, pp. 142–156. doi: 10.1007/978-3-031-63616-5_11.

[135] A. Gundrapally, Y. A. Shah, N. Alnatsheh, and K. K. Choi, "A High-Performance and Ultra-Low-Power Accelerator Design for Advanced Deep Learning Algorithms on an FPGA," *Electronics*, vol. 13, no. 13, p. 2676, 2024, doi: 10.3390/electronics13132676.

[136] V. H. Kim and K. K. Choi, "A Reconfigurable CNN-Based Accelerator Design for Fast and Energy-Efficient Object Detection System on Mobile FPGA," *IEEE Access*, vol. 11, pp. 59438–59445, 2023, doi: 10.1109/ACCESS.2023.3285279.

[137] R. Kayalvizhi, H. Heartlin Maria, S. Malarvizhi, R. Venkatraman and S. Patil, "Hardware deployment of deep learning model for classification of breast carcinoma from digital mammogram images," *Medical & Biological Engineering & Computing*, vol. 61, no. 11, pp. 2843–2857, 2023, doi: 10.1007/s11517-023-02883-2.

[138] K. Inadagbo, B. Arig, N. Alici, and M. Isik, "Exploiting FPGA Capabilities for Accelerated Biomedical Computing," in *2023 Signal Processing: Algorithms, Architectures, Arrangements, and Applications (SPA)*, Sep. 2023, pp. 48–53. doi: 10.23919/SPA59660.2023.10274450.

[139] P. A. Kygonahalli, P. V. Deshpande, A. Hegde, P. Sasikumar, A. Gupta, P. Sharma, and J. Manikandan, "Design and Implementation of Real-Time Fire Segmentation Algorithm on PYNQ Z2 FPGA," in *2023 IEEE International Conference on Industrial Technology (ICIT)*, Apr. 2023, pp. 1–6. doi: 10.1109/ICIT58465.2023.10143090.

[140] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, and R. Buyya, "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software: Practice and Experience*, vol. 41, no. 1, pp. 23–50, 2011, doi: 10.1002/spe.995.

[141] H. Gupta, A. Vahid Dastjerdi, S. K. Ghosh, and R. Buyya, "iFogSim: A toolkit for modeling and simulation of resource management techniques in the Internet of Things, Edge and Fog computing environments," *Software: Practice and Experience*, vol. 47, no. 9, pp. 1275–1296, 2017, doi: 10.1002/spe.2509.

[142] N. K. Giang, M. Blackstock, R. Lea, and V. C. M. Leung, "Developing IoT applications in the Fog: A Distributed Dataflow approach," in *2015 5th International Conference on the Internet of Things (IOT)*, Oct. 2015, pp. 155–162. doi: 10.1109/IOT.2015.7356560.

[143] J. K. Zao, T. T. Gan, C. K. You, S. J. R. Méndez, C. E. Chung, Y. T. Wang, T. Mullen, and T. P. Jung, "Augmented Brain Computer Interaction Based on Fog Computing and Linked Data," in *2014 International Conference on Intelligent Environments*, Jun. 2014, pp. 374–377. doi: 10.1109/IE.2014.54.

[144] K. Hong, D. Lillethun, U. Ramachandran, B. Ottenwälder, and B. Koldehofe, "Mobile fog: a programming model for large-scale applications on the internet of things," in *Proceedings of the second ACM SIGCOMM workshop on Mobile cloud computing*, in MCC '13. New York, NY, USA: Association for Computing Machinery, Aug. 2013, pp. 15–20. doi: 10.1145/2491266.2491270.

[145] R. Mahmud, S. Pallewatta, M. Goudarzi, and R. Buyya, "iFogSim2: An extended iFogSim simulator for mobility, clustering, and microservice management in edge and fog computing environments," *Journal of Systems and Software*, vol. 190, p. 111351, 2022, doi: 10.1016/j.jss.2022.111351.

[146] "Tomato Leaf disease image classification." Accessed: Jun. 07, 2024. [Online]. Available: https://kaggle.com/code/rohanpatnaik/tomato-leaf-disease-image-classification

[147] "Pest Dataset." Accessed: Jun. 07, 2024. [Online]. Available: https://www.kaggle.com/datasets/simranvolunesia/pest-dataset

[148] "Weed-classification." Accessed: Jun. 07, 2024. [Online]. Available: https://www.kaggle.com/datasets/aminelaatam/weed-classification

[149] D. Marković, Z. Stamenković, B. Đorđević, and S. Ranđić, "Image Processing for Smart Agriculture Applications Using Cloud-Fog Computing," *Sensors*, vol. 24, no. 18, p. 5965, 2024, doi: 10.3390/s24185965.

[150] R. Tuli, "Analyzing Network performance parameters using wireshark," *International Journal of Network Security & Its Applications (IJNSA)*, vol. 15, no. 01, pp. 01–13, 2023, doi: 10.5121/ijnsa.2023.15101.

[151] G. Geetharamani and J. Arun Pandian, "Identification of plant leaf diseases using a nine-layer deep convolutional neural network," *Computers & Electrical Engineering*, vol. 76, pp. 323–338, 2019, doi: 10.1016/j.compeleceng.2019.04.011.

[152] A. Abbas, S. Jain, M. Gour, and S. Vankudothu, "Tomato plant disease detection using transfer learning with C-GAN synthetic images," *Computers and Electronics in Agriculture*, vol. 187, p. 106279, Aug. 2021, doi: 10.1016/j.compag.2021.106279.

[153] S. Hossain, M. Tanzim Reza, A. Chakrabarty, and Y. J. Jung, "Aggregating Different Scales of Attention on Feature Variants for Tomato Leaf Disease Diagnosis from Image Data: A Transformer Driven Study," *Sensors*, vol. 23, no. 7, Jan. 2023, doi: 10.3390/s23073751.

[154] M. Agarwal, A. Singh, S. Arjaria, A. Sinha, and S. Gupta, "ToLeD: Tomato Leaf Disease Detection using Convolution Neural Network," *Procedia Computer Science*, vol. 167, pp. 293–301, Jan. 2020, doi: 10.1016/j.procs.2020.03.225.

[155] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," Apr. 2015, doi: 10.48550/arXiv.1409.1556.

[156] M. Tan and Q. Le, "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks," in *Proceedings of the 36th International Conference on Machine Learning*, PMLR, May 2019, pp. 6105–6114. Accessed: Sep. 03, 2024. [Online]. Available: https://proceedings.mlr.press/v97/tan19a.html

[157] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," Apr. 2017, doi: 10.48550/arXiv.1704.04861.

[158] A. Howard, M. Sandler, B. Chen, W. Wang, L.-C. Chen, M. Tan, G. Chu, V. Vasudevan, Y. Zhu, R. Pang, H. Adam, and Q. Le, "Searching for MobileNetV3," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, Oct. 2019, pp. 1314–1324. doi: 10.1109/ICCV.2019.00140.

[159] K. Choi and G. E. Sobelman, "An Efficient CNN Accelerator for Low-Cost Edge Systems," *ACM Trans. Embed. Comput. Syst.*, vol. 21, no. 4, p. 44:1-44:20, 2022, doi: 10.1145/3539224.

[160] L. F. Bittencourt, J. Diaz-Montes, R. Buyya, O. F. Rana, and M. Parashar, "Mobility-Aware Application Scheduling in Fog Computing," *IEEE Cloud Computing*, vol. 4, no. 2, pp. 26–35, 2017, doi: 10.1109/MCC.2017.27.

Биографија

Душан Марковић рођен је у Чачку 1982. године. Основну школу завршио је у Прељини са одличним успехом, а затим и средњу Техничку школу у Чачку са одличним успехом. Основне академске студије уписао је на Техничком факултету (Факултету техничких наука у Чачку) 2001. године на студијском програму Електротехника смер Рачунарска техника. Наведени студијски програм који траје девет семестара завршио је 2006. године и тако стекао звање дипломирани инжењер електротехнике смер рачунарска техника. Студент је докторских студија на Факултету техничких наука у Чачку на студијском програму Електротехничко и рачунарско инжењерство, модул Рачунарска техника.

На Агрономском факултету у Чачку запослио се 2008. године прво на информатичким пословима. Након тога је био запослен у истој установи као сарадник у настави, а затим и као асистент. Тренутно је у звању асистента за ужу научну област Примењено рачунарство. У претходном периоду пружао је информатичку подршку многим активностима које су се спроводиле на Факултету.

Поред наведеног учесник је следећих пројеката.

Пројекат Технолошког развоја под називом: „Развој и моделовање енергетски ефикасних, адаптабилних, вишепроцесорских и вишесензорских електронских система мале снаге”, Министарство просвете и науке Републике Србије који је зепочет 2011. године, а који је касније пренет као институциони пројекат.

Био је учесник ТЕМПУС пројекта „Изградња капацитета српског образовања у области пољопривреде ради повезивања са друштвом, CaSA” (TEMPUS PROJECT: Building Capacity of Serbian Agricultural Education to link with Society, CaSA), 2013.

Учесник је пројекта програма Призма под називом „New biorational methods for stored seed pest control and protection: To serve and prevent”, акронима „SafeSeed”, који је започет 2024. године.

Образац 1

ИЗЈАВА АУТОРА О ОРИГИНАЛНОСТИ ДОКТОРСКЕ ДИСЕРТАЦИЈЕ

Изјављујем да докторска дисертација под насловом:

„ХАРДВЕРСКО-СОФТВЕРСКА ПОДРШКА ПРОЦЕСИРАЊУ НИЗОВА ПОДАТАКА У ОКВИРУ КОНЦЕПТА CLOUD – FOG РАЧУНАРСТВА“

представља *оригинално ауторско дело* настало као резултат *сопственог истраживачког рада*.

Овом Изјавом такође потврђујем:

- да сам *једини аутор* наведене докторске дисертације,
- да у наведеној докторској дисертацији *нисам извршио/ла повреду* ауторског нити другог права интелектуалне својине других лица,

У Чачку, 10.1.2025. године,

потпис аутора

Образац 2

ИЗЈАВА АУТОРА О ИСТОВЕТНОСТИ ШТАМПАНЕ И ЕЛЕКТРОНСКЕ ВЕРЗИЈЕ ДОКТОРСКЕ ДИСЕРТАЦИЈЕ

Изјављујем да су штампана и електронска верзија докторске дисертације под насловом:

„ХАРДВЕРСКО-СОФТВЕРСКА ПОДРШКА ПРОЦЕСИРАЊУ НИЗОВА ПОДАТАКА У ОКВИРУ КОНЦЕПТА CLOUD – FOG РАЧУНАРСТВА“

истоветне.

У Чачку, 10.1.2025. године,

потпис аутора

Образац 3

ИЗЈАВА АУТОРА О ИСКОРИШЋАВАЊУ ДОКТОРСКЕ ДИСЕРТАЦИЈЕ

Ја, Душан Б. Марковић,

☑ дозвољавам

☐ не дозвољавам

Универзитетској библиотеци у Крагујевцу да начини два трајна умножена примерка у електронској форми докторске дисертације под насловом:

„ХАРДВЕРСКО-СОФТВЕРСКА ПОДРШКА ПРОЦЕСИРАЊУ НИЗОВА ПОДАТАКА У ОКВИРУ КОНЦЕПТА CLOUD – FOG РАЧУНАРСТВА“

и то у целини, као и да по један примерак тако умножене докторске дисертације учини трајно доступним јавности путем дигиталног репозиторијума Универзитета у Крагујевцу и централног репозиторијума надлежног министарства, тако да припадници јавности могу начинити трајне умножене примерке у електронској форми наведене докторске дисертације путем *преузимања*.

Овом Изјавом такође

☑ дозвољавам

☐ не дозвољавам[1]

припадницима јавности да тако доступну докторску дисертацију користе под условима утврђеним једном од следећих *Creative Commons* лиценци:

---

[1]Уколико аутор изабере да не дозволи припадницима јавности да тако доступну докторску дисертацију користе под условима утврђеним једном од *Creative Commons* лиценци, то не искључује право припадника јавности да наведену докторску дисертацију користе у складу са одредбама Закона о ауторском и сродним правима.

1) Ауторство

2) Ауторство - делити под истим условима

3) Ауторство - без прерада

4) Ауторство - некомерцијално

5) Ауторство - некомерцијално - делити под истим условима

6) Ауторство - некомерцијално - без прерада[2]

У Чачку, 10.1.2025. године,

потпис аутора

[2]Молимо ауторе који су изабрали дадозволе припадницима јавности да тако доступну докторску дисертацију користе под условима утврђеним једном од *Creative Commons* лиценци да заокруже једну од понуђених лиценци. Детаљан садржај наведених лиценци доступан је на:http://creativecommons.org.rs/