VILNIUS GEDIMINAS TECHNICAL UNIVERSITY

Julius SKIRELIS

EDGE COMPUTING TIED IN ARTIFICIAL NEURAL NETWORK CLASSIFIERS

DOCTORAL DISSERTATION

TECHNOLOGICAL SCIENCES,
ELECTRICAL AND ELECTRONIC ENGINEERING (T 001)

Vilnius, 2021

Doctoral dissertation was prepared at Vilnius Gediminas Technical University in 2016–2021.

Scientific supervisor

Prof. Dr Dalius NAVAKAUSKAS (Vilnius Gediminas Technical University, Electrical and Electronic Engineering – T 001).

The Dissertation Defense Council of Scientific Field of Electrical and Electronic Engineering of Vilnius Gediminas Technical University:

Chairman

Prof. Dr Šarūnas PAULIKAS (Vilnius Gediminas Technical University, Electrical and Electronic Engineering – T 001).

Members:

Prof. Dr Algirdas BAŠKYS (Vilnius Gediminas Technical University, Electrical and Electronic Engineering – T 001),

Assoc. Prof. Dr Olga KURASOVA (Vilnius University, Informatics Engineering – T 007),

Dr Ilias PAPPAS (Agder University, Norway, Informatics – N 009),

Prof. Dr Vytautas URBANAVIČIUS (Vilnius Gediminas Technical University, Electrical and Electronic Engineering – T 001).

The dissertation will be defended at the public meeting of the Dissertation Defense Council of Electrical and Electronic Engineering in the Senate Hall of Vilnius Gediminas Technical University at **10 a. m. on 7 June 2021**.

Address: Saulėtekio al. 11, LT-10223 Vilnius, Lithuania.

Tel. +370 5 274 4956; fax +370 5 270 0112; e-mail: doktor@vilniustech.lt

A notification on the intend defending of the dissertation was send on 6 May 2021. A copy of the doctoral dissertation is available for review at Vilnius Gediminas Technical University repository <http://dspace.vgtu.lt>, at the Library of Vilnius Gediminas Technical University (Saulėtekio al. 14, LT-10223 Vilnius, Lithuania) and the Wroblewski Library of the Lithuanian Academy of Sciences (Žygimantų st. 1, LT-01102, Vilnius, Lithuania).

Vilnius Gediminas Technical University scientific book No. 2021-021-M
doi: 10.20334/2021-021-M

© Vilnius Gediminas Technical University, 2021

© Julius Skirelis, 2021

julius.skirelis@vilniustech.lt

VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS

Julius SKIRELIS

DIRBTINIŲ NEURONŲ TINKLŲ
KLASIFIKATORIAIS SUSIETA
KRAŠTŲ KOMPIUTERIJA

DAKTARO DISERTACIJA

TECHNOLOGIJOS MOKSLAI,
ELEKTROS IR ELEKTRONIKOS INŽINERIJA (T 001)

Vilnius, 2021

Disertacija rengta 2016–2021 metais Vilniaus Gedimino technikos universitete.

Vadovas

prof. dr. Dalius NAVAKAUSKAS (Vilniaus Gedimino technikos universitetas, elektros ir elektronikos inžinerija – T 001).

Vilniaus Gedimino technikos universiteto Elektros ir elektronikos inžinerijos mokslo krypties disertacijos gynimo taryba:

Pirmininkas

prof. dr. Šarūnas PAULIKAS (Vilniaus Gedimino technikos universitetas, elektros ir elektronikos inžinerija – T 001).

Nariai:

prof. dr. Algirdas BAŠKYS (Vilniaus Gedimino technikos universitetas, elektros ir elektronikos inžinerija – T 001),

doc. dr. Olga KURASOVA (Vilniaus universitetas, informatikos inžinerija – T 007),

dr. Ilias PAPPAS (Agderio universitetas, Norvegija, informatika – N 009),

prof. dr. Vytautas URBANAVIČIUS (Vilniaus Gedimino technikos universitetas, elektros ir elektronikos inžinerija – T 001).

Disertacija bus ginama viešame Elektros ir elektronikos inžinerijos mokslo krypties disertacijos gynimo tarybos posėdyje **2021 m. birželio 7 d. 10 val.** Vilniaus Gedimino technikos universiteto senato posėdžių salėje.

Adresas: Saulėtekio al. 11, LT-10223 Vilnius, Lietuva.

Tel. +370 5 274 4956; fax +370 5 270 0112; el. paštas: doktor@vilniustech.lt

Pranešimai apie numatomą ginti disertaciją išsiųsti 2021 m. gegužės 6 d.

Disertaciją galima peržiūrėti Vilniaus Gedimino technikos universiteto talpykloje <http://dspace.vgtu.lt>, Vilniaus Gedimino technikos universiteto bibliotekoje (Saulėtekio al. 14, LT-10223 Vilnius, Lietuva) bei Lietuvos mokslų akademijos Vrublevskių bibliotekoje (Žygimantų g. 1, LT-01102 Vilnius, Lietuva).

Abstract

The dissertation deals with traditional and artificial neural networks based intelligent classifiers, simulations and natural experiments are performed taking into account their specifics and possible specific applications in: cell colony image parametrization and image stitching. Research is conducted to evaluate the potential of developed algorithms and methods application to address edge computing challenges and are therefore examined in different network topologies: centralized, decentralized and distributed. Edge computing and its benefits are compared to the currently relevant but no longer meeting the needs technology – the cloud computing.

The object of the dissertation is the intelligent classifier for edge computing tasks. The aim of the work is to propose and investigate original, artificial neural network classifier based Internet of Things systems for the edge computing tasks. The dissertation is designated to contribute to the development of artificial neural network based edge computing solutions. Analytical review of artificial neural networks for edge computing is performed that underlies the work the relevance of the raise problem and explains the importance of research of the classifier application in edge computing.

The dissertation consists of an introduction, four chapters and general conclusions. In the first chapter essential knowledge and progress on artificial neural networks based classifiers application in edge computing technology is presented, the relationship between the edge computing and Internet of Things is defined, traditional and intelligent classification methods are discussed, self-organizing adaptive resonance theory based classifiers are analyzed and dissertation tasks are formulated. In the second chapter three simulation experiments are described with further analysis of their results revealing quantitative and qualitative Internet of Things characteristics, lastly, Internet of Things, edge computing and sensor network specifics are revealed. The third chapter describes the proposed cell colony images parameterization method, the experiments using constructed stand are performed comparing heuristic, support vectors, and adaptive resonance theory versions 1 and 2 classifiers by evaluation of classification quality based on statistical methods. In the fourth chapter a proposed fixed-structure artificial neural network classifier based adaptive resolution selection stitching system is described, further the arranged constructed stand and obtained experiment results are discussed.

The main results of the thesis were published in 9 scientific publications: three of them were printed in peer-reviewed scientific journals, two of which are listed in *Clarivate Analytics Web of Science* and one of them is with impact factor, six articles – in conference proceedings. The research results were presented in 11 scientific conferences.

Reziumė

Disertacijoje nagrinėjami tradiciniai ir dirbtiniais neuronų tinklais grįsti išmanūs klasifikatoriai, atliekami imitaciniai ir natūriniai eksperimentai siekiant išsiaiškinti jų specifikas ir galimus taikymus specifinėse srityse – ląstelių kolonijų vaizdų analizei ir vaizdų sandaros uždaviniui. Tyrimai atliekami vertinant galimą sukurtų algoritmų ir metodų taikymą kraštų kompiuterijos uždaviniams spręsti, todėl nagrinėjami skirtingi požiūriai tinklų topologijos prasme, centralizuotas, decentralizuotas ir paskirstytas, o kraštų kompiuterija ir jos privalumai lyginami su šiuo metu aktualia, tačiau nebetenkinančia poreikių debesų kompiuterijos technologija.

Disertacijos tyrimo objektas yra išmanus klasifikatorius skirtas kraštų kompiuterijos uždaviniams spręsti. Darbo tikslas – pasiūlyti ir ištirti originalias, dirbtinių neuronų tinklų klasifikatoriais grįstas daiktų interneto sistemas skirtas kraštų kompiuterijos uždaviniams spręsti. Darbas skirtas prisidėti vystant dirbtiniais neuronų tinklais grįstus kraštų kompiuterijos sprendimus. Atliekama analitinė dirbtinių neuronų tinklų kraštų kompiuterijoje apžvalga, kurios pagrindu yra pagrindžiamas darbe iškeltos problemos aktualumas ir iškeliamas svarba tirti klasifikatorių taikymą kraštų kompiuterijoje.

Disertaciją sudaro įvadas, keturi skyriai ir bendrosios išvados. Pirmame skyriuje pristatomos esminės žinios ir pažanga apie kraštų kompiuterijoje taikytinus dirbtinius neuronų tinklus ir jais grįstus klasifikatorius, apibrėžiamas sąryšis tarp kraštų kompiuterijos technologijos ir daiktų interneto, aptariami tradiciniai ir išmanūs klasifikavimo metodai, analizuojami saviorganizuojantys adaptvyviojo rezonanso teorija grįstų klasifikatoriai, bei formuluojami disertacijos uždaviniai. Antrame skyriuje aprašomi trys imitaciniai eksperimentai, jų rezultatų analizės metu išsiaiškintos daiktų interneto kiekybinės ir kokybinės charakteristikos, atkleistos daiktų interneto, kraštų kompiuterijos ir jutiklių tinklų specifikos. Trečiame skyriuje aprašomas sukurtas ląstelių kolonijų vaizdų parametrizavimo metodas, sukonstruotu stendu atliekami natūriniai eksperimentai, tarpusavyje palyginami euristinis, atraminių vektorių ir adaptvyviojo rezonanso teorijos 1 ir 2 versijos klasifikatoriai, atliekama jų klasifikavimo kokybės vertinimas statistiniais metodais. Ketvirtame skyriuje aprašoma pasiūlyta pastoviosios struktūros dirbtinių neuronų tinklo klasifikatoriumi grįsta vaizdų sandaros sistema, sukurta bandomajame stende atliekami natūriniai vaizdo skyros adaptvyvaus parinkimo metodo eksperimentai ir atliekama gautų rezultatų analizė.

Pagrindiniai disertacijos rezultatai paskelbti 9-iose moksliniuose straipsniuose, iš kurių 3 atspausdinti recenzuojamuose mokslo žurnaluose. Iš jų 2 straipsniai paskelbti mokslo žurnaluose, įtrauktuose į *Clarivate Analytics Web of Science* duomenų bazę, vienas jų yra su citavimo indeksu. Kiti 6 straipsniai – konferencijų medžiagoje. Rezultatai viešinti 11 mokslinių konferencijų.

Notations

In General

a	– scalar (number);
$\mathbf{a}$	– vector (column);
$\mathbf{A}$	– matrix;
$\mathcal{A}$	– item in a dataset;
$\mathcal{A}$	– dataset;
$\cdot^{(i)}$	– main index i (layer);
$\cdot_i$	– supplementary index;
$\cdot_f$	– indicator of main feature “f”;
$\cdot^\star$	– indicator of the best (calculated) value;
$\cdot^\star$	– indicator of important area or point in a image;
<i>title</i>	– title of computer software or chip family;
<i>text</i>	– emphasized phrase.

Symbols

$\widehat{D}_f(\mathcal{T}, \mathcal{L})$ – estimated ($f \in \{\text{m}, \text{v}\}$: mean, variance) service duration of investigated network topology $\mathcal{T}$ with link parameter set $\mathcal{L}$;

E_{MAE}	– mean absolute error;
E_{RMSE}	– root mean square error;
$\widehat{E}_f^s$	– estimated ($f \in \{\text{m}, \text{d}, \text{v}\}$: mean, standard deviation, variance) number of events with state $s \in \{\text{S}, \text{F}, \text{T}\}$: success, failure, total;
$f^\top$	– maximum frequency of (circuit) operation;
f_{val}	– value generation frequency;
$\Phi(\cdot)$	– neuron activation function;
θ_i	– boundary parameter of reference frame i ;
i, j, h	– input, output and experiment indices;
λ	– wavelength;
l	– index of ANN layer;
L	– a total number of ANN layers;
n	– time (sample) index;
$N, N^{(l)}$	– a total number of: samples and l th layer neurons;
N_t	– total number of elements of type t ;
ξ	– a random number;
P	– power consumption;
ρ	– vigilance parameter;
$s_{ij}^{(l)}(n)$	– output signal of synapse connecting i th with j th neuron;
τ	– link delay (latency);
$\mathcal{T}$	– a set of network topologies;
ψ	– attenuation factor;
ϖ	– link bandwidth;
$w, \boldsymbol{w}$	– weights or system parameters.

Operators and Functions

$\equiv$	– equivalence;
$\rightarrow$	– substitution;
$\triangleq$	– definition;
$ \cdot $	– absolute value (modulus);
$\ \cdot\ $	– norm (length);
$\widehat{\cdot}$	– estimate;
$\bar{\cdot}$	– mean;
$\max(\cdot)$	– maximum;
$\arg \max(\cdot)$	– maximum of a given arguments;
$\min(\cdot)$	– minimum;
$\text{std}(\cdot)$	– standard deviation;
$\text{var}(\cdot)$	– variance.

Abbreviations

ADC	– analog-to-digital converter;
AI	– artificial intelligence;
ANN	– artificial neural network;
API	– application programming interface;
ARM	– advanced RISC machines;
ART	– adaptive resonance theory;
ART1	– adaptive resonance theory 1;
ART2	– adaptive resonance theory 2;
ART3	– adaptive resonance theory 3;
ASIC	– application-specific integrated circuit;
CDN	– content delivery network;
CNN	– convolutional neural network;
CPU	– central processing unit;
DNN	– deep neural network;
DOR	– diagnostic odd ratio;
DSP	– digital signal processor;
EC	– edge computing;
ECD	– edge computing device;
FNN	– fuzzy neural network;
FPGA	– field-programmable gate array;
GPU	– graphics processing unit;
IO	– input-output;
IoT	– internet of things;
IoV	– internet of vehicles;
MANET	– mobile ad-hoc network;
MEC	– mobile edge computing;
PC	– personal computer;
RAM	– random access memory;
RF	– radio frequency;
RNN	– recurrent neural network;
ROC	– receiver operating characteristic;
SoC	– system on chip;
SOM	– self organizing map;
SVM	– support vector machine;
VHDL	– very high-speed integrated circuit HDL;
VM	– virtual machine;
WSN	– wireless sensor network.

Keywords¹

IEEE Taxonomy	Part(s)
Communications technology	
Communication systems	
Internet – <i>Internet of Things</i>	1.1 2.2
Internet – <i>Internet topology</i>	1.1 2.2 2.3 4.2
Protocols – <i>Routing protocols</i>	2.4
Computational and artificial intelligence	
Artificial intelligence	
Computational intelligence	
Fuzzy systems – <i>Fuzzy neural networks</i>	1.2 4.2
Neural networks	
Artificial neural networks – <i>Self-organizing feature maps</i>	1.3
Artificial neural networks – <i>Feedforward neural networks</i>	1.2 3.3 4.2
Computers and information processing	
Computer Science	
Programming – <i>Performance analysis</i>	2.2 2.3 2.4 3.4 4.4
Software	
Software packages – <i>MATLAB</i>	2.2 2.3 2.4 3.4 4.3
Systems engineering and theory	
System analysis and design	
Distributed processing – <i>Edge computing</i>	1.1 1.2 2.3 4.2

¹Keywords (assigned to parts) are structured and in-line with the latest IEEE taxonomy (see <https://www.ieee.org/documents/taxonomy_v101.pdf>).

Contents

INTRODUCTION	1
Problem Formulation	1
Relevance of the Thesis	2
The Object of the Research	4
The Aim of the Thesis	4
The Objectives of the Thesis	4
Research Methodology	4
Scientific Novelty of the Thesis	5
Practical Value of the Research Findings	6
The Defended Statements	6
Approval of the Research Findings	7
Structure of the Dissertation	8
Acknowledgements	8
1. ANALYTICAL REVIEW ON ARTIFICIAL NEURAL NETWORKS IN EDGE COMPUTING	9
1.1. Edge Computing Technology for Internet of Things	10
1.1.1. Topological Aspects	13
1.1.2. Software Means	14
1.1.3. Hardware Tools	15
1.2. Classifiers in Edge Computing	18

1.2.1. Fields of Application	18
1.2.2. Traditional Classification Methods	19
1.2.3. Intelligent Classifiers	20
1.3. Unsupervised Artificial Neural Networks for Classification	22
1.3.1. General Analysis of Paradigms	22
1.3.2. Adaptive Resonance Theory	23
1.3.3. Deep Learning	25
1.3.4. Self Organizing Maps	26
1.3.5. Support Vector Machines	27
1.4. Artificial Neural Network based Classifier Implementation	30
1.4.1. Architecture Specifics	30
1.4.2. Pipelining	31
1.4.3. Enabling Computations	31
1.4.4. Distribution	34
1.5. Conclusions of the First Chapter and Formulation of the Thesis Objectives	35
2. INTERNET OF THINGS CHARACTERISTICS	37
2.1. Analysis Directions	38
2.2. Internet of Things Specifics	40
2.2.1. Investigation Setup	40
2.2.2. Investigation Results	43
2.3. Edge Computing Specifics	49
2.3.1. Investigation Setup	50
2.3.2. Investigation Results	53
2.4. Sensor Network Specifics	57
2.4.1. Investigation Setup	57
2.4.2. Investigation Results	61
2.5. Conclusions of the Second Chapter	63
3. SELF-ORGANIZING CLASSIFIER FOR CELL COLONIES COUNTING	65
3.1. Application Domain and Approach	66
3.2. Cell Colonies Image Parameterization Technique	67
3.2.1. General Work-Flow	67
3.2.2. Specifics in Image Registration	69
3.2.3. Petri Dish Area Determination	69
3.2.4. Cell Colonies Detection	70
3.2.5. Parameterization of Cell Colonies	70
3.3. Considered Intelligent Classifiers	72
3.3.1. Adaptive Resonance Theory 1	72
3.3.2. Adaptive Resonance Theory 2	75

3.3.3. Support Vector Machines	77
3.4. Experimentation Results	78
3.4.1. Investigation Procedure	79
3.4.2. Reliability of Cell Colony Counting	82
3.4.3. Efficiency of Cell Colony Parameterization	84
3.5. Conclusions of the Third Chapter	85
4. REAL-TIME CLASSIFIER FOR IMAGE STITCHING	87
4.1. Application Domain and Approach	88
4.2. Image Decentralized Stitching Procedure	89
4.2.1. Specifics in Work-Flow	89
4.2.2. Adaptive Image Resolution Selection Technique	91
4.2.3. Algorithm Deployment on Edge Devices Routine	93
4.3. Investigation Setup	94
4.3.1. Experimental Environment	94
4.3.2. Controllable and Varied Parameters	94
4.4. Experimentation Results	95
4.4.1. Investigation Procedure	95
4.4.2. Performance Evaluation	95
4.5. Conclusions of the Fourth Chapter	98
GENERAL CONCLUSIONS	99
REFERENCES	101
LIST OF SCIENTIFIC PUBLICATIONS BY THE AUTHOR ON THE TOPIC OF THE DISSERTATION	115
SUMMARY IN LITHUANIAN	117
SUBJECT INDEX	133
ANNEXES ¹	137
Annex A. Declaration of Academic Integrity	138
Annex B. The Co-authors' Agreements to Present Publications Material in the Dissertation	139
Annex C. The Copies of Scientific Publications by the Author on the Topic of the Dissertation	145

¹The annexes are supplied in the enclosed compact disc.

Introduction

Problem Formulation

Over the past decades centralized cloud computing technology has been greatly developed and is successfully applied to high cost and complex solutions. The rapid development of mobile Internet and Internet of things (IoT) applications generating enormous amounts of data to be processed by cloud servers (Abdelaziz *et al.* 2016) inevitably raised the problems and faced serious challenges on solving high network latency and high network load problems (Chen *et al.* 2018b). These problems are caused by the delay of data transfers, which are unacceptable for particular tasks and can cause bandwidth saturation.

The rapid number increase of high powered, high efficiency and low cost mobile embedded devices (more than 70 billions of devices are expected to be connected wirelessly by 2025) creates a pretext to use up available resources to solve the increased loads on the network and perform the offloading. For that, the shift of the functions of centralized cloud to edge devices needs to be accomplished (Corcoran, Datta 2016; Karim *et al.* 2016). The shift therefore raises the new problems of optimal topology architecture selection, efficient data preprocessing, scheduling of the tasks on decentralized edge devices and the security aspects (Georgakopoulos *et al.* 2016; Li *et al.* 2018). It is currently unknown what in edge network configurations has the greatest influence on network performance, however it is known that edge has to be intelligent – modern machine learning approaches have to be applied (Zhang *et al.* 2020a).

The classification task is found to be one of the most prominent candidates to integrate into the edge to support its intelligence. Unfortunately, limitations of the computational capabilities edge devices prevent complex algorithms to be implemented at the edge (Intharawijitr *et al.* 2017), thus relatively lightweight classifiers is the only option. Artificial neural networks (ANN) based classifiers are found to be the most suitable solution for strict edge computing network limits, however their implementation techniques, performance measures and energy-efficiency remains unknown. It is also unknown whether it is possible to develop self-organized classifier to solve edge computing (EC) specific tasks.

Moreover, the raw data transfers to cloud not only causes increased system response time, but also exposes the private data, therefore requiring additional complex encryption mechanisms growing the loads even more. The security aspect is solved in EC naturally – the data is only transferred locally, that way significantly lowering the risk of the data being compromised and requires minimal encryption mechanisms to be involved.

Thus here the *problem* of the lack of applied knowledge and techniques for the artificial neural network classifier use in effective edge computing is tackled.

Relevance of the Thesis

General needs of the society shows the high demand of IoT related services (Jutila 2016). It includes a variety of different areas, e.g., smart cars, houses and cities, biomedicine, industry (Choochotkaew *et al.* 2019; Tran *et al.* 2017; Wang *et al.* 2016), most of these services currently are cloud computing based. Studies has shown that increasing complexity and amount of services no longer satisfies acceptable service times, therefore results in poor user experience using those services. It is noted that service times are no longer improved by increasing computation capabilities of data centers, the problem arise from the network itself – even if data center process the data instantly, it takes significant amount of time for that data to reach the end user (Sabella *et al.* 2016). It is getting harder and harder to satisfy the real-time criteria for particular applications. Hence, content delivery networks (CDN), fog and edge computing technologies takes places to solve those problems (Salman *et al.* 2015). In recent years the cost of edge devices decreased drastically, enabling them to be used massively.

Based on an reviewed articles, it is found that adaptive EC implementations are not sufficiently investigated, leading to insight that most of current EC systems are fixed architecture and are only suitable for a particular task to solve. Dynamic EC systems are not only able to adapt to the data in the network, but also allows to accelerate the processing of the data through parallelization of computing machines based on the states of the network (Kubota *et al.* 2016; Zhang *et al.* 2019).

Studies have shown that decentralization in EC is one of the hardest part. It includes the synchronization between nodes, data protocols matching, heterogeneous devices matching tasks.

Literature analysis of the ANN classifiers used in cloud, fog or edge computing revealed the fact that adaptive resonance theory (ART) paradigms, being medium computationally complex and possessing attractive self-organizing features, have not been used for EC technology. Despite the method being known for years, recently the adoption in modern applications like securing databases (Brahma, Panigrahi 2021) and generalization of autonomous intelligence (Grossberg 2021) is regained. As well as, Fuzzy neural networks' (FNN) low computational complexity and unique opportunity to inscribe *a priori* knowledge by means of membership functions and fuzzy rules, are promising and looks attractive for EC specific device implementations. Its relevance is confirmed by successful applications in modern wearable technologies for e-health monitoring (Malcangi, Nano 2021) and intelligent embedded space manipulators (Yao 2021).

Cell colonies counting is an important subset of cytometry research (Herberg *et al.* 2015; Lambert *et al.* 2017), demanding high throughput analysis with accurate results at high speed. For parameterization of biological sample images (Horrer *et al.* 2013) intelligent algorithms (Navakauskas *et al.* 2014) bring benefits in both of time consumed and overall cost of investigation. Cell colonies counting is time-intensive and time-sensitive process, since there is limited time range to examine the samples before they changes their state (Choudhry 2016). The aim to create fully automated real-time cell colonies counting system is prevailing for years. Therefore, the contribution in this dissertation is provided by proposing self-organizing classifier based system.

Image stitching is a computationally heavy process thus requires comparatively many hardware resources (Seo *et al.* 2019). At the same time it is the basis for panoramic photography feature available on most modern mobile phones. The trend of multiple cameras use per device highlights the relevance of efficient image stitching for augmented virtual reality applications (La Salandra *et al.* 2020) and research on this method optimization.

In order to solve the previously stated problem in a relevant way such *main hypothesis* were raised and proven:

- the clustering in EC gateway serving middle size user pool can decrease mean service time in comparison to conventional techniques,
- the structural adaptiveness of ART classifiers in comparison to SVM, as good representative of fixed structure classifiers, can improve classification quality in cell colonies counting application,
- the use of a FNN in decentralized EC can improve node performance in image stitching application.

The solution of the problem is two-fold in line with the Lithuania Smart Specialization (BSR 2020). Firstly, according to the priority “ICT – Information and Communication Technologies” theme “IoT – Internet of Things”, the tasks of nature-inspired models are implemented by EC hardware and services. Secondly, according to the priority “Health technologies and biotechnology” theme “Advanced medical engineering for early diagnosis and treatment”, the knowledge generated by the tasks are formalized into new mathematical models, potentially suitable for solution-support systems compatible with the IoT.

The Object of the Research

The *object* of the research is the intelligent classifier for edge computing tasks. The following *main aspects* of the research object are investigated in the thesis: application techniques and characteristics.

The Aim of the Thesis

To propose and investigate original Internet of Things systems, based on artificial neural network classifiers, for solving the tasks of edge computing.

The Objectives of the Thesis

In order to solve the stated problem and reach the aim of the thesis the following *objectives* are formulated:

1. Investigate IoT quantitative and qualitative characteristics and reveal EC technology specifics.
2. Simulate the adaptive resonance theory grounded systems and determine the EC environment influences on the self-organizing classifiers complexity and performance.
3. Implement a fixed structure ANN classifier based system for image stitching purposes and experimentally investigate the EC infrastructure influence on its energy-efficiency.

Research Methodology

The aim of the dissertation is real, it was formulated according to the relevant and widely applied IoT and EC technologies tied with intelligent machine learning

solutions to solve the technology specific tasks through known and proposed new original techniques.

The dissertation aims to reveal clear [EC](#) quantitative and qualitative characteristics and reveal technology specifics, propose technological solutions to solve relevant and practically important tasks which lacks scientific investigations, therefore it may lead to a positive socio-economic effect.

Clear provisions and known indisputable methods are used in the dissertation, raised hypotheses are analyzed, evaluated and proved by experimental studies. These studies are aimed to obtain improved results compared to existing solutions – improved characteristics are reliability and energy-efficiency.

The dissertation investigates and proposes new techniques based on a world-wide knowledge and the following theories:

- machine-to-machine (M2M) communication systems ([IoT](#) and [EC](#));
- [ANN](#).

Known and original techniques of digital image processing and feature extraction, data classification and clustering, [ANN](#) self-organization and optimization, statistical analysis are involved. Techniques of [ART1](#), [ART2](#), [FNN](#) and [SVM](#) classifiers recall and training, also k -means clustering are adopted and implemented.

Original data sets of computer network topologies, network parameters and cell colony images for the experiments are recorded. Two types of controlled experiments are performed – computer simulations and test bench experiments. The computer simulations are carried out with the use of *MATLAB* 2020a and *EdgeNetworkCloudSim* software. Experiments with real hardware are performed on specifically built two test benches, both including digital cameras. The intelligent electronic systems are deployed on [EC](#) devices with the help of *Kubernetes* containers distribution system (Rausch *et al.* 2021). Additionally, the software packages of *OpenCV* and *WebSockets* are used in these experiments.

Scientific Novelty of the Thesis

1. Edge computing topological, parametrical, performance and resilience to external disturbances specifics are revealed, enabling development of effective systems.
2. A work distribution technique based on k -means clustering is proposed, outperforming regular technique in mean service time (in the range from 600 to 1200 simultaneous users).
3. A novel cell colonies image parameterization technique is developed and it is experimentally confirmed that its use with [ART1](#) or [ART2](#) classifiers is more precise in comparison to the use of [SVM](#) or heuristic classifiers.

4. The original adaptive image resolution selection technique based on FNN is developed, decreasing at least 2 times a total service time of distributed image stitching task on EC network.

Practical Value of the Research Findings

Implementation of unsupervised ANN classifier for EC technology is proposed. Practical application for medicine – real-time cell colonies classifier and multimedia – distributed image stitching are presented. Investigation of computer network configurations, network parameters influence on packet transmission probability and communication protocols leads to practically examined and verified recommendations for EC networks architects.

Research results were used taking part in the following scientific projects:

- Technological development project “Smart Optical Resistance Testing System for Laser Technology sector” (Agency for Science, Innovation and Technology, No. TPP-01-011, 2017–2018);
- Innocheck “Research on Intelligent methods based electronic systems implementation on Field Programmable Gate Arrays” (Agency for Science, Innovation and Technology, No. 1981, 2017–2018);
- Future Technology Program project “Electro-magnetoporation mediated biocontrol of the microgravity affected and skin infections causative microorganisms” (Research Council of Lithuania, No. LAT-16001, 2016–2018).

The Defended Statements

1. In Internet of Things network topology with 13 nodes and 15 edges the network delay parameter has the greater impact than bandwidth, with approaching to fully connected topology with 13 nodes and 16 edges grants greater probability of successfully receiving the packet.
2. By the use of Adaptive Resonance Theory 2 classifier it is possible to classify images with cell colonies with not less than 14% better accuracy than unsupervised SVM classifier.
3. Proposed technique for image stitching based on fuzzy neural network in decentralized edge computing grants increase of at least 2 frames/s in average compared to non-adaptive technique, on images resolutions up to 1280×800 pixels.

Approval of the Research Findings

The research results are published in *9 scientific publications*: an article in a peer-reviewed scientific journal with impact factor listed in *Clarivate Analytics Web of Science* (Skirelis, Navakauskas 2020), an article in a peer-reviewed scientific journal without impact factor listed in *Clarivate Analytics Web of Science* (Skirelis, Patlins, Kunicina, Romanovs, Zabasta 2019), an article in peer-reviewed scientific journal listed in other databases (Skirelis 2018), and six publications in international conference proceedings listed in *Clarivate Analytics Web of Science* (Skirelis, Navakauskas, 2019; Skirelis, 2019; Skirelis, Navakauskas, 2018a; Skirelis, Navakauskas, 2018b; Skirelis, Navakauskas, 2017a; Skirelis, Navakauskas, 2017b).

The main results of the thesis are presented in *11 scientific conferences*:

- Skirelis, J. 2020. Embedded Classification in Industry 4.0. 20th Int. Symp. on “Topical problems in the field of Electrical and Power Engineering” and “Doctoral School of Energy and Geotechnology III”. Tartu, Estonia;
- Skirelis, J.; Navakauskas, D. 2019. Network Configuration Impact on IoT Performance. 11th Int. WS on “Data Analysis Methods for Software Systems”. Druskininkai, Lithuania;
- Skirelis, J.; Navakauskas, D. 2019. Classification at the Edge: Implementation and Performance Evaluation. 7th Int. WS on “Advances in Information, Electronic and Electrical Engineering”. Liepaja, Latvia;
- Skirelis, J. 2019. Blockchain Hands-on Experience in MATLAB. Int. Conf. “Electrical, Electronic and Information Sciences”. Vilnius, Lithuania;
- Skirelis, J. 2019. Edge Computing Gateway for Industrial Internet of Things. 19th Int. Symp. on “Topical problems in the field of Electrical and Power Engineering” and “Doctoral School of Energy and Geotechnology II”. Toila, Estonia;
- Skirelis, J. 2018. Structural offload in IoT by Artificial Neural Networks. 24th IFIP World Computer Congress. Poznan, Poland;
- Skirelis, J.; Navakauskas, D. 2018. Classifier based Gateway for Edge Computing. 6th Int. WS on “Advances in Information, Electronic and Electrical Engineering”. Vilnius, Lithuania;
- Skirelis, J.; Navakauskas, D. 2018. Parameterization of Cell Colonies Images by ART2 Classifier. Int. Conf. “Electrical, Electronic and Information Sciences”. Vilnius, Lithuania;
- Skirelis, J.; Navakauskas, D. 2017. Edge Computing in IoT: Preliminary Results on Modeling and Performance Analysis. 5th Int. WS on “Advances in Information, Electronic and Electrical Engineering”. Riga, Latvia;

- Skirelis, J. 2017. Classifier place in Edge Computing for Internet of Things. Int. Conf. of Young Scientists “Science – Future of Lithuania: Electronics and Electrical Engineering”. Vilnius, Lithuania;
- Skirelis, J.; Navakauskas, D. 2017. On Parameterization of Images with Cell Colonies: [ART](#) Classifier Approach. Int. Conf. “Electrical, Electronic and Information Sciences”. Vilnius, Lithuania.

Moreover the results of the research are presented in international seminar – Skirelis, J. Edge Computing for [IoT](#) through Self-organizing Classifier. WS on “IEEE Latvia–Lithuania Section Cooperation”, 2019, Riga, Latvia.

Structure of the Dissertation

The dissertation contains: introduction, four chapters, general conclusions, summary in Lithuanian, list of references with separately presented list of publications by the author. The lists of symbols, abbreviations, keywords and subject index are presented. The dissertation consists of 138 pages, where: 33 displayed equations, 56 figures, 13 tables and 4 algorithms are presented. In total 156 references are cited in the dissertation.

Acknowledgements

I would like to express the greatest appreciation to Prof. Dr Dalius Navakauskas, my supervisor, for his dedication, professionalism and extreme patience by helping to perform the research. Big thanks to all the staff of the Department of Electronic Systems at Vilnius Gediminas Technical University for assistance in all kinds of a problems during this research. Thank you, the administration of Vilnius Gediminas Technical University for the ability to present my own and gain additional knowledge in international conferences, also the Department of Modelling and Simulation at Riga Technical University for acquiring a good practice during my internship.

Special thanks to Assoc. Prof. Dr Artūras Serackis, for directing me to interesting scientific field – artificial neural networks, and longtime colleague Paulius Tumas for encouraging me to start PhD studies. Thanks to the great circle of friends for charging up with positive vibes.

Separate gratitudes for my parents and my sweetheart Justė, for continuous support and comprehensive assistance to keep up on track. Lastly, admiration to my beloved grandmother who inspired me to seek higher education and unfortunately just passed away, leaving me strong to reach the end of the research.

Analytical Review on Artificial Neural Networks in Edge Computing

Internet of things technology is becoming increasingly relevant in daily activities. Effective data processing across distributed **IoT** devices is essential for the research and the application in the industry (Zalieckaitė, Žilinskas 2016). The need of distributed **AI**, databases and cloud computing motivates to optimize big data processing and transferring problems. Centralized cloud computing technology reaches its limits as enormous amount of data is being transferred between the servers and endpoints (Kubota *et al.* 2016). This problem is aimed to be solved with decentralized technology – edge computing (**EC**), where processing tasks are moved to the edge, ensuring non-expensive and secure data transmission characteristics (Georgakopoulos *et al.* 2016). The increasing number of devices with an Internet connection available, e.g., in 2025 it is predicted to be over 75 billion devices, the big data transfers problem becomes relevant (Weisong, Dustdar 2016). The application of **EC** technology, at least on the part of these devices, would significantly help to cope the network congestion and offload the data centers (Sabella *et al.* 2016). Most of the mobile devices these days does not utilize their resources in full capacity, thus it is great opportunity to allocate their resources for distributed computing tasks without affecting the specialized functions of the devices.

EC concept is relatively new in the field of the **IoT**, and its introduction resulted in the implementation of the real-time decentralized systems (Tran *et al.* 2017), e.g., smart cars, houses and cities (Choochootkaew *et al.* 2019), biomedical devices (Wang *et al.* 2016), and even particular industry applications. It is noted that this technology also partly solves problems of the static network connections –

part of data centers functionality is transferred to the **ECDs**. The **EC** technology also provides a higher level of data security while maintaining the same price of the data (Vahid Dastjerdi, Buyya 2016), compared to the cloud computing technology where the data is transferred to geographically remote data centers, the data has a risk of man-in-the-middle attacks. **EC** technology features a low delay, and a better overall quality of service (Alrawais *et al.* 2017; Intharawijitr *et al.* 2017) because the raw data between devices is transferred in local network. Such locality allows for **ECDs** to exchange information of their states instantly, forming autonomous ecosystem. Compared to cloud computing technology, **EC** technology utilizes the same physical structure of the network, this allows seamless migration to **EC** without essentially changes in network topology.

In order to achieve optimal **EC** implementation, it is crucial to introduce intelligent edge devices management system, one of the appropriate is based on **ANN** techniques. Thus, in the chapter a systematic literature review on **ANN** use in **EC** is presented. The following questions are analyzed in separate sections: Section 1.1 – what are implementation options and place of the **EC** technology in the **IoT** stack?; Section 1.2 – how important and where classification methods are used in **EC**?; Sections 1.3 and 1.4 – what **ANN** paradigms and their implementation aspects are relevant to use in **EC** technology? Lastly, the conclusions and formulation of the thesis objectives are presented in Section 1.5.

The summarized research results of this chapter are published in author publication (Skirelis, Navakauskas 2020) and are presented in annual national scientific conference “Science – Future of Lithuania” (Vilnius, 2017).

1.1. Edge Computing Technology for Internet of Things

The **EC** is a model optimizing cloud computing (Barcelo *et al.* 2016), where the data processing is performed as close to the edge as possible. Fig. 1.1 illustrates the structure of the **EC**, its interaction with cloud computing, as well as the principle of the data locality. These transformations are performed in software and hardware, with the aim to increase efficiency and security levels of the **IoT** systems. The figure also emphasizes the fact that **ANN** paradigms are useful at all levels of data processing. The **ANN** tasks vary from sensors data processing, fusion and feature extraction (edge – fog) to collaborative processing, model fusion and autonomous decision making (fog), and to model building, knowledge extraction and intelligent process control (cloud).

For the motivation purposes and willing to be brief let us analyze just one **EC** application area related to a smart transportation. It is known, that in last

Fig. 1.1. ANNs implementation and distribution scheme in cloud computing (servers), fog computing (gateways) and edge computing devices (ECD)

decade the number of vehicles has increased enormously, thus road systems are pushed to their capacity limits. This results in an increased occurrence of traffic congestions and car accidents, especially in urban cities. To overcome these issues, new paradigm evolved – the Internet of connected Vehicles (**IoV**). It described interaction among vehicles (M2M) with an humans intervention. In the **IoV** environment, vehicles are incorporated with variety of sensors, e.g., intelligent cameras, proximity sensors, actuators. Usually these vehicles incorporates transmitters and receivers that can connect to other vehicles and to the city infrastructure. The real-time traffic information is then collected from the vehicles to evaluate the vehicle position, vehicle driving status and identify the vehicle itself. Intelligent **IoV** services allow to reorganize the traffic in real-time to minimize or prevent the congestion completely. With the current solutions, most vehicles does not equip hardware to perform data processing and data storage, thus these computing tasks are performed in remote cloud data centers, which currently are common relay nodes in **IoV**. Despite the on-demand processing ability and barely no limits in resources of cloud platforms, traditional centralized cloud computing has the problem of high transmission delays and unstable connections in the **IoV** (and other **IoT**) network (Chen *et al.* 2016; Xu *et al.* 2019). Specialized technology – mobile edge computing (**MEC**) has the potential to provide a solution for those problems, by shifting cloud services to the edge of the radio access network, the computing tasks of the vehicles are offloaded to the **ECDs** that are in close proximity to the vehicles or in the vehicles itself rather than remote cloud platforms (Chen *et al.* 2018b). By offloading the tasks to the **ECDs**, the quality of experience of the drivers could be greatly improved.

The shift of cloud computing tasks to **ECDs** in **IoV** comes with an potential issues of security (Wang *et al.* 2019; Xu *et al.* 2019). The vehicles transmits sensitive data (speed, location, states of surrounding environment) and the leak of this data could lead to potential disaster, especially for self-driving vehicles. **ECD** hijacking, man-in-the-middle attacks or any other prohibited interaction on these devices may create a direct path for the hackers to control the vehicle or create a undesirable conditions for accident to occur. The leak of the vehicle position may also ease the criminals to trace the vehicle and then perform the criminal activities. It is obvious, that **IoV** based systems has to be standardized according to the possible security flaws and in mind of privacy information avoiding any data leakage.

Due to nature of the limited computing power of the **ECDs**, some tasks must be distributed among other **ECDs** and inevitably leads to slight increase in delays compared to single **ECD** execution, while still resulting in greatly improved performance compared to cloud computing implementations. The main drawback of such task placement is increased energy consumption of vehicles, it should be considered as direct relation to the computational complexity of the task. Multiple **ECDs** placement locally emphasize this problem even more, leading to increased gas emissions and therefore exaggerates greenhouse effect problem making such vehicles environment unfriendly. It raises the need for optimization of the algorithms running on **ECDs**.

In order to analyze how **EC** is related to **IoT**, scientific articles using terms **EC** and **IoT** were filtered within time frame. The terms utility graph for last two decades is presented in Fig. 1.2. It is scaled logarithmically because **EC** term is found to be relatively new, and only reached popularity in the 2014. The **IoT** term

Fig. 1.2. Number of articles using terms **IoT** and **EC** during the last two decades (according to [Clarivate Analytics Web of Science](#) database)

root reaches late 1990s, and started gaining high popularity in around 2008. The decrease of number of articles in the year 2020 may be related to Covid 19 pandemic, resulting in decreased overall scientific activity and cannot be considered as decreasing interest indicative. The increasing relevance of **IoT** terms indicates the society and industry need of intelligent embedded systems and increasing relevance of **EC** term indicates that research of possible and actual shift of cloud services to edge is actively proceeding.

Reviewed aspects of **IoV** are relevant to other **EC** application areas as well. Analyzed research volume shows interest in **EC** technology exponential increase. Thus more thorough analysis is necessary to be carried-out. In the following we will in more details analyze **EC** topological aspects, employed software means and hardware tools.

1.1.1. Topological Aspects

Nevertheless that **IoT** uses all three main types of device interconnection topologies (cf. Fig. 1.3), the **EC** model application moves focus on decentralized and distributed topology employment. In the centralized topology (a) all nodes are served by single data center, in the decentralized case (b) the nodes are connected to micro data centers available in nearby, which have a connection path to other micro data centers, and finally the distributed topology (c), where each node have not only direct route to neighbor nodes, but also an alternative path through other nodes. Thus in **EC** it is important to take care of and characterize topological aspects because it has great impact on data transmission capabilities and allows to increase the reliability.

Fig. 1.3. Three main types of interconnections between **IoT** devices (nodes) (Ehmke *et al.* 2019): a) centralized; b) decentralized; c) distributed

EC utilizes the intermediate calculations, data storage and advanced network services. Since the data sources and their processing equipment in EC are local, the Quality-of-Service assessment usually is performed by approach based on Markov chains (Corno *et al.* 2018) – by comparing distributed events probability to real machine states. Another popular method for direct quality assesment is classic error matrix method (Chen *et al.* 2018a).

There are no strict rules how to describe the EC topology and the relations between the nodes. An essential part of the problem remains in EC network topology is mostly described by graphs, the Waxman being the most popular model. The contemporary tools for describing EC network topology by graphs are:

- Boston Representative of Internet Topology Generator (BRITE) – by far the most common tool that has a graphical user interface and a tool to evaluate the power consumption of the nodes;
- IGen – basically used for router-level topology description;
- INET – features the power of the node measures for intermediate calculations, assesses the number of intermediate nodes, path length, clustering index and other IoT related metrics.

Generally, in the IoT transmission time T of data is described as follows (Tudoran *et al.* 2014):

$$T = \frac{S}{t_m} \times \frac{1}{1 + g(N - 1)}, \quad (1.1)$$

here S – the size of the data in bytes; t_m – the data throughput of the theoretical model; N – the total number of nodes; g – empirically determined constant of parallelization. This expression is inherited to EC and indicates that by increasing the number of edges of the computing device not only provides an opportunity to implement a mobile data centers, shifting the center of district opportunities, however, allows to reduce data transfer time, and at the same time the data price.

1.1.2. Software Means

Most of the EC software means are bundled with specific hardware devices and thus are specialized for particular tasks. Such software is offered by companies like Intel, Corp., Dell, Inc., Cisco Systems, Inc., Google, LLC, Nvidia, Corp. etc. Alternative, hardware independent software packages are: *OpenFlow*, *OpenStack*, *Docker*, and the *KVM* – kernel-based virtual machines (VM). Their most common areas of use are virtual reality, sensors network, Industry 4.0 applications.

EC is usually simulated with *iFogSim*, *CloudAnalyst*, *CloudExp*, *CloudSim*, *EdgeCloudSim* software packages (Bahwairath *et al.* 2016). These software means employ a real-world traffic network model (Gupta *et al.* 2016), i.e., lets to assess the delay, price, as well as to simulate various network bandwidth bottlenecks,

such as skipping and temporary interruptions of the connection. These functions are implemented using the *Java* programming environment, and are available in full source codes. It is worth noting that many of the packages use open-source *CloudSim* engine source code as a base, extending the properties of the original simulator. The biggest disadvantage of the aforementioned software is that simulations are completely virtual, i.e., they are limited by tunable parameters and does not take into account network anomalies that occurs under real network conditions.

Moreover *MATLAB* environment can be used for the *EC* simulation if the following add-ins will be used: *SimEvents*, *NS-2*, or *Complex Networks Package*. These packages are only compatible with older versions of *MATLAB* (7.0). In 2017 a new add-in for *IoT* simulation and analysis was created – *ThingSpeak*. It is based on remote database and allows to visualize and explore the data off-line: data can be loaded into the database directly from the sensor or within the *MATLAB* environment; the data are transmitted through encrypted *REST API* connection; add-in can be used not only for simulation, but also for the actual implementation of the system. However: it is free for home users only (with restrictions) and the academic community, while the commercial activities require the purchase of a license; the data are stored in the *ThingSpeak* database, which may raise doubts as to the safety criteria; the open source part is only *MATLAB* add-in code, the other parts of the software is closed source.

Open source simulation software examination confirmed the possibility of intervention in the execution of the program, e.g., the ability to add edges to simulation model. Thus *EdgeCloudSim* was selected for the following investigations.

Finally one of the popular closed-source platforms – *Azure* was examined, too. Its infrastructure is based on the network of virtual gateways, whose main advantage is compatibility with both *Windows* and *Linux* operating systems. The *Linux* operating system support is grounded by the popularity of the platform as it provides the ability to easily move towards the edges of the cloud computing structure. This infrastructure offers a communication channel with Microsoft, Corp. servers, in order to synchronize status of the *ECDs*, and use the remote databases to store the data. *Azure* novelty in comparison to competitors is in the the ability to encapsulate a service into small-scale datacenters – cloudlets, which can be configured and updated directly from the control panel. Based on *VM* abstraction, cloudlets can be programmed in variety of popular programming languages, e.g., *JavaScript*, *Java*, *.NET*.

1.1.3. Hardware Tools

EC physical equipment is usually inherited from the structures (cloud, fog) and consists of traditional connectivity hardware, routers, switches, gateways, access points. However, when talking about *EC* technology, usually it is described by

Table 1.1. Popular communication technologies used in EC with limiting data rates and ranges (ISO/IEC TR 23188)

Communication technology	Data rate, Kb/s	Range, m
Bluetooth	700	10
WiFi	11,000–750,000	10–100
IEEE 802.15.4	250	10–300
6LoWPAN	250	800
DASH7	9.6–166	≤ 5,000
Zigbee	250	30
Wireless HART	250	200
Bluetooth LE	1,000	5–10

wireless connectivity, the most popular wireless technologies used in EC are listed in the Table 1.1. The main advantage of EC in the hardware perspective is that data-processing equipment becomes independent of the geographical location, which leads to smaller consignment to the center of the data flows, while at the same time, a faster overall system response (Satya 2016).

One of the most popular EC implementation is mobile edge computing (MEC). The main task of the technical level of equipment in the MEC is to dynamically distribute the load in between (Corcoran, Datta 2016), ensuring quality of service (Borcoci 2016) for all remote devices.

The Fig. 1.4 explains how ANN represented by conventional multi layer feed-forward model is applied on IoT devices (nodes), each of them acting as a single neuron (circle). Nodes, depending on the application, can be pure hardware, or virtual. Data flow between nodes is denoted as s for particular neuron at particular layer and in this use case is based on MQTT communication. Such placement of nodes can significantly increase performance of the mentioned ANN through parallelism.

Fig. 1.4. Multilayer feed-forward neural network model application to a network of IoT devices

Fig. 1.5. Structure of j -th neural node in the l -th layer internals: $w_{ij}^{(l)}$ –synaptic weights; $\Phi^{(l)}(.)$ – neuron activation function

Each network node has a set of weights $w_j^{(l)}$ (that represent intra-connection quality), summation element and a neuron activation function $\Phi^{(l)}(.)$. A detailed view of node structure is presented in Fig. 1.5. Input data $s^{(0)}$ to network is shared across all first layer IoT workers and depends on the number of input sensors.

Complexity of multilayer feedforward ANN structure makes cumbersome to implement its training in the hardware, thus training procedure is performed on external tool – *MATLAB*. Later on, weight coefficients are hard-coded in hardware.

Such placement looks very promising in decentralized EC implementation in resource-wise and topological placement-wise aspects, however more research is needed to reveal the possible drawbacks and problems, as well as to evaluate and compare it to current implementations.

Variety of different EC implementations based on different technologies exists, however EC is a strict definition and has clear explanations which need to be met. The following EC specifics for IoT are declared in standards ISO/IEC TR 23188 and ISO/IEC TR 30164:

Security. Data flows between distributed nodes connected through Internet, thus the distributed nature ensure higher security level than cloud solutions, sometimes full ownership is transferred to end-users, however additional special encryption mechanisms are required. Taking into account that EC devices mostly are resource-constrained, it limits the choice of available security methods.

Scalability. As any other type of distributed network, EC ensures high scalability over centralized cloud, however the heterogeneity of the ECDs having different constrains in terms of performance, resources available and energy consumption, scaling process may be slow. This is very true when complex security ensurance solutions introduce parasitic latency.

Reliability. The distributed manner of EC allows to manage failovers in order to maintain system (service) alive, in a situation where node becomes unreachable, the task could be redistributed to adjacent nodes, or even maintain the service in offline mode, with later seamless recovery. Reliability level in EC highly depends on system and network topology design.

Speed. EC brings analytical computational resources close to the edge, therefore increasing the communication speed, a well designed system outperforms traditional cloud-based system, making EC more feasible option when designing modern real-time systems.

Efficiency. Due to increasing hardware capabilities in IoT devices and their cost decrease, it is possible to implement and run sophisticated AI tools directly in ECDs. Such placement increases operational efficiency of the system, allowing intelligent intermediate computing stages, thus avoiding big data transmissions to cloud which results in significant bandwidth savings.

The presented insights on reviewed EC technology for IoT explains the concept of edge computing, its relationship to cloud computing and IoT, the exclusion of fog-edge computing differences and names the key features and technology standards for the correct implementation of EC.

1.2. Classifiers in Edge Computing

Reviewed EC technology showed that there is a variety of data processed across it, and it is seen that the data alone carries little of significance, until they are properly processed and classified. Classification in EC is done in various cases, e.g., in the detection of anomalies in the packages (Pasca *et al.* 2017), in spam detection (Zhang, Sun 2017), or in abnormality detection. In EC the classification step can be directly integrated into the data packet routing block in gateway devices (Hodo *et al.* 2016), where the most popular classifier is a self-organizing map (SOM), ensuring that over 99% of the malicious attacks are detected at packets filtering stage.

1.2.1. Fields of Application

In this subsection, the fields of classifier in EC application are presented revealing the value of data classification. Different fields deals with different type of data, thus different classifiers may be required.

The classification in EC is applied to outline large data (Diallo *et al.* 2017), with the aim to partially or fully structurize data by its meaning, with later reduction of transfer delays, since the only meaningful data is transferred. Classifiers type and parameters are selected according to the data type.

One of example of the practical application is the classification applied in the field of audio, video or documents copyright protection (Alrawais *et al.* 2017) at the software automated level. Decentralized ECDs perform the classification of the samples, with later consensus deciding authenticity of the product.

As for today, one of the main areas of application of the EC is in smart cities and houses. Focus is on the processing of the images and therefore faced with major problems of real data for effective response. Currently, for tasks of image analysis and recognition, ANNs based on complex deep learning, such as the *GoogLeNet*, *VGGNet*, *AlexNet*, and others are used. However they require a lot of computing resources. The distributed EC model enables comprehensive analysis (Salman *et al.* 2015; Samie 2018; Satoh 2016) and decision making by complex algorithms in close to real-time applications. As examined before, IoV generates a lot of data, thus IoV field is a great candidate for ANN classifiers to be used for meaningful traffic data aggregation for further autonomous decision making.

With the future use of 5G technology, the combination of ANN and EC technologies looks like a great duo, a lot of different control data are expected to prevail in 5G networks with an aim to achieve good performance, clearly the control of the network needs to be quick and adaptive. To fulfil this criteria it is clear that machine learning approaches will be used, there is a great opportunity for ANN.

1.2.2. Traditional Classification Methods

Lets briefly discuss traditional classifiers found in literature, that are also used in EC to perform simple tasks.

It is known that raw data has different type and dimensionality, so that classifier needs to be tuned by its parameters to perform with corresponding data optimally (Zalieckaitė, Žilinskas 2016). According to reviews literature, there are several approaches that can be applied in EC:

- data classification, in order to change its dimension to perform the grouping (Hinton, Salakhutdinov 2006);
- classifier is used as an intermediate concentrating unit, carrying out the initial processing of the data;
- the whole EC topology can be used as an emulated ANN classifier.

The traditional classification methods are well proven in various domains for their simplicity and high accuracy. The most studied traditional classification methods could be categorized as follows:

- logical algorithms (decision tree, ruleset learning);
- statistical algorithms (instance based, Bayesian networks, Naive Bayes);
- perceptron based algorithms (single-layered, multi-layered, RBF);

- SVM;
- evolutionary and genetic algorithms.

Traditional classification methods works great on static data, even the big-sized datasets cloud be classified efficiently and with great performance, however, the rapidly changing dynamic data in EC requires more advanced and intelligent classifiers. Some of beforementioned algorithms already incorporates simple machine learning theories, but modern ECD hardware enabled execution of much more complex algorithms without significant performance penalty.

Currently, large sensor data is processed by the data centers – the cloud. The reduction of the data before it reaches data centers is the task for EC. There are many methods used to compress the data sent to a cloud, but they do not solve the problem of excess data sent to remote data centers, nor helps with data availability. To solve this issue, raw data processing methods close to the edge, such as semantic analysis, clustering and classification are applied.

1.2.3. Intelligent Classifiers

In EC, we see the classification process to be the more promising than clustering, literature analysis shows that recognition tasks are more needful, thus the tasks with predefined classes are the most demanded. Depending on the task and the

Fig. 1.6. Augmented Chen *et al.* (2019) classification of artificial neural network paradigms

nature of the data, different ANN are used. The summary of the most popular ANNs is provided in Fig. 1.6. However, not all ANNs are suitable for data classification purposes, the most commonly used classifiers (marked in bold) are: fuzzy neural network (**FNN**); support vector machines (**SVM**); k -nearest neighbor neural network (**kNN**) and **SOM** (Hammadi-Mesmoudi, Korczak 1995).

Two types of ANN learning are considered: supervised and unsupervised. Supervised learning refers to situations in which each instance of input data is tied to desired (target) value for that particular input. In the case when the target values are a set of finite discrete categories – such learning task is known as a classification problem, and when the targets are one or more continuous values – regression problem. Both cases sets the goal to learn a functional connections (mappings) between the input space of the data to a target space. The practical example of supervised learning may be the classification of email messages deciding whether it is a spam or not by comparing the parts of context (phrases, attached files) to learn values, finally assigning the message to particular class (malware or not malware in this case). For the regression problem – the email might be considered malicious based on a prediction of malware occurrences from particular host.

For the case of unsupervised learning, algorithm must learn from raw data alone without any feedback signals (supervision) in the form of targets. Basically it means that learning algorithm needs to group together similar examples (based on undefined features) in the dataset – a task known as clustering. The practical example of unsupervised learning application may be anomaly detection in the computer network traffic based on learnt statistical properties about network connections. In some cases it may be difficult, or even not possible to enumerate all cases, the unsupervised learning is the right way to learn the underlying unlabelled data. Unsupervised learning, particularly pre-training gave an opportunity for one of the most popular ANN – deep learning to rise. It is basically an unsupervised learning based ANN algorithm, which has initial set of weights already pre-trained by supervised learning approach. During the execution, the unsupervised training fine-tunes the network in order to perform classification more accurately.

One more type of intelligent classifiers are the reinforcement learning based classifiers. Method aims to use observations gathered from the interaction with the environment to take actions that would maximize the reward, it continuously learns from the environment in an iterative manner, the process continues until all possible states are explored and learnt. Usually reinforcement learning is defined by a specific type of problem, and all its solutions are classed as reinforcement learning algorithms. In order to produce intelligent decisions, reinforcement learning algorithm goes through the following steps:

- a) input state is observed by the algorithm;
- b) decision making function triggers the algorithm perform an action;

- c) after the action is performed, the reward is received;
- d) the information about state-action interaction and final reward is stored.

The overall reinforcement learning paradigm is based on the feedback response of reward signals to evaluate self performance at each action step. The algorithm itself decides the best action to select based on his current state. Basically, repetition of these steps is known as a Markov decision process.

Beforementioned intelligent classifiers specifics and review of different ANN learning types reveals the suitability of unsupervised learning based ANN classifiers to solve EC specific tasks. A key element in the information processing and control in EC is an intelligent classifier.

1.3. Unsupervised Artificial Neural Networks for Classification

Unsupervised learning is the type of learning when only input data and no corresponding output data are known. The aim of unsupervised learning is to infer the underlying structure or distribution in the data in order to model the data. There are no correct answers nor the teacher. Unsupervised learning problems can be further grouped into data clustering and association problems:

- clustering: a clustering problem is a condition where the aim to discover the inherent groupings in the data is considered;
- association: an association rule learning problem considers the aim to discover rules that describe large portions of data.

1.3.1. General Analysis of Paradigms

The supervised learning types of ANNs suffer from the problems of possible convergence to local minima, instability of memory as the training set drastically increases in size and the large number of iterations are needed to encode even relatively simple patterns. Some of this unpleasant behavior can be minimized by the use of a delta rules for weight updates. Particularly, modified perceptron models to overcome beforementioned problems, causes computationally expensive steps due to the use of exponential functions when determining membership values.

Supervised learning problems are partially solved by using unsupervised learning methods, and the proposal of new designs of self-organizing Kohonen feature maps (Kohonen 1998) and sequel of Adaptive Resonance Theories (Carpenter, Grossberg 1987). These techniques are based on biological studies of human memory organization and its dynamics. The ART design has several versions, notable difference is that starting from second version of ART – ART2, the extension of

functionality to accept continuous valued input vectors (opposing to binary only inputs for ART1). The Kohonen self-organizing feature map system has previously been shown to be equivalent to the k -means clustering algorithm in terms of performance and may be used as a direct replacement. Worth to mention, that Kohonens self-organizing feature maps use a linear update rule for the weights, which makes this model computationally attractive on low-power devices.

However, unsupervised learning algorithms do not fully emulate the way in which humans and other animals learn, even today the power of human and animal learning principles is still main motivator to discover computational principles in machines that could learn with similar capabilities. Humans and animals carry out incremental learning based on the sequential experiences in real-time, and is known to be pre-trained and unsupervised. One difference to be noted, when comparing machines to humans and animals, the latter can learn quickly and the memories can persist for long time, where machines, and especially ECDs is very limited in those terms. ART's solution of this stability-plasticity dilemma (Grossberg 1987; Petkov *et al.* 2016) is found to be the closest candidate, thus further analysis is based on ART family and other popular intelligent classifiers.

1.3.2. Adaptive Resonance Theory

Subclass of ANN based classifiers, that seems to fit EC specifics and also provide an unsupervised learning possibilities – Adaptive Resonance Theory classifiers. It is known that when new data samples enter into the most of ANN classifiers, the previous knowledge is forgotten and the network must be relearned. That repeated learning is a waste of time, moreover, it could even fail (overfit) to use of the learning data properly, resulting in a limited application of these classifiers. To solve this problem, Grossberg and Carpenter from the University of Boston proposed ART, followed by a series of ART network models, such as ART1, ART2, ART3, ARTMAP, Fuzzy ART (FART), and Fuzzy ARTMAP (FAM), which were successively proposed to meet actual application requirements.

ART autonomously carries out fast, yet stable, incremental learning (Carpenter, Grossberg 1987) under both unsupervised and supervised learning conditions in response to a complex non-stationary input data (Huang *et al.* 2014). In contrast, many current ANN algorithms use batch learning in which all the information about the class is available at a single time, the most common example is SVM. Other algorithms algorithms usually are not defined unless all learning trials are supervised. Also other algorithms become unstable in a non-stationary data and become unstable if learning rate is too fast. ART algorithm does not feature these problems.

Fig. 1.7 represents block diagram of ART1 clustering algorithm, $F1$ being the comparison layer, $F2$ – the recognition layer with $N^{(2)}$ output neurons – the vary-

Fig. 1.7. Adaptive Resonance Theory clustering architecture

ing number of clusters produced by the algorithm, generated dynamically. Comparison and recognition layers are fully connected with feed-forward $w^{(1 \rightarrow 2)}$ and weighted feedback $w^{(2 \rightarrow 1)}$ links.

The **ART1** algorithm tries to generate as many output clusters as required by the input data specifics. The number of clusters mainly depends on two parameters – μ and ρ . The algorithm output relies on the aim to maximize (Araújo, Ralha 2012):

$$\frac{\mathbf{s}_i^{(2)} \cdot \mathbf{s}^{(1)}}{\mu + \|\mathbf{s}_i^{(2)}\|_1}, \quad (1.2)$$

here $\mathbf{s}_i^{(2)}$ – an prototype vector; $\mathbf{s}^{(1)}$ – input vector; μ – small positive number (selected manually) to prevent division by zero in particular cases (when input vector length equals 0). The output neurons ($F2$) continuously compete for activation, in the case when input neuron ($F1$) matches the winning output neuron value, the *resonance* event occurs, in the other case the *reset* event is repeated for the next iteration. This process is controlled by *vigilance* control subsystem. It is controlled by dimensionsless parameter ρ ($0 < \rho \leq 1$) by testing the similarity between input and output vectors, and is expressed by vigilance test formula:

$$\frac{\mathbf{s}_i^{(2)}}{\mathbf{s}^{(1)}} < \rho. \quad (1.3)$$

Opposing to probabilistic models, such as Bayesian statistics, that often use non-local computations and are needed to compute an entire probability distribution (shows the best results with stationary data only), the **ART** uses only local

computations, therefore dynamic data could be processed. [ART](#) model is emulating the way how the brain processes new knowledge in terms of probability theory, hypothesis testing, prediction, and decision-making. These local computations enable fast autonomous adaptation to a non-stationary data and changing rules in unexpected ways. These [ART](#) principles and mechanisms outperforms Bayesian analysis and similar methods. The best properties of feedforward and feedback processing, digital and analog processing and data-driven bottom-up processing and hypothesis-driven top-down processing are involved (Grossberg 2003).

1.3.3. Deep Learning

In recent years, deep learning neural networks has been adopted widely in computer vision applications. Because of that, learning reusable feature representations from large unlabeled datasets has become an area of active research. Due to the fact that training of deep learning neural network is usually performed in high performance hardware with very high amount of memory available to store big amount of data representing hidden layers, it is practically unlimited amount of unlabeled images and videos which intermediate representations – features could be learnt. This leads to a variety of practical applications, which of the most popular is image classification.

Variety of different deep learning neural networks are created, the architectures of two most succesful ones – *AlexNet* and *VGGNet* are presented in Fig. 1.8 and Fig. 1.9 correspondingly. As it can be seen, the architectures of different deep learning neural networks is quite similar, most of them are an multi-layer, with very high number of neurons in hidden layers, the main parameters that distinguishes them is the amount of neurons and the type of training. In the most cases the class of convolutional neural networks ([CNN](#)) are used for deep learning. It is also widely applied in image and video recognition, image segmentation, medical image analysis, natural language processing and many more. Similarly to [ART](#) family, the [CNNs](#) were inspired by biological processes. [CNNs](#) use relatively little

Fig. 1.8. AlexNet [CNN](#) based deep learning neural network architecture (Pedraza *et al.* 2017)

Fig. 1.9. VGGNet CNN based deep learning neural network architecture (Ferguson *et al.* 2017)

pre-processing compared to other classification algorithms (image), meaning that the network learns to optimize the kernels through unsupervised learning.

While deep learning neural networks perform very well with variety of relevant problems and there are a lot of already pre-trained weight sets available to use, most of them are trained on specific type of input samples, making the train set usable for specific application. Additionally the network could be fine tuned by perform additional training on new samples without loosing the knowledge of initial input set (Zhao *et al.* 2016). However, the high requirements for processing resources and high amount of memory makes it almost unavalable to use in ECDs. It is almost impossible to run mildly sized deep neural network even in decentralized EC by distributing the processing tasks among EC network. Even if the ECDs would have the possibility to allocate such high resources, the power efficiency of the devices running deep neural network would be terrible. That is why such systems only available on non-generic – specified only ECDs.

1.3.4. Self Organizing Maps

The SOM is certainly one of the most important models of neural networks at present, and is used in a variety of applications such as image processing, precipitation estimations, wind circulation predictions, and seismic interpretation.

The **SOM** is a type of **ANN** based on competitive and unsupervised learning, i.e., no information about the input signal classes is used in the adjustment process for the synaptic weight in the net (Kohonen 1998). Despite being developed to try to model of the human cerebral cortex and the lateral interactions between the neurons, the **SOM** model is considered extremely simple and easy to implement.

The network essentially consists of two layers: an input layer $s^{(1)}$ and an output layer $s^{(2)}$ with neurons generally organized in a two-dimensional topological array. The input to the net corresponds to a $N^{(1)}$ -dimensional vector, $s^{(1)}$. All of the $N^{(1)}$ components of the input vector feed each of the neurons on the map. Each neuron i can be represented by a synaptic weight vector $w_i^{(2)}$, also in the $N^{(1)}$ -dimensional space. For each input pattern $s^{(1)}$ a winner neuron, s^* , is chosen, using the criterion of greatest similarity:

$$\|s^{(1)} - w^*\| = \min \|s^{(1)} - w_i^{(2)}\|, \quad (1.4)$$

where $\|\cdot\|$ represents the *Euclidian* distance. The winner neuron weights w^* , together with the weights of the neighbouring neurons, are adjusted.

The **SOM** has some important properties. Once the training algorithm has converged, the computed feature map displays important statistical characteristics of the input space, summarized as follows (Haykin 1994):

- Approximation of the input space: the basic objective of **SOM** is to store a large set of input vectors by finding a smaller set of prototypes that provides a good approximation to the original input space. This strategy is based on vector quantization theory, the motivation for which is dimensionality reduction, or data compression.
- Topological ordering: the features map computed by **SOM** is ordered topologically. Similar input vectors are mapped close to each other, while dissimilar ones are mapped far apart.
- Density matching: the **SOM** reflects the probability distribution of data in the input space. Regions in the input space in which the input patterns are taken with a high probability of occurrence are mapped onto larger domains of the output space, and thus have better resolution than regions in the output space from which input patterns are taken with a low probability of occurrence.

1.3.5. Support Vector Machines

Another popular classification algorithm – **SVM** was proposed in AT&T Bell Laboratories by Vladimir Vapnik with colleagues, reaching its roots to 1962. It is supervised learning algorithm that showed great performance in data prediction.

Today, **SVM** is one of the most widely used machine learning algorithm at the embedded devices. It is used to solve both, classification and regression problems by discriminating between two or more classes of input data by defining an optimal hyperplane to separate them (Moghaddam *et al.* 2020). Two cases are considered: linearly separable and non linearly separable data. For linear separation, optimization problem to be solved is expressed as follows:

$$\min \|w\|, \text{ in} \quad (1.5)$$

$$\Phi_i(w^T s_i - b) \geq 1, \quad (1.6)$$

here w represents normal vector to the hyperplane; s_i – input vector; b – offset of the hyperplane from the origin.

In the case of non-linear separable data the optimization goal is to minimize:

$$\left[\frac{1}{n} \sum_{i=1}^N \max \left(0, 1 - \Phi_i(w^T s_i - b) \right) \right] + \lambda \|w\|^2, \quad (1.7)$$

where the dataset of N points is separated according to *Hinge loss* function with additional parameter λ setting the trade-off of increased margin size.

The algorithm of **SVM** is relatively simple and is suitable to be used in **EC** low power devices, moreover hardware accelerated versions of the algorithm are presented (Vranjkovic, Struharik 2020) significantly increasing the efficiency of both, training and the recall execution on **ECDs**, showing memory usage reduction from 3% to 85% and a speed-up algorithm execution in a range from 1.17 to 7.92 times.

Summary of the most popular **ANN** classifiers used in **IoT** in cloud, fog and edge computings is presented in Table 1.2. The main terms of training type (supervised or unsupervised), the ability to self-organize (change its output class number or internal structure), the average relative number of neurons and output classes and the accepted input values type are considered. The references of the application in specific computing technology (cloud, fog and edge) are presented as well. Based on the fact, that both **ART1** and **ART2** have not been researched in **EC** technology perspective before (greyed cells), and the review of **ART** family shown that these classifiers may perform well or even outperform current solutions, they were selected as a possible candidates for further research. It is also can be seen that other types of intelligent classifiers are widely reasearch for all three computing technologies allows to perform the comparison between them.

Table 1.2. Summary of the most popular ANN classifiers used in IoT, based on properties like supervised (SL) and unsupervised (UL) learning, self-organization (SO), corresponding application references in cloud (CC), fog (FC) and edge (EC) computing technologies during last two decades

Classifier	Property		Training		SO	Average number		Input	Computing technology: application references
	SL	UL	SL	UL		Neurons	Classes		
ART1	+	+	+	+	+	*	*	bin.	CC: Bhat <i>et al.</i> 2015; Feng <i>et al.</i> 2012; Rangarajan <i>et al.</i> 2004
	—	—	—	—	—	—	—	—	FC: —
	—	—	—	—	—	—	—	—	EC: —
ART2	+	+	+	+	+	**	**	cont.	CC: Bhat <i>et al.</i> 2015; Park <i>et al.</i> 2017; Wan <i>et al.</i> 2018
	—	—	—	—	—	—	—	—	FC: Shete <i>et al.</i> 2016
	—	—	—	—	—	—	—	—	EC: —
FNN	+	—	—	—	—	**	**	cont.	CC: Huang <i>et al.</i> 2012; Kim 2016
	—	—	—	—	—	—	—	—	FC: Aral, Ovatman 2016
	—	—	—	—	—	—	—	—	EC: Gravalos <i>et al.</i> 2018; Souza <i>et al.</i> 2017
Deep Learning	+	+	+	—	—	** ** *	** ** *	cont.	CC: Ben-Zikri <i>et al.</i> 2017; Wang <i>et al.</i> 2021; Zhang <i>et al.</i> 2020b
	—	—	—	—	—	—	—	—	FC: Jazayeri <i>et al.</i> 2020; Lyu <i>et al.</i> 2020; N.G., S. 2020
	—	—	—	—	—	—	—	—	EC: Véstias 2019; Wang <i>et al.</i> 2020; Xu <i>et al.</i> 2021; Zhang <i>et al.</i> 2020a
SOM	+	—	—	—	—	**	**	cont.	CC: Stephanakis <i>et al.</i> 2019; Thilagavathy, Rajagopal 2019; Vesanto, Alhoniemi 2000
	—	—	—	—	—	—	—	—	FC: Bhatia 2020
	—	—	—	—	—	—	—	—	EC: Manihar <i>et al.</i> 2020; Yu <i>et al.</i> 2018
SVM	+	+	—	—	—	**	**	cont.	CC: Valluru, Jeya 2020; Zhang <i>et al.</i> 2019
	—	—	—	—	—	—	—	—	FC: Moghaddam <i>et al.</i> 2020; Wadood <i>et al.</i> 2017
	—	—	—	—	—	—	—	—	EC: Avudaiammal <i>et al.</i> 2020; Merenda <i>et al.</i> 2020; Murshed <i>et al.</i> 2019

1.4. Artificial Neural Network based Classifier Implementation

In this Section the ways of ANN based classifier implementation in various environments (FPGA, software, generic hardware and mixed) are discussed according to the main ANN architectural specifics. Possible algorithm parallelization and pipelining techniques on specified hardware is then discussed. Lastly, the distributed implementations of ANN are reviewed.

1.4.1. Architecture Specifics

ANN models a high density structure of parallel links connected computation units, consisting of many simple self-contained processing elements (Navakauskas *et al.* 2014). ANN implementation complexity and therefore an ability to implement on particular device is always dependent on the number of memory elements and the number of arithmetic operations.

There are two main procedures in the design: forward process and backward process. In the forward process, the ANN architecture is defined: a total number of layers, the number of neurons in each layer, and the type of activation function are chosen. In the backward process, the loss function of the ANN is defined. Through calculation of the gradients in the backward path, weights of the neural network are updated. One forward and one backward process can be called as one epoch, and it is the training of ANN. Usually, a trained neural network model need hundreds of epochs to be available.

Modern ANNs usually have more than 2 layers, that is also true for modern ANN based classifiers. Also modern ANN classifiers features a high number of neurons, perfect example of this is DNN which has a lot of neurons in hidden layer. Modern hardware enables high number of neurons and high number of layers to be calculated fast and efficiently, however a lot of resources are needed for that. In terms of the direction of data flow, in RNN data flows between adjacent cells in the same layer and in feedforward neural networks data flows only forwardly across the network. Likewise, CNN is mainly based on convolutional operation, which is a computationally complex step as well. The non-linear transformation function is the key for the mentioned CNN, DNN, RNN, and similar networks, with an aim to extract high dimension features of original data set.

The mentioned architecture specifics points out to viewpoint that not only the data size and complexity is increasing rapidly, but computational complexity of relevant classifiers are increasing as well. Some of stages, like neuron adjacent value computations shows an acceleration possibility by parallelizing or distributing them and performing these computation in single instance of time.

1.4.2. Pipelining

The pipelining of the complex classifiers algorithms is essential. Lets take an CNN as an example for pipelining. CNN is a sequential architecture algorithm, that means that the output of one layer becomes the input for the next layer. It is very important to pipeline or schedule the CNN structure correctly in order to reduce the wait time between data flow across the layers. In order to achieve efficient pipelining, methods such as Backward Pipeline Scheduling can be used, resulting in dramatic latency improvement of processing one image which is considered to be critical for an edge device. Due to backward pipeline scheduling, data that is computed in one layer does not have to wait for all other data in the same layer to be computed before they can be used immediately for the next layer. This process propagates in the pipeline in such a way that all the layers can execute in an overlapped fashion to reduce overall latency. The recent rapid growth of applications based on deep learning and CNN algorithms, has dramatically improved both industrial and academic research on optimized implementations of CNNs on accelerators such as GPUs, FPGAs and ASICs, as general purpose processors can hardly meet the increasing performance and energy-efficiency requirements for those applications (Bacis *et al.* 2017). As for hardware pipelining, FPGAs are one of the most possible alternative, as they allow to parallelise the algorithm, also provide extreme flexibility to reconfigure the hardware based on particular algorithm. The algorithmic design is written for FPGA is written with VHDL language, that permits the design of complex circuits with structural or behavioral description. The synthesis tools allow to simulate and pipeline the algorithm through logic optimization.

1.4.3. Enabling Computations

The ANN could be trained and executed on various devices, thus different approach in implementation is used. Four different ways of ANN based classifier implementation are discussed. Briefly about each in the following subsections.

A. Field Programmable Gate Arrays

FPGA have become one of the key digital circuit implementation media over the last decade (Alberto de Albuquerque Silva *et al.* 2015). A crucial part lies in their architecture, it combines the nature of reprogrammable logic functionality and their programmable links. FPGA architecture has a dramatic effect on the quality of the final device's speed performance (Cassidy *et al.* 2013), area efficiency and power consumption.

Resources Estimation. FPGA resource specifications often include the number of configurable logic blocks, number of fixed specific function logic blocks

such as multipliers and size of RAM. Of the many FPGA specifications, these are typically the most important (Biookaghazadeh *et al.* 2018) when selecting and comparing FPGAs for a classifier application, also four main hardware resources measures for FPGA are: the size of Look Up Table, Flip-flop, arithmetic Digital Signal Processor, and Block Random Access Memory counts.

Hardware Supply Market. Microsoft, Corp. was the first company started using FPGAs in mass IoT applications implementing it into Bing and Azure infrastructure to accelerate its own workloads, like search indexing, and the long-term aims was to make hardware acceleration available to customers in acceptable way. Thus, a first step for IoT development for custom platforms is to begin with prototyping using a platform or combination of platforms that matches the desired target as closely as possible. This may be an existing CPU platform, a GPU platform such as the Tegra K1 or Tegra X1, or FPGA platforms such as the Xilinx, Inc. Zynq-based Parallela. These platforms contain standard communications, sensor interfaces, and general purpose IO connections which is convenient for the user to integrate sensors, actuators, communications, and other peripherals to prototype the system. Since that, variety of solutions for EC based on Intel/Altera SoC FPGAs were presented. Currently the aim of the system architects is to use the growing capabilities of FPGAs with embedded powerful CPUs and high data bandwidth memory access to implement, manage and reconfigure the EC solutions based on Deep Learning algorithms (Barreto *et al.* 2019). Basically, the IoT is revealing an important need in programmable hardware technology. Development typically begins with application and device prototyping, the first prototypes assist in identifying platform computation needs in terms of performance, power and efficiency.

Scalability and Flexibility of Solutions. The FPGA consists of a matrix of reconfigurable gates, when the logic arrays are configured, these gates gets connected with a way that builds a hardware implementation of a software application. Variety of tools available are enabling the designers of embedded systems to create and adapt FPGA-based applications more quickly and more easily.

Although the FPGA is a family of reconfigurable hardware, the digital computing tasks are developed in software and compiled into a bit stream file that contains information about how the components should be wired together. FPGAs combine the best features of ASICs and processor based systems, in fact FPGAs execution is parallel in nature. Advantage of using a software programmed processor is that software is very flexible to change while a disadvantage is that performance can suffer if the clock is not fast enough. However, the advantage of an ASICs is that it can provide very high performance because of its dedicated type of operation, but its disadvantages are:

- high cost to volume ratio;

- high time range between design to end product;
- incapability to include any changes after the system is fabricated;
- difficulties in debugging the errors.

FPGAs overcomes these disadvantages and fill the gap between hardware and software, offering such advantages: flexibility, reliability, low cost, fast time-to-market and long term maintenance. For the case of **ANN** classifier implementation, **FPGA**-based prototype is superior for producing an early validation of the design, this helps to make prototype fast and efficient and to simplify the prototype translation into a design of the final **SoC**.

Paralellization of the Algorithms. **FPGAs** are unlike processors, they do not have an operating system and are using hardware which is dedicated solely for logic processing. Worth to note, that different operations do not have to compete for the same processing resources since the processing is parallel (Sharma, Nair 2020), it ensures very fast and multiple control loops that can run on a single **FPGA** device at different rates.

B. Software

ANN classifier implementation in software is the easiest way to do. Numerous software packages by known corporations are released, most notable ones are: *Keras*, *Neural Designer*, *Neuroph*, *TFLearn*. Most of them allow to design and train various classifiers with following packing the trained **ANN** solution to be deployed on the target system. For convenience, almost all software **ANN** design tools compile to *Python* or *Java* executables, enabling the compiled software classifier to be run almost on machine (device) (Agrawal, Gopalan 2020). Software **ANN** classifier can be even launched (and trained) on any web browser – using recent *ConvNetJS* software library (Kieu *et al.* 2018).

C. Hardware

Pure hardware implementations of **ANN** is rarely met nowadays and usually refers to old designs, it is due to the fact, that dedicated hardware or application-specific integrated circuit (**ASIC**) is non versatile and is suitable for one specific problem to solve (Lee *et al.* 2019). However with recently proposed and implemented **AI** based accelerators for personal computers (*Intel Optane*) and already showed great performance gain in computer memory bandwidths (Hildebrand *et al.* 2020), it is possible that hardware processors may return to the market and will occupy the accelerators role for ordinary gear that requires a performance boost.

D. Mixed

Most of leading integrated circuit manufacturing corporations offers the devices which could be considered as a mixed architecture – software controlled hard-

ware. The closest representative of such integrated circuit is a system-on-chip package where part of the package is an **FPGA**, the secondary part of chip consists of **ARM** microprocessor (Xue *et al.* 2020). Such combination enables to use utilize the advantages of reprogrammable hardware technology (**FPGA**), as well as the advantages of pure software implementation running on **ARM** cores, on a single machine.

The most popular devices available on the market for **ANN** classifiers implementation in mixed (software and hardware) way can be grouped according to the type of the device, key physical types of those devices are:

- Coprocessor (CoP): a processor used to additionally supplement the primary **CPU** functions. Usually packed as accessory, to be easily attached to the host, for example, USB, particularly **FPGA** based development boards used as coprocessors;
- System-on-Chip (**SoC**): system integrates processor(-s), memory, **IO** controllers on a single chip. High integration level of **SoC** allows to build a complete system with optimized size and power efficiency, the example of such system for **EC** – Tegra development board;
- System-on-Module (SoM): a single module composed of microprocessor, memory and **IO** controllers, together with the memory itself. Often SoM incorporates SoC, but does not offer full functionality without host device, e.g., Jetson Nano development board;
- Single Board Computer (SBC): a complete computer system built on a single board and does not depend on any external hardware, incorporates **CPU**, **GPU**, memory (RAM, Flash), networking, and other peripherals in compact size, the most known example of SBC is Raspberry Pi.

1.4.4. Distribution

Distributed computing is currently widely applied to meet the upcoming 5G technology (McClellan *et al.* 2020) needs. Various distributed **ANN** based systems takes place as well, one of most common application is an sensor network processing, particularly image sensors (cameras) based decisions, i.e., distributed image stitching (Sapijaszko, Mikhael 2018). The industry need for smart imaging devices, for example aerial, **IoV**, smart glasses, medical equipment and others, that requires image stitching (Wang *et al.* 2017) for its primary functions, urges to implement embedded distributed image analysis techniques (Seo *et al.* 2019). Parallel acceleration, feature extraction, feature matching and image fusion algorithms are already known and are widely used on single multi-core machines or **DSPs**, however these techniques are not widely adopted in **EC** and requires more scientific research on it. Current local implementations of image stitching task with single machine

shows processing time acceleration of 6.66 times on images with resolutions up to 1024×768 pixels while increasing execution core count from 1 to 8. The distribution and external parallelization of mentioned algorithm could result in massive performance gains and make image analysis much more efficient (Satoskar *et al.* 2017).

1.5. Conclusions of the First Chapter and Formulation of the Thesis Objectives

The analytical literature review on Artificial Neural Networks (ANN) place in Edge Computing (EC) was done by: the description of EC technology place in the Internet of Things (IoT) stack and it's implementation options; the elucidation of classification methods role and it's importance in EC; an analysis of ANNs paradigms and their implementation aspects relevant to use in EC technologies. The results of performed analysis support and let to draw the following conclusions:

1. Only optimized for application and ad hoc re-programmable hardware solutions can meet current EC demands, thus IoT systems complexity, performance and efficiency must be well assessed.
2. One of the EC key elements in the information processing and control is a classifier, thus the ways to make classifiers intelligent (adaptive or self-organizing or both) must be explored.
3. The artificial neural network theory is a strong candidate to fulfill needs for intelligent classifiers however unsupervised paradigms for EC are not sufficiently analyzed and thus require more thorough research.

Therefore, in the thesis the following main objectives will be fulfilled:

1. Investigate IoT quantitative and qualitative characteristics and reveal Edge Computing technology specifics.
2. Simulate the adaptive resonance theory grounded systems and determine the EC environment influences on the self-organizing classifiers complexity and performance.
3. Implement a fixed structure ANN classifier based system for image stitching purposes and experimentally investigate the EC infrastructure influence on its energy-efficiency.

Internet of Things Characteristics

Performed analytical literature review on the artificial neural networks place in a edge computing confirmed the need and importance in the development of ANN based classifiers for Internet of Things devices. In order to propose ANN classifier based IoT systems for EC tasks first of all it is necessary to investigate the IoT quantitative and qualitative characteristics and reveal a EC technology specifics. That is why firstly here, in the Section 2.1 the analysis directions and investigation methodology are outlined. Then in the Section 2.2 the experiment on IoT performance investigation focused on network control is described. It is intended to reveal the impact of different network parameters on different types of topologies – Internet and IoT, and to find out, whether network parameters or topology itself has the greatest impact. Later on, in the Section 2.3 according to derived results, simulation of EC gateway with incorporated classifier for live video streaming is described. The aim of this simulation is to reveal the technique of classifier implementation in EC related hardware, with further investigation of unsupervised classifier performance and comparison to existing techniques. Lastly, in the Section 2.4 a sensor network data communication protocols investigation is presented. The most popular communication protocols are considered with different weather condition models, revealing the resilience and the performance of particular protocols. The aim of this study is to find out wireless sensor network specifics under quickly changing conditions. Finally, the chapter finishes with the conclusions where the recommendations of IoT systems for EC tasks implementation are highlighted, too.

The corresponding main research results are published in author publications (Skirelis 2018; Skirelis, 2019; Skirelis, Navakauskas, 2018b; Skirelis, Navakauskas, 2017b) and are presented in multiple issues of international scientific conference “Advances in Information, Electronic and Electrical Engineering” (AIEEE) (Vilnius, Riga, 2017, 2018, 2019).

2.1. Analysis Directions

The **IoT** commonly is described by the “things” (devices) configuration and the network characteristics. To reveal the **EC** specifics, we chose to focus on the **IoT** devices and network characteristics, since most of them are inherited to **EC**, with additional, the sensors data term included. Three main aspects are used to characterize the **IoT**: network topology, link characteristics and node configuration. Lets discuss each of them in more detail.

Network topology. Topology knowledge, i.e., the physical relation (edges) of the nodes enables to clusterize the network segments and classify them in a useful way, so that decision on optimal link selection can be done. Depending on the task, we see the ability to apply the graph theory for fast alternative route selection in the cases when node becomes unavailable, thus increasing the reliability of the network itself when compared to non-**IoT** topologies. Moreover, topology knowledge enables intelligent role (master–slave) switching techniques to be applied, which is an important advantage for **EC**. **IoT** network topology is highly related with an fault resiliency, thus it is especially important for wireless sensor networks for scenario where a node connectivity may be interrupted, the **IoT** technology ensures maximal chance of recover or reconfiguration with further successful packet delivery to its destination.

One of the main aspects of network topology in **IoT** is to optimize the hops – the number of intermediate nodes through which a data packets from one node to another needs to travel, and the range – the overall distance a data packet needs to travel to reach its destination. What is more described by the **IoT** network topology – the scalability, a total number of nodes that can be included in a single network. Accordingly, three main topological architectures are considered: point-to-point, star and mesh networks. The mesh network is most commonly found in **IoT**, depending on if it is fully connected (each node has a failback route) or not, the performance of the network is found to be different. That is the direction we used for topological analysis: construct multiple mesh topologies with at least 10 nodes, having multi-hop architecture, with at least one topology to be fully connected and having edges with different characteristics.

Link characteristics. Most important parameters of the link for **IoT** are: the bandwidth the link is capable to transfer, measured in amount of unit data per unit

time (b/s) and the delay – the parasitic unit of time which network hardware takes to initialize necessary buffers for packet transmission, usually measured in the order of ms. Combining those link parameters, derivative parameters like link jitter, round-trip-time and others can be calculated. Based on beforementioned parameters, with the help of intelligent algorithms, optimal route for data transmission may be selected dynamically.

The link characteristics highly depend on the way things are connected (wired or wireless) and on communication protocol used ([Ethernet](#), [WiFi](#), [Bluetooth](#), [Zig-Bee](#), [LoRa](#), [LTE](#) and etc.). Each communication protocol targets to optimize particular layers of abstraction layers (OSI, TCP/IP and others) and has different priorities on the link characteristics, i.e., security, power, throughput constraints. It is known that modern data transmission protocols (like [HTTP](#), [MQTT](#), [CoAP](#) and others) introduce some deflection in link characteristics through data compression, data validity checks and other features, however we do not take that into account and use only raw data transmission as a reference for all experiments in this chapter except for Section 2.4. As for today, wired [Ethernet](#) and wireless [WiFi](#) communications are still found to be most popular for [IoT](#) applications, that is why we only concentrate on these standardized link characteristics to perform the following experiments and analysis.

Node configuration. [IoT](#) network usually consists of different types of devices, those are described by its connectivity subsystem (bandwidth), computation resources ([CPU](#) performance and core count, [RAM](#) amount), onboard storage, power requirements and special features. The first three parameters are found to be general for both Internet and [IoT](#), that is why we select those parameters for analysis to maintain comparability of the results. Energy efficiency and cost effectiveness plays the most important role for [IoT](#) specifically, analysis of it is provided in the Chapter 4.

Worth to note that the node configuration knowledge is meaningful in [IoT](#) (and [EC](#)) routers, gateways, since normally, the node device configuration is not stored, nor shared across the network. However, network related control data can have unique features, for example DHCP server in router stores the node network interface vendor ID, which can represent specific devices would allow to evaluate all connected nodes and identify [IoT](#) devices among them. Furthermore, identification of devices in the network allows to know more meaningful information about the node, like its hardware resources, on the basis of which, the assignment of tasks for the nodes may be altered automatically.

Based on the mentioned specifics and characteristics, further directions of analysis are constructed: experimentally investigate and verify [IoT](#) and [EC](#) specifics through performance evaluation of Internet and [IoT](#) varying the network parameters or the network topology, likewise perform analysis on proposed optimal link selection technique performance for live video streaming application.

2.2. Internet of Things Specifics

The upcoming 5G technology and the demand for near real-time response services raise the need for optimizing current **IoT** solutions. The demand of cloud technologies for big data processing (Kurasova *et al.* 2014) is tremendous, while the insights show that with the realization of Industry 4.0 it would not be possible to process such enormous data capacities. The **EC** technology is highly suitable for low-power devices, such as wearables, mobile gadgets or even sensors themselves to adopt computing tasks from common cloud technology. Local or even global internetworking of physical devices (things) introduces new constraints on the **IoT** paradigm. The most important undertaking in superior **EC** realization is an appropriate job scheduling strategy selection aiming to fulfill timeliness criteria, i.e., to perform a task before its data became obsolete. However, to support this, the proper law of reorganization of a dynamic network architecture must be determined. Numerous studies investigate the job-scheduling problem by proposing algorithms for the optimal **VM** placement across the computer network. However, most of them do not consider the optimization of network architecture. Therefore, in this section, we describe modelling and performance analysis of different topologies with an aim to determine what has greater impact on **IoT**: the network parameters or the network topology.

2.2.1. Investigation Setup

The virtual experimental environment based on *EdgeNetworkCloudSim* tool was set up to perform simulations. The two cases of the simulation were arranged:

- the Internet,
- the **IoT**.

A. Experimental Environment

Simulations were performed in *Java* virtual machine using *Ubuntu* ver. 16.04 operating system. A *Java* **VM** ensures isolation, thus guarantees that random initiation and simulation execution is operating system independent. The analysis of results was performed with *MATLAB* ver. 2017 software. The network topology was generated by *BRITE* tool.

B. Controllable Parameters

Network topologies, including delay and network capacity parameters between nodes, are defined by *BRITE* structure. In the simulation software, each node resource for Internet and **IoT** is defined by the following parameters: million instructions per second (MIPS), the capacity of random-access memory (**RAM**), the

Table 2.1. Edge computing device configurations used in the study

Name	MIPS	RAM, MB	Bandwidth, kB/s	Storage, MB
$\mathcal{M}_0$	10,000	4,096	5,120	10,240
$\mathcal{M}_1$	2,000	512	1,024	1,024
$\mathcal{M}_2$	1,000	128	1,024	128
$\mathcal{M}_3$	100	16	512	64

bandwidth, and amount of data that can be stored in a particular node. For the Internet case, $\mathcal{M}_0$ device configuration was used for simulation on all nodes, while for IoT devices three types of configurations $\mathcal{M}_1$ – $\mathcal{M}_3$ were applied. Device parameter values (cf. Table 2.1) were selected on the basis of the real hardware that was used in industrial EC and cyber-physical systems for studying connected automated vehicles and large scale video analysis platforms.

Device $\mathcal{M}_1$ represents Raspberry Pi 3 hardware, $\mathcal{M}_2$ – a LinkIt Smart 7688 Du IoT device, and $\mathcal{M}_3$ – ESP32 SoC. Selected values represent the maximum values of the parameters these devices can reach. The assignment of configurations to IoT devices was done in random order.

The ECD configuration parameters were selected to represent typical low-efficiency IoT devices realistically, as well as an average consumer PC.

C. Varied Parameters

For the Internet and IoT comparison purposes, the three sets of network topologies were defined: $\mathcal{A}_0$ (according to Seufert *et al.* (2017)), $\mathcal{A}_1$ and $\mathcal{B}$.

Fig. 2.1a illustrates the IoT network topology with 13 nodes (devices) and 15 edges denoted by $\mathcal{A}_0$. This topology serves as a reference since it has been analyzed by others. In order to reveal the sensitivity of the IoT network structure, the additional 16th edge was added (note a brown dotted link in Fig. 2.1a).

Consequently, the 8th node device is offloaded and the topology denoted by $\mathcal{A}_1$ becomes fully connected. Given that numerous topologies can be defined, the purpose of this study is not to discover the best one. Thus only a single alternative topology denoted by $\mathcal{B}$ with 22 nodes and 27 edges (Fig. 2.1b), was generated. Topology $\mathcal{B}$ is 30% more complex than previous topologies ($\mathcal{A}_0$ and $\mathcal{A}_1$).

In all considered network topologies, device 0 is a packet initiator device exclusively having following parameters: up to 1 Gbps throughput with 5 ms delay for Internet and 54 Mbps throughput with 100 ms delay (cf. IEEE 802.11g) for IoT, respectively. As a network load, data chunks with a size of 1 GB are used.

Three edge (link) parameter sets are employed:

- baseline, $\mathcal{P}_0$ – having three types of links with varying bandwidth (ϖ) and delay (τ) (cf. Fig. 2.1): $\varpi_0^{E1} = 10$ Gb/s, $\tau^{E1} = 0$ ms; $\varpi_0^{E1} = 50$ Gb/s, $\tau^{E2} = 10$ ms; and $\varpi_0^{E3} = 100$ Gb/s, $\tau^{E3} = 50$ ms;

Fig. 2.1. Pictorial representation of network topologies used in investigation:

a) $\mathcal{A}_0$ and $\mathcal{A}_1$ network topologies having 13 nodes (devices) with 15 and 16 edges; b) $\mathcal{B}$ network topology having 22 nodes with 27 edges

- double bandwidth, $\mathcal{P}_{2B}$: $\varpi_0^{E1} = 20 \text{ Gb/s}$, $\tau^{E1} = 0 \text{ ms}$; $\varpi_0^{E2} = 100 \text{ Gb/s}$, $\tau^{E2} = 10 \text{ ms}$; and $\varpi_0^{E3} = 200 \text{ Gb/s}$, $\tau^{E3} = 50 \text{ ms}$
- double delay, $\mathcal{P}_{2D}$: $\varpi_0^{E1} = 10 \text{ Gb/s}$, $\tau^{E1} = 0 \text{ ms}$; $\varpi_0^{E2} = 50 \text{ Gb/s}$, $\tau^{E2} = 20 \text{ ms}$; and $\varpi_0^{E3} = 100 \text{ Gb/s}$, $\tau^{E3} = 100 \text{ ms}$.

A dynamic fluctuation of the network bandwidth, a jitter, was imposed between nodes. Random values ξ_h^G were generated using Gaussian distribution ($\mu = 0$ and $\sigma = 1$) and then used as a jitter multiplier in order to simulate the fluctuations in the all edge types:

$$\varpi_{h,i}^{Ej} = \xi_h^G \times \varpi_{0,i}^{Ej}, \quad (2.1)$$

where $\varpi_{0,i}^{Ej}$ – the bandwidth of i th node, ξ_h^G – a jitter coefficient following Gaussian distribution at h th experiment; i – a node index; h – an experiment index, j – a edge type.

All specified experiments were repeated a one hundred times. Fig. 2.2 presents an exemplary histogram of used jitter coefficients selected within a reasonable $[-0.03, 0.03]$ range.

Fig. 2.2. Exemplary histogram of bandwidth jitter coefficients used during simulation experiments

2.2.2. Investigation Results

After performing a total of 1,800 simulations for each of two network configurations and three different topologies with varying bandwidth and delay parameters (baseline, double bandwidth, and double delay) repeated 100 times, the investigation procedure was performed.

A. Procedural Steps

The distribution of individual experiment estimates of packet delivery success ratios were found to be Gaussian. That is why the success of packet delivery events is evaluated by estimating the ratio values:

$$\widehat{E}_h^S(\mathcal{T}, \mathcal{L}) = \frac{E_h^S(\mathcal{T}, \mathcal{L})}{E_h^T(\mathcal{T}, \mathcal{L})} \times 100\%; \quad (2.2)$$

together with mean and standard deviation of success ratio values as follows:

$$\widehat{E}_m^S(\mathcal{T}, \mathcal{L}) = \frac{1}{N} \sum_{h=1}^N \widehat{E}_h^S(\mathcal{T}, \mathcal{L}); \quad (2.3)$$

$$\widehat{E}_d^S(\mathcal{T}, \mathcal{L}) = \text{std}_{h \in [1, N]} \widehat{E}_h^S(\mathcal{T}, \mathcal{L}), \quad (2.4)$$

where $E_h^S(\mathcal{T}, \mathcal{L})$ and $E_h^T(\mathcal{T}, \mathcal{L})$ are the numbers of successful packet delivery and total events during h th experiment given a specific network configuration $\mathcal{T} \in \{\mathcal{A}_0, \mathcal{A}_1, \mathcal{B}\}$ and specific set of parameters $\mathcal{L} \in \{\mathcal{P}_0, \mathcal{P}_{2B}, \mathcal{P}_{2D}\}$, $\widehat{E}_h^S$ is an estimate of packet delivery success ratio during h th experiment, $\widehat{E}_m^S$ and $\widehat{E}_d^S$ are mean and

dispersion values of packet delivery success ratio estimates, h is an experiment index, and N is a total number of experiments, $N = 100$.

The distribution of individual experiment estimates of packet delivery failure ratios was found to be exponential. As a result, the failure of packet delivery events is evaluated with a 95% confidence interval by estimating the ratio values:

$$\widehat{E}_h^F(\mathcal{T}, \mathcal{L}) = \frac{E_h^F(\mathcal{T}, \mathcal{L})}{E_h^T(\mathcal{T}, \mathcal{L})} \times 100\%, \quad (2.5)$$

and mean together with variance values of failure ratio as follows:

$$\widehat{E}_m^F(\mathcal{T}, \mathcal{L}) = \frac{1}{N} \sum_{h=1}^N \widehat{E}_h^F(\mathcal{T}, \mathcal{L}); \quad (2.6)$$

$$\widehat{E}_v^F(\mathcal{T}, \mathcal{L}) = \text{var}_{h \in [1, N]} \widehat{E}_h^F(\mathcal{T}, \mathcal{L}), \quad (2.7)$$

where E_h^F is the number of packet delivery failures, while $\widehat{E}_h^F$ is an estimate of packet delivery failure ratio during the h th experiment, $\widehat{E}_m^F$ and $\widehat{E}_v^F$ are mean and variance values of packet delivery failure ratio estimates.

The status difference for success (2.8) and failure (2.9) cases between **IoT** and Internet network configurations is expressed as an algebraic difference between mean ratio values:

$$\Delta \widehat{E}_m^S(\mathcal{T}, \mathcal{L}) = \widehat{E}_m^{S_{\text{IoT}}}(\mathcal{T}, \mathcal{L}) - \widehat{E}_m^{S_{\text{Int}}}(\mathcal{T}, \mathcal{L}); \quad (2.8)$$

$$\Delta \widehat{E}_m^F(\mathcal{T}, \mathcal{L}) = \widehat{E}_m^{F_{\text{IoT}}}(\mathcal{T}, \mathcal{L}) - \widehat{E}_m^{F_{\text{Int}}}(\mathcal{T}, \mathcal{L}). \quad (2.9)$$

By analogy, service duration statistics were analyzed, the inverse exponential distribution was obtained, corresponding estimation distribution was applied, and statistical measures of mean and variance values were calculated as follows:

$$\widehat{D}_m(\mathcal{T}, \mathcal{L}) = \frac{1}{N} \sum_{h=1}^N D_h(\mathcal{T}, \mathcal{L}); \quad (2.10)$$

$$\widehat{D}_v(\mathcal{T}, \mathcal{L}) = \text{var}_{h \in [1, N]} \widehat{D}_h(\mathcal{T}, \mathcal{L}), \quad (2.11)$$

where D_h is the duration of service execution during the h th experiment given specific network configuration $\mathcal{T}$ and specific set of parameters $\mathcal{L}$, $\widehat{D}_m$, and $\widehat{D}_v$ are the mean and variance values of service execution durations.

Fig. 2.3. The results of success and failure ratios represented as mean and standard deviation values for 100 simulation repetitions over different configurations and varying parameter sets of networks

B. The Analysis of Results

Packet delivery states for three different topologies within the two different network configurations, Internet and IoT, are presented in Fig. 2.3.

The mean success and failure ratios together with their dispersions are compared to different parameter sets: base, double bandwidth, and double delay. The results show that:

- Double bandwidth or delay for the Internet configuration results in narrowing a distribution of successful packets, while the mean ratio value changes slightly;
- The failure ratio on the Internet configuration is more sensitive to the doubled bandwidth or delay and results in an increase of the packet failure ratio;
- Dispersion value is dependent on the mean ratio value and shows the lowest distribution for base parameters, whereas the double delay has the highest impact on the dispersion increase.

Fig. 2.4 presents the integral difference between IoT and Internet network configurations in terms of mean success and failure ratios of packet delivery. It is seen that an increase in success rate decreases the failure rate in 5 of the 9 cases; these events are unrelated and are not the opposite measures.

Fig. 2.4. Mean packet delivery status (success, failure) difference results over 100 simulation repetitions emphasizing network topology and influence of parameter sets on different networks

Fig. 2.5. Mean service duration and variance results of inverse exponential distribution estimation per different topology and parameter sets over 100 simulation repetitions

Fig. 2.6. Service duration difference results between the Internet and IoT configuration, including duration difference on different topologies and different parameter sets

Mean service duration and variance values presented in Fig. 2.5 further demonstrate that, in most cases, worsening network conditions for IoT configurations results in the decreased mean value; however, the distribution of extremums is increasing and in mean value it is much higher than in Internet configuration. It is also evident that the IoT configuration is more sensitive to network topology than to parameter fluctuation. In the majority of cases, the Internet configuration duration is dependent more on the bandwidth available than on the increased delay.

Fig. 2.6 reveals the service duration differences between IoT and Internet network configurations and resulting insights:

- topology with 22 nodes and 27 edges is more sustainable to fluctuations of network parameters across all simulated cases;
- Fully connected IoT topology is more resistant to the fluctuation of network parameters and results in better mean packet delivery duration than Internet configuration;
- The fluctuation of packet delivery duration difference in a base configuration is negligible; it confirms that under ideal conditions, there exists an insignificant difference between IoT and Internet topologies.

C. Discussion of the Results

The first investigation i.e., on data packet delivery, has supported the idea that device configuration has a little impact on successfully delivered packets, because its mean ratio value to total events that occurred in the routing activity fluctuated near 50% for both Internet and IoT configurations. To reveal the differences, the distribution value (Fig. 2.3 error bars) has been considered. It is evident that varied network parameters start to reflect in successful packets ratio value, thus doubling the delay; the probability of successful packet delivery increases to 65% and even more. The failure ratio statistics confirm the principle: the more bandwidth and fewer delays, the fewer failing packets. Our contribution has not submitted any new insights based on distribution differences in terms of failing packets ratio. Such a behavior summarized in a differential view (cf. Fig. 2.4) leads to a recommendation on using fully connected network topologies in the EC systems for increasing the success packet count and decreasing the failing packet count. It should be noted here, that apart from success and failure, there exist additional packet delivery states, which have not been taken into account, and which, in our opinion, do not have an impact on the system performance.

The results of the second investigation, i.e., on service duration analysis, partially coincide with the results obtained in the related research stating that the double delay parameter increases the total service time in the same topology. However, we have obtained slightly different results in terms of double bandwidth parameter; the related simulation experiments have shown that it has no effect on the service time, but that is true only when investigating the single device configuration. That is why the two device categories, Internet and IoT, have been established. It has been revealed that on highly resourced Internet configuration exploits a fully connected network contributes to decreasing the service time, while in IoT case, as we assume, hardware limits occur, and the service time decrease does not happen in topology with double bandwidth. This raises a hypothesis that network stack is capable of adapting itself to an exhausted network and/or device resource limit conditions producing maximum performance with priority on hardware resources. For this reason, with Fig. 2.6 we wanted to show how the node device hardware affects on the same network topology under exact same conditions. Surprisingly, the service duration value of baseline conditions varied in a $\pm 5\%$ margin, and the best network utilization scenario is seen over small network topologies confirming the recommendation from the related research on reducing and segmenting the schema (topology in our case) in order to increase the system performance.

Additionally, it is notable that the experiments have been performed in a virtual environment with real network and device parameters, together with related studies that prove conformity with the simulator used. Close-to-real conditions have been assured to be simulated, and the results acquired have been very close

to reality. In order to confirm the results of these experiments on real hardware, further research is needed.

2.3. Edge Computing Specifics

This section describes proposed technique of classifier incorporation into **EC** gateway, used simulation environment and varied parameters.

Most investigated **EC** gateways solve problem of limited hardware capabilities by adaptive scheduling focusing on the response time between the service components. That is why different approach is researched – we propose adaptive most optimal link between microservice and end user selection technique based on unsupervised k -means clustering of links statistics and **VM** available resources, with further classification and decision making.

A live video streaming service simulation process is here considered. This work-flow is selected because of high demand (Bilal, Erbad 2017) and active development by corporations like Facebook, Inc., Google, LLC or Twitter, Inc. Basic work-flow (Fig. 2.7) consists of single capture and multiple end user devices. Single input video stream requires encoding and transcoding procedures of initial stream to satisfy criteria for different bitrates and resolutions of particular devices, thus it requires device in the middle to perform these tasks. Currently, the cloud based architectures are used for this purpose, the servers with high resources available are performing the encoding-transcoding task, however the static physical geolocation of these servers introduces high latency because it involves multiple hops to reach the user. Partial solution for high latency issue is the use of dynamically geolocated content delivery networks (**CDNs**) as the buffers caching the transcoded video streams, therefore decreasing the latency of transcoded streams only.

The actual streaming is performed by sending video chunks (using **HLS**, **HDS** protocols) and aiming to guarantee the best quality experience for each user while maintaining highest buffering ratio and lowest latency. The main claim is that by the application of k -means clustering in **EC** gateway it is possible to reduce end-to-end latency, with them main aim to increase user experience quality by application of classifier based stream distribution gateway on **EC** topology.

Fig. 2.7. Simulated live-video streaming service work-flow

2.3.1. Investigation Setup

Three different topologies were simulated, each providing different path from video source to user multiple devices (Fig. 2.8):

- cloud computing, denoted by $\mathcal{C}_0$;
- cloud computing using buffering, denoted by $\mathcal{C}_{\text{CDN}}$;
- edge computing, denoted by $\mathcal{D}$.

For the cloud data stream down-link buffering a well known and widely used Content Delivery Network (CDN) techniques was selected.

Fig. 2.8. Three main network topologies used in simulations: a) cloud computing ($\mathcal{C}_0$); b) cloud computing with **CDN** buffering ($\mathcal{C}_{\text{CDN}}$); c) edge computing ($\mathcal{D}_0, \mathcal{D}_{\text{ANN}}$)

A. Experimental Environment

EdgeCloudSim simulation software together with *WEKA* *k*-means clustering library was used to perform simulation on *Java VM* running *Ubuntu* ver. 16.10 operating system. Simulation results are saved to text files, then analysis using *MATLAB* ver. 2017 software package is performed.

B. Employed Techniques

Data stream transcoding task in the first two cloud topologies (Fig. 2.8a and 2.8b) is performed by cloud servers, while in **EC** topology (Fig. 2.8c), transcoding is

performed locally by gateway device. Edge gateway communicates with cloud servers only by control data and state reports, all main tasks performing locally.

In order to pilot ANN based simple classifier's benefits two techniques of work (transcoding) distribution in edge gateway were implemented: regular and novel (that is based on k -means clustering). That yields in total four different processing cases that were researched: $\mathcal{C}_0$, $\mathcal{C}_{\text{CDN}}$, $\mathcal{D}_0$, and $\mathcal{D}_{\text{ANN}}$.

A regular work distribution technique is based on the scheduling algorithm proposed by Scoca *et al.* (2018). Algorithm evaluates each i th VM by a quality score q_i that is a harmonic mean of connection scores q_i^L (latency), q_i^{BW} (bandwidth) and VM resources score q_i^{RES} , recalculated on each connection instance. Later on, the VM with highest quality score from a set $Q = \{q_1, q_2, \dots, q_n\}$ is selected by the scheduler for particular end user.

A novel work distribution technique is based on the use of k -means clustering. Proposed technique relies on most suitable link between edge gateway and ECD (user device) selection based on round trip time (RTT). It is determined by regularly sending ICMP (ping) packet to user device. The time it takes to respond back represents device to gateway latency. To prevent network flooding by ping packets, latency query is performed each 1,000-th data packet, and last 50 latency values are kept in a buffer.

Fig. 2.9 represents local connection diagram, where edge gateway consists of infinite number of micro data centers – VM based cloudlets, with only limiting factor being total hardware resources of gateway.

Fig. 2.9. Edge gateway internal structure and connection links

Optimal link connections for all links between edge gateway and user devices are determined by performing classification. Firstly, unsupervised clustering into $k = 3$ clusters (interpreted as a most suitable $\mathcal{S}_1$, mediocre $\mathcal{S}_2$, and undesirable $\mathcal{S}_3$)

Fig. 2.10. View of VM clustering using k -means algorithm in particular instance of time

of each link s_i based on it's estimated latency $\hat{\tau}_i$ and remaining CPU resources $\hat{\rho}_i$ by k -means algorithm is done. Mathematically, the used clustering operation can be expressed by:

$$\arg \min \sum_{j=1}^3 \sum_{s \in \mathcal{S}_j} \|s - c_j\|^2, \quad (2.12)$$

here $s_i = \{\hat{\tau}_i, \hat{\rho}_i\}$ – i th link data; $\mathcal{S}_j$ – j th cluster set; c_j – cluster centroid (Wu 2012). VM clustering exemplary view is presented in Fig. 2.10. It is a snapshot of total available 302 VMs clustered into three clusters according to the resources available in corresponding VM and the link latency from perspective of end-user candidate to join the stream.

The random VM that falls into the cluster $\mathcal{S}_1$ is kept as a potentially suitable VM to serve the end user candidate. If selected VM does not establish connection in expected latency period τ_0 , connection is attempted to VM from the next cluster ($\mathcal{S}_2$). Similarly, if this attempt also fails, the random representative from the last cluster $\mathcal{S}_3$ is tried. The sequence is repeated until connection is established.

In order to handle situation when link may be switched to other source after each clustering operation, check for last connected source is performed to maintain connection with the same VM. Scenario when no VM can serve link is solved by opportunistic await time (in ms range) while at least one VM finish current tasks.

Assignment of link to cluster is performed by calculating sum-of-squares dis-

tance to cluster centroid – this is specified in k -means algorithm. Worth to note, that initially, at the start of the algorithm there are no latency values measured, so that clustering is impossible, thus links are initially assigned in a random order.

C. Controllable and Varied Parameters

Each network topology used in simulation is defined by *BRITE* tool, describing nodes and edges connection rules together with network delay and maximum bandwidth parameters. Network devices are described by their resource parameters, main of them are: number of (virtual) CPU, MIPS (million instructions per second) value, the capacity of RAM and a amount of storage for buffered video stream data. For simulations, video streaming service with 5 GB data chunk size is used.

After the simulation, only the following simulation software outputs are taken into account: network delay, processing time and service time. Network delay in our case represents amount of time for a bit of data to pass from capture device to end user device. Processing time represents a time frame between packet arrival to processing (encoding and transcoding) node and outset from it. Service time is a total time data packet elapses to pass full route. All measures were evaluated as a mean value. Under small loads of ≤ 400 users, all three topologies result in constant results, to reveal advantages and disadvantages, user count in range from 400 to 1600 was selected.

Table 2.2. Machine parameters used for simulation

Name	CPU	MIPS	RAM, GB	Storage, GB
$\mathcal{M}_1$	4	2,400	8	840
$\mathcal{M}_2$	8	4,800	16	1,680
$\mathcal{M}_3$	26	20,000	70	1,680
$\mathcal{M}_4$	2	2,400	16	500

During simulation of cloud computing topology ($\mathcal{C}_0$), non-restricted on the total number of $\mathcal{M}_3$ type VMs configuration was applied. The cloud computing with CDN buffering topology ($\mathcal{C}_{\text{CDN}}$) was configured with up to 10 geographically distributed $\mathcal{M}_4$ VMs. Edge computing topology ($\mathcal{D}_0$ and $\mathcal{D}_{\text{ANN}}$) simulation was configured to allow up to 20 $\mathcal{M}_1$ and $\mathcal{M}_2$ types VMs to be allocated, limiting 10 instances per node. Corresponding to real hardware parameters values were used for VM configuration (Table 2.2).

2.3.2. Investigation Results

After performing a total of 84 simulations on each of four different network configurations with varying number of users using the streaming service (in the range

of 400 to 1600) and repeating each instance for 3 times, the investigation procedure of the results was performed.

A. Procedural Steps

Proposed technique results are compared to cloud, CDN and score based scheduler solutions in main user experience representing terms: mean network delay, mean processing time and mean service time accordingly. Network delay and processing time (the VM execution time) values were extracted from the simulation software generated reports, derivative metric of service time was calculated as the sum of network delay and processing time value. Overall service time characteristic is seen as the most valuable indicator for video streaming quality estimation in EC environment.

Trustworthiness of results was guaranteed by repeating each simulation 3 times and confirming the same results, that ensures no error is introduced by random generated initial variables in simulation software internally.

B. The Analysis of Results

First measure in analysis – mean network delay, its values over different topologies are provided in Fig. 2.11. It can be seen, that results are pretty constant, indicating that increase of load on system does not affect network delay mean value. However different topologies have different values because of different physical length between nodes, as it affects data bit propagation time dramatically.

Results on mean service time (see Fig. 2.12) show constant values over whole range for Cloud topology, while hardware resources limited topologies reach their

Fig. 2.11. Mean network delay over number of users for different topologies

Fig. 2.12. Mean service time over number of users for different topologies

throughput limits at 1200 simultaneous users. It is seen how k -means classifier initially performs worse than scheduler based approach, but later, when more and more values of ping entries are sampled to classifier, its performance improves.

Results on processing time (Fig. 2.13) are expected to be similar to service time trends, since they are derived measures, although absolute processing time mean values differs dramatically over topologies. It is seen that cloud solution itself has best processing time, but considering very limited hardware resources provided for [CDN](#) and edge gateway, they are performing quite well, until bottleneck is reached.

Fig. 2.13. Mean processing time over number of users for different topologies

C. Discussion of the Results

The investigation of mean network delay showed that incorporation of intermediate buffer device in the middle of the path constructing dynamically placed **CDN** network results in highly improved delay value compared to conventional cloud topology where all traffic is passed through static cloud servers. This is more confirmed by the incorporation of **EC** gateway instead of **CDN** – routing all source-destination data traffic through locally located gateways resulted in even higher improvement in delay value compared to cloud. The results confirms the idea of local geographical placement of processing device for live video streaming applications reduces mean network delay drastically. However, the value of network delay is more or less constant for different number of connections (users) concluding that the delay value solely depends on the link characteristics and is not affected by the load. This may be false for higher number of users, but we tested reasonable range of users that are usually served by a single server. The proposed classifier based inherits same performance gain, and shows similar, or a little better results than Edge Scheduler technique concluding that **EC** gateway solutions provide similar results in mean network delay term and the specifics of the algorithm have little impact.

Mean service time results showed that the computational resources take place for this measure – cloud architecture with high resources available can serve any load (users in the range of this simulation) serving the live video streaming service with constant processing duration. In the case of computational resource limited devices incorporated in the path (**CDN**, **EC** gateway) the performance gain in service time is only available when low load conditions are met (resource limit is not reached), with load increase the performance gain decreases, and reaching a certain limit – results are even worse than cloud. Both **EC** gateway techniques perform similarly, however classifier based technique allows to reach hardware limits maintaining better performance (case of 1000 users), it may be the specifics of the Edge Scheduler algorithm to predict the increasing load and adapt to it evenly.

Lastly, results of mean processing time is found to be the difference of mean service time and mean delay values, and is the direct indicator of actual hardware processing time. It confirms all beforementioned specifics found from the results, additionally concluding that the actual processing time is considered as real-time (≤ 100 ms) for analysed case, and is true for **CDN** and **EC** architectures up to certain load (1100 users), higher loads for these architectures results in near real-time (> 100 ms) processing. It can be stated, that the network delay for most cases has greater impact than processing when considering overall service time – the user experience measure.

To summarize, cloud topology results in the lowest processing time, ensures stable network delay and service time because of static network characteristics and high resources available. Placing resource limited nodes close to the edge results

in greatly improved service time and network delay, which are main parameters representing overall user experience in live video streaming service.

2.4. Sensor Network Specifics

IoT is tightly related to wireless sensor networks (WSN), that are often used as data sensing and acquiring layer (Bruzgiene *et al.* 2017). To reveal the sensor network specifics, the dynamically reconfigurable Mobile Ad-hoc Network (MANET) with parameters of a real system is simulated and investigated. The main feature of MANET – it does not have any centralized management mechanism, nor strict topological infrastructure, hence each node acts as a router for remaining nodes in the network. Adverse weather conditions drastically impact MANET performance, special attention should be paid on the deployment of this solution in a certain environment where different weather conditions may cause delays or complete data loss. Performance degradation in terms of overall system performance is valid for most popular protocols. An integrated water resource management system in a municipality could be considered as a real-life example of data transmission over a wireless ad hoc network links in adverse weather conditions. Different automated reading metering systems have been used in the last years, which use smart WSN based meters. The reliability of the monitoring and control systems in municipalities is an important issue, however, very few research projects have been devoted to reliability and resilience to the adverse weather conditions.

2.4.1. Investigation Setup

The basic block scheme (Fig. 2.14) represents the simulated WSN system that collects and transmits data on water consumption and water temperature at the premises, also water flow and pressure values in the water distribution network.

Fig. 2.14. The system architecture of the investigated wireless sensor network

Basic data flow consists of the sensor itself, which registers and converts analogue water flow and pressure signal values to digital ones internally, then data is processed into messages and is sent to the gateway (concentrator) using Short Range Devices ISM unlicensed telemetry band, frequency of 868 MHz. Later, the data is forwarded through the classic Internet network, where they are processed, stored and managed by the final user in a cloud server through a web interface. The communication protocol between the sensors and the gateway is the main object of analysis in this Section.

A. Experimental Environment

Simulation and analysis using *MATLAB* software package running on *Windows* 10 x64 machine was performed. *MANET* simulation toolbox with external routing protocols libraries was used. The ability to substitute real *RF* sensors and network parameters together with the possibility of further information analysis were the main criteria for the selection of such a simulation environment. The nature of open-source toolbox permits simulation to be tuned close to the real environment by substituting the weather disruption parameters and alter the simulation scenario directly in the toolbox code.

B. Employed Techniques

Several popular tree and mesh topology-based protocols were used for *MANET* simulation. Lets briefly discuss main features of these protocols.

1-Hop Neighbour Discovery and Clustering protocol. The technique of continuous discovering neighbour nodes' presence is involved. Node polling is performed by broadcasting advertisement messages, with further registration of corresponding returned packets representing the alive node. The Neighbour Discovery mechanism is a part of the IPv6 standard.

On-Demand Multicast Routing Protocol (ODMRP). It is a soft state mesh-based protocol, where a subset (group) of particular nodes performs forwarding of the multicast packets. On-demand driving mode ensures mobility robustness and low power consumption because the whole network topology knowledge is not stored, the acquired data is routed dynamically.

Hop Limited Multicast Routing Protocol (HLMRP). It uses classical tree-based topology with software controlled limited hops available for selecting the route. Nodes presence in the tree data is collected through fixed time interval heartbeat messages broadcasted in the network.

Dynamic Source Routing (DSR). DSR protocol incorporates two mechanisms: Route Maintenance and Route Discovery, maintaining the route caches in the nodes itself. Packet sender caches complete ordered route list of the nodes through which packet will be sent to the receiver, the cache is updated as nodes learn new routes. Check of the route is incorporated in the packet's header, and

at route failure, the Route Discovery mechanism is initiated. Verification of route links is performed by querying acknowledgement (ACK) packets.

C. Controllable and Varied Parameters

During the simulation, the packets with a data length of 512 bytes with the period of periodical data transmission is set to 100 ms, thus maintaining constant bitrate of send data. Node locations are depicted in Fig. 2.15. Simulation time for each protocol is algorithmically fixed to 15 s. The network consisting geographically randomly placed 10 nodes is simulated, five of them are data transmitters (red, indexed by 3, 4, 5, 6 and 7) and three are receivers (green, indexed by 1, 2 and 8). The remaining two nodes (yellow, indexed by 9 and 10) serve as data transceivers in the case when they are present in the direct route path between communicating nodes. The main parameters of the RF network and the sensors were selected considering the characteristics of the real hardware. The carrier frequency of 868 MHz was selected because it meets the requirements set for a public band regulated by the norms of the EU, it also meets ERC-REC-70-3E requirements and is widely used by low-power technologies, i.e., LoRa and Sigfox.

Fig. 2.15. Field view of the geographic distribution of nodes used in MANET simulation

The physical network layer is characterized by the parameters obtained from the Friis transmission equation (Huang *et al.* 2020):

$$\frac{P_r}{P_t} = \frac{A_r A_t}{d^2 \lambda^2}, \quad (2.13)$$

where P_r and P_t denote power at receiving and transmitting antenna terminals, A_r and A_t – the effective aperture of antennas, d represents the distance between

antennas, λ denotes the wavelength of the radio frequency. The values used for simulation are presented in Table 2.3.

Table 2.3. Physical parameters used in simulation

Parameter	Value
Carrier frequency, MHz	868
Modulation type	BPSK
Bitrate, Mb/s	10
Transmission power P_t , dBm	20
Receiver sensitivity P_r , dBm	−95
Geographical area $W \times H$, m	2000×2000

Modulation, bitrate and power characteristics were derived from industrial sensor characteristics, a square geographical area of average size was selected for the sake of simplicity, as area imperfections are not the main object of the present analysis.

Three different weather disruptions models (fog, rain, and snow) were applied to four investigated protocols by translating their attenuation equations to node positions and physical distance between neighbor nodes. Disruption models used for the simulation are characterized by specific attenuation for RF signals, measured in dBm/km.

The fog model takes into account the attenuation of signals that propagate through fog or clouds:

$$\gamma_f = K(f)M, \quad (2.14)$$

here $K(f)$ – specific attenuation coefficient, dependent on frequency of the signal f ; M – the liquid water density.

The rain attenuation model is the following:

$$\gamma_r = 1.076R^{0.67}, \quad (2.15)$$

here R – the rain rate.

The snow attenuation model used for the simulation:

$$\gamma_s = 0.00349 \frac{R^{1.6}}{\lambda^4} + 0.00224 \frac{R}{\lambda}, \quad (2.16)$$

here R – the snow rate; λ – the wavelength of signal transmitted through RF link.

Corresponding frequency and wavelength values are substituted according to simulation parameters described in Table 2.3. The rain and snow rate value in all case is fixed to 100 mm/hr. The simulation scenario is then tuned up according the calculated attenuation coefficient.

2.4.2. Investigation Results

Total of 16 different simulations for each data protocol and different disruption models applied, the analysis of results was performed. Worth to note, that each simulation was limited in time programatically, so that different number of total sent packets per protocol is achieved, we only take into account average absolute number of dropped packets, this allows to evaluate overall data protocol performance in fixed range of time.

A. Procedural Steps

A first simulation was launched without any disruption model applied, to figure out what is the baseline (natural) dropped packet rate for particular protocols on specified geographical placement. Later simulation results were grouped by the disruption model and were analyzed group-wise.

B. The Analysis of Results

After the simulation, the analysis of results was performed. The baseline results have been taken into account as a reference value for the evaluation of the sustainability of the protocols. The results of the simulation are presented in Figure 2.16.

It is evident that baseline (reference) values for all protocols are more or less evenly distributed and are non-zero, which implies that even without disruptions

Fig. 2.16. The dependency of average number of dropped packet on weather-based disruptions within different protocols and baseline values

such network is highly loaded and some packet drops occur. With the rain attenuation model applied, the *ODMRP* protocol shows the highest number of dropped packets (35.84%). For fog and snow models, all protocols except *ODMRP* operate with good confidence. *HLMRP* protocol results show the best performance for this particular environment in terms of a number of dropped packets, it has proved to be very sustainable to rain attenuation factors – 8.56% dropped packets.

Table 2.4. Summary of dropped packets ratio per protocol over all disruptions

Protocol	Average with baseline, %	Average without baseline, %
Neighbor	37.89	46.01
HLMRP	27.09	27.76
ODMRP	39.51	40.36
DSR	43.37	50.24

However, after the analysis of the overall protocol performance over all disruptions as the ratio of the dropped packets to total sent packets, it is evident that the *HLMRP* protocol demonstrated the best performance. The results, where the average is calculated considering the baseline (no disruptions) value and without it, are provided in Table 2.4.

C. Discussion of the Results

Snow disruption model is found to be the most stressful for all four analysed protocols, however fog causes a little lower impact but is still very disruptive. The rain model caused the highest variation between protocols; thus, it may be concluded that rain is the most disruptive factor within this particular simulation. Different protocols features different conditions on collecting the knowledge about network topology and neighbor nodes, that is mostly seen from the rain disturbance results – *HLMRP* and *DSR* protocols queries the network, and in the case of high ratio of disturbance (big rain drops physically blocks or reflects the portion of the signal) the packet is rerouted through alternative path quickly delivering the packet without significant packet drops. Worth to note, that baseline results (no disruptions applied) shows that some natural packet drops occur in *WSN* and are around the magnitude of 50 dropped packets and is protocol independent, that may be related to wireless *RF* specifics, where various signal collisions or disruptions due to external factors may occur. This phenomenon confirms that the reliability of *IoT* network depends not only on software means, but depends on the hardware connection type as well. The analysis of different *WSN* data routing protocols and evaluation of their performance in term of dropped packets confirmed that the most appropriate communication protocol needs to be selected for particular environmental conditions while implementing *WSN* based *IoT* systems.

2.5. Conclusions of the Second Chapter

Results of performed investigation of **IoT** quantitative and qualitative characteristics confirms the **EC** suitability for the real-time applications and affirms that:

1. Even simple clustering in **EC** gateway serving middle size user pool can decrease mean service time in comparison to conventional techniques: the proposed work distribution technique based on k -means clustering outperformed regular technique based on the scheduling algorithm in mean service time in the range from 600 to 1200 simultaneous users.
2. The following specifics are relevant to the **EC**:
 - delay has the bigger than bandwidth influence on a data packet routing success;
 - the use of **EC** in comparison to cloud computing topology reduces mean network delay and service time;
 - **EC** hardware resource limit cause data processing bottlenecks that are highly manifested with the increase of the users number.
3. The **CDN** and **EC** technologies enable real-time (≤ 100 ms) video streaming applications for relatively small number of users, opposing to cloud technology where such short processing time is not possible.
4. In the **RF WSN** for **IoT** use case the natural packet drops are inevitable, furthermore **HLMRP** data communication protocol performs the best of all considered protocols.

Revealed **EC** technology specifics ground following *recommendations*:

- a fully connected network topology for sensors data to gateway level is preferable because it is less susceptible to network bottlenecks;
- in order to increase the performance of the system in jittery conditions (in terms of bandwidth) the network has to be scaled down or the whole topology splitted into segments (i.e., subnets) or both.

Self-Organizing Classifier for Cell Colonies Counting

In the Chapter 2 performed pilot investigations on Internet of Things characteristics revealed edge computing specifics and even enabled formulation of some recommendations. That empowers here to tackle the second objective of the Thesis – simulate the adaptive resonance theory grounded systems and determine the EC environment influences on the self-organizing classifiers complexity and performance. The main working hypothesis – structural adaptiveness of ART classifiers in comparison to support vector machines, as good representative of fixed structure classifiers, can improve classification quality in cell colonies counting application. Thus this chapter starts with the uncommon research application domain short presentation and introduction to selected approach (Section 3.1). Then in the Section 3.2 the original cell colonies image parameterization technique is presented in details. It lays down a framework for a different classifiers comparison. Later on, in Section 3.3, considered intelligent classifiers: ART1, ART2 and SVM, are algorithmically characterized with further brief discussion of principles of operation. The experimentation procedure and comparative investigation results for cell colonies counting with the considered classifiers are in details presented in the Section 3.4. Finally, the chapter is completed with the conclusions on the fulfillment of the second task of the Thesis.

The main research results are published in author publications (Skirelis, Navakauskas, 2018b, Skirelis, Navakauskas, 2017b) and are presented in multiple issues of international scientific conferences “Electrical, Electronic and Information Sciences” (eStream) (Vilnius, 2017, 2018).

3.1. Application Domain and Approach

Cell colonies counting is an important subset of cytometry research (Herberg *et al.* 2015; Lambert *et al.* 2017), demanding high throughput analysis with accurate results at high speed. While manual counting persists to be the paragon, automation interest has been increasing for many years, exhibiting much consistent results.

Manual counting of cell colonies is the slowest and sometimes least reliable method, due to human error factor, especially while dealing with large number of cell colonies, as well as complex patterns of biological samples. Technological systems are required to replace staff interventions by automated intelligent solutions. This is very true when spoken about modern medicine and biology, where people contact to hazardous substances should be minimized (Dursun *et al.* 2016; Savickiene *et al.* 2009), and further investigation performed in safe environment. Studies showed (Grainys *et al.* 2016; Lucinskis *et al.* 2014) that electroporation is a decent technique for treatment of cancer (Merzviniskyte *et al.* 2006; Miyashita *et al.* 2017; Savickiene *et al.* 2012; Zhang *et al.* 2016). Cell colonies counting is critical to evaluate electroporation impact on microorganisms *in vivo*. To perform intelligent examination of biological samples, complex computer vision solutions take place (Choudhry 2016). For parameterization of biological sample images (Hoerr *et al.* 2013) intelligent algorithms (Navakauskas *et al.* 2014) bring benefits in both of time consumed and overall cost of investigation. Time consumption is expressed and compared as interval of cell colonies detection by assessor manually to interval of developed technique, taking into account count of false colonies detected, as well as undetected colonies. Research on automated cell colonies analysis has taken up several decades and today it is essential tool in every biomedical laboratory. Current computational power and capability of algorithms allows indirect analysis to be accomplished with accuracy near to direct measurements (Rawat *et al.* 2017; Suga *et al.* 2015). Dozens of solutions, in both, hardware (Bourbeau, Ledebouer 2013; Chen, Zhang 2009; Jung, Lee 2016; Kwee *et al.* 2017) and software (Kato *et al.* 2016) configurations are provided to count cell colonies, however cost of those solutions is high. Importance in profit is expressed as demand in software solutions at low-cost (Chalfoun *et al.* 2016).

To fulfill EC criteria of fast and accurate implementation, local image processing approach is followed. Simple way of automatic cell colonies counting by heuristic image processing here is comprised by substitution of common thresholding with multi-class neural network classifier, that stipulates intelligent classification approached by classifier training procedure. ART classifiers are considered, since they are able in an automatic manner to identify and recognize patterns, through incremental learning, solving stability problem by acquiring new knowledge without disrupting existing one. They are compared with heuristic and SVM classifiers.

Biological samples investigated in this research are *Candida albicans* cells, isolated out from patient with clinical diagnosis of atopic dermatitis, which were plated on the *YPD agar* with incubation period of 48 hours at constant 30 °C temperature. For the intelligent classifiers training and testing purposes, cell colonies contours were extracted from these biological sample images according to (McInerney, Terzopoulos 2000).

The main *working hypothesis* – structural adaptiveness of **ART** classifiers in comparison to **SVM**, as good representative of fixed structure classifiers, can improve classification quality in cell colonies counting applications.

3.2. Cell Colonies Image Parameterization Technique

The original technique for cell colonies image parameterization is developed in order to act as a framework for different (simple and intelligent) classifiers comparison. This section in details presents the technique: general work-flow, image registration specifics, Petri dish area determination, cell colonies detection and parameterization.

3.2.1. General Work-Flow

Developed cell colonies image parameterization technique generally consist of three aggregated processing stages:

- image registration,
- cell colonies detection,
- cell colonies verification.

Image processing in each stage is performed separately, in total 16 different algorithms are employed. The structure of the created cell colonies image parameterization system is shown in Fig. 3.1.

Fig. 3.1. Structure of developed cell colonies image parameterization system

Fig. 3.2. Data flow of developed technique for parameterization of cell colonies images

The system accepts up to full HD (1920×1080 px) resolution, colored or grey scale images of biological samples with even background, containing Petri dish up to 100 mm in diameter placed in the center. Output of the system is set to result in true cell colonies count, for particular application, in addition it can be configured to output true colonies position in image.

Fig. 3.2 reveals complete data flow of developed image parameterization technique containing 16 steps divided to three main execution stages. Specifics of most important image processing stages are described in more details in further subsections.

3.2.2. Specifics in Image Registration

Leftmost part of Fig. 3.2 explains sequence of procedures used in registration stage.

Light intensity histograms are calculated in order to determine optimal parameters for a precise registration. Lighting distortions are compensated in a following way. Regions, where Petri dish is absent, i.e., corners of image rectangle, are used in order to estimate four mean background intensity P_{raw} values. Then a plane of “mean” intensity of image is fitted on these values and consequently subtracted from the original sample image. More specifically, the following intermediate steps are carried on.

Supporting image creation. In order to reduce unnecessary computational load associated with normalization factors computation, original image I_{raw} is re-sampled by a factor k :

$$I_{\text{res}}(i, j) = I_{\text{raw}}(ki, kj), \quad \forall i \in [1, N_x/k], [1, N_y/k], \text{ with } k = \text{const}, \quad (3.1)$$

where N_x and N_y are limits of the raw image, I_{raw} .

Image segments creation. A central part image $I_{\text{res}}^{(1)}$, that consists of re-sampled image I_{res} data taken from a maximum size circle inscribed in it (outside of the circle area is masked, i.e., pixels are set to zero value), is created. Then four corner image segments $C_{\text{res}}^{(1)}, C_{\text{res}}^{(2)}, C_{\text{res}}^{(3)}, C_{\text{res}}^{(4)}$ are created. They consists of smaller segments of re-sampled image I_{res} data taken from circles inscribed in four image corners, correspondingly. The outside of the circle areas is masked similarly. Moreover, it is ensured, that corner image segments and the central part image do not have the same raw data.

Image normalization. Raw image intensity is normalized by plane fitting of corner segments data:

$$I_{\text{raw}}^{(1)} = I_{\text{raw}} - \text{plane} \left(I_{\text{res}}^{(1)}; C_{\text{res}}^{(1)}, C_{\text{res}}^{(2)}, C_{\text{res}}^{(3)}, C_{\text{res}}^{(4)} \right), \quad (3.2)$$

where $\text{plane}(I; \cdot)$ – a linear approximation (plane fitting) of mean intensity values of the image I , given the central part and four image corner segments data.

3.2.3. Petri Dish Area Determination

The next step in the technique aims to detect Petri dish area in the image. Petri dish displacement in respect to camera shooting direction is compensated by the image geometric transformation T_{res} . It is estimated based on ovalness, i.e., differences of Petri dish contour from the circle. Based on the inherited Petri dish feature – round shape, inscription of circle into the image is used to estimate cell

colony place – working area. Along that, regions outside working area are used to calculate average background intensity by 2D Otsu algorithm (Sha *et al.* 2016). Additionally, estimated working area is checked whether geometric transformation is necessary to compensate camera misalignment to Petri dish and to find the center of an image. Exact Petri dish region K_{pd} is found by application of Hough transform with up to 10 circles.

3.2.4. Cell Colonies Detection

Initial presumed cell colonies are found by concatenating intensity peaks in Petri dish area, which later on are filtered by *heuristic classification* step. It constitutes logic of separation by boundary length of concatenated intensity peaks (3.4) where threshold t_0 represents maximal length of true cell colony boundary and is set to experimentally found value of 250 px.

In order to perform translation of initial cell colonies data to an array of inputs for classifier, the following actions are taken. First of all each cell colony spot area is cropped preserving size, shape and pixel intensity of initial sample image (Fig. 3.3).

Fig. 3.3. Illustration of initial cell colonies segmentation results

Extraction of cell spots C_s is performed by applying intensity threshold t_{otsu} only to Petri dish area of image. From obtained cell cores collection false spots are repealed by performing check if spot is in closed polygon – likely cell boundary C_c . Since some false and stucked together cell colonies are still captured, parameterization is necessary.

3.2.5. Parameterization of Cell Colonies

Prior to parameterization of cell colonies, the intensity normalization for all cropped areas C_{sr} and C_{cr} to range from 0 to 1 is done. This is crucial in order to meet ART inputs specification (Sumathi, Vijayalakshmi 2015) and to preserve relational

intensity differences of all cell colony spots. Such transformation allows to substitute cropped areas for multiple classifiers and compare the results among them. For the case of **ART1** classifier, values are also converted to binary values by performing round operation, then region outside the cell colony is filled with black (zeros) and each cell colony image matrix is reshaped into a vector form, with the following aggregation into **ART1** classifier input array. To perform parameterization, **ANN** based classifiers – **ART1** and **ART2** are used. For comparison purposes heuristic and **SVM** classifiers are implemented as well.

In *heuristic classification* case, simple separation by cell colony boundary length is done:

$$H_o(n) = \begin{cases} 1, & \text{length}(n) < t_o; \\ 0, & \text{otherwise,} \end{cases} \quad (3.3)$$

here threshold t_o is set manually by expert and represents maximal length of true cell colony boundary. As **ART1** and **SVM** classifiers input data, cell colony regions are segmented (Marmanis *et al.* 2018) and transformed to binary pattern (Fig. 3.4), plane is represented as 0, boundary – as 1. Binary representation of colony is converted to vector and combined in colonies array.

SVM classifier with linear kernel separates categories dividing plane of input vectors by a clear gap that is as wide as possible, resulting in binary decision. Two class learning function is used:

$$\Phi_k(s_s, s) = s_s^T s, \quad (3.4)$$

where s_s – the support vector; s – the input vector.

Fig. 3.4. Extraction of binary patterns for their following classification by the use of **ART1** classifier: a) image of Petri dish with detected cell colonies; b) binary patterns of in (a) outlined cell colonies

ART1 classifier characterizes cell colonies by their shape and size, which gives the opportunity to exclude non-colony classes from total colony count and increases overall accuracy of the process. However trained *ART1* classifier outputs many, e.g., 29–33, different automatically generated classes, assigning each colony region to particular class. Such classification does not reduce output data in comparison to input data, since no input sample is removed yet. After the training of *ART1* classifier, classes that represent true cell colonies have to be found in order to yield a two-class classification. Calculation result is considered to be an array of relationally indexed position of cell colony spots detected by classifier P_c .

ART2 classifier is used to classify initially detected cell colonies. Preliminary experimental study showed that *ART2* classifier for this particular application produces 28–33 different output classes. Since *ART2* incorporates ability to change classifier structure, we have minimized this phenomenon by performing supervised training step. During training step top-down and bottom-up weights are calculated. Trained classifier weights are saved and used for new classifier instances thus reducing new class generation to minimum, herewith maintaining known class count even if no sample falls in particular class (producing empty class).

The parameter to be extracted from *ART* classifiers results is a total number of cell colonies G_c . It is estimated by dividing *ART* classes into two groups: true and false cell colonies, and summing up the number of cell colonies that belong to the true class. By varying *ART* vigilance ρ parameter different classification results can be achieved (Nelson Jr 2011), e.g., cell colonies can be separated according size, roundness. Thus care must be taken not only in selection of the threshold of binary classifier but also selection of vigilance parameter's value.

3.3. Considered Intelligent Classifiers

The main component of in previous section introduced original cell colonies image parameterization technique is a classifier, that determines a cell colonies boundaries (contours). Thus this section briefly at the algorithmic level presents considered to investigate intelligent classifiers: *ART1*, *ART2* and *SVM*.

3.3.1. Adaptive Resonance Theory 1

Adaptive resonance theory refers to the self-organizing neural network based architecture that clusters the pattern space by producing weight vector, used to address problems such as pattern recognition and prediction. The first version of *ART* algorithm is developed for binary vectors clustering (*ART1*). It's basic principle of operation is presented in the Algorithm 3.1.

Algorithm 3.1 (Adaptive Resonance Theory 1)

Let: $s_i^{(0)}$ – input binary vector, $i \in [1, N^{(0)}]$; $N^{(0)}$ – a total number of neurons in l -th layer, note $N^{(1)} = N^{(0)}$; $N_{\max}^{(2)}$ – a maximum number of outputs (classes); $w_{ij}^{(1 \rightarrow 2)}(t)$ – feedforward (bottom-up) synaptic weight value at the time instance t , $j \in [1, N^{(2)}]$; $w_{ji}^{(2 \rightarrow 1)}(t)$ – feedback (top-down) synaptic weight value at the time instance t ; $G1$ and $G2$ are gain signals; R – a reset signal; μ – a learning rate; ρ – a vigilance parameter; T – maximum number of epochs.

```

1   $\mu > 1$ ;    $0 < \rho \leq 1$ ;    $t = 0$ ;
2   $0 < w^{(1 \rightarrow 2)}(0) < \mu / (\mu - 1 + N^{(0)})$ ;    $w^{(2 \rightarrow 1)}(0) = 1$ ;
3  for each  $s^{(0)}$  do
4     $G1 = G2 = 1$ ;    $R = 0$ ;    $t = t + 1$ ;
5     $F1 = s^{(1)} = s^{(0)}$ ;    $F2 = 0$ ;
6    while  $(R = 0) \wedge (\Delta w^{(1 \rightarrow 2)} \neq 0) \wedge (\Delta w^{(2 \rightarrow 1)} \neq 0) \wedge (t \leq T)$  do
7      for  $j = 1, 2, \dots, N^{(2)}$  do
8        if  $F2_j \neq -1$  then  $s_j^{(2)} = \sum_{i=1}^{N^{(1)}} w_{ij}^{(1 \rightarrow 2)}(t) s_i^{(1)}$ ; end if
9      end for  $j$ 
10     Find  $j^* \in [1, 2, \dots, N^{(2)}]$  such that  $s_{j^*}^{(2)} \geq s_j^{(2)}$ .
11     if  $\|w_{j^*}^{(2 \rightarrow 1)}(t) s^{(1)}\| / \|s^{(1)}\| \geq \rho$  then
12       for  $i = 1, 2, \dots, N^{(1)}$  do
13          $w_{j^*,i}^{(2 \rightarrow 1)}(t+1) = w_{j^*,i}^{(2 \rightarrow 1)}(t) s_i^{(1)}$ ;
14          $w_{i,j^*}^{(1 \rightarrow 2)}(t+1) = \frac{\mu w_{j^*,i}^{(2 \rightarrow 1)}(t) s_i^{(1)}}{\mu - \frac{1}{2} + \sum_{k=1}^{N^{(1)}} w_{j^*,k}^{(2 \rightarrow 1)}(t) s_k^{(1)}}$ ;
15       end for  $i$ 
16        $R = 1$ ;
17     else
18        $F2_{j^*} = -1$ ;
19     end if
20   end while
21 end for

```

The ART1 is principally a two-layer neural network architecture (cf. Fig. 3.5) framed with two – comparison ($F1$) and recognition ($F2$) layers of binary neurons (outputting values 0 or 1). Each neuron in $F1$ layer has a forward synaptic links to $F2$ layer (weighted by $w^{(1 \rightarrow 2)}$) and backward synaptic links from the same $F2$ layer (weighted by $w^{(2 \rightarrow 1)}$). Each neuron in $F2$ layer has lateral connections with the other neurons in the same layer in order to implement competitive behavior. Additionally there are two gain ($G1$ and $G2$) and one reset (R) subsystems.

Comparison layer receives three signals: input pattern $s^{(0)}$, feedback pattern $s^{(3)}$, and gain $G1$. A neuron in this layer will activate (output 1) if and only if all inputs are high. Recognition layer receives three signals: forwardly by $w^{(1 \rightarrow 2)}$ weighted input pattern $s^{(0)}$, lateral connections stimuli (winning neuron inhibition to others, in Fig. 3.5 shown as dashed synaptic links) and gain $G2$.

Fig. 3.5. The ART1 architecture

Gain $G2$ is computed as the logical OR of all the elements in the input pattern $s^{(0)}$. Usually $G1 = G2$ except when the feedback pattern from $F2$ layer contains at least one non-zero value, then $G1 = 0$. The reset signal is sent to the active neuron in $F2$ layer if the input vector $s^{(0)}$ and the output of $F1$ layer differs by more than selected vigilance parameter ρ value.

ART1 operation (cf. the Algorithm 3.1) starts from at E1–2 shown initialization. Each new input pattern $s^{(0)}$ is applied to the network sequentially (E3–5). Then activation values of neurons in $F2$ layer are computed (E7–9) and the winning neuron by E10 is selected. Gain $G1$ is inhibited, and only $F1$ layer neurons that receive “1” from both $s^{(1)}$ and $F2$ remain active. The vigilance test at E11 determines if the winning neuron will be disabled for the current input pattern processing and the procedure to find next best neuron winner will be repeated (E18) or ART1 weights accordingly adjusted and by reset the processing continued with the next input pattern (E12–16).

3.3.2. Adaptive Resonance Theory 2

To overcome the limitation of [ART1](#) algorithm, structurally much more complex algorithm accepting discrete or continuous time analog inputs – [ART2](#) is analyzed.

Fig. 3.6. The ART2 architecture

The main concept is similar, as it can be seen from the [ART2](#) architecture (Fig. 3.6), the F_1 and F_2 layers connections to weight vectors and aims to find a sufficiently close match of the input pattern to encoded pattern associated with a class, and if no is found – the new class is created. The main structural difference compared to [ART1](#) structure is the 6 internal compute layers (denoted as $w^{(1)}$, $u^{(1)}$, $q^{(1)}$, $v^{(1)}$, $x^{(1)}$ and $p^{(1)}$) in the F_1 layer. Computation of those layers values are presented in the Algorithm 3.2 lines E5–6 and E14–15. However, there are two ways [ART2](#) classifier can learn – slow and fast. [ART1](#) can only be learning in fast learning mode, equivalently [ART2](#) classifier in fast learning mode is characterized by weights reaching equilibrium in each learning trial and is more appropriate for data in which the primary information is contained in the pattern of components that are small. In slow learning mode only one weight update iteration is performed on each learning trial, that is why it need more epochs for weights to converge to average of learned patterns. For the simplicity and comparability of [ART1](#) and [ART2](#) implementation performance, we only consider fast learning mode.

Algorithm 3.2 (Adaptive Resonance Theory 2)

Let: $s_i^{(0)}$ – the input vector, $i \in [1, N^{(0)}]$; $N^{(l)}$ – a total number of neurons in l -th layer, note $N^{(1)} = N^{(0)}$; $N_{\max}^{(2)}$ – a maximum number of outputs (classes); $w_{ij}^{(1 \rightarrow 2)}(t)$ – feedforward (bottom-up) synaptic weight value at the time instance t ,

$j \in [1, N^{(2)}]$; $w_{ji}^{(2 \rightarrow 1)}(t)$ – feedback (top-down) synaptic weight value at the time instance t ; R – a reset signal; ρ – a vigilance parameter; T – maximum number of epochs; a, b, c, d, α – positive constants.

```

1   $0 < \rho \leq 1$ ;    $t = 0$ ;
2   $w^{(1 \rightarrow 2)}(0) = 1 / ((1 - d)N^{(0)})$ ;    $w^{(2 \rightarrow 1)}(0) = 0$ ;
3  for each  $s^{(0)}$  do
4     $R = 0$ ;    $t = t + 1$ ;    $F1 = s^{(1)} = s^{(0)}$ ;    $F2 = 0$ ;
5     $w^{(1)} = s^{(0)} + a w^{(1)}$ ;    $x^{(1)} = \frac{w^{(1)}}{c + \|w^{(1)}\|}$ ;    $v^{(1)} = \Phi^{(1)}(x^{(1)}) + b \Phi^{(1)}(q^{(1)})$ ;
6     $u^{(1)} = \frac{v^{(1)}}{c + \|v^{(1)}\|}$ ;    $q^{(1)} = p^{(1)} = u^{(1)}$ ;
7    while  $(R = 0) \wedge (\Delta w^{(1 \rightarrow 2)} \neq 0) \wedge (\Delta w^{(2 \rightarrow 1)} \neq 0) \wedge (t \leq T)$  do
8      for  $j = 1, 2, \dots, N^{(2)}$  do
9        if  $F2_j \neq -1$  then  $s_j^{(2)} = \sum_{i=1}^{N^{(1)}} w_{ij}^{(1 \rightarrow 2)}(t) s_i^{(1)}$ ; end if
10     end for  $j$ 
11      $u^{(1)} = v^{(1)} / \|v^{(1)}\|$ ;    $p^{(1)} = u^{(1)} + d w_j^{(2 \rightarrow 1)}(t)$ ;
12     Find  $j^* \in [1, 2, \dots, N^{(2)}]$  such that  $s_{j^*}^{(2)} \geq s_j^{(2)}$ .
13     if  $\|w_{j^*}^{(2 \rightarrow 1)}(t) s^{(1)}\| / \|s^{(1)}\| \geq \rho$  then
14        $v^{(1)} = \Phi^{(1)}(x^{(1)}) + b \Phi^{(1)}(q^{(1)})$ ;  $u^{(1)} = v^{(1)} / \|v^{(1)}\|$ ;  $w^{(1)} = s_i^{(0)} + a u^{(1)}$ ;
15        $p^{(1)} = u^{(1)} + d w_j^{(2 \rightarrow 1)}(t)$ ;    $x^{(1)} = w^{(1)} / \|w^{(1)}\|$ ;    $q^{(1)} = p^{(1)} / \|p^{(1)}\|$ ;
16       for  $i = 1, 2, \dots, N^{(1)}$  do
17          $w_{j^*,i}^{(2 \rightarrow 1)}(t+1) = \alpha d u + (1 - \alpha d + \alpha d^2) w_{j^*,i}^{(2 \rightarrow 1)}(t)$ ;
18          $w_{i,j^*}^{(1 \rightarrow 2)}(t+1) = \alpha d u + (1 - \alpha d + \alpha d^2) w_{i,j^*}^{(1 \rightarrow 2)}(t)$ ;
19       end for  $i$ 
20        $R = 1$ ;
21     else
22        $F2_{j^*} = -1$ ;
23     end if
24   end while
25 end for

```

As can be seen from [ART2](#) (Algorithm 3.2), it introduces few more parameters a, b, d – positive constants and small value c , those are used for preventing neural activities becoming infinite in the cases when no signal is present on the layers. For the sake of simplicity, we do not investigate those values and their impact on the actual results, leaving them auto tuned by the [ART2](#) training algorithm.

3.3.3. Support Vector Machines

The [SVM](#) classifier with only linear kernel is used in following investigations, because it ensures fast algorithm execution compared to non-linear kernels, complex calculations would not only make the algorithm inoperable in low power [ECDs](#), but also it cannot be compared to relatively simple (computational-wise) [ART1](#) and [ART2](#) algorithms.

The [SVM](#) (cf. Algorithm 3.3) is a single hidden layer architecture (Fig. 3.7) with each neuron in this layer performing kernel function $\Phi_k^{(1)}(\cdot, \cdot)$ calculations on each input signal $s^{(0)}$ and the neuron related support vector s_{sj} . Each neuron output $s_j^{(1)}$ is weighted by $w_j^{(2)}$ and the results are summed. Additionally, the output neuron has a bias weight $w_0^{(2)}$.

Fig. 3.7. The linear SVM architecture

Algorithm 3.3 (Support Vector Machine)

Let: $s_i^{(0)}$ – input vector, $i \in [1, N^{(0)}]$; $N^{(0)}$ – a total number of inputs;
 $s_{si}^{(1)}$ – support vectors, $i \in [1, N^{(1)}]$; $N^{(1)}$ – a total number of hidden neurons;

$w_i^{(2)}$ – a second layer weights values, $i \in [0, N^{(1)}]$.

```

1  for each  $s^{(0)}$  do
2    for  $j = 1, 2, \dots, N^{(1)}$  do
3       $s_j^{(1)} = \sum_{i=1}^{N^{(0)}} s_{ji}^{(0)} s_{ij}^{(1)}$ ;
4    end for  $j$ 
5    for  $i = 1, 2, \dots, N^{(1)}$  do
6       $s^{(2)} = \sum_{i=1}^{N^{(1)}} w_i^{(2)} s_i^{(1)} - w_0^{(2)}$ ;
7    end for  $i$ 
8  end for

```

Only recall of **SVM** classifier is considered in Algorithm 3.3, because training of **SVM** classifier, opposing to **ART** family classifiers is not performed during the recall, and is thought as separate process. For this pasticular investigation, the training of **SVM** was performed in *MATLAB* R2020a software using machine learning toolbox, the values of trained **ANN** (weights and support vectors) then were substituted into recall algorithm.

As for the recall algorithm, due to simplicity (therefore fast execution) of linear-kernel, it basically consists of three loops, one of which iterates over input signals Algorithm 3.3 line E1, second loop iterates over supplied support vectors on the line E2 calculating the product of input value to corresponding support vector value (hence the linear-kernel), finally, the third loop on the line E5 iterates on signal values of neurons in hidden layer and calculates the output layer value by applying the weight of corresponding synapse. Worth to note, that with different kernel used, the algorithm length and complexity can be increased in orders of magnitude due to exponent or even polynomial computations may be required.

3.4. Experimentation Results

Experimental investigation of developed technique was performed in order to check reliability of cell colony count as also as efficiency of cell colony parameterization. For all investigations the best **ART** classifiers were found in the steps more thoroughly described in following subsection.

3.4.1. Investigation Procedure

Images of biological samples being placed in Petri dish were gathered using constructed stand installation (Fig. 3.8) incorporated with SMX-12A2x digital camera. It additionally consists of the stand with the ability to to change the distance to the sampling object and the artificial focused ligh system – incadescent bulb with wide spectrum of visible light. The camera is connected to workstation computer using 1 Gb/s ethernet interface, to ensure high uncompressed data stream of images and takes images up to 60 frames/s on 1280×720 px resolution with 8–12 bits/px RGB intensity.

Dataset. For the investigation the dataset of over 70 biological sample images that manifest different light intensities, Petri dish positions, biological samples and even cell colony growth stages is collected. All images contain 100 mm diameter Petri dish with cell colonies placed inwards. Randomly selected 30 images, having different light intensities, cell colony numbers, shapes and sizes, are used in the experiment. The snaphots of the image processing algorithm at main steps are presented in Figure 3.9.

Controllable parameters. During registration and detection stages, for each biological sample image specific thresholds of intensity gradient and it's derivative were elucidated. That guarantees correct image registration and Petri dish area determination. For heuristic classifier maximum length of cell colony boundary t_0 is set to 250 px. **ART1** classifier parameter – vigilance ρ_{ART1} is experimentally tuned to value 0.001, in such a way that ensures minimal number **ART1** classes to be constructed during training phase. For the **ART2** classifier the vigilance parameter ρ_{ART2} is experimentally tuned to value of 0.7. For **SVM** classifier a linear type kernel function is selected.

Fig. 3.8. Camera stand installation used for the images of cell colony capture

Fig. 3.9. Snapshots of the image processing main steps: a) original in gray, I_{raw} ; b) registered in norm. gray, I_{gsn} ; c) raw contours and peaks, C_{cr} , C_{sr} ; d) masks applied, K_{sp} ; e) determined working area, I_{pd} ; f) cell colonies classified, G_{c}

Assessment parameters. All results of different cell colonies parameterization approaches were compared against true number of cell colonies, manually counted by the experts. Based on confusion matrix data, sensitivity, specificity, accuracy, diagnostic odd ratio (DOR), precision and F1 score (harmonic mean) were estimated. Moreover Youden's J index was calculated. These parameters were chosen in order to ensure classifiers results to be assess in terms of prevalence dependent and independent characteristics.

Procedural steps in ART1 case. The best ART1 classifier was found out in two steps optimization procedure. Chosen sample images (I_i , $i = 1, \dots, 30$) were divided into two sets ($\mathcal{T}$ and $\mathcal{V}$) for ART1 training and validation, correspondingly. In order to reduce final results dependency on the dataset, three classifier cases (ART1 _{n} , $n = \{1, 2, 3\}$) were investigated:

1. $\mathcal{T}_1 = \{I_1, \dots, I_{10}\}$ and $\mathcal{V}_1 = \{I_{11}, \dots, I_{30}\}$;
2. $\mathcal{T}_2 = \{I_{11}, \dots, I_{20}\}$ and $\mathcal{V}_2 = \{I_1, \dots, I_{10}, I_{21}, \dots, I_{30}\}$;
3. $\mathcal{T}_3 = \{I_{21}, \dots, I_{30}\}$ and $\mathcal{V}_3 = \{I_1, \dots, I_{20}\}$.

The selection from the ART1_n classifiers the best one using assessment parameters comprises the first step in the procedure.

The aim of the second step is to find out the best set of ART1_n learned classes that combined together yields the best estimate of cell colonies number. This is a combinatorial problem with, e.g., $2^{30} \approx 1$ billion combinations to be examined (in case ART1 learned 30 classes). In order to reduce computational burden and simplify the search procedure following actions were taken. ART classes that most probably represent true cell colony $k_{\text{CC}+}$ were elucidated by picking 10–12 first classes in the set after the sorting it in descending order according to the number of detected cell colonies. Very similarly, ART classes that most probably don't represent true cell colony $k_{\text{CC}-}$ were elucidated by picking 10–12 first classes in a set after the sorting it in descending order according to the number of detected false cell colonies. Given in total 29–33 classes learned by ART1_n classifiers, such approach enabled to reduce the combinatorial problem to a moderate ($2^9 = 512$ combinations) size and solve it by the full search procedure.

Procedural steps in ART2 case. For the best ART2 classifier determination, two steps optimization procedure was performed, too. Image dataset $I_j, j \in [1, 30]$ was split into thirds (10 images each), taking 1/3 of samples for training and remaining 2/3 – for validation. Depending on which third of dataset was taken for training, a set of three classifiers ($\text{ART2}_n, n = \{1, 2, 3\}$) was received. Thus assessment of the set of ART2 classifiers and selection of the best one constitutes the first step optimization.

Fig. 3.10. Illustration of the best ART2 classifier selection procedure

The second step of optimization is done on a basis of individual ART2_n representatives, searching for a best set of learned classes that combined together yields the best estimate of cell colonies number. It allows to increase overall reliability of cell colonies counting. For computationally practical second step optimization initially: a) classes were sorted in descending order of true positives and 12 first class indexes Idx_P were assigned to belong to cell colony set; b) similarly classes

were sorted in ascending order of true negative and 12 last class indexes Idx_N were assigned to belong to not cell colony set. Fig. 3.10 illustrates how true positive TP_i and true negative TN_i were processed. In such way remaining classes index Idx_R size was reduced up to 9, enabling to perform full search procedure on $2^9 = 512$ combinations and elucidate which remaining classes has to be assigned to a cell colony set.

3.4.2. Reliability of Cell Colony Counting

Table 3.1 shows confusion matrix results of reliability measures. Based on reliability measures best $ART1_n$ and $ART2_n$ classifiers are selected (indicated by grey background). It can be seen that while for $ART1$ best classifier is constructed with second training subset, for $ART2$ classifier first subset of samples used for training yields better results. The most important measures for classifier evaluation selected to be Diagnostic Odds ratio (DOR) and Youden's J index, which are single indicators of classifier performance. Higher values of those measures denotes better performance, thus are main selectors for best in class representatives.

Table 3.1. Investigated classifiers reliability results for cell colony classification

Classifier*	F1 score	Accuracy	Precision	DOR	Youden's J
heuristic	0.906	0.890	0.840	209.89	0.764
SVM	0.879	0.873	0.902	48.85	0.748
ART1 ₁	0.654	0.683	0.795	6.21	0.388
ART1 ₂	0.939	0.933	0.928	200.90	0.864
ART1 ₃	0.705	0.721	0.818	8.46	0.458
ART2 ₁	0.907	0.890	0.834	506.29	0.765
ART2 ₂	0.879	0.863	0.834	48.18	0.714
ART2 ₃	0.793	0.788	0.837	14.75	0.582

* – grey background outlines best in class representatives.

Sensitivity versus specificity graph (see Fig. 3.11) presents results of all considered classifiers. It is known that topmost left position indicates perfect classifier – the higher point above diagonal is, the better classification it results, so it can be concluded that $ART2_1$ classifier outperforms heuristic and SVM classifiers in terms of sensitivity versus specificity ratio – Youden's J index, and it is superseded by $ART1_2$ classifier showing the best result.

Fig. 3.12 presents reliability measures, top numbers indicates best to worst classifier result. It is evident that $ART1_2$ shows the best results in all presented measures – F1 score, accuracy and precision. The $ART2_1$ classifier shows the second overall performance in these terms, concluding that ART family classifiers outperforms heuristic and SVM classifiers. One more examined measure (not in-

Fig. 3.11. ROC analysis results for all considered classifier classes: heuristic, SVM, ART1_n and ART2_n

Fig. 3.12. Results on F1, accuracy and precision for all considered best classifiers: heuristic, SVM, ART1₂ and ART2₁. Numbers on the top of bars indicate the order: from the best to the worst

cluded in the figure) – **DOR** shows **ART2**₁ classifier as an absolute dominant, yet is excluded from reliability measure graph, due to the fact of being scale less.

3.4.3. Efficiency of Cell Colony Parameterization

To evaluate and compare efficiency of heuristic, **ART1**, **ART2** and **SVM** classifiers, test system in *MATLAB* environment was set up. Measures of total code length and total processing time of 30 biological samples were chosen. Average execution duration is being calculated as independent measure. Experiment was performed on PC with Intel i7 875K CPU (running at 4.3 GHz speed) having 16 GB of **RAM** with memory virtualization feature disabled (to ensure no slow virtualization to disk drive was imposed). Test results are depicted in Table 3.2.

Table 3.2. Investigated classifiers code complexity and execution duration

Classifier type	Code length, lines	Processing duration of 30 images, s	Average execution duration, s
heuristic	36	29.55	0.99
SVM	372	36.38	1.21
ART1	141	34.63	1.15
ART2	284	35.44	1.18

Since each biological sample image contains different number of cell colonies, average execution duration is calculated. It can be clearly seen, that **ART1** classifier, because of it's binary fashion, outperforms **SVM** classifier, in both, algorithm complexity and execution time, for the case of **ART2** classifier it is similar to **SVM** classifier in terms of complexity, but results in faster algorithm execution.

Table 3.3. Comparison of classifiers adjustable parameters

Classifier type	Adjustable parameters
heuristic	maximum boundary length
SVM	alpha; cost; kernel
ART1, ART2	vigilance; maximum boundary length

Each investigated classifier has it's own number of adjustable parameters. Table 3.3 presents parameters that are necessary to tune up, in order to get valid classification results. From Table 3.3 it is seen that **ART1** and **ART2** classifiers have two adjustable parameters, opposing to one parameter for heuristic and three parameters for binary **SVM** classifiers. It is evident that **ART2** classifier being twice as complex as **ART1**, still carries off classification task per image only 30 ms slower.

3.5. Conclusions of the Third Chapter

Research results presented in the chapter prove the main working hypothesis that structural adaptiveness of **ART** classifiers in comparison to **SVM** improves the classification quality in cell colonies counting application and ground the following conclusions:

1. The original cell colonies image parameterization technique is developed enabling to experimentally compare quality of different classifiers.
2. The use of **ART1** classifier increases trustworthiness of the parameterization of images with cell colonies in terms of sensitivity and specificity.
3. The **ART1** classifier outperforms the **SVM** and heuristic classifiers in terms of F1 score, accuracy and precision.
4. The **SVM** and **ART1** classifiers are up to 10 times more complex than heuristic classifier, however their execution is slower only by 15–25%.
5. The best according to **ROC** analysis and Youden's J index classifier is **ART2₁** detecting cell colonies in images with biological samples with up to 99.35% true positives and 76.89% true negatives.
6. The **ART2** classifier brings up similar results to heuristic classifier in terms of F1 score, accuracy and precision, yet the **DOR** value is increased more than 2 times.
7. The **ART2** classifier in terms of true positives outperforms other considered classifiers: **ART1** – by 4%, **SVM** – by 14% and heuristic – by 2%.

Real-Time Classifier for Image Stitching

In line with the main aim to propose and investigate original, artificial neural network classifier based Internet of Things systems for edge computing, in the previous Chapter 3 a adaptive resonance theory based system for cell colonies counting was researched. The key aspects of carried investigations were: a single EC node use scenario, a powerful self-organization nature of the ANN classifier and a still image segmentation functional domain. Here we tackle the third objective of the Thesis – implement a fixed structure ANN classifier based system for image stitching purposes and experimentally investigate the EC infrastructure influence on its energy-efficiency. It intends to extend previously acquired knowledge on different key aspects: a multi (co-operative) EC node use scenario, a simple fix-structure nature of the ANN classifier and a more demanding real-time image streaming functional domain. The main working hypothesis – the use of a fuzzy neural network in decentralized EC can improve node performance in image stitching application.

The chapter starts with the short presentation of the research application domain (Section 4.1). Then in Section 4.2 the image stitching process is examined in order to find algorithm stages that could be decentralized, define the adaptive image resolution selection technique based on the FNN classifier, and to briefly discuss the developed algorithm placing to the ECDs routine. Later on, in the Section 4.3 the investigation setup based on originally prepared test bench is shortly presented. The experimentation procedure and performance evaluation results of FNN classifier based image stitching test bench for EC are presented in the Section 4.4. At last, the chapter completes with the conclusions.

The research results are published in author publications (Skirelis, Navakauskas, 2019) and are presented in international scientific conference “Advances in Information, Electronic and Electrical Engineering” – AIEEE (Liepaja, 2019).

4.1. Application Domain and Approach

Reduction of data processing cost, such as expenditures on data bandwidth or hardware, is a priority task nowadays. These issues are addressed by [EC](#) technology and are especially emphasized for applications like Smart City (Rathnayake, Seneviratne 2018), Virtual Reality (Lee, Lee 2018; Nakajima, Saito 2017), [IoV](#), industrial [IoT](#). No less important aspect is the processing reliability. [EC](#) enabled solutions demonstrate ease maintenance (because of locally owned hardware), allow local off-line operations and ensure high security level when configured properly (since sensitive data is shared among local devices). Automated live image stitching is considered as demanded research area in computer vision (Gonçalves *et al.* 2008; Seo *et al.* 2019), that is why this process is selected as service scenario.

Image stitching is a computationally heavy process thus requires comparatively many hardware resources (Seo *et al.* 2019). At the same time it is the basis for panoramic photography feature available on most modern mobile phones. Current implementation techniques rely on a single image input at a time and post-processing (stitching) done not in a real-time (Omar *et al.* 2018). In the following we consider more complex task when employing multiple cameras and their computational resources real-time image stitching is done (Fig. 4.1). Novelty of our approach lies behind the implementation of decentralized image stitching algorithm and the process distribution technique that is implemented by industry standard deployment system – *Kubernetes*, that places *Docker* containers (Ismail *et al.* 2015) to supported hardware.

Fig. 4.1. Illustration of image stitching procedure

EC technology has shown high success in committing decentralized algorithms (Wang *et al.* 2017) for high resolution image processing (Han *et al.* 2019), introducing techniques like Pyramid-block technology based image reconstruction. Image clustering on IoT devices by classic approaches (Abdel Magid *et al.* 2019) confirms that performance is highly dependent on the data.

Simple, yet effective algorithm for adaptive camera resolution switching was selected – fuzzy neural network. Based on the experience of other researchers, FNN features good universality for EC tasks, e.g., task offloading (Hossain *et al.* 2020), orchestration (Sonmez *et al.* 2019), and great applicability on ECDs and IoT in general. The main *working hypothesis* – the use of a fuzzy neural network in decentralized EC can improve node performance in image stitching application.

4.2. Image Decentralized Stitching Procedure

In order to experimentally investigate EC infrastructure influence on energy efficiency of classifier based system for image stitching, the basic image stitching procedure is modified. Thus this section presents: the image stitching process decentralization set-up, the original adaptive image resolution selection technique based on the FNN use and the proposed algorithm deployment on ECD routine.

4.2.1. Specifics in Work-Flow

In the following two main computer network configurations are considered:

- typical cloud computing, denoted by $\mathcal{C}_0$ (Fig. 4.2a), when raw images are gathered via specified hardware and then are transferred to cloud data center, where analysis of data is performed;

Fig. 4.2. Considered main computer network structures for image acquisition: a) cloud computing, $\mathcal{C}_0$ (image pass-through); b) edge computing, $\mathcal{D}_0$ (image preprocessing)

- typical edge computing, denoted by $\mathcal{D}_0$ (Fig. 4.2b), when complete analysis is performed locally, only state-defining data is shared with cloud.

With the further focus on **EC** implementation, *basic image stitching procedure* consists of the following steps:

- a) features extraction using local invariant descriptors like SIFT (scale-invariant feature transform), SURF (speeded up robust features), KAZE, ORB (oriented FAST and rotated BRIEF) or other;
- b) features matching between features sets of different images;
- c) homography estimation and homography matrix generation;
- d) actual image perspective warping using geometrical transform.

Feature extraction. Preliminary research revealed that the feature extraction is the most computationally intensive task that takes about one third of the whole process execution duration. That is why for the comparison purposes two **EC** approaches are considered. They differ in implementation of feature extraction:

- centralized, denoted by $\mathcal{D}_C \equiv \mathcal{D}_0$ (Fig. 4.3), when master node is assigned and raw images are transmitted from slave node with further analysis steps performed on master;
- decentralized, denoted by $\mathcal{D}_D$ (Fig. 4.4), when slave node performs features extraction step on its own and transmits initial image with corresponding feature set to master.

Fig. 4.3. Illustration of image acquisition in the centralized edge computing, $\mathcal{D}_C$ – a one master node processes all images gathered by the slave nodes

Fig. 4.4. Illustration of of image acquisition in the decentralized edge computing, $\mathcal{D}_D$ – each slave node extracts features and together with image fragment share them to a master node

Feature matching. Taking into account that the algorithm is designed to run on low power SoC, the high performance local feature matching algorithm – Accelerated KAZE (AKAZE) is selected (Sharma, Jain 2020).

Homography calculation. Homography matrix is calculated using the mathematical relation: $H' = HX$, and in expanded form is represented as:

$$\begin{bmatrix} x^* \\ y^* \\ 1 \end{bmatrix} = \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}, \quad (4.1)$$

here pairs (x, y) and (x^*, y^*) represents the coordinates of corresponding points matched in two analyzed images.

Warping. The coordinate system (x, y) of the target image is then rotated to a new coordination system (u, v) according to the relationship between them (Xiang *et al.* 2016):

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} \cos \Phi & -\sin \theta & 0 \\ \sin \Phi & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}, \quad (4.2)$$

where rotation angle θ is calculated as: $\theta = \arctan(h_8/h_7)$.

4.2.2. Adaptive Image Resolution Selection Technique

The Fig. 4.5 reveals proposed adaptive selection of the common (for all slave ECD employed for a particular image stitching) image resolution technique. It relies on the fuzzy neural network classifier that makes the decision based on individual nodes battery charge levels.

Fig. 4.5. The structure of developed fuzzy neural network classifier for adaptive selection of the common image resolution for multiple edge computing devices

The adaptive selection of image resolution principle of operation is elaborated in the Algorithm 4.1.

Algorithm 4.1 (Fuzzy Neural Network Classifier)

Let: $s_i^{(0)}(t)$ – input vector of **ECD** battery charge levels at the time instance t , $i \in [1, N^{(0)}]$; $N^{(0)}$ – a total number of **ECD** slave devices, $N^{(0)} \equiv N^{(1)} \equiv N^{(3)}$; $w_i^{(1)}$ – feedforward synaptic weight value (normalized to 1); $s_{i,\max}^{(0)}$ – a maximum battery charge level of i -th **ECD**; $\phi_j(\cdot)$ – image typical resolutions membership functions, $j \in [1, N^{(2)}]$; $s_i^{(l)}(t)$ – a output signal of i -th neuron in l -th layer.

```

1   $w^{(1)} = 1/s_{\max}^{(0)}$ ;
2  for each  $t$  do
3    for  $i = 1, 2, \dots, N^{(0)}$  do
4       $s_i^{(1)}(t) = s_i^{(0)}(t)w_i^{(1)}$ ;
5    end for  $i$ 
6    for  $j = 1, 2, \dots, N^{(2)}$  do
7      for  $i = 1, 2, \dots, N^{(1)}$  do
8         $s_{ij}^{(2)}(t) = \phi_j(s_i^{(1)}(t))$ ;
9      end for  $i$ 
10   end for  $j$ 
11   for  $i = 1, 2, \dots, N^{(3)}$  do
12      $s_i^{(3)}(t) = \arg \max(s_{i,1}^{(2)}(t), s_{i,2}^{(2)}(t), \dots, s_{i,N^{(2)}}^{(2)}(t))$ ;
13   end for  $i$ 
14    $s^{(4)}(t) = \min(s_1^{(3)}(t), s_2^{(3)}(t), \dots, s_{N^{(3)}}^{(3)}(t))$ ;
15 end for

```

The **FNN** classifier has four feedforward connection layers. The input layer accepts information from slave nodes about their battery charge, $s_i^{(0)}(t)$. It is normalized in the first layer according to individual node battery maximum charge level, $w_i^{(1)}$ (cf. **E1–E5** of Algorithm 4.1). Then, given a image typical resolutions membership functions, $\phi_j(\cdot)$ (see Fig. 4.6 and Section 4.3.2 for specific values), in the second **FNN** classifier layer all nodes information is fuzzified (**E6–E10**). The maximum possible to handle resolutions by corresponding slave nodes are estimated in the third layer (**E11–E13**). Based on the need to select a single image resolution for all the nodes, a minimal resolution from estimated resolutions pool is selected by the fourth layer neuron (**E14**). The $\phi_{\max}$ is a variable expressing the count of total

Fig. 4.6. Membership functions $\phi_j(\cdot)$ used in the developed fuzzy neural network classifier

camera resolutions available in the corresponding system. The intersection point of two membership functions indicate the resolution switching point, to resolve the uncertainty when two ϕ_j functions triggers the output – the mathematical round operation selects the winning function.

The battery charge level read-up function is not a standard one. Thus for the test bench experimentation purposes the SoC hardware was supplemented with ability to measure voltage by means of additional ADC module.

4.2.3. Algorithm Deployment on Edge Devices Routine

Placement of in the previous subsection described stitching procedure as a service on EC nodes is done using *Kubernetes* deployment system with *Docker* containers, the architecture is provided in Fig. 4.7.

Fig. 4.7. Architecture of container placement $\mathcal{K}$ and ECDs WebSockets inter-signalling mechanisms

Such routine ensures the same algorithm to be deployed to any supported architecture Ogbuachi *et al.* (2020); Rausch *et al.* (2021). Based on secure *WebSockets* communication channel to perform data exchange between distributed containers was implemented. Deployment mechanism and orchestration of EC is managed by master *Kubernetes* instance on cloud server.

4.3. Investigation Setup

Experimental investigation of modified image decentralized stitching procedure that employs original adaptive image resolution selection technique based on use of [FNN](#) was performed. This section presents investigation setup: shortly describes the experimental environment, controllable and varied parameters. Noticeably, a developed test bench is presented, too.

4.3.1. Experimental Environment

To perform pilot tests, the test bench was built up. The Fig. 4.8 shows the main components of the test bench: two HD cameras (LifeCam) mounted on servo motors (to ensure equal opposite viewing angles); servo motors controller; two Raspberry Pi 3B computers powered by own lithium-ion (Samsung 25R 2500 mAh) batteries. Initially test was performed on Xilinx Zynq-7000 [SoC](#), but for the sake of simplicity it was replaced with similar *BCM2837 SoC* running *Linux* operating system.

Fig. 4.8. View of developed test bench for automated decentralized real-time image stitching technique investigation

Algorithm was written in *Python* programming language, which later was packed to *Docker* container with all required prerequisites, namely: *OpenCV* library, *Keras* neural network packages, *WebSockets* library. *Kubernetes* environment was installed for seamless application deployment. Battery voltage measurement on each node was performed by 18 bits [ADC](#) connected to [SoC](#) I^2C bus. Power consumption was measured by *Ruideng UM34* USB power meters.

4.3.2. Controllable and Varied Parameters

On the test bench, distance of 30 cm between the cameras was set. Such composition allowed to acquire overlapping viewing angle at cameras rotation of $\alpha = 8^\circ$ precisely set up by servo motors.

Cameras were set to fixed 25 frames/s capture rate and varied over five different resolutions for each camera: 160×120 , 176×144 , 320×240 , 800×600 and 1280×800 pixels. Voltage measurement frequency was set to 3.75 samples/s, with 18 bit resolution introducing quantization error of $15.625 \mu\text{V}$.

4.4. Experimentation Results

This section presents experimentation procedure and performance evaluation results of FNN classifier based image stitching for EC test bench.

4.4.1. Investigation Procedure

Performance of the proposed technique is evaluated in such terms: stitching procedure rate (in frames per second, FPS), average node power usage (in W), average CPU load (in %) and node battery charge level (in %). To increase the trustworthiness of the results and eliminate instantaneous error, each term was measured for 10 min at each resolution.

To evaluate the performance of adaptive resolution selection, total service time of node was measured. Each measurement was started with a fully charged lithium-ion battery (4.20 V level). The end of measurement was triggered when complete battery drain occurred and node was shut down.

There are no known objective stitching quality evaluation algorithms, thus in the investigation expert evaluated the rightness of perspective warp. It was found that big difference between the angle value in homography matrix and actual angle of camera rotation indicates whether correct stitching is possible (however camera rotation value as an input of the system is required to be known).

4.4.2. Performance Evaluation

Results of image stitching average rate dependences on employed image resolution are presented in Fig. 4.9.

Three different network configurations were investigated: typical cloud computing, $\mathcal{C}_0$ (when *Docker* container is deployed to *Azure* server and data for stitching is sent over remote channel), centralized and decentralized EC ($\mathcal{D}_C$ and $\mathcal{D}_D$, correspondingly). It is seen that the image resolution value has a little impact on the stitching rate while performed on remote hardware with high resources available. However the image stitching rate decreases drastically when it is executed on MEC hardware (the difference between centralized and decentralized case consisted of average 2.08 frames/s). Results confirm that for this particular task, hardware resources have more impact than network connection properties. Moreover,

the decentralization of the stitching process by distributing features extraction task leads to increase of stitching rate.

Fig. 4.9. Image stitching average rate dependence on image resolution for three different network configurations: cloud computing, $\mathcal{C}_0$; centralized EC, $\mathcal{D}_C$; decentralized EC, $\mathcal{D}_D$

For correct FNN classifier rules adaptation, terms of average node power consumption and CPU load were investigated (Fig. 4.10).

Fig. 4.10. Averaged power consumption (left bars) and CPU load (right bars) dependences on selected camera resolution

Table 4.1. Relation between cameras resolution to image stitching service execution duration until device shut-done

Camera resolution, px	Data size, pixels	Battery discharge time, min
160×120	19,200	203
176×144	25,344	171
320×240	76,800	82
800×600	480,000	61
1280×800	1024,000	58
Adaptive	40,991*	124

* – estimated value.

From the results is seen, that average power and average CPU load of SoC terms have nearly direct relation and are directly dependent on image resolution. Since analyzed pixels count on available camera resolution changes non-linearly (Table 4.1 middle column), it is seen that average power consumption increases drastically when switching to 800×600 px and higher resolutions. Thus it can be concluded that the most optimal resolution in image quality to power consumption ratio is 320×240 px. The analysis of average CPU load showed that despite of the use of lightweight feature matching algorithm (AKAZE), the lowest load is still as high as 65%, this reading should be optimized in order to prolong the functioning time of the ECDs.

Analysis of relation between camera resolution to battery discharge duration was performed to reveal the tendency, the results are presented in Figure 4.11.

Fig. 4.11. Relation of total pixels in particular camera resolutions to battery discharge time, with fitted two-term exponential model

As an representative value for different resolutions and the computational load needed for it to process – number of analyzed pixels was chosen. The logarithmic axis is selected because number of pixels of available resolutions in the camera used are non-linear. To find the average estimate number of pixels of the proposed adaptive technique, firstly the two-term exponential model ($y = a \exp(bx) + c \exp(dx)$) was fitted on the results data, then the value of 40,991 pixels was found representing the measured 124 minutes of service time.

Battery discharge duration analysis results are provided in Table 4.1 and manifest that service duration is directly dependent on camera resolution. Adaptive resolution selection technique results in more than 2 h of continuous image stitching service, maintaining maximum quality of the image while utilizing the battery charge efficiently.

4.5. Conclusions of the Fourth Chapter

Research results presented in the chapter prove the main working hypothesis that the use of a fuzzy neural network in decentralized EC improves nodes performance in image stitching application and ground the following conclusions:

1. The original adaptive image resolution selection technique based on FNN is developed, increasing at least 1.5 times a total service time of distributed image stitching task on EC network, compared to fixed resolutions greater than 320×240 px.
2. The distribution of images features extraction task on edge nodes can be done, resulting in average 2.08 frames/s increase that is a small performance growth.
3. The decentralized EC infrastructure provides better performance than centralized in case of image stitching task, that causes node the high CPU load, > 95% on resolutions greater than 320×240 px.

General Conclusions

The aim to propose and investigate the original artificial neural network classifier based Internet of things systems for edge computing tasks has been reached. The following significant results for scientific field of Electrical and Electronic engineering are obtained:

1. Results of performed computer simulations grounds the following EC specifics:
 - in EC delay has the bigger than bandwidth influence on a data packet routing success;
 - the use of EC in comparison to cloud computing topology reduces mean network delay and service time;
 - EC hardware resource limit cause data processing bottlenecks that are highly manifested with the increase of the users number;
 - a fully connected network topology for sensors data to gateway level is preferable because it is less susceptible to network bottlenecks;
 - in order to increase the performance of the system in jittery conditions (in terms of bandwidth) the network has to be scaled down or the whole topology splitted into segments (i.e., subnets) or both.
2. The work distribution technique based on k -means clustering is proposed, outperforming regular technique in mean service time (in the range from 600 to 1200 simultaneous users).

3. A novel cell colonies image parameterization technique is developed and it is experimentally confirmed that its use with [ART1](#) or [ART2](#) classifiers is more precise in comparison to the use of [SVM](#) or heuristic classifiers:
 - The use of [ART1](#) classifier increases trustworthiness of the parameterization of images with cell colonies in terms of sensitivity and specificity.
 - The [ART1](#) classifier outperforms the [SVM](#) and heuristic classifiers in terms of F1 score, accuracy and precision.
 - The best according to [ROC](#) analysis and Youden's J index classifier is [ART2₁](#) detecting cell colonies in images with biological samples with up to 99.35% true positives and 76.89% true negatives.
 - The [ART2](#) classifier brings up similar results to heuristic classifier in terms of F1 score, accuracy and precision, yet the [DOR](#) value is increased more than 2 times.
 - The [ART2](#) classifier in terms of true positives outperforms other considered classifiers: [ART1](#) – by 4%, [SVM](#) – by 14% and heuristic – by 2%.
4. The original adaptive image resolution selection technique based on [FNN](#) is developed, increasing at least 1.5 times a total service time of distributed image stitching task on [EC](#) network, compared to fixed resolutions greater than 320×240 px.

In the future the work presented in this thesis can be continued in parallelizing the proposed methods and implementing them in Field Programmable Gate Arrays, with further performance experiments and comparison to the results described in this dissertation.

References

- Abdel Magid, S.; Petrini, F.; Dezfouli, B. 2019. Image classification on iot edge devices: profiling and modeling, *Cluster Computing* 1–14. ISSN 15737543. [see 89 p.]
- Abdelaziz, J.; Adda, M.; McHeick, H. 2016. Toward service aggregation for edge computing, *Procedia Computer Science* 94: 424–428. ISSN 18770509. [see 1 p.]
- Agrawal, V.; Gopalan, A. 2020. Energy Optimisation of Cascading Neural-network Classifiers, in *Proceedings of the 9th International Conference on Smart Cities and Green ICT Systems (SMARTGREENS)*, 149–158. [see 33 p.]
- Alberto de Albuquerque Silva, C.; Duarte Doria Neto, A.; Alberto Nicolau Oliveira, J.; Dantas Melo, J.; Simonetti Barbalho, D.; Medeiros Avelino, A. 2015. Definition of an architecture to configure artificial neural networks topologies using partial reconfiguraton in fpga, *IEEE Latin America Transactions* 13(7): 2094–2100. ISSN 1548-0992. [see 31 p.]
- Alrawais, A.; Alhothaily, A.; Hu, C.; Cheng, X. 2017. Fog computing for the internet of things: Security and privacy issues, *IEEE Internet Computing* 21(2): 34–42. ISSN 10897801. [see 10, 19 p.]
- Araújo, G.; Ralha, C. 2012. Using art1 neural networks for clustering computer forensics documents, *The International Journal of Forensic Computer Science* 7: 24–41. [see 24 p.]
- Aral, A.; Ovatman, T. 2016. Network-aware embedding of virtual machine clusters onto federated cloud infrastructure, *Journal of systems and software* 120: 89–104. [see 29 p.]
- Avudaiammal, R.; Elaveni, P.; Selvan, S.; Rajangam, V. 2020. Extraction of buildings in urban area for surface area assessment from satellite imagery based on morphological building index using svm classifier, *Journal of the Indian Society of Remote Sensing* 48(9): 1325–1344. [see 29 p.]

- Bacis, M.; Natale, G.; Sozzo, E. D.; Santambrogio, M. D. 2017. A pipelined and scalable dataflow implementation of convolutional neural networks on fpga, in *2017 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, 90–97. [see 31 p.]
- Bahwaireth, K.; Tawalbeh, L.; Benkhelifa, E.; Jararweh, Y.; Tawalbeh, M. A. 2016. Experimental comparison of simulation tools for efficient cloud and mobile cloud computing applications, *Eurasip Journal on Information Security* 2016(1): 1–14. ISSN 2510523X. [see 14 p.]
- Barcelo, M.; Correa, A.; Llorca, J.; Tulino, A. M.; Vicario, J. L.; Morell, A. 2016. Iot-cloud service optimization in next generation smart environments, *IEEE Journal on Selected Areas in Communications* 34(12): 4077–4090. ISSN 07338716. [see 10 p.]
- Barreto, R.; Lobo, J.; Menezes, P. 2019. Edge computing: A neural network implementation on an iot device, in *2019 5th Experiment International Conference*, 244–246. [see 32 p.]
- Ben-Zikri, Y. K.; Mendez, S.; Linte, C. A. 2017. Anatomical based registration of multi-sector x-ray images for panorama reconstruction, in *Medical Imaging 2017: Biomedical Applications in Molecular, Structural, and Functional Imaging*, 1–6. ISBN 9781510607194. ISSN 16057422. [see 29 p.]
- Bhat, J. A.; George, V.; Malik, B. 2015. Cloud computing with machine learning could help us in the early diagnosis of breast cancer, in *2015 Second International Conference on Advances in Computing and Communication Engineering*, 644–648. [see 29 p.]
- Bhatia, M. 2020. Fog computing-inspired smart home framework for predictive veterinary healthcare, *Microprocessors and Microsystems* 78: 1–16. [see 29 p.]
- Bilal, K.; Erbad, A. 2017. Edge computing for interactive media and video streaming, *2017 2nd International Conference on Fog and Mobile Edge Computing, FMEC 2017* 68–73. [see 49 p.]
- Biokaghazadeh, S.; Ren, F.; Zhao, M. 2018. Are FPGAs suitable for edge computing?, in *USENIX Workshop on Hot Topics in Edge Computing (HotEdge 18)*, 1–6. [see 32 p.]
- Borcoci, E. 2016. *Fog Computing, Mobile Edge Computing, Cloudlets - which one?* [see 16 p.]
- Bourbeau, P. P.; Ledebor, N. A. 2013. Automation in clinical microbiology, *Journal of Clinical Microbiology* 51(6): 1658–1665. [see 66 p.]
- Brahma, A.; Panigrahi, S. 2021. Role-Based Profiling Using Fuzzy Adaptive Resonance Theory for Securing Database Systems, *International Journal of Applied Metaheuristic Computing* 12(2): 36–48. [see 3 p.]
- Bruzgiene, R.; Narbutaite, L.; Adomkus, T. 2017. MANET Network in Internet of Things System, in *Ad Hoc Networks*, IntechOpen, 89–114. [see 57 p.]

- BSR, S. U. 2020. *An Overview of Smart Specialization Strategy in Lithuania*. Available online at: <<https://smartup-bsr.eu/wp-content/uploads/2020/12/O2.6-Lithuania-RIS3.pdf>>. [see 4 p.]
- Carpenter, G. A.; Grossberg, S. 1987. Art 2: self-organization of stable category recognition codes for analog input patterns, *Appl. Opt.* 26(23): 4919–4930. [see 22, 23 p.]
- Cassidy, A. S.; Georgiou, J.; Andreou, A. G. 2013. Design of silicon brains in the nanocmos era: Spiking neurons, learning synapses and neural architecture optimization, *Neural Networks* 45: 4–26. ISSN 08936080. [see 31 p.]
- Chalfoun, J.; Majurski, M.; Dima, A.; Halter, M.; Bhadriraju, K.; Brady, M. 2016. Lineage mapper: A versatile cell and particle tracker, *Scientific Reports* 6: 1–9. ISSN 2045-2322. [see 66 p.]
- Chen, C. H.; Lin, M. Y.; Liu, C. C. 2018a. Edge computing gateway of the industrial internet of things using multiple collaborative microcontrollers, *IEEE Network* 32(1): 24–32. ISSN 08908044. [see 14 p.]
- Chen, D.; Cong, J.; Gurumani, S.; Hwu, W.-m.; Rupnow, K.; Zhang, Z. 2016. Platform choices and design demands for iot platforms: cost, power, and performance tradeoffs, *IET Cyber-Physical Systems: Theory & Applications* 1(1): 70–77. [see 11 p.]
- Chen, M.; Challita, U.; Saad, W.; Yin, C.; Debbah, M. 2019. Artificial Neural Networks-Based Machine Learning for Wireless Networks: A Tutorial, *IEEE Communications Surveys Tutorials* 21(4): 3039–3071. [see 20 p.]
- Chen, W.-B.; Zhang, C. 2009. An automated bacterial colony counting and classification system, *Information Systems Frontiers* 11(4): 349. [see 66 p.]
- Chen, Y.; Zhang, N.; Zhang, Y.; Chen, X. 2018b. Dynamic computation offloading in edge computing for internet of things, *IEEE Internet of Things Journal* PP(c): 1. ISSN 23274662. [see 1, 11 p.]
- Choochootkaew, S.; Yamaguchi, H.; Higashino, T. 2019. A self-organized task distribution framework for module-based event stream processing, *IEEE Access* 7: 6493–6509. ISSN 2169-3536. [see 2, 9 p.]
- Choudhry, P. 2016. High-throughput method for automated colony and cell counting by digital image analysis based on edge detection, *Plos One* 11(2): 1–23. [see 3, 66 p.]
- Corcoran, P.; Datta, S. K. 2016. Mobile-edge computing and the internet of things for consumers: Extending cloud computing and services to the edge of the network., *IEEE Consumer Electronics Magazine* 5(4): 73–74. ISSN 21622256. [see 1, 16 p.]
- Corno, F.; De Russis, L.; Marcelli, A.; Montanaro, T. 2018. An unsupervised and noninvasive model for predicting network resource demands, *IEEE Internet of Things Journal* 5(6): 4342–4350. ISSN 23274662. [see 14 p.]

- Diallo, L.; Hassan, A.; Hashim, A.; Eyiomika, M. J.; Babiker, S.; Elagib, O.; Zarir, A. A. 2017. The rise of internet of things and big data on the cloud : Challenges and future trends, *International Journal of Future Generation Communication and Networking* 10(3): 49–56. [see 18 p.]
- Dursun, E.; Endelev, M.; Musumeci, A.; Failmezger, H.; Wang, S.-H.; Tresch, A.; Schroeder, T.; Krug, A. B. 2016. Continuous single cell imaging reveals sequential steps of plasmacytoid dendritic cell development from common dendritic cell progenitors, *Scientific Reports* 6: 1–12. ISSN 2045-2322. [see 66 p.]
- Ehmke, C.; Blum, F.; Gruhn, V. 2019. Properties of decentralized consensus technology - why not every blockchain is a blockchain, *ArXiv abs/1907.09289*: 1–38. [see 13 p.]
- Feng, W.; Kazi, T.; Hu, G. 2012. Web prefetching by art1 neural network, *Studies in Computational Intelligence* 413: 29–40. [see 29 p.]
- Ferguson, M.; ak, R.; Lee, Y.-T.; Law, K. 2017. Automatic localization of casting defects with convolutional neural networks, in *IEEE International Conference on Big Data*, 1726–1735. [see 26 p.]
- Georgakopoulos, D.; Jayaraman, P. P.; Fazio, M.; Villari, M.; Ranjan, R. 2016. Internet of things and edge cloud computing roadmap for manufacturing, *IEEE Cloud Computing* 3(4): 66–73. ISSN 23256095. [see 1, 9 p.]
- Gonçalves, M.; Netto, M.; Costa, J. A.; Zullo Junior, J. 2008. An unsupervised method of classifying remotely sensed images using kohonen self-organizing maps and agglomerative hierarchical clustering methods, *International Journal of Remote Sensing* 29: 3171–3207. [see 88 p.]
- Grainys, A.; Novickij, V.; Novickij, J. 2016. High-power bipolar multilevel pulsed electroporator, *Instrumentation Science & Technology* 44(1): 65–72. ISSN 1073-9149. [see 66 p.]
- Gravalos, I.; Makris, P.; Christodouloupoulos, K.; Varvarigos, E. A. 2018. Efficient network planning for internet of things with qos constraints, *IEEE Internet of Things Journal* 5(5): 3823–3836. [see 29 p.]
- Grossberg, S. 1987. Competitive learning: From interactive activation to adaptive resonance, *Cognitive Science* 11(1): 23–63. ISSN 0364-0213. [see 23 p.]
- Grossberg, S. 2003. From working memory to long-term memory and back: Linked but distinct, *Behavioral and Brain Sciences* 26(6): 709–777. [see 25 p.]
- Grossberg, S. 2021. Toward Autonomous Adaptive Intelligence: Building Upon Neural Models of How Brains Make Minds, *IEEE Transactions on Systems Man Cybernetics-Systems* 51(1): 51–75. [see 3 p.]
- Gupta, H.; Dastjerdi, A.; Ghosh, S.; Buyya, R. 2016. ifogsim: A toolkit for modeling and simulation of resource management techniques in internet of things, edge and fog computing environments, *Software: Practice and Experience* 47: 1275–1296. [see 14 p.]

- Hammadi-Mesmoudi, F.; Korczak, J. J. 1995. An unsupervised neural network classifier and its application in remote sensing, in *Proc. International Conference on Image Processing*, 4–6. [see 21 p.]
- Han, Y.; Ye, L.; Zhong, W.; Fang, L. 2019. A gigapixel image browsing scheme in edge computing environment, in *Communications in Computer and Information Science*, 324–333. ISBN 9789811381379. ISSN 18650937. [see 89 p.]
- Haykin, S. 1994. *Neural Networks: A Comprehensive Foundation*. 1st edition. USA: Prentice Hall PTR. ISBN 0023527617. [see 27 p.]
- Herberg, M.; Zerjatke, T.; de Back, W.; Glauche, I.; Roeder, I. 2015. Image-based quantification and mathematical modeling of spatial heterogeneity in ESC colonies, *Cytometry Part A* 87A(6, SI): 481–490. ISSN 1552-4922. [see 3, 66 p.]
- Hildebrand, M.; Khan, J.; Trika, S.; Lowe-Power, J.; Akella, V. 2020. AutoTM: Automatic Tensor Movement in Heterogeneous Memory Systems using Integer Linear Programming, in *25th International Conference on Architectural Support for Programming Languages and Operating Systems*, 875–890. [see 33 p.]
- Hinton, G. E.; Salakhutdinov, R. 2006. Reducing the dimensionality of data with neural networks, *Science* 313: 504–507. [see 19 p.]
- Hodo, E.; Bellekens, X.; Hamilton, A.; Dubouilh, P.-L.; Iorkyase, E.; Tachtatzis, C.; Atkinson, R. 2016. Threat analysis of iot networks using artificial neural network intrusion detection system, *2016 International Symposium on Networks, Computers and Communications (ISNCC)* 1–6. [see 18 p.]
- Hoerr, V.; Tuchscher, L.; Hueve, J.; Nippe, N.; Loser, K.; Glyvuk, N.; Tsytsyura, Y.; Holtkamp, M.; Sunderkoetter, C.; Karst, U.; Klingauf, J.; Peters, G.; Loeffler, B.; Faber, C. 2013. Bacteria tracking by in vivo magnetic resonance imaging, *BMC Biology* 11: 1–13. ISSN 1741-7007. [see 3, 66 p.]
- Hossain, M. D.; Sultana, T.; Nguyen, V.; Rahman, W.; Nguyen, T.; Huynh, L.; Huh, E.-n. 2020. Fuzzy based collaborative task offloading scheme in the densely deployed small-cell networks with multi-access edge computing, *Applied Sciences* 10: 1–23. [see 89 p.]
- Huang, H.; Shi, J.; Wang, F.; Zhang, D.; Zhang, D. 2020. Theoretical and Experimental Studies on the Signal Propagation in Soil for Wireless Underground Sensor Networks, *SENSORS* 20(9): 1–17. [see 59 p.]
- Huang, Y.-P.; Lai, S.-L.; Sandnes, F. E.; Liu, S.-I. 2012. Improving classifications of medical data based on fuzzy art 2 decision trees, *International Journal of Fuzzy Systems* 14(3): 444–453. [see 29 p.]
- Huang, Y.-T.; Cheng, F.-T.; Shih, Y.-H.; Chen, Y.-L. 2014. Advanced ART2 scheme for enhancing metrology-data-quality evaluation, *Journal of the Chinese Institute of Engineers* 37(8): 1064–1079. [see 23 p.]

- Intharawijitr, K.; Member, S.; Iida, K.; Member, S. 2017. Simulation study of low latency network architecture using mobile edge computing, *IEICE TRANSACTIONS on Information and Systems* E100.D(5): 963–972. [see 2, 10 p.]
- Ismail, B. I.; Mostajeran Goortani, E.; Ab Karim, M. B.; Ming Tat, W.; Setapa, S.; Luke, J. Y.; Hong Hoe, O. 2015. Evaluation of Docker as Edge computing platform, in *2015 IEEE Conference on Open Systems (ICOS)*, IEEE, 130–135. ISBN 978-1-4673-9434-5. [see 88 p.]
- ISO/IEC TR 23188. Information technology – Cloud computing – Edge computing landscape. 2020. [see 16, 17 p.]
- ISO/IEC TR 30164. Internet of Things (IoT) – Edge computing. 2020. [see 17 p.]
- Jazayeri, F.; Shahidinejad, A.; Ghobaei-Arani, M. 2020. Autonomous computation offloading and auto-scaling the in the mobile fog computing: a deep reinforcement learning-based approach, *Journal of Ambient Intelligence and Humanized Computing* 1–20. [see 29 p.]
- Jung, J. H.; Lee, J. E. 2016. Real-time bacterial microcolony counting using on-chip microscopy, *Scientific Reports* 6: 1–8. ISSN 2045-2322. [see 66 p.]
- Jutila, M. 2016. An adaptive edge router enabling internet of things, *IEEE Internet of Things Journal* 3(6): 1061–1069. ISSN 2327-4662. [see 2 p.]
- Karim, M. B. A.; Ismail, B. I.; Tat, W. M.; Goortani, E. M.; Setapa, S.; Luke, J. Y.; Ong, H. 2016. Extending cloud resources to the edge: Possible scenarios, challenges, and experiments, *Proceedings - International Conference on Cloud Computing Research and Innovation 2016, ICCCRI 2016* 78–85. [see 1 p.]
- Kato, R.; Matsumoto, M.; Sasaki, H.; Joto, R.; Okada, M.; Ikeda, Y.; Kanie, K.; Suga, M.; Kinehara, M.; Yanagihara, K.; Liu, Y.; Uchio-Yamada, K.; Fukuda, T.; Kii, H.; Uozumi, T.; Honda, H.; Kiyota, Y.; Furue, M. K. 2016. Parametric analysis of colony morphology of non-labelled live human pluripotent stem cells for cell quality control, *Scientific Reports* 6: 1–12. ISSN 2045-2322. [see 66 p.]
- Kieu, P. N.; Tran, H. S.; Le, T. H.; Le, T.; Nguyen, T. T. 2018. Applying Multi-CNNs model for detecting abnormal problem on chest x-ray images, in *Proceedings of 10th International Conference on Knowledge and Systems Engineering (KSE)*, 300–305. [see 33 p.]
- Kim, K. B. 2016. Extraction of canine cataract object for developing handy pre-diagnostic tool with fuzzy stretching and ART2 learning, *International Journal of Fuzzy Logic and Intelligent Systems* 16(1): 21–26. [see 29 p.]
- Kohonen, T. 1998. The self-organizing map, *Neurocomputing* 21(1-3): 1–6. ISSN 09252312. [see 22, 27 p.]
- Kubota, M.; Fukuta, S.; Yoshihide, N.; Kenichi, A. 2016. Dynamic resource controller technology to accelerate processing and utilization of iot data, *FUJITSU Scientific & Technical Journal* 52(4): 41–51. [see 2, 9 p.]

- Kurasova, O.; Marcinkevicius, V.; Medvedev, V.; Rapecka, A.; Stefanovic, P. 2014. Strategies for big data clustering, *2014 IEEE 26th International Conference on Tools with Artificial Intelligence* 740–747. ISSN 1082-3409. [see 40 p.]
- Kwee, E.; Herderick, E. E.; Adams, T.; Dunn, J.; Germanowski, R.; Krakosh, F.; Boehm, C.; Monnich, J.; Powell, K.; Muschler, G. 2017. Integrated colony imaging, analysis, and selection device for regenerative medicine, *SLAS Technology* 22(2): 217–223. ISSN 2472-6303. [see 66 p.]
- La Salandra, A.; Frajberg, D.; Fraternali, P. 2020. A virtual reality application for augmented panoramic mountain images, *Virtual Reality* 24(1): 123–141. [see 3 p.]
- Lambert, B. S.; Olson, R. J.; Sosik, H. M. 2017. A fluorescence-activated cell sorting subsystem for the imaging flowcytobot, *Limnology and Oceanography-Methods* 15(1): 94–102. ISSN 1541-5856. [see 3, 66 p.]
- Lee, K.; Park, J.; Yoo, H.-J. 2019. A Low-power, Mixed-mode Neural Network Classifier for Robust Scene Classification, *Journal of Semiconductor Technology and Science* 19(1): 129–136. [see 33 p.]
- Lee, S.-H.; Lee, S.-J. 2018. Development of remote automatic panorama vr imaging rig systems using smartphones, *Cluster Computing-The Journal of Networks Software Tools and Applications* 21(1, SI): 1175–1185. ISSN 1386-7857. [see 88 p.]
- Li, G.; Xu, S.; Wu, J.; Ding, H. 2018. Resource scheduling based on improved spectral clustering algorithm in edge computing, *Scientific Programming* 2018: 1–13. ISSN 1058-9244. [see 1 p.]
- Lucinskis, A.; Novickij, V.; Grainys, A.; Novickij, J.; Tolvaisiene, S. 2014. Modelling the cell transmembrane potential dependence on the structure of the pulsed magnetic field coils, *Elektronika ir Elektrotechnika* 20: 9–12. [see 66 p.]
- Lyu, L.; Bezdek, J. C.; Jin, J.; Yang, Y. 2020. Foreseen: Towards differentially private deep inference for intelligent internet of things, *IEEE Journal on Selected Areas in Communications* 38(10): 2418–2429. [see 29 p.]
- Malcangi, M.; Nano, G. 2021. Biofeedback: e-health prediction based on evolving fuzzy neural network and wearable technologies, *Evolving Systems* 1–9. [see 3 p.]
- Manihar, S.; Bano, T.; Patel, R.; Agrawal, S. 2020. Intelligent and scalable iot edge-cloud system, *International Journal of Advanced Computer Science and Applications* 11: 359–364. [see 29 p.]
- Marmanis, D.; Schindler, K.; Wegner, J. D.; Galliani, S.; Datcu, M.; Stilla, U. 2018. Classification with an edge: Improving semantic image segmentation with boundary detection, *ISPRS Journal of Photogrammetry and Remote Sensing* 135: 158–172. ISSN 09242716. [see 71 p.]
- McClellan, M.; Cervelló-Pastor, C.; Sallent, S. 2020. Deep learning at the mobile edge: Opportunities for 5g networks, *Applied Sciences* 10(14): 4735. ISSN 2076–3417. [see 34 p.]

- McInerney, T.; Terzopoulos, D. 2000. T-snakes: Topology adaptive snakes, *Medical Image Analysis* 4(2): 73–91. [see 67 p.]
- Merenda, M.; Porcaro, C.; Iero, D. 2020. Edge machine learning for ai-enabled iot devices: A review, *Sensors* 20: 2533. [see 29 p.]
- Merzviniskyte, R.; Treigyte, G.; Savickiene, J.; Magnusson, K.-E.; Navakauskiene, R. 2006. Effects of histone deacetylase inhibitors, sodium phenyl butyrate and vitamin B3, in combination with retinoic acid on granulocytic differentiation of human promyelocytic leukemia HL-60 cells., *Annals of the New York Academy of Sciences* 1091: 356–367. ISSN 0077-8923. [see 66 p.]
- Miyashita, T.; Higuchi, Y.; Kojima, M.; Ochiai, A.; Ishii, G. 2017. Single cell time-lapse analysis reveals that podoplanin enhances cell survival and colony formation capacity of squamous cell carcinoma cells, *Scientific Reports* 7: 1–12. ISSN 2045-2322. [see 66 p.]
- Moghaddam, M. Z.; Umili, G.; Messina, V.; Bonetto, S.; Ferrero, A. M.; Bollini, G.; Gandreau, D. 2020. An svm-based scheme for automatic identification of architectural line features and cracks, *Applied Sciences (Switzerland)* 1–20. ISSN 20763417. [see 28, 29 p.]
- Murshed, M. G. S.; Murphy, C.; Hou, D.; Khan, N.; Ananthanarayanan, G.; Hussain, F. 2019. *Machine Learning at the Network Edge: A Survey*. [see 29 p.]
- Nakajima, Y.; Saito, H. 2017. Robust camera pose estimation by viewpoint classification using deep learning, *Computational Visual Media* 3(2): 189–198. ISSN 2096-0433. [see 88 p.]
- Navakauskas, D.; Serackis, A.; Matuzevičius, D.; Laptik, R. 2014. *Specializuotos elektroninės intelektualiosios sistemos garsams ir vaizdams apdoroti. Teorija ir taikymai*. Vilniaus Gedimino technikos universiteto leidykla Technika. 580 p. ISBN 9786094577673. [see 3, 30, 66 p.]
- Nelson Jr, R. 2011. Vigilance, expectancy, and noise: Attention in second language lexical learning and memory, *Second Language Research* 27(2): 153–171. [see 72 p.]
- N.G., B. A.; S., S. 2020. Anomaly detection framework for internet of things traffic using vector convolutional deep learning approach in fog environment, *Future Generation Computer Systems* 113: 255–265. ISSN 0167-739X. [see 29 p.]
- Ogbuachi, M.; Reale, A.; Suskovics, P.; Kovacs, B. 2020. Context-aware kubernetes scheduler for edge-native applications on 5g, *Journal of Communications Software and Systems* 16: 85. [see 93 p.]
- Omar, T.; Nehdi, M. L.; Zayed, T. 2018. Infrared thermography model for automated detection of delamination in rc bridge decks, *Construction and Building Materials* 168: 313–327. ISSN 0950-0618. [see 88 p.]
- Park, J.; Song, D. H.; Han, S.-S.; Joo Lee, S.; Baek Kim, K. 2017. Automatic extraction of soft tissue tumor from ultrasonography using ART2 based intelligent image analysis, *Current Medical Imaging Reviews* 13(4): 447–453. [see 29 p.]

- Pasca, T. V.; Dama, S.; Sathya, V.; Kuchi, K. 2017. A feasible cellular internet of things, *IEEE Consumer Electronics Magazine* 66–72. [see 18 p.]
- Pedraza, A.; Gallego, J.; Lopez, S.; Gonzalez, L.; Laurinavicius, A.; Bueno, G. 2017. Glomerulus classification with convolutional neural networks, in *Communications in Computer and Information Science*, 839–849. [see 25 p.]
- Petkov, T.; Sotirov, S.; Surchev, S. 2016. Generalized net model of person recognition using ART2 neural network and viola-jones algorithm, in *Novel Developments in Uncertainty Representation and Processing*, Springer, 257–265. [see 23 p.]
- Rangarajan, S.; Phoha, V.; Balagani, K.; Selmic, R.; Iyengar, S. 2004. Adaptive neural network clustering of web users, *Computer* 37: 34–40. [see 29 p.]
- Rathnayake, R. M. N. B.; Seneviratne, L. 2018. An efficient approach towards image stitching in aerial images, in *2018 3rd International Conference on Information Technology Research (ICITR)*, 1–6. ISBN 978-1-7281-1470-5. 3rd International Conference on Information Technology Research (ICITR), Moratuwa, SRI LANKA, DEC 05-07, 2018. [see 88 p.]
- Rausch, T.; Rashed, A.; Dustdar, S. 2021. Optimized container scheduling for data-intensive serverless edge computing, *Future Generation Computer Systems-The International Journal of Esience* 114: 259–271. [see 5, 93 p.]
- Rawat, J.; Singh, A.; Bhadauria, H. S.; Virmani, J.; Devgun, J. S. 2017. Classification of acute lymphoblastic leukaemia using hybrid hierarchical classifiers, *Multimedia Tools and Applications* 1. ISSN 1573-7721. [see 66 p.]
- Sabella, D.; Vaillant, A.; Kuure, P.; Rauschenbach, U.; Giust, F. 2016. Mobile-edge computing architecture: The role of mec in the internet of things., *IEEE Consumer Electronics Magazine* 5(4): 84–91. ISSN 21622256. [see 2, 9 p.]
- Salman, O.; Elhadj, I.; Kayssi, A.; Chehab, A. 2015. Edge computing enabling the internet of things, *IEEE World Forum on Internet of Things, WF-IoT 2015 - Proceedings* 603–608. [see 2, 19 p.]
- Samie, F. 2018. *Resource Management for Edge Computing in Internet of Things (IoT)*: Doctoral Dissertation. Karlsruhe Institute of Technology (KIT). [see 19 p.]
- Sapijaszko, G.; Mikhael, W. B. 2018. An overview of recent convolutional neural network algorithms for image recognition, in *2018 IEEE 61st International Midwest Symposium on Circuits and Systems (MWSCAS)*, 743–746. [see 34 p.]
- Satoh, I. 2016. Edge data processing, in *2016 30th International Conference on Advanced Information Networking and Applications Workshops (WAINA)*, IEEE, 410–415. ISBN 978-1-5090-2461-2. [see 19 p.]
- Satoskar, A.; Nambiar, V.; Mohan, A.; Rainy, C.; Gadhikar, L. 2017. Parallelization of algorithms for image stitching using distributed memory architecture, in *2017 International Conference on Nascent Technologies in Engineering (ICNTE)*, 1–4. [see 35 p.]

- Satya, M. S. 2016. Edge computing: Vision and challenges, *IEEE Internet of Things Journal* 3(5): 637–646. ISSN 23274662. [see 16 p.]
- Savickiene, J.; Treigyte, G.; Jonusiene, V.; Bruzaite, R.; Borutinskaite, V.-V.; Navakauskiene, R. 2012. Epigenetic changes by zebularine leading to enhanced differentiation of human promyelocytic leukemia NB4 and KG1 cells., *Molecular and Cellular Biochemistry* 359: 245–261. ISSN 1573-4919. [see 66 p.]
- Savickiene, J.; Treigyte, G.; Magnusson, K.-E.; Navakauskiene, R. 2009. Response of retinoic acid-resistant KG1 cells to combination of retinoic acid with diverse histone deacetylase inhibitors., *Annals of the New York Academy of Sciences* 1171: 321–333. ISSN 1749-6632. [see 66 p.]
- Scoca, V.; Aral, A.; Brandic, I.; De Nicola, R.; Uriarte, R. B. 2018. Scheduling latency-sensitive applications in edge computing, in *Proceedings of the 8th International Conference on Cloud Computing and Services Science*, SCITEPRESS - Science and Technology Publications, 158–168. ISBN 978-989-758-295-0. [see 51 p.]
- Seo, J.-H.; Yang, S.; Kang, M.-S.; Her, N.-G.; Nam, D.-H.; Choi, J.-H.; Kim, M. H. 2019. Automated stitching of microscope images of fluorescence in cells with minimal overlap, *Micron* 126: 1–8. [see 3, 34, 88 p.]
- Seufert, M.; Kwam, B. K.; Wamser, F.; Tran-Gia, P. 2017. Edgenetworkcloudsim: Placement of service chains in edge clouds using networkcloudsim, in *2017 IEEE Conference on Network Softwarization (NetSoft)*, 1–6. [see 41 p.]
- Sha, C.; Hou, J.; Cui, H. 2016. A robust 2D Otsu's thresholding method in image segmentation, *Journal of Visual Communication and Image Representation* 41: 339–351. ISSN 1047-3203. [see 70 p.]
- Sharma, P.; Nair, R. (eds.) 2020. *FPGA Algorithms and Applications for the Internet of Things: Advances in Systems Analysis, Software Engineering, and High Performance Computing*. IGI Global. ISBN 9781522598060. [see 33 p.]
- Sharma, S.; Jain, K. 2020. Image stitching using akaze features, *Journal of the Indian Society of Remote Sensing* 48: 1–13. [see 91 p.]
- Shete, P. P.; Sarode, D. M.; Bose, S. K. 2016. Real-time panorama composition for video surveillance using gpu, in *2016 International Conference on Advances in Computing, Communications and Informatics, ICACCI 2016*, 137–143. ISBN 9781509020287. [see 29 p.]
- Sonmez, C.; Oztgovde, A.; Ersoy, C. 2019. Fuzzy workload orchestration for edge computing, *IEEE Transactions on Network and Service Management* 16(2): 769–782. [see 89 p.]
- Souza, V. B.; Gomez, A.; Masip-Bruin, X.; Marin-Tordera, E.; Garcia, J. 2017. Towards a fog-to-cloud control topology for qos-aware end-to-end communication, in *2017 IEEE/ACM 25th international symposium on Quality of Service (IWQOS)*, 0. [see 29 p.]

- Stephanakis, I. M.; Chochliouros, I. P.; Sfakianakis, E.; Shirazi, S. N.; Hutchison, D. 2019. Hybrid self-organizing feature map (som) for anomaly detection in cloud infrastructures using granular clustering based upon value-difference metrics, *Information Sciences* 494: 247–277. [see 29 p.]
- Suga, M.; Kii, H.; Niikura, K.; Kiyota, Y.; Furue, M. K. 2015. Development of a monitoring method for nonlabeled human pluripotent stem cell growth by time-lapse image analysis, *Stem Cells Translational Medicine* 4(7): 720–730. ISSN 2157-6564. [see 66 p.]
- Sumathi, M.; Vijayalakshmi, P. 2015. Efficient implementation of dynamic hierarchical structure for ART2 network, in *International Conference on Electrical, Computer and Communication Technologies (ICECCT), 2015 IEEE*, IEEE, 1–6. [see 70 p.]
- Thilagavathy, R.; Rajagopal, S. 2019. Using cloud effectively in concept based text mining using grey wolf self organizing feature map, *Cluster Computing* 22: 1–11. [see 29 p.]
- Tran, T. X.; Hajisami, A.; Pandey, P.; Pompili, D. 2017. Collaborative mobile edge computing in 5g networks: New paradigms, scenarios, and challenges, *IEEE Communications Magazine* 55(April): 54–61. ISSN 0163-6804. [see 2, 9 p.]
- Tudoran, R.; Costan, A.; Wang, R.; Bouge, L.; Antoniu, G. 2014. Bridging data in the clouds: An environment-aware system for geographically distributed data transfers, *Proceedings - 14th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing, CCGrid 2014* 92–101. [see 14 p.]
- Vahid Dastjerdi, A.; Buyya, R. 2016. Fog computing: Helping the internet of things realize, *IEEE Computer Society* 112–116. [see 10 p.]
- Valluru, D.; Jeya, I. 2020. Iot with cloud based lung cancer diagnosis model using optimal support vector machine, *Health Care Management Science* 23: 1–11. [see 29 p.]
- Vesanto, J.; Alhoniemi, E. 2000. Publication 7 clustering of the self-organizing map clustering of the self-organizing map, *IEEE Transactions on Neural Networks* 11(3): 586–600. [see 29 p.]
- Vranjkovic, V.; Struharik, R. 2020. Hardware acceleration of sparse support vector machines for edge computing, *Elektronika ir Elektrotechnika* 26(3): 42–53. [see 28 p.]
- Véstias, M. P. 2019. A survey of convolutional neural networks on edge with reconfigurable computing, *Algorithms* 12(8): 154. ISSN 1999–4893. [see 29 p.]
- Wadood, A.; ar Ali, Z.; Ghouzali, S.; ALfawaz, B.; Muhammad, G.; Hossain, M. S. 2017. Biometric security through visual encryption for fog edge computing, *IEEE Access* 5: 5531–5538. ISSN 2169-3536. [see 29 p.]
- Wan, X.-J.; Liu, L.; Xu, Z.; Xu, Z. 2018. Gearbox fault diagnosis based on selective integrated soft competitive learning fuzzy adaptive resonance theory, *Journal of Computing and Information Science in Engineering* 19(1): 011 008. ISSN 1530-9827. [see 29 p.]

- Wang, G.; Zhai, Z.; Xu, B.; Cheng, Y. 2017. A parallel method for aerial image stitching using orb feature points, in *Proceedings - 16th IEEE/ACIS International Conference on Computer and Information Science, ICIS 2017*, 769–773. ISBN 9781509055074. [see 34, 89 p.]
- Wang, J.; Pan, J.; Esposito, F.; Calyam, P.; Yang, Z.; Mohapatra, P. 2019. Edge cloud offloading algorithms: Issues, methods, and perspectives, *ACM Comput. Surv.* 52(1): 2:1–2:23. ISSN 0360-0300. [see 12 p.]
- Wang, T.; Chen, Y.; Lv, H.; Teng, J.; Snoussi, H.; Tao, F. 2021. Online detection of action start via soft computing for smart city, *IEEE Transactions on Industrial Informatics* 17(1): 524–533. [see 29 p.]
- Wang, X.; Han, Y.; Leung, V. C. M.; Niyato, D.; Yan, X.; Chen, X. 2020. Convergence of edge computing and deep learning: A comprehensive survey, *IEEE Communications Surveys Tutorials* 22(2): 869–904. [see 29 p.]
- Wang, Y.; Li, H.; Li, X. 2016. Re-architecting the on-chip memory sub-system of machine-learning accelerator for embedded devices, in *Proceedings of the 35th International Conference on Computer-Aided Design - ICCAD '16*, New York, New York, USA: ACM Press, 1–6. ISBN 9781450344661. [see 2, 9 p.]
- Weisong, S.; Dustdar, S. 2016. The promise of edge computing, *IEEE Computer Society* 59(5): 78–81. [see 9 p.]
- Wu, J. 2012. *Advances in K-Means Clustering: A Data Mining Thinking*. Springer Publishing Company, Incorporated. ISBN 3642298060. [see 52 p.]
- Xiang, T.-Z.; Xia, G.-S.; Zhang, L. 2016. Image stitching with perspective-preserving warping, *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences* III-3: 1–8. [see 91 p.]
- Xu, X.; Xue, Y.; Qi, L.; Yuan, Y.; Zhang, X.; Umer, T.; Wan, S. 2019. An edge computing-enabled computation offloading method with privacy preservation for internet of connected vehicles, *Future Generation Computer Systems* 96: 89–100. ISSN 0167739X. [see 11, 12 p.]
- Xu, Z.; Zhao, L.; Liang, W.; Rana, O. F.; Zhou, P.; Xia, Q.; Xu, W.; Wu, G. 2021. Energy-aware inference offloading for dnn-driven applications in mobile edge clouds, *IEEE Transactions on Parallel and Distributed Systems* 32(4): 799–814. [see 29 p.]
- Xue, Z.; Wei, J.; Guo, W. 2020. A Real-Time Naive Bayes Classifier Accelerator on FPGA, *IEEE Access* 8: 40 755–40 766. [see 34 p.]
- Yao, Q. 2021. Adaptive fuzzy neural network control for a space manipulator in the presence of output constraints and input nonlinearities, *Advances in Space Research* 67(6): 1830–1843. [see 3 p.]
- Yu, W.; Liang, F.; He, X.; Hatcher, W. G.; Lu, C.; Lin, J.; Yang, X. 2018. A survey on the edge computing for the internet of things, *IEEE Access* 6: 6900–6919. [see 29 p.]

- Zalieckaitė, L.; Žilinskas, R. 2016. Daiktų interneto technologijos taikymo versle nauda ir rizika, *Informacijos mokslai* 72(72): 102. ISSN 1392-0561. [see 9, 19 p.]
- Zhang, M.; Sun, D.; Liu, G.; Wu, H.; Zhou, H.; Shi, M. 2016. Real-time in vivo imaging reveals the ability of neutrophils to remove *Cryptococcus neoformans* directly from the brain vasculature, *Journal of Leukocyte Biology* 99(3): 467–473. ISSN 0741-5400. [see 66 p.]
- Zhang, P.; Liu, F.; Vashisht, S.; Singh Mann, R. 2020a. A robust network measurement and feature selection strategy for software-defined edge computing environment, *Transactions on Emerging Telecommunications Technologies* 1–16. ISSN 21613915. [see 1, 29 p.]
- Zhang, P.; Shu, S.; Zhou, M. 2019. Adaptive and dynamic adjustment of fault detection cycles in cloud computing, *IEEE Transactions on Industrial Informatics* PP: 1–10. [see 2, 29 p.]
- Zhang, W.; Sun, H.-M. 2017. Instagram spam detection, in *2017 IEEE 22nd Pacific Rim International Symposium on Dependable Computing (PRDC)*, 227–228. [see 18 p.]
- Zhang, Y.; Zhou, Y.; Lu, H.; Fujita, H. 2020b. Traffic network flow prediction using parallel training for deep convolutional neural networks on spark cloud, *IEEE Transactions on Industrial Informatics* 16(12): 7369–7380. [see 29 p.]
- Zhao, W.; Fu, H.; Luk, W.; Yu, T.; Wang, S.; Feng, B.; Ma, Y.; Yang, G. 2016. F-CNN: An FPGA-based framework for training Convolutional Neural Networks, in *27th International Conference on Application-specific Systems, Architectures and Processors*, IEEE, 107–114. [see 26 p.]

List of Scientific Publications by the Author on the Topic of the Dissertation

Papers in the Reviewed Scientific Journals

Skirelis, J.; Navakauskas, D. 2020. Performance Analysis of Edge Computing in IoT, *Elektronika ir Elektrotechnika* 26(1): 72–77. ISSN 1392-1215. DOI: [10.5755/j01.eie.26.1.23235](https://doi.org/10.5755/j01.eie.26.1.23235).

Skirelis, J.; Patlins, A.; Kunicina, N.; Romanovs, A.; Zabasta, A. 2019. Wireless Sensor Networks: Towards Resilience Against Weather-Based Disruptions, *Electrical, Control and Communication Engineering* 15(2): 79–87. ISSN 2255-9159. DOI: [10.2478/eccc-2019-0011](https://doi.org/10.2478/eccc-2019-0011).

Skirelis, J. 2018. Klasifikatoriaus vieta daiktų interneto kraštų kompiuterijoje, *Mokslas – Lietuvos ateitis* 10: 1–6. ISSN 2029-2341. DOI: [10.3846/mla.2018.2760](https://doi.org/10.3846/mla.2018.2760).

Papers in Other Editions

Skirelis, J.; Navakauskas, D. 2019. Classification at the Edge: Implementation and Performance Evaluation, in *Proceedings of IEEE 7th Workshop on Advances in Information, Electronic and Electrical Engineering*, 1–4. DOI: [10.1109/AIEEE48629.2019.8977122](https://doi.org/10.1109/AIEEE48629.2019.8977122).

Skirelis, J. 2019. Blockchain Hands-on Experience in MATLAB, in *Proceedings of Workshop on Industry 4.0: Methodologies, Tools and Applications in Electronic and Information Sciences (eStream)*, 1–4. DOI: [10.1109/eStream.2019.8732150](https://doi.org/10.1109/eStream.2019.8732150).

Skirelis, J.; Navakauskas, D. 2018. Classifier based Gateway for Edge Computing, in *Proceedings of IEEE 6th Workshop on Advances in Information, Electronic and Electrical Engineering*, 1–4. DOI: [10.1109/AIEEE.2018.8592162](https://doi.org/10.1109/AIEEE.2018.8592162).

Skirelis, J.; Navakauskas, D. 2018. Classification of Cell Colonies Images by ART2 Classifier, in *Proceedings of Open Conference of Electrical, Electronic and Information Sciences (eStream)*, 1–4. DOI: [10.1109/eStream.2018.8394123](https://doi.org/10.1109/eStream.2018.8394123).

Skirelis, J.; Navakauskas, D. 2017. Edge Computing in IoT: Preliminary Results on Modeling and Performance Analysis, in *Proceedings of IEEE 5th Workshop on Advances in Information, Electronic and Electrical Engineering*, 1–5. DOI: [10.1109/AIEEE.2017.8270555](https://doi.org/10.1109/AIEEE.2017.8270555).

Skirelis, J.; Navakauskas, D. 2017. On Parameterization of Cell Colonies Images: ART Classifier Approach, in *Proceedings of Open Conference of Electrical, Electronic and Information Sciences (eStream)*, 1–5. DOI: [10.1109/eStream.2017.7950323](https://doi.org/10.1109/eStream.2017.7950323).

Summary in Lithuanian

Įvadas

Problemos formulavimas

Pastarųjų dviejų dešimtmečių pažanga debesų kompiuterijos technologijoje lėmė sėkmingą jos taikymą brangioje įrangoje sudėtingiems uždaviniams spręsti. Sparčiai besivystančios mobilaus interneto ir daiktų interneto (angl. *Internet of things, IoT*) technologijos generuoja milžiniškus duomenų kiekius, kurie yra apdorojami debesų kompiuterijos duomenų centruose (Abdelaziz *et al.* 2016). Tai neišvengiamai iškelia didelės kompiuterių tinklo delsos ir srauto užkraudimo problemas, bei iššūkius šioms problemoms spręsti (Chen *et al.* 2018b). Šios problemos kyla dėl natūralios duomenų perdavimo delsos, kuri priklausomai nuo taikymo srities yra nepriimtina ir gali lemti duomenų srauto tinkle įsisotinimą.

Staigus našių, efektyvių ir prieinamų mobilių įrenginių skaičiaus augimas (2025 m. prognozuojama daugiau nei 70 mlrd. įrenginių veikiančių belaidžiam tinkle) sukuria prielaidas mažinti išaugusias tinklų apkrovas, perkelti centralizuotos debesų kompiuterijos užduotis į kraštų kompiuterijos (angl. *edge computing*) įtaisus (Corcoran, Datta 2016; Karim *et al.* 2016). Perėjimas prie kraštų kompiuterijos iškelia papildomų klausimų susijusių su tinkamu topologinės architektūros parinkimu, efektyviu pirminių duomenų apdorojimu, racionalių užduočių decentralizuotuose įrenginiuose planavimu ir saugumo aspektais (Georgakopoulos *et al.* 2016; Li *et al.* 2018). Šiuo metu nėra žinoma kas kraštų kompiuterijoje turi didžiausią įtaką tokio tipo tinklo našumui, tačiau esami moksliniai rezultatai suponuoja kraštų išmanumą įgyvendinti per mašininį mokymąsį (Zhang *et al.* 2020a).

Klasifikavimas yra viena pagrindinių užduočių įgyvendinant kraštų kompiuterijos išmanumą, tačiau ribotos kraštų įrenginių skaičiavimo galimybės riboja sudėtingų klasifi-

kavimo algoritmų įgyvendinimo galimybę (Intharawijitr *et al.* 2017), o paprastesni algoritmai nepatenkina lūkesčių. Aiškėja, kad dirbtinių neuronų tinklų (angl. *artificial neural networks*) klasifikatoriai yra labiausiai tinkami taikyti ribotų išteklių kraštų kompiuterijos įrenginiuose, tačiau jų įgyvendinimo metodai, greitaveika ir energetinio efektyvumo rodikliai nėra žinomi, taip pat nėra aišku, ar įmanoma įgyvendinti saviorganizuojantį klasifikatorių kraštų kompiuterijos uždaviniams spręsti.

Pirminių jutiklių duomenų persiuntimas į nutolusius debesų kompiuterijos duomenų centrus ne tik ilgina bendrą paslaugos aptarnavimo laiką, tačiau ir kelia grėsmę duomenų saugumui, šiai problemai spręsti šiuo metu naudojami sudėtingi šifravimo metodai, kurie didina įrenginių apkrovą, taip pat daro neigiamą įtaką energijos sąnaudoms. Kraštų kompiuterijoje ši problema išsispindžia natūraliai – duomenys persiunčiami lokaliems įrenginiams, tai stipriai sumažina riziką duomenų nutekimui ir reikalauja tik nesudėtingų šifravimo sprendimų.

Dėl išvardintų priežasčių, šiame darbe spręsta *problema* – dirbtinių neuronų tinklais grįstų klasifikatorių naudojimo efektyvioje kraštų kompiuterijoje taikomųjų žinių ir metodų trūkumas.

Darbo aktualumas

Bendrieji visuomenės poreikiai rodo didelę daiktų internetu grįstų paslaugų paklausą (Jutla 2016) įvairiose srityse, pvz. išmaniųjų automobilių, namų ir miestų, biomedicinoje ar pramonėje. Šiuo metu, dauguma šių paslaugų yra grįsti debesų kompiuterija. Tyrimai rodo, kad sparčiai augantis paslaugų skaičius ir jų sudėtingumas lemia išaugusį paslaugos aptarnavimo laiką, o tai lemia prastą paslaugos vartotojo patirtį. Pažymėtina, kad duomenų centrų skaičiavimo pajėgumų didinimas nebeteikia pozityvaus efekto – paslaugos laikas gerėja nežymiai. Šie faktai leidžia daryti prielaidą, kad problema yra kompiuterių tinkle – net jei duomenų centras sugeba apdoroti duomenis akimirksniu, apdorotų duomenų perdavimo laikas jų gavėjui yra ženkliai didesnis (Sabella *et al.* 2016). Todėl tam tikros taikymo sritys nebegali užtikrinti realaus laiko kriterijaus savo paslaugoms. Šiai problemai spręsti buvo sukurtas turinio pristatymo tinklų (angl. *CDN*), rūko ir kraštų kompiuterijos technologijos (Salman *et al.* 2015). Pastaraisiais metais stipriai sumažėjus kraštų kompiuterijos įrenginių kainai, masinis šių įrenginių ir technologijos taikymas tapo galimas.

Atsižvelgus į atliktą adaptyvios kraštų kompiuterijos taikymų literatūros apžvalgą, išaiškinta, kad šioje srityje nepakanka mokslinių tyrimų ir didžioji dauguma įgyvendinamų sistemų yra fiksuotos architektūros, todėl šios sistemos tinka tik specializuotam uždaviniui spręsti. Dinaminės krašto kompiuterijos sistemos pasižymi ne tik prisitaikymo prie tinkle vyraujančių duomenų savybe, bet leidžia spartinti duomenų apdorojimą skirstant lygiagrečius uždavinius tarp skaičiavimo įrenginių atsižvelgiant į tinklo būseną (Kubota *et al.* 2016; Zhang *et al.* 2019).

Moksliniai tyrimai parodė, kad decentralizavimo įgyvendinimas kraštų kompiuterijoje yra sudėtingas žingsnis, nes reikalauja papildomai spręsti mazgų sinchronizavimo, duomenų protokolų suderinamumo, nevienalyčių įrenginių suderinamumo problemas.

Dirbtinių neuronų tinklais grįstų klasifikatorių taikymo daiktų, rūko ir debesų kompiuterijoje apžvalga atskleidė, kad adaptyviojo rezonanso teorijos (angl. *adaptive resonance theory, ART*) šeimos klasifikatoriai yra vidutinio sudėtingumo ir pasižymi patraukliais privalumais, tačiau nebuvo rasta literatūros aprašančios jų taikymą kraštų kompiuterijos

kontekste. Pats *ART* metodas yra žinomas daugelį metų, tačiau pastaraisiais metais pastebimas jų taikymo skaičiaus augimas saugių duomenų bazių (Brahma, Panigrahi 2021) ir savarankiškų išmaniųjų sistemų (Grossberg 2021) srityse. Taip pat neraiškiųjų neuronų tinklų (angl. *fuzzy neural network*, *FNN*) sąlyginis skaičiavimo lengvumas ir išskirtinė galimybė įrašyti neraiškiasias taisykles ir narystės funkcijas yra patrauklios taikymui kraštų kompiuterijos įrengiuose. Šių tinklų aktualumas yra pagrindžiamas sėkmingu jų taikymu išmaniose sveikatos stebėsenos (Malcangi, Nano 2021) ir išmaniose įterptinėse kosminių manipuliatorių (Yao 2021) sistemose.

Lašelių kolonijų skaičiavimo užduotis yra svarbi citometrijos mokslų dalis (Herberg *et al.* 2015; Lambert *et al.* 2017), reikalaujanti greitos ir tikslios biologinių bandinių analizės. Išmaniųjų algoritmų skirtų biologinių bandinių (Navakauskas *et al.* 2014) vaizdų (Horrer *et al.* 2013) parametrizavimui taikymas sutrumpina tyrimų trukmę ir sumažina bendrą tyrimų kainą. Įvertinant tai, kad lašelių kolonijų skaičiavimas yra sudėtingas ir ilgas procesas, o bandinių būseną yra stabili tik ribotame laiko intervale (Choudhry 2016), tikslas sukurti automatizuotą realaus laiko lašelių kolonijų skaičiavimo sistemą yra iškeltas jau daugelį metų, tačiau aiškus sprendimas iki šiol nežinomas. Todėl, disertacijoje yra tiriami ir siūlomi saviorganizuojantys klasifikatoriai kaip galimi kandidatai minėtam tikslui pasiekti, taikant juos kraštų kompiuterijos technologijoje, atliekant palyginimą su jau pasiūlytų klasifikatorių rezultatais.

Vaizdų sandara (angl. *image stitching*) skaičiavimo prasme yra sudėtingas procesas, kurio metu kiekvienam sudaromam vaizdui būtina atlikti visų pavienių taškų analizę, tai reikalauja daug įrenginių skaičiavimo išteklių (Seo *et al.* 2019). Kita vertus, tai yra pagrindinis metodas kuriant panorامينius vaizdus daugumoje mobiliųjų įrenginių. Šiuolaikinė tendencija didinti vaizdo kamerų skaičių mobiliuose įrenginiuose išryškina aktualumą efektyvių vaizdo sandaros algoritmų taikymui tokiose srityse kaip papildyta virtuali realybė (La Salandra *et al.* 2020).

Siekiant išspręsti praeitame poskyryje suformuluotą problemą, darbe išsiskeltos ir patvirtintos šios pagrindinės *hipotezės*:

- klasterizavimo algoritmas, veikiantis kraštų kompiuterijos koncentratoriuje, lyginant su įprastiniais metodais, sumažina bendrą paslaugos aptarnavimo laiką vidutinės apkrovos sąlygomis,
- adaptyviojo rezonanso teorija grįsto klasifikatoriaus struktūrinės dalies prisitaikymo savybė lemia geresnę klasifikavimo kokybę, lyginant su fiksuotos struktūros atraminų vektorių klasifikatoriumi sprendžiant lašelių kolonijų skaičiavimų uždavinį,
- neraiškiųjų neuronų tinklo taikymas decentralizuotoje kraštų kompiuterijoje gali pagerinti mazgo efektyvumą sprendžiant vaizdų sandaros uždavinį.

Problemos sprendimas yra susijęs su Lietuvos išmaniaja specializacija (BSR 2020) dviem aspektais. Visų pirma, remiantis prioritetinės transporto, logistikos ir informacinių ir ryšių technologijų (IRT) krypties technologija – daiktų internetu, realių modelių uždaviniai yra įgyvendinti kraštų kompiuterijos įrangoje atkartojant realias paslaugas. Antra, problemos sprendimas yra susijęs su sveikatos technologijų ir biotechnologijų kryptimi – pažangi medicinos inžinerija ankstyvai diagnostikai ir gydymui, nes problemos sprendimo metu įgautos žinios yra apibendrintos matematiniais modeliais, kurie yra tinkami taikyti daiktų interneto sistemose.

Tyrimų objektas

Darbo tyrimų *objektas* – išmanus klasifikatorius skirtas kraštų kompiuterijos uždaviniams spręsti. Šie svarbiausi tyrimų objekto *dalykai* tirti disertacijoje: taikymo metodika ir charakteristikos.

Darbo tikslas

Darbo *tikslas* – pasiūlyti ir ištirti originalias, dirbtinių neuronų tinklų klasifikatoriais grįstas daiktų interneto sistemas skirtas kraštų kompiuterijos uždaviniams spręsti.

Darbo uždaviniai

Siekiant išspręsti aktualią problemą ir pasiekti tikslą, išsikelti šie uždaviniai:

1. Ištirti daiktų interneto kokybines ir kiekybines charakteristikas siekiant atskleisti kraštų kompiuterijos specifiką.
2. Imituoti adaptyviojo rezonanso teoriją grįstas sistemas ir nustatyti kraštų kompiuterijos aplinkos įtaką saviorganizuojančio klasifikatoriaus sudėtingumui ir našumui.
3. Įgyvendinti pastoviosios struktūros dirbtinių neuronų tinklo klasifikatoriumi grįstą vaizdų sandaros sistemą ir bandymais ištirti kraštų kompiuterijos įtaką jos energetiniam efektyvumui.

Tyrimų metodika

Disertacijos tikslas suformuotas atsižvelgiant į plačiai naudojamas daiktų interneto ir kraštų kompiuterijos technologijas susietas su išmanaus mašininio mokymo sprendimais.

Darbe siekiama atskleisti aiškias kraštų kompiuterijos kiekybines ir kokybines charakteristikas, išsiaiškinti technologijos specifiką ir pasiūlyti techninius sprendimus aktualiams ir praktiškai svarbiems uždaviniams, kurie yra nepakankamai moksliskai ištirti, siekiant teigiamo socio-ekonominio efekto.

Disertacijoje naudojamosi neginčijamais nusistovėjusiais metodais, iškeltos hipotezės patvirtinamos eksperimentiniais tyrimais. Siekiama gauti geresnius, lyginant su esamais sprendimais, rezultatus – patikimumą ir energijos sąnaudas. Darbe tiriami ir siūlomi nauji metodai yra grindžiami pasaulinėmis žiniomis ir teorijomis:

- įrenginis-įrenginiui (angl. *M2M*) komunikacijų sistemomis (daiktų interneto ir kraštų kompiuterija);
- dirbtinių neuronų tinklai.

Darbe pasitelkiami žinomi ir pateikimi originalūs skaitmeninio vaizdų apdorojimo, požymių išskyrimo, klasifikavimo ir klasterizavimo, dirbtinių neuronų tinklų saviorganizavimo bei optimizavimo, o taip pat statistinės analizės metodai. Įgyvendinti adaptyviojo rezonanso teorijos 1 ir 2 versijos, neraiškiųjų neuronų tinklo ir atraminių vektorių mašinų

klasifikatorių mokymo ir vykdymo algoritmai, taip pat panaudotas k -vidurkių klasterizavimo algoritmas.

Darbe aprašytiems eksperimentiniams tyrimams atlikti buvo sudaryti originalūs tinklo topologijų, tinklo parametrų ir ląstelių kolonijų vaizdų duomenų rinkiniai. Buvo atliekami dviejų tipų parametriškai valdomi eksperimentai – imitaciniai ir natūriniai. Imitaciniai eksperimentai atlikti *MATLAB* 2020a ir *EdgeNetworkCloudSim* programiniais paketais, natūriniais eksperimentams atlikti sukonstruoti 2 bandomieji standai, abiejų jų pagrindinis sudedamasis elementas yra skaitmeninė vaizdo kamera. Išmanūs elektroniniai sprendimai talpinami į įterptines kraštų kompiuterijos sistemas pasitelkus *Kubernetes* konteinerizavimo ir paskirstymo metodą (Rausch *et al.* 2021).

Darbo mokslinis naujumas

1. Atskleista kraštų kompiuterijos topologinė, parametrinė, greitaveikos priklausomybės ir atsparumo išorinėms poveikio priemonėms specifika leidžianti kurti efektyvias sistemas.
2. Pasiūlytas k -vidurkių klasterizavimo algoritmu grįstas kraštų kompiuterijos koncentratoriaus darbo skirstymo metodas pranoksta įprastus metodus vidutinio paslaugos vykdymo laiko prasme (apkraunant sistemą 600 – 1200 vartotojų tuo pačiu metu.).
3. Sukurtas naujas ląstelių kolonijų vaizdų parametrizavimo metodas ir eksperimentų rezultatai įrodyta, kad jo naudojimas su adaptyviojo rezonanso teorijos 1 ir 2 versijos klasifikatoriais yra tikslesnis nei su atraminių vektorių ar euristiniais klasifikatoriais.
4. Pasiūlytas neraiškiųjų neuronų tinklu grįstas adaptyvaus vaizdo skyros parinkimo metodas pagerina paskirstyto realaus laiko vaizdų sandaros sistemos kraštų kompiuterijoje vykdymo laiką ne mažiau kaip 2 kartus.

Darbo rezultatų praktinė reikšmė

Disertacijoje pateikiami saviorganizuojančio ir fiksuotos struktūros dirbtinių neuronų tinklais grįstų klasifikatorių praktinio taikymo pavyzdžiai kraštų kompiuterijoje. Tirti taikymai medicinos srityje – ląstelių kolonijų vaizdų parametrizavimui ir daugialypėje terpėje – paskirstyto realaus laiko vaizdų sandaros sistemose. Praktiniu būdu gauti tinklo konfigūracijų, parametrų ir protokolų tarpusavio sąveikos tyrimų rezultatai bei rekomendacijos yra reikšmingi kraštų kompiuterijos technologijos architektams kuriant naujas sistemas.

Disertacijoje pateiktais tyrimų rezultatais, rekomendacijomis ir sukurtais metodais buvo vadovautasi vykdant šiuos projektus:

- Mokslo, inovacijų ir technologijų agentūros technologinės plėtros projektas „Išmanioji optinio atsparumo testavimo sistema lazerinių technologijų sektoriui“, Nr. TPP-01-011, 2017–2018;
- Mokslo, inovacijų ir technologijų agentūros inočekis „Intelektualiaisiais metodais grįstų ir lauku programuojamų loginių matricių integriniuose grandynuose įgyvendinamų elektroninių sistemų tyrimas“, Nr. 1981, 2017–2018;

- Lietuvos mokslo tarybos programos „Link ateities technologijų” projektas „Mikrogravitacijos sąlygose suaktyvėjančių ir odos infekcijas sukeliančių mikroorganizmų biokontrolė naudojant elektro-magnetoporaciją”, Nr. LAT-16001, 2016–2018.

Ginamieji teiginiai

1. Daiktų interneto tinklo topologijoje su 13 mazgų ir 15 jungčių, artėjimas link pilnų ryšių topologijos su 13 mazgų ir 16 jungčių lemia didesnę sėkmės tikimybę gauti paketą, pabrėžiant, kad tinklo vėlinimo parametras turi didesnę įtaką nei tinklo pralaidumas.
2. Naudojant saviorganizuojantį adaptyviojo rezonanso teoriją 2 versija grįstą klasifikatorių, galima klasifikuoti ląstelių kolonijų vaizdus su didesniu nei 14 % tikslumu negu naudojant atraminių vektorių klasifikatorių.
3. Pasiūlytas neraiškiųjų neuronų tinklu grįstas adaptyvaus vaizdo skyros parinkimo metodas taikomas decentralizuotoje kraštų kompiuterijoje, sprendžiant vaizdų sandaros uždavinį, pagerina greitaveiką vidutiniškai ne mažiau nei 2 kadr/s, lyginant su ne adaptyviu metodu, kai nuotraukų skyra mažesnė nei 1280×800 taškų.

Disertacijos struktūra

Disertaciją sudaro: įvadas, keturi skyriai, bendrosios išvados, literatūros sąrašas su atskirai pateiktomis autoriaus publikacijomis ir trys priedai. Darbo apimtis yra 138 puslapiai, kuriuose yra pateikta: 33 formulė, 56 paveikslai, 13 lentelės ir 4 algoritmai. Disertacijoje remtasi 156 kitų autorių literatūros šaltiniais.

1. Dirbtinių neuronų tinklų kraštų kompiuterijoje apžvalga

Daiktų interneto technologija tampa vis labiau sutinkama kasdienėje veikloje, efektyvus duomenų apdorojimas paskirstytose įterptinėse sistemose tampa vienu populiariausiu tyrimų objektu (Zalickaitė, Žilinskas 2016). Augantis dirbtiniu intelektu (angl. *artificial intelligence*) grįstų paslaugų poreikis išryškina aktualias debesų kompiuterijos problemas – technologija pasiekia savo ribas, nes vyrauja milžiniškas kiekis duomenų, kurie yra siunčiami tarp serverių ir galinių vartotojų.

Šią problemą siekiama išspręsti naudojant decentralizuotą technologiją – kraštų kompiuteriją, kai apdorojimo užduotys perkeliama į kraštų įrenginius, užtikrinant nebrangias ir saugias duomenų perdavimo charakteristikas (Georgakopoulos *et al.* 2016). Prognozuojama, kad vis daugiau prieinamų įrenginių, naudosis interneto ryšiu, 2025 m. tikimasi daugiau nei 75 mlrd. įrenginių, taigi efektyvaus duomenų perdavimo problema taps labai aktuali. Kraštų kompiuterijos technologijos taikymas šiuose įrenginiuose padėtų įveikti tinklo perkrovos ir perpildytų duomenų centrų problemą (Sabella *et al.* 2016). Dauguma mobiliųjų įrenginių šiai dienai neišnaudoja visų išteklių, taigi tai puiki proga paskirstyti jų išteklius skaičiuoti užduotis, nedarant įtakos specifinėms šių prietaisų funkcijoms. Pilna sąsaja taip skirtingų technologijų ir jose taikomų dirbtinių neuronų tinklų schema pateikta S1.1 paveiksle.

S1.1 pav. Dirbtinių neutronų tinklų (DNT) įgyvendinimo ir skirstymo schema debesų kompiuterijos, rūko kompiuterijos ir kraštų kompiuterijos įrenginiuose (KKI)

Literatūros apžvalgos etape aptarta kraštų kompiuterijos technologijos sąsaja su daiktų interneto technologija topologiniais, programiniais ir įrangos prasmėmis, išsiaiškinti aktualūs tyrimų metodai. Žinoma, kad egzistuoja įvairių kraštų kompiuterijos taikymo būdų, tačiau egzistuojantys [ISO/IEC TR 23188](#) ir [ISO/IEC TR 30164](#) standartai nurodo šias pagrindines kraštų kompiuterijos taikymo taisykles:

- **Saugumas.** Duomenys perduodami tarp prijungtų paskirstytų mazgų per interneto tinklą, todėl paskirstytas pobūdis užtikrina aukštesnį saugumo lygį nei debesų kompiuterijos sprendimai, nes visi duomenys vyrauja tik tarp galinių vartotojų, tačiau poreikis papildomiems šifravimo mechanizmas išlieka. Atsižvelgiant į tai, kad kraštų kompiuterijos įrenginiai dažniausiai yra ribotų išteklių, tai riboja taikytinų saugumo metodų pasirinkimą.
- **Mastas.** Kaip ir bet kurio kito paskirstytojo tinklo technologija, kraštų kompiuterija užtikrina aukšto masto paskirstymą lyginant su debesų kompiuterijos technologija, tačiau įrenginių nevienalytiškumas įveda papildomus našumo ir energijos suvartojimo ribojimus, todėl didelio masto tinkle procesai gali vykti lėčiau. Tai labai aktualu, jei saugumo užtikrinimui naudojami sudėtingi šifravimo algoritmai, kurie didina duomenų apdorojimo trukmę.
- **Patikimumas.** Paskirstyta kraštų kompiuterijos technologija leidžia valdyti situacijas kai mazgas(-ai) tampa nepasiekiami taip užtikrinant tarnybos gyvavimą persikrstant užduotis gretimiems mazgams, taip pat yra galimybė vykdyti paslaugą neprisijungus prie visuotinio interneto tinklo. Kraštų kompiuterijos technologijos patikimumo lygis labai priklauso nuo visos sistemos ir tinklo topologijos dizainų.
- **Greitaveika.** Perkėlus analitinius skaičiavimo uždavinius arčiau tinklo krašto pasiekiamas didesnis duomenų perdavimo greitis, todėl kraštų kompiuterija yra pranašesnė nei debesų kompiuterijos technologija, tai leidžia įgyvendinti paskirstytas realaus laiko sistemas.
- **Efektyvumas.** Dėl augančių daiktų interneto įrenginių našumų ir jų prieinamos kainos atsiveria galimybė taikyti kraštų kompiuterijos įrenginius dideliu mastu. Tokia pertvarka lemia sumažėjusius duomenų srautus interneto tinkle, taip išsaugoma energija, kuri būtų sunaudota duomenims perduoti.

S1.2 pav. Straipsnių kuriuose vartojamos daiktų interneto (DI) ir kraštų kompiuterijos (KK) sąvokos per pastaruosius du dešimtmečius sąvadas (remiantis [Web of Science](#) duomenų baze)

Siekiant ištirti kaip kraštų kompiuterijos sąvoka siejasi su daiktų interneto sąvoka, buvo atlikta straipsnių su šiomis sąvokomis kiekybinė paieška, filtruojant pastarųjų dviejų dešimtmečių intervale. Sąvokų vartojimo grafikas (S1.2) suformuotas logaritmiškai, nes nustatyta, kad kraštų kompiuterijos sąvoka yra santykinai nauja ir populiarumą pasiekė tik 2014 metais. Daiktų interneto šaknys siekia 1990-ųjų pabaigą ir įgijo didelį populiarumą maždaug nuo 2008 metų. Straipsnių skaičiaus sumažėjimas 2020 metais gali būti susijęs su Covid 19 pandemija, dėl kurios sumažėjo bendra mokslinė veikla, todėl tai negali būti traktuojama kaip populiarumo mažėjimo indikacija. Vis didėjantis susidomėjimas daiktų interneto sąvoka rodo, kad visuomenės ir pramonės poreikiai intelektualioms įterptinėms sistemoms auga, o didėjantis kraštų kompiuterijos sąvokos populiarumas rodo, kad perėjimas iš debesų kompiuterijos į kraštų kompiuterijos technologiją jau yra prasidėjęs ir yra aktyviai vystomas.

Populiariausių dirbtinių neuronų tinklais grįstų klasifikatorių, naudojamų daiktų interneto debesų, rūko ir kraštų kompiuterijos technologijose sąvadas pateiktas S1.1 lentelėje. Apžvelgti pagrindinės savybės, tokios kaip mokymo tipas (su mokytoju ar be mokytojo), gebėjimas savarankiškai organizuotis (pakeisti jo išvesties klasių skaičių arba struktūrą), vidutinis santykinis neuronų skaičius ir klasių skaičius, o taip pat įvesties reikšmių tipas. Pateiktos taikymo debesų, rūko ir kraštų kompiuterijoje nuorodos. Remiantis tuo, kad tiek ART1, tiek ART2 klasifikatoriai nebuvo tiriami ir taikyti kraštų kompiuterijos technologijoje (pilkos ląstelės), o adaptiviojo rezonanso teorijos šeimos klasifikatorių apžvalga parodė, kad šie klasifikatoriai gali klasifikuoti duomenis aukšta kokybe ar net pranokti esamus sprendimus, todėl buvo atrinkti kaip galimi kandidatai tolesniems tyrimams. Taip pat galima pastebėti, kad kiti intelektualieji klasifikatoriai yra plačiai tyrinėjami visuose trijuose kompiuterijos technologijose, todėl tai leidžia atlikti jų palyginimą.

Atlikta analitinė literatūros apžvalga patvirtino dirbtinių neuronų tinklų taikymo galimybę kraštų kompiuterijoje, išryškino jų poreikį ir svarbą, taip pat pagrindė klasifikatorių poreikį kraštų kompiuterijos uždaviniams spręsti.

S1.1 lentelė. Populiariausių dirbtinių neuronų tinklais grįstų klasifikatorių taikymo daiktų internete apibendrinimas, remiantis su mokytoju (SM) ir be mokytojo (BM) mokymosi būdo, savi-organizuotumo (SO), įėjimo tipo (devejetainių ir tęstinių verčių) savybėmis ir taikymo debesų (DK), rūko (RK) ir kraštų (KK) kompiuterių technologijose per pastaruosius du dešimtmečius nuorodomis

Sąlybė Klasif.		Mokymas		SO	Vid. skaičius		Įėjimo tipas	Skaiciavimo technologija: taikymo nuorodos
		SM	BM		Neuronų Klasių			
ART1		+	+	+	*	*	dvej.	DK: Bhat <i>et al.</i> 2015; Feng <i>et al.</i> 2012; Rangarajan <i>et al.</i> 2004
								RK: – KK: –
ART2		+	+	+	**	**	tęst.	DK: Bhat <i>et al.</i> 2015; Park <i>et al.</i> 2017; Wan <i>et al.</i> 2018
								RK: Shete <i>et al.</i> 2016 KK: –
FNN		+	–	–	**	**	tęst.	DK: Huang <i>et al.</i> 2012; Kim 2016
								RK: Aral, Ovatman 2016 KK: Gravalos <i>et al.</i> 2018; Souza <i>et al.</i> 2017
Gilaus mokymo		+	+	–	*	*	tęst.	DK: Ben-Zikri <i>et al.</i> 2017; Wang <i>et al.</i> 2021; Zhang <i>et al.</i> 2020b
					*	*		RK: Jazayeri <i>et al.</i> 2020; Lyu <i>et al.</i> 2020; N.G., S. 2020
								KK: Vėstias 2019; Wang <i>et al.</i> 2020; Xu <i>et al.</i> 2021; Zhang <i>et al.</i> 2020a
SOM		+	–	+	**	**	tęst.	DK: Stephanakis <i>et al.</i> 2019; Thilagavathy, Rajagopal 2019; Vesanto, Alhoniemi 2000
								RK: Bhatia 2020 KK: Manihar <i>et al.</i> 2020; Yu <i>et al.</i> 2018
								DK: Valluru, Jeya 2020; Zhang <i>et al.</i> 2019
SVM		+	+	–	**	**	tęst.	RK: Moghaddam <i>et al.</i> 2020; Wadood <i>et al.</i> 2017
								KK: Avudaiammal <i>et al.</i> 2020; Merenda <i>et al.</i> 2020; Murshed <i>et al.</i> 2019

2. Daiktų interneto charakteristikos

Daiktų internetas paprastai apibūdinamas daiktų (įrenginių) konfigūracija ir tinklo charakteristikomis. Siekiant atskleisti kraštų kompiuterijos specifiką, mes nusprendėme sutelkti dėmesį į daiktų interneto ypatybes, nes dauguma jų yra paveldimos, taip pat nagrinėjamas jutiklių duomenų aspektas. Trys pagrindiniai aspektai naudojami charakterizuojant daiktų internetą: tinklo topologija, jungties charakteristikos ir mazgų konfigūracija.

Egzistuoja daugybė tyrimų nagrinėjančių darbo planavimo problemą, juose siūlomi optimalaus virtualių mašinų talpinimo kompiuterių tinkle algoritmai. Tačiau dauguma jų nesvarsto tinklo architektūros optimizavimo, todėl atliekamas skirtingų tinklo topologijų modeliavimas kurio tikslas nustatyti, kas daiktų internete turi didesnę įtaką: tinklo parametrai ar tinklo topologija. Tam nustatyti buvo atlikti imitaciniai eksperimentiniai tyrimai su realių tinklų konfigūracija, kurių metu buvo varijuojami tinklo parametrai bei topologijos, o gauti rezultatai patvirtina iškeltas hipotezes bei pateikia rekomendacijas skirtas daiktų interneto architektams naudoti projektuojant sistemas.

Dauguma tirtų kraštų kompiuterijos koncentratorių sprendžia ribotų aparatinės įrangos galimybių problemą pritaikant tracinį planavimą, sutelkiant dėmesį į atsako laiką tarp paslaugos mazgų. Tiesioginio vaizdo transliacijos paslauga yra puikus pavyzdys šiai problemai išryškinti. Dėl didelės šios paslaugos paklausos ir aktyvaus korporacijų, tokių kaip „Facebook, Inc.“ ir „Google“ vaizdo transliacijos paslaugų vystymo, poreikis šiems tyrimams yra aktualus. Vienos įvesties vaizdo srautui reikalingi pradiniai srauto kodavimo ir perkodavimo žingsniai, tam, kad atitiktų skirtingų įrenginių skyros poreikį, todėl šioms užduotims atlikti reikalingas tarpinis įrenginys. Šiuo metu naudojamos debesų kompiuterijos technologijos serveriai, turintys didelius išteklius, kodavimo-perkodavimo užduotį atlieka sparčiai, tačiau jų statinė fizinė geografinė padėtis lemia didelį duomenų vėlavimą. Dalinis sprendimas, sprendžiantis didelio vėlavimo problemą yra turinio pristatymo tinklai (angl. *CDN*), tačiau tai pilnai neeliminuoja vaizdo srauto siuntimo į nutolusius serverius. Siekiant atskleisti būtent kraštų kompiuterijos specifikas, tirtas kitoks požiūris – atliktas tyrimas vykdamas imitacinį eksperimentą, kurio metu pasiūlytas adaptyvus aptarnaujančių virtualių mašinų metodas parenka optimalų kelią pagal du parametrus: delsą tarp vartotojo ir gali-

a)

b)

S2.1 pav. Vidutinės tinklo delsos (a) ir vidutinės paslaugos trukmės (b) priklausomybės nuo vartotojų skaičiaus pagal skirtingų topologijų atvejų

mos aptarnaujančios virtualios mašinos, ir išteklių toje mašinoje likutį. Toks sprendimas atliekamas automatiškai remiantis k -vidurkių klasterizavimo algoritmu. Tirtų topologijų rezultatai pateikiami S2.1 paveiksle.

Daiktų internetas yra glaudžiai susijęs su belaidžių jutiklių tinklais (angl. *WSN*), kurie dažnai naudojami kaip sluoksnis tarp duomenų šaltinio ir jų gavėjo. Norėdami atskleisti jutiklių tinklų specifikas, modeliuojamas dinamiškai pertvarkomas (angl. *Ad-hoc*) mobilusis tinklas (angl. *MANET*) ir tiriami realios sistemos parametrai. Pagrindinė *MANET* tinklo savybė – ji neturi jokio centralizuoto valdymo mechanizmo ir nėra apibrėžtas topologine struktūra, todėl kiekvienas mazgas veikia kaip maršrutizatorius likusiems mazgams tinkle. Nepalankios oro sąlygos smarkiai veikia *MANET* tinklo veikimą, ypatingas dėmesys turėtų būti skiriamas šio sprendimo diegimui tam tikroje aplinkoje. Dėl skirtingų oro sąlygų trikdžių gali vėluoti duomenų persiuntimas arba duomenų paketai gali apskritai būti nepristatyti. Skirtingi duomenų perdavimo protokolai skirtingai reaguoja į trikdžius, todėl buvo atliktas jutiklių tinklų specifikų tyrimas, kurio metu atlikta belaidžių, radijo ryšiu komunikuojančių mazgų analizė, varijuojant skirtingais populiariais duomenų perdavimo protokolais. Siekiant atskleisti skirtingų protokolų savybes, belaidis tinklas buvo trikdytas trimis skirtingais gamtoje sutinkamų trikdžių modeliais: rūko, lietaus ir sniego.

3. Saviorganizuojantis klasifikatorius ląstelių kolonijoms skaičiuoti

Ląstelių kolonijų skaičiavimas yra svarbi citometrijos mokslų dalis (Herberg *et al.* 2015; Lambert *et al.* 2017), reikalaujanti didelio našumo analizės ir tikslios peržiūros dideliu greičiu. Nors rankinis skaičiavimas išlieka naudojamu, susidomėjimas automatiniais metodais yra vis labiau tyrinėjamas, atrandama daug nuoseklių rezultatų. Rankinis ląstelių kolonijų skaičiavimas yra lėčiausias ir kartais mažiausiai patikimas metodas būtent dėl žmogaus klaidų faktoriaus, ypač kai susiduriama su dideliais ląstelių kolonijų rinkiniais, o taip pat sudėtingais biologinių mėginių modeliais. Siekiama, kad rankinis skaičiavimas būtų pilnai pakeistas automatiniais sprendimais. Tai ypač reikšinga, kai kalbama apie šiuolaikinę mediciną ir biotyra, kur žmonių kontaktas su pavojingomis medžiagomis turėtų būti sumažintas iki minimumo (Dursun *et al.* 2016; Savickiene *et al.* 2009), o tolimesni tyrimai būtų atliekami saugioje aplinkoje.

S3.1 pav. Sukonstruotas skaitmenine vaizdo kamera grįstas maketas skirtas ląstelių kolonijų vaizdams gauti

Pasiūlytas originalus ląstelių kolonijų vaizdų parametrizavimo metodas, leidžiantis palyginti skirtingų (tradicinių ir išmanių) klasifikatorių rezultatus. S3.2 pav. iliustruoja sukurto metodo žingsnius, jie yra yra apibendrinti trimis pagrindiniais etapais: registracija, detektavimas ir tikrinimas.

Biologinių bandinių Petri lėkštelėje vaizdai buvo gauti naudojant sukonstruotą maketą (S3.1 pav.). Maketą surado SMX-12A2x skaitmeninė kamera, tiksliai derinamo aukščio stovas leidžiantis nustatyti tinkamą atstumą nuo kameros iki bandinio, taip pat fokusuotas dirbtinės šviesos šaltinis – platus matomos šviesos spektro kaitrinė lempuė.

Siekiant išsiaiškinti ar šios užduoties sprendimo įgyvendinimas yra galimas kraštų kompiuterijos technologijoje, buvo atlikti tyrimai įgyvendinant automatinę ląstelių kolonijų skaičiavimo sistemą kuri grįsta euristiniu, atraminių vektorių (angl. *support vector machine*, *SVM*) ir adaptyviojo rezonanso teorija grįstais klasifikatoriais. Tirtų klasifikatorių rezultatai pateikti S3.1 lentelėje, jie gauti remiantis susimaišymo matrica (angl. *confusion matrix*). Remiantis gautais rezultatais išrinkti geriausi $ART1_n$ ir $ART2_n$ klasifikatoriai (jie pažymėti pilku fonu). Išsiaiškinta, kad $ART1$ klasifikatorius mokytas antruoju mokymo pavuždžių rinkiniu, o $ART2$ klasifikatorius – pirmuoju rinkiniu parodė geriausius klasifikavimo kokybės rezultatus, remiantis svarbiausiais statistiniais klasifikatorių vertinimo

S3.2 pav. Pasiūlyto ląstelių kolonijų vaizdų parametrizavimo metodo žingsninė schema

S3.1 lentelė. Tirtų klasifikatorių statistinių indeksų rezultatai

Klasifik.*	F1 balas	Tikslumas	Preciz.	DOR	Youden's J
euristinis	0,906	0,890	0,840	209,89	0,764
<i>SVM</i>	0,879	0,873	0,902	48,85	0,748
<i>ART1</i> ₁	0,654	0,683	0,795	6,21	0,388
<i>ART1</i> ₂	0,939	0,933	0,928	200,90	0,864
<i>ART1</i> ₃	0,705	0,721	0,818	8,46	0,458
<i>ART2</i> ₁	0,907	0,890	0,834	506,29	0,765
<i>ART2</i> ₂	0,879	0,863	0,834	48,18	0,714
<i>ART2</i> ₃	0,793	0,788	0,837	14,75	0,582

* - pilka spalva pažymėti geriausi atitinkamos šeimos klasifikatoriai.

indeksais – *DOR* ir Youdeno *J*. Didėjančios šių indeksų reikšmės rodo geresnę klasifikavimo kokybę, lyginant su etaloninėmis klasifikavimo reikšmėmis.

Jautrumo ir specifiškumo grafike (S3.3 pav.) pateikiami visų tirtų klasifikatorių rezultatai. Yra žinoma, kad viršutinė kairė padėtis rodo tesingiausią klasifikaciją, kuo aukščiau įstrižainės yra išsidėstę klasifikatoriai, tuo geresni jo klasifikavimo kokybės rezultatai, todėl yra daroma išvada, kad *ART2*₁ klasifikatorius lenkia euristinį ir *SVM* klasifikatorius jautrumo ir specifiškumo prasmėmis – Youdeno *J* indeksu, o kokybiškiausiai klasifikuojantis iš tirtų klasifikatorių yra *ART1*₂. Kadangi kiekviename biologinio mėginio vaizde yra skirtingas ląstelių kolonijų skaičius, skaičiuota vidutinė vykdymo trukmė. Rezultatai rodo, kad *ART1* klasifikatorius lenkia *SVM* klasifikatorių algoritmo sudėtingumo ir vykdymo laiko prasmėmis, o *ART2* klasifikatorius būdamas panašaus kompleksiskumo kaip *SVM* klasifikatorius, vykdo klasifikavimą tiksliau ir sparčiau.

S3.3 pav. ROC kreivės rezultatai visiems tirtiems klasifikatoriams: euristinis, *SVM*, *ART1*_n ir *ART2*_n

4. Realaus laiko klasifikatorius vaizdai sudaryti

Vaizdo sandara yra sudėtingas procesas, reikalaujantis daug aparatinės įrangos skaičiavimo išteklių (Seo *et al.* 2019). Tuo pačiu, tai yra panoraminės fotografavimo funkcijos moderniose mobiliuose telefonuose pagrindas. Šiuo metu siūlomi sprendimai leidžia efektyviai atlikti vaizdo sandaros veiksmą kai vaizdai su postūmiu gaunami ta pačia vaido kamera (Omar *et al.* 2018). Šiame darbe svarstomas sudėtingesnis uždavinys – naudojamos keletą kamerų, o vaizdų nuskaitymas ir vaizdų sandara yra atliekami realiu laiku.

Vaizdo sandaros metodą sudaro šie žingsniai: požymių išskyrimas; požymių atitikmenų paieška; homografijos apskaičiavimas; vaizdų geometrinė transformacija. Preliminarūs tyrimai atskleidė, kad požymių išskyrimas yra daugiausiai skaičiavimų reikalaujanti užduotis, sudaranti maždaug trečdaliį viso vykdymo trukmės. Pasiūlyti du būdai, kaip vaizdų sandaros metodas gali būti įgyvendintas kraštų kompiuterijoje: centralizuotas, kai vienas iš įrenginių priskiriamas valdančiuoju mazgu, jam perduodami neapdoroti vaizdai iš valdomųjų įrenginių, tolesni metodo žingsniai veiksmai atliekami valdančiajame mazge; decentralizuotas, kai valdomieji įrenginiai atlieka požymių išskyrimo žingsnį ir valdančiajam įrenginiui perduodami neapdorotas vaizdas ir su juo susijęs požymių rinkinys, tolesniu žingsnius atlieka valdantysis įrenginys.

Pasiūlytas adaptyvus vaizdo skyros parinkimo metodas, jo struktūra pavaizduota S4.1 paveiksle. Neraiškiųjų neuronų tinklo klasifikatoriaus pagalba parenkama maksimali galima skyra pagal baterijos įkrovos reikšmę; tuomet iš visų įrenginių maksimalių skyrų išrenkama minimali skyra ir visų įrenginių kameroms priskiriama ši skyra. Algoritmas įgyvendintas *Python* programavimo kalba, vėliau programa pakuojama į *Docker* konteinerį kartu su papildomomais programiniais paketais: *OpenCV* biblioteka, *Keras* neurų tinklų programinis paketas, *WebSockets* ryšių biblioteka. *Kubernetes* aplinka buvo pasirinkta siekiant užtikrinti sklandų algoritmo talpinimą įrenginiuose nepriklausomai nuo jų tipo ir jose veikiančios operacinės sistemos. Baterijos įtampos matavimas kiekviename mazge atliekamas 18 skilčių ASK keitikliu prijungtu naudojant I²C protokolą. Elektriniai galios parametrai buvo matuojami *Ruideng UM34* galios matavimo prietaisu.

S4.1 pav. Sukurto neraiškiųjų neuronų tinklo grįsto klasifikatoriaus skirto adaptyvaus vaizdo skyros parinkimui struktūra daugeliui krašto kompiuterijos įrenginių

S4.2 pav. Vidutinių energijos sąnaudų (kairieji stulpeliai) ir CPI apkrovos priklausomybė nuo vaizdo skyros

Siūlomo metodo veikimas vertinamas tokiais būdais: vaizdų sandaros greitimeika (kadrai/s), vidutinės mazgo energijos sąnaudos (W), vidutinė CPI apkrova (%) ir mazgo akumuliatoriaus įkrovos lygis (%). Šių matavimų rezultatai pateikti S4.2 paveiksle. Norėdami padidinti rezultatų patikimumą ir eliminuoti atsitiktinių klaidų galimybę, matavimų rezultatai buvo kaupiami 10 min intervale kiekvienai skyrai. Kiekvienas matavimas buvo pradėtas visiškai įkrautu ličio jonų akumuliatoriumi (4,20 V lygis). Matavimai buvo tęsti iki bent vieno įrenginio akumuliatoriaus išsekimo. Apibendrinti metodo rezultatai pateikti S4.1 lentelėje, paslaugos trukmė tiesiogiai priklauso nuo kameros skyros. Adaptivus skyros parinkimo metodas lemia daugiau nei 2 valandų nepertraukiamą įrenginių veikimą išlaikant maksimalią vaizdo kokybę, efektyviai išnaudojama baterijos įkrova. Rezultatai rodo, kad vaizdo skyros vertė turi nedidelį poveikį sandaros algoritmo greitimeikai kai algoritmas vykdomas nutolusiuose įrenginiuose su daug skaičiavimo išteklių. Tačiau vaizdo sandaros greitimeika stipriai mažėja kai algoritmas vykdomas kraštų kompiuterijos aparatinėje įrangoje, skirtumas tarp centralizuotos ir decentralizuotos topologijos vidutiniškai siekė 2,08 kadrus/s. Rezultatai patvirtina, kad būtent aparatinės įrangos ištekliai turi daugiau įtakos nei tinklo ryšio ypatybės. Be to, sandaros algoritmo decentralizavimas paskirstant funkcijas kraštų kompiuterijos įrenginiams didina sandaros greitimeiką.

S4.1 lentelė. Baterijos išsikrovimo trukmės priklausomybė nuo taškų skaičiaus skirtingoms kameros skyroms, matuota iki pilno įrenginio išsijungimo

Skyra, taškų	Duomenų dydis, taškų	Baterijos išsikrovimas, min
160×120	19 200	203
176×144	25 344	171
320×240	76 800	82
800×600	480 000	61
1280×800	1024 000	58
Adaptivi	40 991*	124

* – aproksimuota vertė.

Bendrosios išvados

Tikslas pasiūlyti originalias dirbtinių neuronų tinklų klasifikatoriumi grįstas daiktų interneto sistemas kraštų kompiuterijos uždaviniams spręsti buvo pasiektas. Gauti šie elektros ir elektronikos inžinerijos mokslo kryptčiai reikšmingi rezultatai:

1. Atlikti imitaciniai eksperimentai atskleidė šias kraštų kompiuterijos specifikas:
 - kraštų kompiuterijoje, tinklo delsos parametras turi didesnę įtaką sėkmingam duomenų paketų pristatymui, nei tinklo pralaidos parametras;
 - kraštų kompiuterijos topologijos taikymas mažina vidutinę tinklo delsą ir paslaugos vykdymo laiką, lyginant su debesų kompiuterijos topologija;
 - perduodant jutiklių duomenis kraštų kompiuterijos tinklų sąsajai-koncentratoriui, rekomenduojama taikyti pilnų ryšių tinklo topologiją, tai užtikrina mažesnę tikimybę iširemti į tinklo ribas (angl. *bottleneck*);
 - siekiant pagerinti nestabilaus tinklo (pralaidos prasme) sistemos našumą, reikėtų mažinti mazgų skaičių arba skaldyti topologiją segmentais-potinkliais.
2. Pasiūlytas k -vidurkių klasterizavimu grįstas darbo skirstymo metodas leido sumažinti vidutinį paslaugos aptarnavimo laiką, lyginant su įprastais metodais, apkraunant sistemą 600 – 1200 vartotojų tuo pačiu metu.
3. Sukurtas naujas ląstelių kolonijų parametrizavimo metodas ir eksperimentų rezultatais patvirtinta, kad adaptivityvojo rezonanso teorijos 1 ir 2 versijos klasifikatoriai atlieka klasifikavimą tiksliau nei atraminių vektorių ar euristinis klasifikatoriai:
 - *ART1* klasifikatorius padidina klasifikavimo patikimumą remiantis statistiniais jautrumo ir specifiskumo matmenimis.
 - *ART1* klasifikatorius pralenkia atraminių vektorių ir euristinių klasifikatorių statistinių F1 balo ir tikslumo matmenimis.
 - Remiantis *ROC* kreivės ir Youden J indekso rezultatais, išrinktas geriausias *ART2*₁ klasifikatorius, klasifikuojantis biologinių bandinių rinkinį iki 99,35 % teisingai teigiamų ir iki 76,89 % teisingai neigiamų pavyzdžių.
 - Bendrai, *ART2* klasifikatoriaus rezultatai yra labai panašūs į euristinio klasifikatoriaus rezultatus F1 balo ir tikslumo prasmėmis, tačiau *DOR* matmuo yra daugiau nei 2 kartus didesnis.
 - *ART2* klasifikatorius yra pranašesnis teisingai teigiamų pavyzdžių prasme nei šie klasifikatoriai: *ART1* – 4 %, atraminių vektorių – 14 % ir euristinių – 2 %.
4. Originalus neraiškiųjų neuronų tinklu grįstas adaptivityvus vaizdo skyros parinkimo metodas leidžia sumažinti bendrą paslaugos laiką ne mažiau kaip 2 kartus, kai metodas vykdomas kraštų kompiuterijos tinklo įrenginiuose.

Ateityje šiame darbe pristatytus tyrimus reikėtų tęsti lygiagrečiai pasiūlytus metodus ir įgyvendinant juos laukų programuojamose loginėse matricose, toliau atliekant natūrinius eksperimentus ir palyginant su šiame darbe aprašytais rezultatais.

Subject Index

A

accuracy 72, 80, 82, 85, 100
ADC 93, 94
AI 9, 18, 33
algorithm 3, 8, 12, 19–24, 27, 30, 34, 51–53, 56,
63, 66, 67, 70, 75, 77, 79, 84, 88, 89, 91,
93, 95, 97
ANN. viii, 2–6, 10, 11, 16, 17, 19, 21–23, 27, 28,
30, 31, 33–35, 37, 51, 71, 78, 87
API 15
ARM 34
ART . 3, 8, 22–25, 28, 65–67, 70, 72, 78, 81, 82,
85
ART1 . . . 5, 23, 24, 28, 29, 65, 71, 72, 74, 75, 77,
79–82, 84, 85, 100
ART2 5, 7, 22, 23, 28, 29, 65, 71, 72, 75, 77, 79,
81, 82, 84, 85, 100
ART3 23
ASIC 31–33

B

bandwidth . . 1, 14, 18, 38, 39, 41–43, 45, 47, 48,
51, 53, 63, 99

C

CDN 2, 49, 50, 53–56, 63
cell colonies . . 3, 5, 6, 65–68, 70, 72, 79–81, 84,
85, 87, 100

cell colony 5
chip family
 BCM2837 94
 Ruideng UM34 94, 130
classification 2, 3, 5, 18–21, 65, 70–72
classifier . 2–6, 19, 21, 22, 28, 35, 55, 56, 65–67,
70–72, 75, 77–82, 84, 85, 87, 89, 91, 92,
95, 96, 99, 100
cloud . . 1–3, 9–13, 15, 17, 18, 20, 28, 40, 56, 58,
60, 89, 90, 93, 95
cloudlets 15, 51
clustering . 3, 5, 14, 20, 23, 49, 50, 52, 53, 89, 99
CNN 25, 26, 30, 31
company
 Advanced Assessment Systems, Inc. 41
 Altera, Corp. 32
 Cisco Systems, Inc. 14
 Clarivate, Comp. v, vi, 7
 Dell, Inc. 14
 Espressif Systems 41
 Facebook, Inc. 49
 Google, LLC 14, 19, 49
 Intel, Corp. 14, 32, 33
 MathWorks, Inc. 5, 15, 17, 40, 50, 58, 78, 84,
 121
 Microsoft, Corp. 15, 32, 58, 94, 95
 Nvidia, Corp. 14, 32, 34
 Oracle, Corp. 15, 40, 50

- Rackspace Technology 14
- Raspberry Pi Foundation 34, 41, 94
- Red Hat Software 14
- Ruideng Technology 94, 130
- Samsung Electronics 94
- Sumix, Corp. 79
- Twitter, Inc. 49
- Xilinx, Inc. 32, 94
- complexity 2, 12, 65
- counting 3, 65–67, 87
- CPU 32, 34, 39, 52, 53, 95–98
- cyber-physical systems 41
- D**
- delay 1, 10, 12, 14, 39–43, 45, 47, 48, 53, 54, 56, 57, 63, 99
- detection 18, 21, 66, 67, 79
- diagnostic odd ratio 80
- DNN 30
- DOR 80, 82, 84, 85, 100
- DSP 34
- E**
- EC 2–6, 9, 10, 12–20, 22, 23, 26, 28, 32, 34, 35, 37–41, 48–50, 56, 63, 65, 87–90, 93, 95, 98–100
- ECD 10–12, 15, 17–20, 23, 26, 28, 41, 51, 77, 87, 89, 91–93, 97
- edge 2, 3, 10, 18, 28
- efficiency 1, 2, 4, 5, 10, 18, 31, 34, 35, 39, 41, 87
- extraction 10, 34, 90, 96, 98
- F**
- F1 score 80, 82, 85, 100
- FNN .. 3, 5, 6, 21, 29, 87, 89, 92, 94–96, 98, 100
- fog 2, 3, 10, 15, 18, 28, 60
- FPGA 30–34
- G**
- gateway . 3, 15, 18, 37, 39, 49, 51, 55, 56, 63, 99
- GPU 31, 32, 34
- H**
- heuristic 5, 71, 79, 82, 84, 85, 100
- Hough transform 70
- hypothesis 48, 67, 85, 87, 89, 98
- I**
- image descriptor
 - AKAZE 91
 - BRIEF 90
 - FAST 90
 - KAZE 90
 - ORB 90
 - SIFT 90
 - SURF 90
- image stitching 3
- implementation . 9, 10, 16, 31, 32, 34, 35, 66, 75, 88, 90
- Industry 4.0 7, 14, 40
- Internet 1, 6–9, 11, 14, 17, 35, 37, 39–41, 44, 45, 47, 48, 58, 65, 87
- IO 32, 34
- IoT 1, 2, 4, 5, 7–18, 28, 32, 35, 37–41, 44, 45, 47, 48, 57, 62, 63, 88, 89
- IoV 11–13, 19, 34, 88
- J**
- jitter 39, 42, 63, 99
- K**
- k*-means 5, 23, 49–53, 55, 63, 99
- L**
- latency 1, 49, 51–53
- link 38, 39, 41, 49, 51, 52, 56
- load 1, 2, 7, 9, 16, 41, 53, 54, 56, 62, 69, 89, 95–98
- M**
- M2M 11
- MANET 57, 58
- matching 91
- MEC 11, 16, 95
- microcomputer
 - ESP32 41
 - Jetson Nano 34
 - LifeCam 94
 - LinkIt Smart 7688 Du 41
 - Parallela 32
 - Raspberry Pi 34, 41, 94
 - Samsung 94
 - SMX-12A2x 79
 - Tegra 34
 - Tegra K1 32
 - Tegra X1 32
 - Xilinx Zynq-7000 94
 - Zynq 32
- model 4, 14–16, 24–27, 60–62

N

node 14, 16–18, 38–42, 48, 53, 54, 56–60, 91–93, 98

P

parameterization 3, 5, 66–68, 70–72, 80, 85, 100
 PC 41
 performance 1–4, 12, 16, 17, 27, 31, 32, 35, 37–40, 48, 55, 56, 61–63, 65, 95, 99
 Petri dish 67–70, 79
 precision 80, 82, 85, 100
 processing 11, 79, 89
 protocol
 Bluetooth 39
 CoAP 39
 DSR 58, 62
 Ethernet 39
 HDS 49
 HLMRP 58, 62, 63
 HLS 49
 HTTP 39
 LoRa 39
 LTE 39
 MQTT 16, 39
 ODMRP 58, 62
 WiFi 39
 ZigBee 39

R

RAM 32, 39, 40, 53, 84
 real-time 11
 registration 69, 79
 REST 15
 RF 58–60, 62, 63
 RNN 30
 ROC 85, 100

S

sensitivity 80, 85, 100
 service time 48, 53–57, 63, 95, 99
 simulation 15, 37, 40, 41, 48–50, 53, 54, 56, 58–62
 SoC 32–34, 41, 91, 93, 94, 97
 software 58
 .NET 15
 AlexNet 19, 25
 Altera 32
 Azure 15, 32, 95
 Bing 32
 BRITE 40, 53

Clarivate Analytics Web of Science ... v, vi, 7
 CloudAnalyst 14
 CloudExp 14
 CloudSim 14, 15
 Complex Networks Package 15
 ConvNetJS 33
 Docker 14, 88, 93–95, 130
 EdgeCloudSim 14, 15, 50
 EdgeNetworkCloudSim 5, 40, 121
 GoogLeNet 19
 iFogSim 14
 Intel 32, 33
 Java 15, 33, 40, 50
 JavaScript 15
 Keras 33, 94, 130
 Kubernetes 5, 88, 93, 94, 121, 130
 KVM 14
 Linux 15, 94
 MATLAB ... 5, 15, 17, 40, 50, 58, 78, 84, 121
 Neural Designer 33
 Neuroph 33
 NS-2 15
 ODMRP 62
 OpenCV 5, 94, 130
 OpenFlow 14
 OpenStack 14
 Python 33, 94, 130
 SimEvents 15
 TFLearn 33
 ThingSpeak 15
 Ubuntu 40, 50
 VGGNet 19, 25
 WebSockets 5, 93, 94, 130
 WEKA 50
 Windows 15, 58
 SOM 18, 21, 26, 27, 29
 specificity 80, 85, 100
 stitching . . 3, 4, 6, 34, 35, 87–91, 93–96, 98, 100
 SVM . . 3, 5, 6, 20, 21, 23, 27–29, 65–67, 71, 72, 77–79, 82, 84, 85, 100

T

technique . 2, 5, 6, 34, 37, 56, 58, 63, 67, 68, 85, 87, 98–100
 test bench 5, 87, 94
 topology . 1, 10, 13, 14, 18, 19, 37–41, 47–50, 53, 54, 56, 63, 99

V

verification 67

VHDL 31
VM 14, 15, 40, 49–54

W

WSN 57, 62, 63

Y

Youden’s J index 80, 82, 85, 100

Annexes¹

Annex A. Declaration of Academic Integrity

Annex B. The Co-authors' Agreements to Present Publications Material in the Dissertation

Annex C. The Copies of Scientific Publications by the Author on the Topic of the Dissertation

¹The annexes are supplied in the enclosed compact disc.

Julius SKIRELIS

EDGE COMPUTING TIED IN
ARTIFICIAL NEURAL NETWORK CLASSIFIERS

Doctoral Dissertation

Technological Sciences,
Electrical and Electronic Engineering (T 001)

DIRBTINIŲ NEURONŲ TINKLŲ KLASIFIKATORIAIS
SUSIETA KRAŠTŲ KOMPIUTERIJA

Daktaro disertacija

Technologijos mokslai,
elektros ir elektronikos inžinerija (T 001)

2021 05 06. 13,0 sp. l. Tiražas 20 egz.

Leidinio el. versija <https://doi.org/10.20334/2021-021-M>

Vilniaus Gedimino technikos universitetas

Saulėtekio al. 11, 10223 Vilnius,

Spausdino BĮ UAB „Baltijos kopija“,

Kareivių g. 13B, 09109 Vilnius