VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS

Eldar ŠABANOVIČ

SĄSŪKOS DIRBTINIŲ NEURONŲ TINKLŲ ĮGYVENDINIMAS SPARTINANČIUOSIUOSE ĮRENGINIUOSE VAIZDAMS ANALIZUOTI REALIUOJU LAIKU

DAKTARO DISERTACIJA

TECHNOLOGIJOS MOKSLAI,
ELEKTROS IR ELEKTRONIKOS INŽINERIJA (T 001)

VGTU
LEIDYKLA
Vilnius TECHNIKA 2019

Disertacija rengta 2015–2019 metais Vilniaus Gedimino technikos universitete.

Vadovas

doc. dr. Dalius MATUZEVIČIUS (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. Vytautas URBANAVIČIUS (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),
doc. dr. Gytis MYKOLAITIS (Vilniaus Gedimino technikos universitetas, elektros ir elektronikos inžinerija – T 001),
habil. dr. Krzysztof Zbigniew PATAN (Zelionos Guros universitetas, Lenkija, elektros ir elektronikos inžinerija – T 001).

Disertacija bus ginama viešame Elektros ir elektronikos inžinerijos mokslo krypties disertacijos gynimo tarybos posėdyje **2019 m. gruodžio 6 d. 9 val.** Vilniaus Gedimino technikos universiteto senato posėdžių salėje.

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

Tel.: (8 5) 274 4956; faksas (8 5) 270 0112; el. paštas doktor@vgtu.lt

Pranešimai apie numatomą ginti disertaciją išsiųsti 2019 m. lapkričio 5 d.

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

VGTU leidyklos TECHNIKA 2019-051-M mokslo literatūros knyga
http://leidykla.vgtu.lt

ISBN 978-609-476-204-8

eldar.sabanovic@vgtu.lt

VILNIUS GEDIMINAS TECHNICAL UNIVERSITY

Eldar ŠABANOVIČ

IMPLEMENTATION OF CONVOLUTIONAL NEURAL NETWORKS IN ACCELERATING UNITS FOR REAL-TIME IMAGE ANALYSIS

DOCTORAL DISSERTATION

TECHNOLOGICAL SCIENCES,
ELECTRICAL AND ELECTRONIC ENGINEERING (T 001)

VGTU
LEIDYKLA
TECHNIKA
Vilnius 2019

Doctoral dissertation was prepared at Vilnius Gediminas Technical University in 2015–2019.

Supervisor

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

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

Chairman

Prof. Dr Vytautas URBANAVIČIUS (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),
Assoc. Prof. Dr Gytis MYKOLAITIS (Vilnius Gediminas Technical University, Electrical and Electronic Engineering – T 001),
Dr Habil. Krzysztof Zbigniew PATAN (University of Zielona Gora, Poland, Electrical and Electronic Engineering – T 001).

The dissertation will be defended at the public meeting of the Dissertation Defence Council of Electrical and Electronic Engineering in the Senate Hall of Vilnius Gediminas Technical University at **9 a. m. on 6 December 2019**.

Address: Saulėtekio al. 11, LT-10223 Vilnius, Lithuania. Tel.: +370 5 274 4956; fax +370 5 270 0112; e-mail: doktor@vgtu.lt

A notification on the intend defending of the dissertation was send on 5 November 2019.

A copy of the doctoral dissertation is available for review at VGTU repository http://dspace.vgtu.lt, at the Library of Vilnius Gediminas Technical University (Saulėtekio al. 14, LT-10223 Vilnius, Lithuania), and at the Library of the Lithuanian Academy of Sciences (Žygimantų st. 1, LT-01102 Vilnius, Lithuania)

Reziumė

Matomojo šviesos spektro vaizdų analizė įgalina intelektualiąsias sistemas gauti informaciją taip, kaip žmogus rega. Daugelyje sričių taikoma sąsūkos dirbtinių neuronų tinklais (SDNT) grįsta vaizdų analizė. Tačiau dėl didelės skaičiavimų apimties kyla sunkumų įgyvendinant įterptinėse sistemose lokaliai vykdomus SDNT grįstus algoritmus vaizdų analizei realiuoju laiku. Šiuo metu vaizdų analizei įterptinėse sistemose galima taikyti spartinančiuosius įrenginius, tačiau trūksta suderinamų SDNT elementų sąrašų ir SDNT pritaikymo spartinantiesiems įrenginiams įterptinėse sistemose metodikos.

Darbo tyrimų objektas – sąsūkos dirbtinių neuronų tinklai vaizdams analizuoti realiuoju laiku. Disertacijoje tiriami šie, su tiriamuoju objektų susiję dalykai: įgyvendinimas spartinančiuosiuose įrenginiuose ir projektavimo metodika.

Disertacijoje pateikiami vaizdams apdoroti skirtų SDNT įgyvendinimo tyrimai, kuriais remiantis sudarytas SDNT elementų sąrašas ir metodika, skirta SDNT pritaikymui įgyvendinti spartinančiuosiuose įrenginiuose pasirinktiems vaizdų analizės uždaviniams spręsti. Taip pat pateikiami dviejų taikymo sričių vaizdų analizės algoritmų, sukurtų taikant pasiūlytą elementų sąrašą ir metodiką, aprašymai. Viena iš sričių – žmogaus akies tinklainės vaizdų požymių aprašymas. Kita sritis – kelio dangos tipo ir būklės įvertinimas, analizuojant vaizdus iš automobilio priekyje sumontuotos vaizdo kameros.

Pagrindiniai disertacijos rezultatai yra paskelbti septyniuose moksliniuose straipsniuose recenzuojamuose mokslo leidiniuose: vienas straipsnis mokslo žurnale, referuojamame *Clarivate Analytics Web of Science* duomenų bazėje, turintis citavimo indeksą 1,524, vienas straipsnis mokslo žurnale, referuojamame *Clarivate Analytics Web of Science* duomenų bazėje, vienas straipsnis mokslo žurnale, referuojamame *Index Copernicus* duomenų bazėje, trys straipsniai tarptautinių konferencijų medžiagose, referuojamose *Clarivate Analytics Web of Science Proceedings* duomenų bazėje, vienas straipsnis konferencijos medžiagoje, referuojamoje kitose duomenų bazėse. Disertacijoje atliktų tyrimų rezultatai buvo pristatyti devyniose mokslinėse konferencijose Lietuvoje ir užsienyje.

Disertaciją sudaro įžanga, trys skyriai, bendrosios išvados, literatūros šaltinių sąrašas ir trys priedai.

Abstract

The visible spectrum image analysis enables intelligent systems to derive information as human vision. Image analysis based on Convolutional Neural Networks (CNN) is used in many areas. The application of locally executed CNN algorithms in embedded systems still limited due to the high amount of required calculations. Currently, there are useful accelerating units for CNN calculations in embedded systems, but a list of useful and supported CNN elements and methodology for the creation of specialized CNN structures are needed for implementation.

The object of work research is convolutional neural networks for real-time image analysis. The following main aspects of the research object are investigated in the present thesis: implementation in accelerating units and design methodology.

The thesis presents the implementation of convolutional neural networks for image analysis based on the proposed list of CNN elements and the methodology to solve selected tasks of image analysis. There are two application areas of CNN-based image analysis algorithms, which have been presented in this work. The first area is the local future description of the retinal images, and the second one is the evaluation of the road type and conditions using images received from the camera in front of the car.

The main results of the thesis were published in seven scientific publications – three in peer-reviewed scientific journals, four in conference proceedings: one in journal, referenced by the Clarivate Analytics Web of Science database, with a citation index of 1.524, one in journal, referenced in the Clarivate Analytics Web of Science database, without citation index, one in journal referenced in Index Copernicus database, three articles in international conference proceedings, referenced in the Clarivate Analytics Web of Science Proceedings database, one article in conference proceedings, referenced in other databases. The results of the research that have been defined in the dissertation have been presented at nine scientific conferences in Lithuania and abroad.

The dissertation consists of an introduction, three chapters, general conclusions, references, and three annexes.

Žymėjimai

Simboliai

a	–	neigiamos vertės koeficientas (konstanta);
a	–	neigiamos vertės koeficientas (kintamasis);
$\mathbf{A}$	–	visiškai sujungtųjų dirbtinių neuronų sluoksnio įėjimo vektorius;
A_{K}	–	klasifikavimo tikslumas;
A_m	–	įėjimo vektoriaus reikšmė, kurios indeksas m;
$A_{\mathrm{s}i+w,\mathrm{s}j+h,c}$	–	įėjimo masyvo vertė, su indeksu $\mathrm{s}i+w, \mathrm{s}j+h, c$, čia s – filtro lango žingsnis, c – įėjimo masyvo sluoksnių skaičius; h – sąsūkos filtro eilutės indeksas; w – sąsūkos filtro stulpelio indeksas; i – požymių žemėlapio masyvo eilutės indeksas; j – požymių masyvo stulpelio indeksas;
$\mathbf{B}$	–	visiškai sujungtųjų dirbtinių neuronų sluoksnio poslinkio vektorius;
B_n	–	n-ojo neurono poslinkio reikšmė;
C	–	sąsūkos filtro sluoksnių skaičius;
d	–	Euklido atstumas;
E_{K}	–	klasifikavimo klaidingumas;

$f(\cdot)$	–	aktyvavimo funkcija;
$f_{\mathrm{ReLU}}(\cdot)$	–	*ReLU* aktyvavimo funkcija;
$f_{\mathrm{LReLU}}(\cdot)$	–	*LReLU* aktyvavimo funkcija;
$f_{\mathrm{ELU}}(\cdot)$	–	ELU aktyvavimo funkcija;
H	–	sąsūkos filtro eilučių skaičius;
I	–	įėjimo masyvo, pateikiamo į sąsūkos sluoksnį, eilučių skaičius;
J	–	įėjimo masyvo, pateikiamo į sąsūkos sluoksnį, stulpelių skaičius;
l	–	tikslo funkcijos vertė;
m	–	pasirinkta siektino atstumo tarp nepanašių vaizdo iškarpų konstanta;
M	–	įėjimo vektoriaus ilgis;
M_{SA}	–	mokymo metu derinamų kintamųjų skaičius sąsūkos sluoksnyje;
M_{VS}	–	mokymo metu derinamų kintamųjų skaičius visiškai sujungtųjų dirbtinių neuronų sluoksnyje.
N	–	neuronų arba sąsūkos filtrų skaičius;
N_{F}	–	neteisingai klasifikuotų pavyzdžių skaičius;
N_{FN}	–	neteisingai nepriskirtų klasėms vaizdų skaičius;
N_{FP}	–	neteisingai priskirtų klasėms vaizdų skaičius;
N_{GSPV}	–	globaliojo sutelkimo skaičiavimo operacijų skaičius;
N_{LReLU}	–	*LReLU* aktyvavimo funkcijos skaičiavimo operacijų skaičius;
N_{ReLU}	–	*ReLU* aktyvavimo funkcijos skaičiavimo operacijų skaičius;
N_{SA}	–	sąsūkos dirbtinių neuronų sluoksnio skaičiavimo operacijų skaičius;
N_{SPL2}	–	skaičiavimo operacijų skaičius sutelkimui pagal Euklido atstumą;
N_{SPM}	–	sutelkimo pagal maksimumą sluoksnio skaičiavimo operacijų skaičius;
N_{SPV}	–	visiškai sujungtųjų neuronų sluoksnio skaičiavimo operacijų skaičius;
N_{T}	–	teisingai klasifikuotų pavyzdžių skaičius;
N_{TN}	–	teisingai nepriskirtų klasėms vaizdų skaičius;
N_{TP}	–	teisingai priskirtų klasėms vaizdų skaičius;
N_{V}	–	visų klasifikuotų pavyzdžių skaičius;
s	–	sutelkimo lango arba sąsūkos filtro žingsnis;
t	–	tikslas;
W	–	sąsūkos filtro stulpelių skaičius;
$\mathbf{W}$	–	visiškai sujungtųjų dirbtinių neuronų sluoksnio ryšio svorių matrica;

W_{mn} – dirbtinių neuronų ryšio svorių matricos vertė, kurios indeksas mn, čia m – įėjimo indeksas; n – dirbtinio neurono indeksas;

$W_{n,h,w}$ – sąsūkos filtrų masyvo vertė, su n,h,w indeksu, čia n – sąsūkos filtro indeksas; h – sąsūkos filtro eilutės indeksas; w – sąsūkos filtro stulpelio indeksas;

$\mathbf{X}$ – visiškai sujungtųjų dirbtinių neuronų sluoksnio grąžinamas vektorius prieš aktyvavimą;

X_n – n-ojo neurono išėjimo reikšmė prieš aktyvavimą;

$X_{n,i,j}$ – sąsūkos sluoksnio išėjime gaunamo požymių žemėlapio masyvo vertė (prieš aktyvavimo funkciją), su n,i,j indeksais, čia n – sąsūkos filtro indeksas; i – požymių žemėlapio masyvo eilutės indeksas; j – požymių masyvo stulpelio indeksas;

$\mathbf{Y}$ – visiškai sujungtųjų dirbtinių neuronų sluoksnio grąžinamas vektorius po aktyvavimo;

Y_n – n-ojo neurono išėjimo reikšmė po aktyvavimo;

Operatoriai ir funkcijos

$\cdot$ – masyvų sandauga panariui;

$\times$ – matricų sandauga.

Santrumpos

BPGPĮ – bendrosios paskirties ir grafikos procesorinis įrenginys;

BRISK – dvejetainiai patvarieji invariantiški keičiamo dydžio būdingieji taškai (angl. *Binary Robust Invariant Scalable Keypoints*);

CAN – valdiklio tinklas (angl. *Controller Area Network*);

CA WoS – (angl. *Clarivate Analytics Web of Science*);

CPĮ – centrinis procesorinis įrenginys;

DNT – dirbtinių neuronų tinklas;

DRAM – dinaminė atsitiktinės kreipties atmintis (angl. *Dynamic Random-Access Memory*);

FFT – greitoji Furjė transformacija (angl. *Fast Fourier Transform*);

FREAK – greitieji tinklainės būdingieji taškai (angl. *Fast Retina Keypoint*);

GDNT – gilusis dirbtinių neuronų tinklas;

GFLOPS – milijardas slankiojo kablelio operacijų per sekundę (angl. *Giga Floating Point Operations Per Second*);

GPĮ – grafikos procesorinis įrenginys;

HBM – didelio pralaidumo atmintis (angl. *High Bandwidth Memory*);

I^2C	–	sąsaja tarp mikroschemų (angl. *Inter-Integrated Circuit*);
LAN	–	lokaliojo atsako normalizavimas (angl. *Local Response Normalization*);
LPLM	–	lauku programuojama loginė matrica;
MNIST	–	modifikuota Nacionalinio standartų ir technologijos instituto duomenų bazė (angl. *Modified National Institute of Standards and Technology database*);
MSER	–	maksimaliai stabilios ekstremumo sritys (angl. *Maximally Stable Extremal Regions*);
NPĮ	–	neuronų procesorinis įrenginys;
PDK	–	puslaidininkinis duomenų kaupiklis;
PN	–	paketinis normalizavimas (angl. *Batch Normalization*);
ReLU	–	tiesinė detektorinė aktyvavimo funkcija (angl. *Linear Rectified Unit*);
SATA	–	nuoseklioji pažangiosios technologijos įrenginių sąsaja (angl. *Serial Advanced Technology Attachment*);
SIFT	–	keičiamam dydžiui invariantiška požymių transformacija (angl. *Scale-Invariant Feature Transform*);
SDNT	–	sąsūkos dirbtinių neuronų tinklas;
SDNT-VS	–	sąsūkos dirbtinių neuronų tinklas su visiškai sujungtaisiais sluoksniais;
SIMD	–	viena instrukcija, daug duomenų (angl. *Single Instruction, Multiple Data*);
SGN	–	stochastinis gradientinis nuolydis;
SPIG	–	specialiosios paskirties integrinis grandynas;
SRAM	–	statinė atsitiktinės kreipties atmintis (angl. *Static Random-Access Memory*);
SURF	–	pagreitinti patvarieji požymiai (angl. *Speeded Up Robust Features*);
TPĮ	–	tenzorių procesorinis įrenginys;
VS	–	visiškai sujungtieji sluoksniai;
VSDNT	–	visiškai sąsūkos dirbtinių neuronų tinklai;
USB	–	universalioji nuoseklioji sąsaja (angl. *Universal Serial Bus*);
SD	–	atminties kortelės tipas (angl. *Secure Digital*);
TFLOPS	–	trilijonas slankiojo kablelio operacijų per sekundę (angl. *Tera Floating Point Operations Per Second*);
TOPS	–	trilijonas operacijų per sekundę (angl. *Tera Operations Per Second*);
UART	–	universalusis asinchroninis siųstuvas-imtuvas (angl. *Universal Asynchronous Receiver-Transmitter*);
VLIW	–	labai ilgas instrukcijos žodis (angl. *Very Long Instruction Word*).

Turinys

[1] Priedai pateikiami pridėtoje kompaktinėje plokštelėje

Contents

[2] The annexes are supplied in the enclosed compact disc

Įvadas

Problemos formulavimas

Sąsūkos dirbtinių neuronų tinklų (SDNT) įgyvendinimas spartinančiuosiuose įrenginiuose vaizdams analizuoti realiuoju laiku reikalauja priimti daug sprendimų. Pasirenkamas spartinantysis įrenginys SDNT algoritmams vykdyti, SDNT struktūra, mokymo algoritmas ir paruošiamas mokymo, validavimo ir testavimo duomenų rinkinys. Sprendžiamas uždavinys apibrėžia maksimalią vykdymo trukmės ribą ir minimalų rezultato tikslumą. Spartinantieji įrenginiai turi ribotą skaičiavimų spartą ir prieinamą atminties talpą bei gali būti skirti vykdyti tik tam tikras skaičiavimo operacijas. Būtina, užtikrinti, kad SDNT struktūra galėtų būti vykdoma pasirinktame spartinančiajame įrenginyje, o skaičiavimų trukmė neviršytų užduotyje numatytos ribos, tačiau būtų pasiekta reikiama algoritmo veikimo kokybė. Dažnai SDNT struktūra ir parametrai parenkami rankiniu būdu, dėl to neišvengiama klaidų, dėl kurių sumažėja algoritmo efektyvumas. Kuo efektyvesnis algoritmas, tuo mažesnio našumo spartinantieji įrenginiai reikalingi jam įgyvendinti ir mažiau suvartojama elektros energijos. Tai ypač svarbu įterptinėse sistemose, apdorojančiose vaizdinę informaciją taikant algoritmus, grįstus SDNT. Šiuo metu kuriamos metodikos, leidžiančios aprašyti SDNT struktūros parinkimo procesą, bet vis dar trūksta projektavimo metodikų SDNT įgyvendinimui spartinančiuosiuose įtaisuose.

Disertacijoje sprendžiama sąsūkos dirbtinių neuronų tinklų projektavimo problema jų įgyvendinimui spartinančiuosiuose įrenginiuose vaidams analizuoti realiuoju laiku. Darbe iškelta hipotezė, kad galima sudaryti sąsūkos dirbtinių neuronų tinklų elementų sąrašą ir sukurti projektavimo metodiką, kuriuos naudojant galima kurti specializuotus sąsūkos dirbtinių neuronų tinklus įgyvendinti spartinančiuosiuose įtaisuose vaizdams apdoroti realiuoju laiku.

Darbo aktualumas

Šiuo metu didėja sąsūkos dirbtinių neuronų tinklų (SDNT) pagrindu sukurtų vaizdų analizės algoritmų taikymo įterptinėse sistemose paklausa. Yra sukurti spartinantys įrenginiai, leidžiantys realiuoju laiku atlikti didelės apimties SDNT skaičiavimus. Daugelis SDNT tyrimų yra skirti tikslumui didinti, mažai nagrinėjamas efektyvumo užtikrinimas energijos suvartojimo požiūriu, neatsižvelgiama į konkrečių spartinančiųjų įtaisų charakteristikas ir apribojimus. Dėl to dažnai skaičiavimo resursai naudojami neefektyviai, ir suvartojama daugiau energijos, tenka naudoti našesnę, brangesnę, didesnių matmenų įrangą. Sudarytas SDNT elementų sąrašas leidžia įvertinti SDNT skaičiavimų apimtį ir užtikrinti SDNT suderinamumą su spartinančiaisiais įtaisais. Taikant sukurtą projektavimo metodiką, galima iš dalies automatizuoti projektavimą, taip paspartinti projektavimo procesą, išvengti dažniausiai pasitaikančių klaidų. Metodikai patikrinti sukurti du, SDNT grįsti, vaizdo analizės algoritmai. Pirmasis yra lokaliojo požymių aprašymo algoritmas, kuris specializuotas akies tinklainės vaizdams sutapdinti. Šis algoritmas aktualus atliekant oftalmologinius akies tinklainės tyrimus. Antrasis – algoritmas kelio dangos tipui ir būklei nustatyti. Šis algoritmas aktualus automobilių saugumui ir komfortui užtikrinti.

Tyrimų objektas

Tyrimų objektas yra sąsūkos dirbtinių neuronų tinklai vaizdams analizuoti realiuoju laiku. Disertacijoje tiriami šie, su tiriamuoju objektų susiję dalykai: įgyvendinimas spartinančiuosiuose įrenginiuose ir projektavimo metodika.

Darbo tikslas

Sukurti ir pritaikyti projektavimo metodiką sąsūkos dirbtinių neuronų tinklams įgyvendinti spartinančiuosiuose įrenginiuose vaizdams analizuoti realiuoju laiku.

Darbo uždaviniai

Darbo tikslui pasiekti iškelti šie uždaviniai:

1. Ištirti sąsūkos dirbtinių neuronų tinklų (SDNT) struktūros elementų ir parametrų įtaką algoritmų tikslumui ir jų mokymo bei vykdymo spartinančiuosiuose įrenginiuose spartai.
2. Sudaryti SDNT elementų sąrašą, iš kurio galima parinkti elementus projektuojant specializuotą SDNT, skirtą įgyvendinti spartinančiuosiuose įrenginiuose.
3. Sukurti SDNT projektavimo metodiką pasirinktai vaizdų analizės užduočiai spręsti, kai žinomi minimalūs reikalavimai greitaveikai ir algoritmą vykdančio spartinančiojo įrenginio techninės charakteristikos.
4. Taikant sudarytą metodiką, sukurti ir spartinančiuosiuose įrenginiuose įgyvendinti SDNT, skirtus šiems vaizdų analizės uždaviniams spręsti:
   - akies tinklainės vaizdų požymiams aprašyti;
   - kelio dangos tipui ir būklei nustatyti.

Tyrimų metodika

Darbe taikomos skaitmeninio vaizdų apdorojimo, dirbtinių neuronų tinklų, giliojo mokymo, statistinės analizės teorijos. Pritaikyti ir įgyvendinti vaizdų pirminio apdorojimo ir papildymo, sąsūkos dirbtinių neuronų tinklų (SDNT) mokymo ir vykdymo, SDNT spartos ir tikslumo vertinimo, vaizdų požymių sutapdinimo tikslumo vertinimo ir sunkiai išmokstamų mokymo rinkinio pavyzdžių atrankos ir naudojimo SDNT mokyti metodai.

Eksperimentams surinkti ir sudaryti specializuoti vaizdų rinkiniai, skirti mokyti SDNT akies tinklainės vaizdų požymiams aprašyti, mokyti ir tikrinti SDNT kelio tipui ir būklei nustatyti. SDNT buvo tiriami ir įgyvendinami taikant įvairią programinę įrangą, daugiausia naudojant DIGITS, *Tensorflow* ir *Matlab*. SDNT įgyvendinami taikant centrinį procesorinį įrenginį *Intel i7-6900K* ir įvairius spartinančiuosius įtaisus: grafikos procesorinius įrenginius (GPĮ) *Nvidia Geforce GTX 960* ir *Nvidia Geforce GTX 1080 Ti*, tenzorių procesorinį įrenginį *Google Coral Edge*, įterptinę sistemą su GPĮ *Nvidia Jetson TX2*.

Darbo mokslinis naujumas

1. Sudarytas sąsūkos dirbtinių neuronų tinklų (SDNT) elementų sąrašas, kurio elementai tinkami įgyvendinti spartinančiuosiuose įrenginiuose, o pateiktos skaičiavimo operacijų apimties formulės leidžia įvertinti elementų ir jų parametrų įtaką skaičiavimo spartai.
2. Sukurta projektavimo metodika SDNT, kuri leidžia iš dalies automatizuoti specializuotų SDNT struktūros ir parametrų parinkimą atsižvelgiant į vaizdo analizės užduoties reikalavimus ir spartinančiojo įrenginio apribojimus.
3. Sukurtas naujas, SDNT grįstas, lokaliojo požymių aprašymo algoritmas, specializuotas akies tinklainės vaizdams sutapdinti, įgyvendintas taikant grafikos procesorinį įrenginį.
4. Sukurtas naujas, SDNT grįstas, kelio dangos tipo ir būklės nustatymo, atliekant vaizdų analizę realiuoju laiku, algoritmas, įgyvendintas įterptinėje sistemoje su grafikos procesoriniu įrenginiu *Nvidia Jetson TX2*.

Darbo rezultatų praktinė reikšmė

Pasiūlytas sąsūkos dirbtinių neuronų tinklų (SDNT) elementų sąrašas leidžia atlikti SDNT tinklų struktūros elementų ir parametrų parinkimą atsižvelgiant į skaičiavimų apimtį ir įvertinant vykdymo trukmę konkrečiame spartinančiajame įrenginyje.

Sukurto lokaliojo požymių aprašymo algoritmo pritaikomumas akies tinklainės nuotraukoms sutapdinti. Vaizdų sutapdinimas palengvina oftalmologinius tyrimus, kai būtina palyginti akies tinklainės pokyčius, įvertinant skirtumus nuotraukose, darytose skirtingų paciento vizitų metu. Be to, sukurtas algoritmas gali būti pritaikomas kitiems medicininiams vaizdams analizuoti, pavyzdžiui, baltymų dėmėms dvimačiuose elektroforezės gelių vaizduose sutapdinti.

Algoritmo, nustatančio kelio dangos tipą ir būklę realiuoju laiku iš vaizdo, pritaikomumas kelių transporto priemonių saugumui ir komfortui pagerinti per aplinkos suvokimą. Taip pat yra galimybė patobulinti šį algoritmą atstumui iki objektų kelio vaizde nustatyti, naudojant binokuliarinės regos modelį.

Gautus rezultatus ir surinktus vaizdus planuojama naudoti tęsiant SDNT grįstų algoritmų vaizdams analizuoti mokslinius tyrimus ir taikant moksliniuose projektuose.

Ginamieji teiginiai

1. Taikant sudarytą sąsūkos dirbtinių neuronų tinklų (SDNT) elementų sąrašą ir pasiūlytą projektavimo metodiką, galima SDNT įgyvendinti spartinančiuosiuose įrenginiuose vaizdams analizuoti realiuoju laiku.
2. Taikant sukurtą metodiką, įgyvendintas vaizdų lokaliojo požymių aprašymo algoritmas, grįstas SDNT, leidžia sutapdinti FIRE duomenų rinkinio požymių poras panašiai kaip tradiciniai požymių aprašymo algoritmai.
3. Taikant sukurtą metodiką, įgyvendintas algoritmas, iš pavienių vaizdo kadrų nustato kelio dangos tipą ir būklę pasiekiant ne mažesnį nei 84 % tikslumą.
4. Taikant sukurtą metodiką, įterptinėje sistemoje su grafikos posisteme *Nvidia Jetson TX2* įgyvendintas algoritmas kelio dangos tipui ir būklei nustatyti vieną kadrą apdoroja greičiau nei per 30 ms.

Darbo rezultatų aprobavimas

Disertacijos tema yra publikuoti 7 moksliniai straipsniai: vienas – mokslo žurnale, įtrauktame į *Clarivate Analytics Web of Science* (*CA WoS*) duomenų bazę, su citavimo indeksu 1,524 (Žuraulis, Surblys, Šabanovič 2019); vienas – mokslo žurnale, įtrauktame į *CA WoS* duomenų bazę (Serackis *et al*. 2017), vienas – mokslo žurnale, įtrauktame į *Index Copernicus* duomenų bazę (Šabanovič, Matuzevičius, Vaitkevičius 2016); trys – recenzuojamose tarptautinių konferencijų medžiagose, įtrauktose į *CA WoS Proceedings* duomenų bazę (Šabanovič, Matuzevičius 2015; Šabanovič, Matuzevičius 2017; Šabanovič, Stankevičius, Matuzevičius 2018), vienas – kitoje tarptautinės konferencijos medžiagoje (Šabanovič, Matuzevičius, Serackis 2016).

Disertacijoje atliktų tyrimų rezultatai buvo pristatyti devyniose mokslinėse konferencijose Lietuvoje ir užsienyje:

- Tarptautinėje konferencijoje „Advances in Information, Electronic and Electrical Engineering“ 2015, 2017 m. Rygoje, Latvijoje ir 2018 m. Vilniuje;
- Jaunųjų mokslininkų konferencijoje „Mokslas – Lietuvos ateitis. Elektronika ir elektrotechnika“ 2016 m. Vilniuje;
- Tarptautinėje konferencijoje „Biomedical Engineering“ 2016 m. Kaune;
- Tarptautinėje konferencijoje „Duomenų analizės metodai programų sistemoms“ 2016 m. Druskininkuose;

- Tarptautinėje konferencijoje „Electrical, Electronic and Information Sciences“ 2018 m., Vilniuje;
- Tarptautinėje konferencijoje „Vision Zero For Sustainable Road Safety in Baltic Sea Region“ 2018 m., Vilniuje;
- Tarptautinėje konferencijoje „Advanced Microwave Devices and Systems“ 2018 m., Vilniuje.

Disertacijos struktūra

Disertaciją sudaro įvadas, trys skyriai ir bendrosios išvados. Taip pat yra trys priedai. Darbo apimtis yra 117 puslapių, neskaitant priedų, tekste panaudota 18 numeruotų formulių, 23 paveikslai ir 6 lentelės. Rašant disertaciją, buvo panaudoti 182 literatūros šaltiniai.

Padėka

Noriu padėkoti darbo vadovui doc. dr. Daliui Matuzevičiui už pasiūlytą darbo temą, galimybę tęsti studijas doktorantūroje, kantrybę, motyvaciją, pagalbą ir naudingas rekomendacijas atliekant tyrimus. Taip pat dėkoju Vilniaus Gedimino technikos universiteto Elektronikos fakulteto ir Elektroninių sistemų katedros bei Transporto ir logistikos kompetencijos centro darbuotojams ir kolegoms už naudingus patarimus rengiant disertaciją. Esu labai dėkingas darbo recenzentams ir redaktorei už pastabas ir rekomendacijas, kurios leido pagerinti disertacijos kokybę. Labai dėkoju savo šeimai, artimiesiems, draugams ir kolegoms už skatinimą, paramą ir supratingumą.

1

Vaizdų analizės, grįstos sąsūkos dirbtinių neuronų tinklais, apžvalga

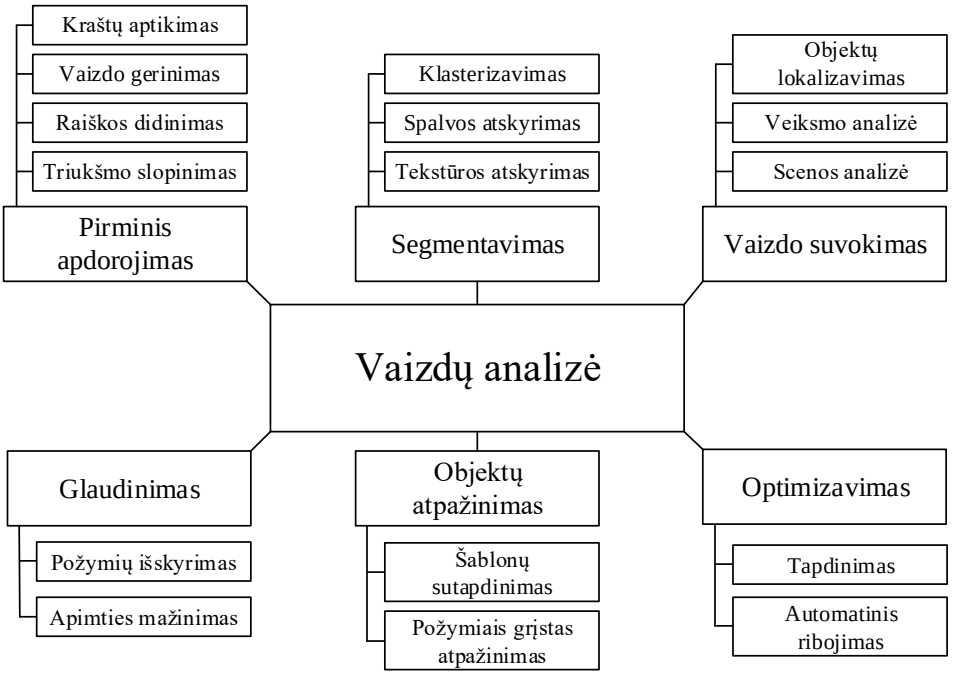
Šiame skyriuje apžvelgta vaizdų analizė, grįsta sąsūkos dirbtinių neuronų tinklais (SDNT), SDNT struktūrų evoliucija, mokymo algoritmai, mokymo ir taikymo aparatinė ir programinė įranga. Skyrius suskirstytas į 4 poskyrius, paskutiniame pateiktas apžvalgos skyriaus apibendrinimas.

Apžvalgoje minimi tradiciniai ir SDNT grįsti vaizdų analizės algoritmai viešai skelbti mokslinėse konferencijose ir mokslinėse publikacijose (Šabanovič, Matuzevičius 2015; Šabanovič, Matuzevičius, Serackis 2016; Serackis *et al*. 2017; Šabanovič, Matuzevičius 2017; Šabanovič, Stankevičius, Matuzevičius 2018).

1.1. Vaizdų analizė, grįsta sąsūkos dirbtinių neuronų tinklais

Dirbtinės regos sistemos yra viena iš plačiausiai studijuojamų ir tiriamų sričių, pritraukianti daug dėmesio paskutiniuosius du dešimtmečius. Be to, vis daugiau dėmesio sulaukia Vaizdų analizės užduotis galima išskaidyti į kategorijas: pirminį

apdorojimą, glaudinimą, segmentavimą, objektų atpažinimą, vaizdų suvokimą ir optimizavimą (1.1 pav.).



1.1 pav. Vaizdų analizės uždavinių kategorijos ir uždaviniai
Fig. 1.1. Categories and tasks of image analysis

Naudojant stereoskopinės regos modelį ir sukalibravus kameras naudojant genetinį algoritmą, galima kurti trimačius paviršiaus atvaizdus ir trimačius erdvės modelius, prieš tai neturint jokios pradinės informacijos apie aplinką (Zhou, Du ir Tang 2011). Taip pat požymių detektoriai, tokie kaip *Hessian-Affine*, *Harris-Affine* (Mikolajczyk, Schmid 2002) ir MSER (Matas *et al*. 2002), yra naudojami trimačiams objektams atpažinti iš dviejų dvimačių vaizdų, gautų skirtingu kampu, nepriklausomai nuo kampo, mastelio (pasikeitusio atstumo iki objekto), apšvietimo, suliejimo, perdengimo ir vaizdų kokybės (Hsu 2011). Po to išrinkti požymių taškai apdorojami taikant SIFT algoritmą (Lowe 2004).

Biologinės regos veikimo savybes bandoma realizuoti ir vaizdo jutikliuose. Žmogaus akis sugeba prisitaikyti gerai matyti šviesias ir tamsias vietas regos lauke. Pritaikius tokį principą vaizdo jutiklyje, kiekvienas jautrusis elementas arba jų grupės gali individualiai adaptuoti jautrumo šviesai lygį. Pasiektas rezultatas, kai kiekvienas jutiklio taškas gali turėti individualų jautrumą šviesai. Taip pat

vaizdui filmuoti naudojamos stereoskopinės kameros, kurios sustatomos viena kitos atžvilgiu kaip žmogaus akys ir leidžia išnaudoti stereoskopinio matymo galimybę vaizdo gyliui nustatyti. Stereoskopinių vaizdų analizei nustatant gylį taikomi skaitmeninių signalų apdorojimo įrenginiai (SSAĮ) ir lauku programuojamos loginės matricos (LPLM) (Kogler *et al*. 2011).

Yra tirtos objektų orientacijos sekimo sistemos, grįstos principu, kad objekto orientacija erdvėje gali būti nustatoma pagal objekto geometrijos pokytį jam sukantis. Pavyzdžiui, apskritimas pasisukęs virsta elipse. Žinant objekto formą ir stebint jos pokytį judant, galima nustatyti, kaip tas objektas yra pasisukęs trimatėje erdvėje (Canessa, Gibaldi 2012).

Vaizdams filtruoti ir transformacijoms atlikti mokslininkai pritaiko sudėtingus dirbtinių neuronų tinklus (DNT) (Ogawa 2012). Juos moko atlikti konkrečią sudėtingą transformaciją teikdami įėjimo ir išėjimo vaizdų poras. Mokytas tinklas naudojamas tiek transformacijai tiesiogine kryptimi (angl. *feed-forward*), tiek ir atgaliniam klaidos sklidimui atlikti.

Kuriant objektų aptikimo sistemą, remiamasi žmogaus regos sistemos modeliais, aprašytais psichofiziologų, psichologų ir neurobiologų straipsniuose (Kweon, Kim 2017). Juose nustatyta, kad žmonių regos sistema leidžia įsiminti trimačius objektų vaizdus taip, kad žmogus, turėdamas daikto atvaizdą atmintyje, žino, kaip jį tinkamai paimti ir laikyti rankoje. Taip pat žmogus įsimena objekto išvaizdą nuo smulkių detalių iki bendresnių detalių, ir atvirkščiai. Yra naudojami keli efektyvūs vaizdo būdingųjų požymių išskyrimo būdai, kurie inspiruoti žmogaus regos sistemos:

1. Hierarchinis dėmesys yra būdingas žmogaus regos sistemai (Treisman 1998). Skiriami erdvinis, požymių ir trimačio objekto hierarchiniai dėmesio lygiai. Erdvinį dėmesį geriausiai atitinka Hariso kampų algoritmas (Harris, Stephens 1988), požymių dėmesį atitinka lokaliųjų Zernikio momentų algoritmas (Bigorgne, Achard, Devars 2000), o trimačio objekto dėmesio lygmenį atitinka procesas nuo abstrakcijos iki detalių.

2. Požymių susiejimas yra taikomas erdvinio dėmesio atskirų požymių žemėlapiams susieti (Treisman 1998). Kraštų žemėlapis susiejamas su kampų žemėlapiu naudojant nuo detalių iki abstrakcijos procesą. Gradientų orientacijos žemėlapis sujungiamas su gradientų amplitudžių žemėlapiu.

3. Kontrastas tarp taško ir jo aplinkos yra svarbi vaizdinė informacija žmogaus regos sistemoje (VanRullen 2003).

4. Dydžio sugretinimas yra taikomas atpažįstant objektą grindžiantis erdvine apimtimi, o ne erdviniu dažnumu (Fiser *et al*. 2001).

5. Dalimis grįstas vaizdo suvokimas yra grįstas atskirų objekto dalių atpažinimu (Biederman 1987). Tai leidžia patikimiau atpažinti objektus, kurie yra iš dalies matomi.

Remiantis šiais principais, yra sukurta nemažai dirbtinės regos modelių. Tyrėjai, remdamiesi išvardytais žmogaus regos modeliais, pasiūlė trimačių objektų atpažinimo būdą (Kweon, Kim 2017).

Žmogaus regos modeliai taip pat buvo pritaikyti skaitmeninių vaizdų raiškos didinimui atlikti (Song *et al*. 2007). Remiantis žinoma žmogaus akies tinklainės savybe atlikti pradinį vaizdų apdorojimą prieš perduodant signalą į smegenis, sudarytas raiškos didinimo algoritmas. Pagal algoritmą, vaizde aptinkami kraštai ir kampai, o jais remiantis suformuojamas didesnės raiškos vaizdas. Šis algoritmas yra spartesnis nei bitiesinis ir kubinis interpoliavimas, o gaunamas rezultatas geresnis.

Dirbtinių neuronų tinklai (DNT) yra matematinės funkcijos, modeliuojančios biologinio neurono veikimą. DNT dažniausiai sudaro įėjimo, paslėptieji ir išėjimo sluoksniai. Sąsūkos dirbtinių neuronų tinklai (SDNT) yra biologinės regos inspiruotas giliųjų dirbtinių neuronų tinklų (GDNT) tipas. Sąsūkos dirbtinių neuronų tinklai (SDNT) plačiai naudojami vaizdų analizei atlikti. SDNT naudojami tokioms vaizdų apdorojimo operacijoms atlikti:

- trimačių vaizdų rekonstrukcijai (Dou, Shah, Kakadiaris 2017);
- vaizdo būdingiesiems bruožams išskirti (Hertel *et al*. 2015; Ng, Yang, Davis 2015);
- vaizdams segmentuoti (Shelhamer, Long, Darrell 2017);
- objektams atpažinti ir klasifikuoti (LeCun *et. al*. 1998, Krizhevsky, Sutskever, Hinton 2012);
- vaizde esantiems objektams atskirti ir nustatyti (Eitel *et al*. 2015);
- vaizdui suvokti (Yuan, Mou, Lu 2015);
- vaizdui aprašyti (Nugraha, Anditya, Suyanto 2019);
- vaizdų gylio žemėlapiams sudaryti (Afifi, Hellwick 2016; Smolyanskiy, Kamenev, Birchfield 2018);
- padėties ir orientacijos pokyčiui nustatyti iš vaizdų srauto (Smolyanskiy *et al*. 2017);
- veiksmui atpažinti iš vaizdų srauto (Yao *et al*. 2015);
- atsakyti į klausimus, susijusius su vaizdu (Noh, Seo, Han 2015).

SDNT gali būti taikomi įvairiausioms vaizdų analizės užduotims spręsti. Tačiau kiekvienai užduočiai atlikti reikia skirtingų SDNT. Dažnai yra pritaikomi esamos SDNT struktūros arba projektuojamos naujos, tačiau trūksta bendros metodikos SDNT projektuoti pasirinktam uždaviniui.

Nagrinėjant SDNT taikymą vaizdų analizės užduotims spręsti pastebėta, kad dažnai šie algoritmai naudojami vaizdų būdingųjų požymių taškams aprašyti (Yi *et al*. 2016; Simo-Serra *et al*. 2015; Rosten, Porter, Drummond 2010;

Dosovitskiy *et al*. 2016), kurie gali būti lyginami su tradiciniais algoritmais SIFT (Lowe 2004), SURF (Bay *et al*. 2008), BRISK (Leutenegger, Chli, Siegwart 2011), FREAK (Alahi, Ortiz, Vandergheynst 2012), KAZE (Alcantarilla, Artoli, Davison 2012), kurie nenaudoja SDNT. Viena iš sričių, kurioje požymių deskriptoriai yra reikalingi, tai medicininių vaizdų požymių taškų sutapdinimas, siekiant palyginti skirtingus vaizdus. Sutapdinimas gali būti taikomas dvimačių elektroforezės gelių vaizdams (Serackis *et al*. 2017) ir akies tinklainės vaizdams (Abramoff, Garvin, Sonka 2010; Hernandez-Matas *et al*. 2017). SDNT grįstų vaizdo požymių aprašymo algoritmų taikymas akies tinklainės vaizdams dar nėra tirtas, todėl tai būtų tinkama užduotis išbandyti SDNT projektavimo metodiką.

Šiuo metu plačiai naudojamos įvairios pažangiosios vairuotojo pagalbos sistemos (Krasner, Katz 2016; Chandran *et al*. 2014) ir savivaldžių automobilių sistemos (Yang et al. 2018; Mahmud, Arafat, Zuhori 2012; Milz *et al*. 2018), kurios gerina automobilių saugumą ir komfortą surinkdamos aplinkos informaciją taikant įvairius jutiklius. Vaizdų analizė leidžia nustatyti prasto matomumo, orų ir apšvietimo sąlygas (Gimonet, Cord, Saint Pierre 2015; Cheng, Zheng, Murase 2018), rasti kelią ir aptikti kliūtis (Nadav ir Katz 2016), atpažinti kelio juostas ir kelkraščius (Yang et al. 2018; Hamme, Veelaert, Philips 2013; Zhang, Wu 2009). Tačiau tarp vaizdų analizės naudojimo sričių trūksta kelio dangos tipo ir būklės nustatymo. Kelio dangos tipas ir būklė, nustatomi iš automobilio priekyje sumontuotos kameros, leistų iš anksto dar prieš automobiliui užvažiuojant ant konkrečios kelio dangos, žinant jos tipą ir būklę, pasirinkti saugius automobilio stabdžių sistemos parametrus ir pakabos parametrus, geriausiai tinkančius dangos tipui, kuriuo bus važiuojama. SDNT dar nebuvo taikomi kelio dangos tipui ir būklei nustatyti, todėl tai būtų tinkama užduotis išbandyti SDNT projektavimo metodiką.

Šiame poskyryje apžvelgta vaizdų analizė, grįsta sąsūkos dirbtinių neuronų tinklais. Vaizdų analizė susieta su dirbtinės regos sistemomis ir jų tyrimais. Žmogaus regos sistemos modeliai paskatino naujų vaizdų analizės algoritmų kūrimą. Vienas iš pažangiausių ir plačiai taikomų vaizdų analizėje algoritmų yra sąsūkos dirbtinių neuronų tinklai. Nustatyta, kad trūksta šių tinklų projektavimo konkrečiai užduočiai metodikos ir būtų naudinga ją sukurti. Pasirinktos dvi vaizdų analizės užduotys, tinkamos sukurtai SDNT projektavimo metodikai išbandyti, projektuojant specializuotus SDNT vaizdų analizei realiuoju laiku, įgyvendinamus spartinančiuosiuose įrenginiuose.

Kuriant SDNT projektavimo metodiką, būtina suprasti, kaip veikia SDNT tinklai, kaip keitėsi jų struktūros ir naudojami elementai. Siekiant SDNT įgyvendinti spartinančiuosiuose įrenginiuose vaizdams analizuoti realiuoju laiku, būtina išsiaiškinti, kokia programinė įranga ir spartinantieji įrenginiai gali būti naudojami, bei kokią spartą ir energijos sąnaudų efektyvumą jie užtikrina. Šie dalykai apžvelgiami ir nagrinėjami tolesniuose poskyriuose.

1.2. Sąsūkos dirbtinių neuronų tinklai

Matematikai nuo seno visus procesus bando aprašyti matematinėmis formulėmis ir algoritmais. Per XX a. sukurta itin naši skaitmeninė skaičiavimo technika, grįsta dvejetaine logika. Šios technikos našumas lenkia žmogaus mąstymą greičiu, tačiau lankstumo prasme gerokai atsilieka. Taip yra dėl to, kad vykdomos tik anksčiau užprogramuotos operacijos.

Gyvų būtybių smegenų lankstumo raktas yra jų struktūros mažiausioji dalelė – neuronas. Šie neuronai jungiasi daugybe jungčių į neuronų tinklus, kurie sudaro smegenis. Pirmąjį neurono modelį 1943 m. paskelbė Varenas Makulokas ir Valteris Pitsas (Rojas 1996; *Artificial neural network* 2015). 1958 m. Frankas Rosenblatas sukūrė kelių sluoksnių perceptroną (Rosenblatt 1957). 1959 m. Hubelis ir Vieselis atrado, kad ląstelės gyvūnų smegenų žievės regos srityje yra atsakingos už šviesos aptikimą akių matymo laukuose (Hubel, Wiesel 1968). DNT tyrimai buvo sulėtėję po 1969 m. Minskio ir Paperto tyrimo (Minsky, Papert 1969). Jie nustatė, kad vieno sluoksnio neuronų tinklai negali atlikti išskirtinio ARBA loginės operacijos, o didelių matmenų DNT skaičiavimams vykdyti buvo nepakankamas to laikotarpio kompiuterių skaičiavimų našumas. Tyrimai buvo sulėtėję iki 1974 m., kai Verbosas savo disertacijoje pasiūlė DNT mokyti taikant atgalinio klaidų sklidimo algoritmą (Werbos 1974), kuris leido įgyvendinti Rosenblato pasiūlytus daugiasluoksnius DNT. 1986 m. Rumelhartas ir Maklilandas aprašė lygiagrečiųjų paskirstytųjų skaičiavimų principo taikymą neuronų procesams modeliuoti (Rumelhart, McClelland 1986). Tai leido sujungti daugybės kompiuterių pajėgumą ir sudėtingesnių DNT skaičiavimams vykdyti.

1980 m. Fukushima, remdamasis Hubelio ir Vieselio darbu, pasiūlė neurokognitroną, kuris laikomas giliųjų dirbtinių neuronų tinklų (GDNT) pirmtaku (Fukushima 1980). 1990 m. ir 1998 m. Lekunas ir bendraautoriai publikavo darbus, kurie padėjo pamatus sąsūkos dirbtinių neuronų tinklams (SDNT) (LeCun *et al*. 1990; LeCun *et al*. 1998). Šiuose darbuose aprašomas SDNT pavadintas *LeNet-5*, skirtas ranka rašytiems skaičiams atpažinti ir tekstui skaitmeninti. Aprašytas daugiasluoksnis tinklas, o mokymui naudojamas atgalinio klaidos sklidimo algoritmas. Šis tinklas skyrėsi nuo įprastų daugiasluoksnių DNT tuo, kad įėjime buvo pateikiami ne požymiai, o vaizdai. Vaizdams apdoroti taikyti sąsūkos sluoksniai, kurie reikiamus vaizdų požymius išskirti išmoksta mokymo proceso metu. Taip pat tais pačiais metais Žiangas ir bendraautoriai pristatė vaizdo poslinkiui invariantišką DNT, pavadintą poslinkiui invariantišku dirbtinių neuronų tinklu (angl. *Shift-invariant Artificial Neural Network*), ranka rašytiems ar spausdintiems skaičiams vaizduose atpažinti, net jei jie pasukti iki 90 laipsnių (Zhang *et al*. 1990).

Šiuo metu vadinamasis gilusis mokymas sulaukė daug investicijų ir mokslinių tyrimų (Jones 2014; *Machine Learning* 2015). GDNT gali būti naudojami klasifikavimo, grupavimo, regresijos, anomalijų aptikimo, sąryšių nustatymo, optimalių sprendimų priėmimo, prognozavimo, valdymo, vaizdų, rašybos, teksto, kalbos atpažinimo uždaviniams spręsti (Jones 2014; *Machine Learning* 2015; *Machine learning* 2019; Morgan 2014; Coates *et al*. 2013; Krizhevsky, Sutskever, Hinton 2012; Yadan *et al*. 2013). Rengiamuose vaizdų apdorojimo konkursuose dominuoja SDNT grįsti algoritmai (Sermanet *et al*. 2013).

Sąsūkos dirbtinių neuronų tinklai (SDNT) – tai tiesioginio sklidimo DNT su sąsūkos sluoksniais, kuriuose atskiri neuronai apdoroja įvesties vaizdo masyvo reikšmes slenkančiu langu, apskaičiuojant lange esančių verčių sandaugų su sąsūkos filtrų vertėmis sumą. SDNT yra inspiruoti biologinių procesų ir yra daugiasluoksnio perceptrono variacija, naudojanti daug mažiau skaičiavimo resursų bei leidžianti iš vaizdo išgauti dvimačius arba trimačius požymius. SDNT naudojami kompiuterinės regos, automatinio kalbos atpažinimo ir natūralios kalbos apdorojimo, garso atpažinimo ir bioinformatikos srityse. Čia jos pasiekia itin gerų rezultatų.

1.2.1. Sąsūkos dirbtinių neuronų tinklų evoliucija

Sąsūkos dirbtinių neuronų tinklai (SDNT) turi daugiau nei 20 metų evoliucijos istoriją – nuo pirmojo SDNT pasiūlyto ranka rašytiems skaičiams klasifikuoti *LeNet-5* (LeCun *et al*. 1998) iki dabartinių, ypač sudėtingų arba efektyvių tinklo architektūrų. Motyvaciją tobulinti SDNT architektūras suteikė tokie atvirieji vaizdų rinkiniai ir viešieji klasifikavimo ir objektų atpažinimo tikslumo konkursai, kaip *ImageNet* (*ImageNet* 2016). Pradedant *Alexnet* (Krizhevsky, Sutskever, Hinton 2012) kiekvienais metais *ImageNet* vaizdų klasifikavimo konkurso rezultatų lentelės aukščiausioje pozicijoje pagal tikslumo metriką pasirodydavo vis nauji, SDNT grįsti algoritmai. Nors iš pradžių daugelis tinklų pasiekdavo didesnį tikslumą tiesiog dėl didesnės tinklo apimties, atsirado esminių tinklo struktūros inovacijų, kurios leisdavo sumažinti mokymo metu derinamų kintamųjų skaičių, SNDT algoritmų vykdymo trukmę ir permokymo tikimybę bei padidinti maksimalų tikslumą. Šiame skirsnyje apžvelgtos inovatyvios tinklų struktūros – nuo pirmųjų iki naujausių.

LeNet-5 (1998) – seniausia 7 sluoksnių sąsūkos neuronų tinklo architektūra (LeCun *et al*. 1998), panaudota ranka rašytiems skaičiams klasifikuoti. Tinklui mokyti, validuoti ir testuoti naudotas vaizdų rinkinys vadinamas *Modified National Institute of Standards and Technology database* (MNIST). Įėjime pateikiamas skaitmeninis nespalvotas 32×32 pikselių dydžio vaizdas. Visi neuronų sluoksniai naudoja hiperbolinio tangento aktyvavimo funkcijas. Šį SDNT sudaro:

sąsūkos (angl. *convolution*), sutelkimo pagal vidurkį (angl. *average pooling*), visiškai sujungtųjų neuronų sluoksniai. Mokymui taikytas stochastinio gradientinio nuolydžio (SGN) algoritmas, įgyvendintas skaičiavimus atliekant centriniu procesoriniu įrenginiu.

Alexnet (2012) – tai tinklas, *ImageNet* duomenų bazės vaizdų klasifikavimo konkurse 2012 m. pelnęs reikšmingai geresnį rezultatą, nei prieš tai jame dalyvavę ne DNT grįsti sprendimai. Šis SDNT klaidingai klasifikavo tik 15,3 % pavyzdžių (Krizhevsky, Sutskever, Hinton 2012). Tinklo architektūra nuo *LeNet-5* skiriasi nežymiai, tačiau tinklą sudaro daugiau neuronų. Siekiant efektyviau išnaudoti du grafikos procesorinius įrenginius (GPĮ), tinklo operacijos padalytos į dvi grupes ir vykdomos lygiagrečiai, o jų rezultatai susiejami visiškai sujungtųjų (VS) neuronų sluoksniu. Šiame SDNT aktyvavimui naudojama detektorinė linijinė (angl. *Rectified Linear Unit – ReLU*) funkcija, kuri palygti su sigmoidine (angl. *sigmoid*) ir hiperbolinio tangento funkcijomis yra paprastesnė ir spartesnė. *Alexnet* tinklu įrodyta, kad *ReLU* aktyvavimo funkcija nesumažina klasifikavimo tikslumo. *Alexnet* pasižymėjo naudojamais didesniais sąsūkos filtrais. SDNT mokymo metu vaizdai buvo papildomi (angl. *augmented*) iškraipymais. Šį SDNT sudaro sąsūkos, sutelkimo pagal maksimumą (angl. *max pooling*), lokaliojo atsako normalizavimo (LAN), visiškai sujungtųjų (VS) dirbtinių neuronų ir išmetimo (angl. *dropout*) sluoksniai. Mokymo metu taikytas SGN algoritmas su momentu.

ZFNet (2013) – tai tinklo architektūra, laimėjusi *ImageNet Large Scale Visual Recognition Competition* (ILSVRC) 2013 m. (Zeiler, Fergus 2014). Šis tinklas klaidingai klasifikuoja tik 14,8 % pavyzdžių. Iš esmės šis tinklas yra panašios į *Alexnet* struktūros, bet be išskyrimo į dvi grupes ir tiksliau parinktomis hiperparametrų reikšmėmis. Šio tinklo pasiektas rezultatas parodo, kad SDNT rezultatas gali būti pagerinamas tiksliau parenkant struktūros parametrus.

Inception (2014) – tai SDNT, laimėjęs ILSVRC 2014 konkursą (Szegedy *et al*. 2015), klaidingai klasifikuoti tik 6,67 % pavyzdžių. Šiame tinkle pirmą kartą pritaikytas naujas struktūros blokas, pavadintas *Inception*. Šio bloko esmė – lygiagrečiai naudojant skirtingo dydžio sąsūkos filtrus atliekamos sąsūkos, po kurių gauti to paties dydžio masyvų sluoksniai sujungiami į bendrą masyvą. Toks jungimas leidžia atrinkti įvairaus dydžio ir sudėtingumo vaizdo požymius tame pačiame bloke. Taikant kelis tokius blokus vieną po kito, kitas blokas įėjime gauna išsamesnę informaciją apie vaizdo požymius. Tinkle pritaikytas paketinis normalizavimas (PN) (angl. *batch normalization*). SDNT mokymo metu vaizdai buvo papildomi iškraipymais. Naudotas *RMSprop* mokymo algoritmas, kuris geriau tinka itin sudėtingai šio tinklo architektūrai, kuriai sąlygas sudarė *Inception* modulinė struktūra. Mokymo metu tinklas išveda atsakymą trijose skirtingose vietose, tai leidžia išvengti silpstančio klaidos gradiento efekto mokymo metu. Ši architektūra įdomi tuo, kad nors joje yra 22 sluoksniai, o palyginimui *Alexnet* – 12 sluoksnių, tačiau jo mokymo metu derinamų kintamųjų skaičius sumažintas

nuo 60 mln. iki 4 mln. Atliekant *ImageNet* rinkinio klasifikavimą, *Inception* klaidingai klasifikuoja daugiau nei dvigubai mažiau pavyzdžių. Vėliau *Inception* blokas patobulintas taip, kad vietoje lygiagrečiųjų šakų su didesnių filtrų sąsūkos sluoksniu būtų naudojami 3×3 dydžio nuosekliai sujungti sąsūkos sluoksniai. Toks tinklo variantas įrodė, kad vienas po kito einantys sąsūkos sluoksniai su mažo dydžio sąsūkos filtrais gali tiksliau ir sparčiau atlikti užduotį, nei vienas sluoksnis su dideliais sąsūkos filtrais. Šis tinklas pirmasis priartėjo prie tuo metu tik pradėtų tirti šio rinkinio žmogaus daromų klaidų lygmens (5,1 %) ir žmonių grupės daromų klaidų lygmens (3,6 %).

VGGNet (2014) – tai antrąją vietą ILSVRC 2014 konkurse užėmęs SDNT (Simonyan, Zisserman 2015). Jis sudarytas iš 16 sąsūkos sluoksnių ir pasižymi struktūros vientisumu. Kaip ir *Alexnet*, jame naudojamos tik 3×3 dydžio filtrų sąsūkos, tačiau su daugiau filtrų, nei iki to laiko naudota. Šis tinklas mokytas 2–3 savaites naudojant 4 grafikos procesorinius įrenginius (GPĮ). Šiuo metu tai yra viena iš dažniausiai naudojamų SDNT struktūrų vaizdo požymiams aprašyti. Šio SDNT mokymo metu derinama 138 mln. kintamųjų, dėl to jis yra laikomas sudėtingai mokomu, nes reikalauja itin didelių mokymo vaizdų rinkinių, tokių kaip *ImageNet*, kuriame yra apie 14 milijonų vaizdų. Šis tinklas turi daug modifikacijų, turinčių skirtingą sluoksnių skaičių.

ResNet (2015) – tai tinklas (He *et al*. 2016), laimėjęs ILSVRC 2015 konkurse. Tai viena iš itin sudėtingų SDNT architektūrų, naudojančių blokinę liekamosios vertės tinklo (angl. *residual neural network*) struktūrą. Šioje struktūroje duomenys, pateikti įėjime, sujungiami su duomenimis, gaunamais bloko išėjime. Toks sujungimas leidžia mokyti tinklus, kurie turi daug sluoksnių išvengiant klaidos gradiento nykimo reiškinio. *ResNet* originalus tinklas sudarytas net iš 152 sluoksnių, tačiau jo sudėtingumas yra žemesnis nei *VGGNet*. *ResNet* blokus galima sugrupuoti pagal jų hiperparametrus į keturias konfigūracijų grupes po 3. Šio tipo tinklas *ImageNet* rinkinyje klaidingai klasifikuoja tik 3,57 % pavyzdžių, tai pranoksta žmogaus ir žmonių grupės tikslumo lygmenį.

Xception (2016) – tai tinklas, kuriame pirmą kartą panaudota gyliu padalyta sąsūkos operacija (Chollet 2016). Gyliu padalyta sąsūka – tai sąsūkos operacijos variantas, kai iš pradžių atliekamos sąsūkos operacijos su plokščiu 2×2 ar didesniais dvimačiais sąsūkos filtrais kiekviename įėjimo masyvo sluoksnyje, o po to atliekama 1×1 sąsūka, kuri apdoroja visų sluoksnių vaizdo duomenis, esančius tose pačiose įėjimo masyvo sluoksnio koordinatėse, tai yra vadinama gilumine sąsūka. Palyginimui, vykdant įprastą sąsūką įėjimo masyvo duomenų sąsūka atliekama ne su plokščiu filtru, o su trimačiu 2×2×N, čia N yra įėjimo sluoksnių skaičius. Atlikus abi sąsūkos operacijas. gaunami vienodo dydžio masyvai, tačiau sąsūkos filtrų verčių skaičius ir skaičiavimo operacijų apimtis atliekant gyliu padalytą sąsūką yra beveik tiek kartų mažesnė, kiek įėjimo duomenų masyvas turi

sluoksnių. Dėl tokio mokymo metu derinamų kintamųjų ir operacijų skaičiaus sumažėjimo gyliu padalyta sąsūka leidžia padidinti mokymo ir vykdymo spartą.

ResNeXt-50 (2017) – lyginant šio tinklo struktūrą (Xie *et al*. 2016) su *Resnet-50* yra pridėtos lygiagrečios sąsūkos operacijos šakos kiekviename modulyje, panašiai kaip *Inception*. Tokia struktūra leido sujungti geriausias *Inception* ir *ResNet* tinklų savybes.

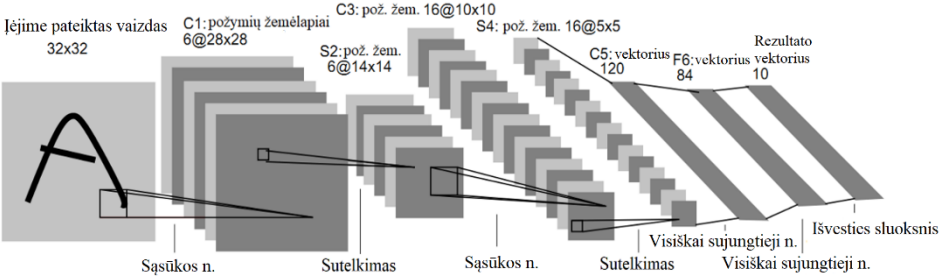
DenseNet (2016) – tai tinklo architektūra (Huang *et al*. 2016), kuri užtikrina ankstesnių sąsūkos operacijų duomenų prieinamumą kitiems sluoksniams. Taigi kiekvieno iš sąsūkos sluoksnių požymių žemėlapiai yra sujungiami su tolesnių sluoksnių įėjimo požymių žemėlapiais taikant matricų sujungimo (angl. *concatenation*) funkciją. Autoriai teigia, kad požymių žemėlapių, išmoktų skirtinguose sluoksniuose, sujungimas leidžia padidinti kitų sluoksnių įėjimo duomenų variaciją ir pagerinti efektyvumą. Jos privalumas – galimybė išlaikyti gana aukštą tikslumą naudojant itin mažus sąsūkos filtrų kiekius, nes kiekviename sluoksnyje prieinama informacija gauta panaudojus ankstesniuose sluoksniuose panaudotus sąsūkos filtrus, bendras mokymo metu derinamų kintamųjų skaičius gerokai sumažėja. Lyginant su *ResNet*, *DenseNet* pasiekia geresnių rezultatų naudojant mažiau mokymo metu derinamų kintamųjų. Paties taikomojo tinklo architektūra primena *ResNet*, joje *Dense* blokai išdėstyti grupėmis, ir tik tose grupėse užtikrinamos minėtos jungtys tarp sluoksnių.

EfficientNet (2019) – 2019 m. *ArXiV* paskelbtas Tano ir Le straipsnis (Tan, Le 2019), kuriame aprašytas efektyvus tinklo struktūros parinkimas, parenkant santykį tarp tinklo gylio (sluoksnių skaičiaus), pločio (filtrų skaičiaus) ir raiškos (filtrų dydžio). Jų pasiūlyto sprendimo rezultatas – *EfficientNet* SDNT, kuris pasiekia aukštą tikslumą klasifikuojant *ImageNet* vaizdus esant itin mažam mokymo metu derinamų kintamųjų skaičiui.

Apžvelgus SDNT struktūrų evoliuciją, pastebima tinklų gylio didėjimo tendencija (Gu *et al*. 2018). 2015 m. pristatytas *ResNet* yra 20 kartų gilesnis lyginant su *Alexnet*. DNT gylio didėjimas paaiškinamas tuo, kad gilesnis tinklas geba tiksliau aproksimuoti funkciją tarp įėjimo duomenų ir reikalaujamų rezultatų, išmokti sudėtingesnius vaizdo požymius. Kita vertus, didesni tinklai linkę į persimokymą, daugelyje SDNT struktūrų daug dėmesio skiriama mokymo algoritmų ir specialių reguliarizavimo elementų naudojimui apsaugai nuo persimokymo.

1.2.2. Sąsūkos dirbtinių neuronų tinklų struktūros elementai

Sąsūkos dirbtinių neuronų tinklai (SDNT) sudaryti iš sąsūkos, sutelkimo (angl. *pooling*) arba atrinkimo (angl. *subsampling*), visiškai sujungtųjų (VS) neuronų, saviorganizuojančių požymių žemėlapių (angl. *self-organizing map*) ar kitokio tipo sluoksnių (1.2 pav.).



1.2 pav. Sąsūkos dirbtinių neuronų tinklo struktūra (pritaikyta iš LeCun *et al*. 1998)
Fig. 1.2. Structure of Convolutional Neural Network (adapted from LeCun *et al*. 1998)

Sąsūkos sluoksniuose per įėjimo duomenų matricą yra slenkamas pasirinkto dydžio $N \times M$ filtro langas su tam tikromis vertėmis žingsniu n per stulpelius ir m per eilutes. Kiekvienas filtras išėjime suformuoja požymių žemėlapį, kurio dydis $(N/n) \times (M/m)$ taškų. Naujai gauti požymių žemėlapiai apdorojami sutelkimo ar atrinkimo sluoksniuose. Paprasčiausi sutelkimo sluoksnių variantai – tai lango verčių vidurkis, mediana, minimumas, maksimumas ar kitokia funkcija. Gaunamas sutelktas požymių žemėlapis, kuris gali būti apdorojamas kitu sąsūkos sluoksniu.

Kai požymių žemėlapių masyvų dydis pakankamai sumažėja, jie gali būti pateikiami į VS neuronų sluoksnį. Šie sluoksniai atlieka galutinę algoritmo užduotį, pavyzdžiui, klasifikavimą.

1.2.3. Sąsūkos dirbtinių neuronų tinklų mokymo algoritmai

Sąsūkos dirbtinių neuronų tinklams (SDNT) mokyti taikomi įvairūs giliojo mokymo algoritmai, kurie yra atgalinio klaidos sklidimo algoritmo variantai. SDNT gali būti mokomi taikant mokymo algoritmus su mokytoju ir be mokytojo (Sathya, Abraham 2013). Pavyzdžiui, klasifikavimo uždaviniui galima taikyti mokymą su mokytoju. Tokiu atveju, mokymo metu tinklui pateikiami įvesties vaizdas ir klasė, kuriai jis turi būti priskiriamas, todėl, sudarant duomenų rinkinį, būtina pateikti ir siektinus apdorojimo rezultatus. Pavyzdžiui, klasterizavimui galima taikyti mokymą be mokytojo, tada iš anksto numatomas tik klasifikacijos kategorijų skaičius ir tinklas pats atlieka mokymo duomenų klasterizavimą į numatytą dalių skaičių. Mokymas be mokytojo sutaupo laiką pradinių duomenų paruošimo etape, tačiau ne visoms užduotims tinka. Giliojo mokymo algoritmai keičia tradicinius požymių paieškos ir aprašymo algoritmus dėl didesnio efektyvumo ir galimybės taikyti mokymą be mokytojo ir mokymą su pastiprinimu.

Giliųjų dirbtinių neuronų tinklų (GDNT) ryšio svoriams keisti naudojamas atgalinio skleidimo algoritmas (Werbos 1974) ir įvairios jo modifikacijos.

Paprasčiausias yra gradientinio nuolydžio algoritmas. Tačiau tinklams didėjant ir mokymo rinkiniams, tokiems kaip *ImageNet* (*ImageNet* 2016), perkopiant šimtus tūkstančių pavyzdžių apimtį buvo pradėti taikyti stochastiniai gradientinio nuolydžio ir kiti mokymo algoritmai. Toliau apžvelgti įvairūs GDNT mokymo algoritmai ir jų ypatybės.

Gradientinio nuolydžio algoritmas yra vienas pirmųjų tinklo svorių parinkimo algoritmų, taikytų atgaliniam sklidimui (LeCun *et al*. 1998). Ieškant GDNT tikslo funkcijos lokaliojo minimumo atliekami žingsniai, priešingi funkcijos gradientui dabartiniame taške. Dar šis algoritmas vadinamas stačiausiojo nuolydžio algoritmu. Taikant šį algoritmą, gradientinis nuolydis skaičiuojamas visam duomenų rinkiniui, o stochastinio gradientinio nuolydžio (SGN) – tik vienam pavyzdžiui arba nedideliam pavyzdžių paketui (angl. *mini-batch*), kuris lengvai telpa procesorinio įrenginio atmintyje (Bottou 2010). SGN algoritmas yra ypač naudingas tais atvejais, kai mokymas vyksta pateikiant į tinklą duomenis, gaunamus realiuoju laiku. Vaizdų paketų apdorojimu grįstas SGN algoritmas yra dažniausiai naudojamas GDNT mokymo metu, kadangi dideli mokymo rinkiniai nebetelpa į procesorinių įrenginių atmintį, tačiau keliolikos pavyzdžių apdorojimas lygiagrečiai leidžia paspartinti skaičiavimus ir padidinti atgaliniam klaidos sklidimui naudojamos tikslo funkcijos stabilumą tiek kartų, kiek pavyzdžių yra apdorojamame pakete.

Stochastinio gradientinio nuolydžio su momentu algoritmas – tai GDNT mokymo algoritmas su nuolydžio momento skaičiavimu (Qian 1999). DNT ryšio svorių atnaujinimas vyksta taikant akumuliuotą klaidos reikšmę. Yra kelios šio algoritmo atmainos: tankioji ir retoji. Pagal tankiąją versiją, akumuliuota reikšmė atnaujinama ir taikoma nepriklausomai nuo gradiento vertės, o pagal retąją – atnaujinama tik ta akumuliuotų verčių masyvo dalis, kuri naudota tiesioginio skleidimo metu. Dažnai naudojama Nesterovo momento algoritmo versija (Sutskever *et al*. 2013), kuri veikia greičiau nei įprasta versija.

Adadelta algoritmas yra adaptyviojo mokymosi koeficiento algoritmas (Zeiler *et al.* 2012). Šis algoritmas dinamiškai adaptuojasi pagal pirmosios eilės išvestines ir reikalauja nedaug papildomų skaičiavimų lyginant su SGN algoritmu. Šis algoritmas adaptyviai parenka mokymo koeficientą, be to, yra atsparus triukšmingiems gradientams, skirtingoms modelio struktūroms ir duomenų specifikai bei hiperparametrų parinkimui.

Adagrad algoritmas realizuoja adaptyviuosius subgradientinius metodus interaktyviajam GDNT mokymui ir stochastiniam optimizavimui (Duchi, Hazan, Singer 2011). Šie algoritmai dinamiškai taiko žinias apie apdorojamų duomenų geometriją ankstyvosiose iteracijose. Tai yra proksimalus algoritmas išgaubtoms funkcijoms minimizuoti.

Adagrad dvigubo vidurkinimo (*AdagradDA*) (McMahan 2017) algoritmas yra skirtas retiesiems tiesiniams modeliams. Šis algoritmas pasirūpina nematytų

požymių reguliarizavimu rinkinyje atnaujinant juos tada, kai jie atitinka uždaros formos atnaujinimo taisyklę, kuri ekvivalenti jų atnaujinimui tik gavus naują rinkinį. *AdagradDA* įprastai naudojamas, kai yra didelis mokomo modelio retumas. Šis algoritmas garantuoja retumą tik tiesiniams modeliams. Jis reikalauja įdėmaus gradientų akumuliatorių inicializavimo mokymui paleisti.

ADAM yra gradientu grįstas algoritmas, skirtas optimizuoti stochastines tikslo funkcijas su adaptyviaisiais žemesnės eilės įverčiais (Kingma, Ba 2015). Algoritmo skaičiavimai yra efektyvūs ir reikalauja mažai atminties resursų, be to, jis pasižymi invariantiškumu įstrižajam gradientų mastelio kitimui ir gerai tinka užduotims, turinčioms daug parametrų ar duomenų, spręsti. Šis algoritmas tinkamas nestacionariosioms tikslo funkcijoms ir užduotims su triukšmingais ir retais gradientais.

FTLR-Proksimalinis (McMahan 2011) mokymo algoritmas, pasižymintis puikiomis retumo (angl. *sparsity*) ir konvergavimo savybėmis bei turintis atskirai nustatomus mokymosi koeficientus kiekvienai koordinatei.

Proksimalinis stačiausio nuolydžio algoritmas (Duchi ir Singer 2009) yra paprastas ir efektyvus tikslo funkcijos vertei mažinti rinkinio mastu ir interaktyviajam SDNT mokymui. Tai dviejų pakopų algoritmas, kuris leidžia retą sprendimą, kai naudojamas kartu su reguliarizavimo funkcijomis, kurios skatina retumą, tokios kaip Manheteno atstumo metrika. Ši metrika atstumą nustato pagal absoliučių skirtumų sumą. Taip pat šis algoritmas gerai veikia ir su kitais reguliarizavimo metodais.

Proksimalinis *Adagrad* algoritmas yra panašus į proksimalinį stačiausio gradiento algoritmą ir pristatytas tame pačiame straipsnyje (Duchi ir Singer 2009).

RMSprop algoritmas (Tieleman ir Hinton 2012), kuris yra alternatyva momento algoritmui, jame taikomi atskiri adaptyvieji mokymosi koeficientai kiekvienam parametrui siekiant paspartinti tinklo mokymą naudojant vaizdų paketus. Šio algoritmo esmė yra dalyti mokymosi koeficientą iš ryšio svorio, remiantis vidutine paskutinių šio ryšio svorio gradientų amplitude.

Tyrimų, atliktų doktorantūros metu, išbandyti SGN, SGN su momentu, *Adadelta*, *Adagrad*, *ADAM* algoritmai. *Adadelta* ir *Adagrad* buvo naudingi, kai būdavo sunku pasirinkti tinkamą tinklo mokymo koeficientą. SGN su momentu algoritmas, yra realizuotas įvairiose programose SDNT įgyvendinti.

1.3. Sąsūkos dirbtinių neuronų tinklų mokymo ir taikymo įrankiai

Šiuo metu sąsūkos dirbtinių neuronų tinklų (SDNT) algoritmams įgyvendinti naudojami kompiuteriai su centriniu procesoriniu įrenginiu (CPĮ) ir koprocesoriais, bendrosios paskirties ir grafikos procesoriniais įrenginiais (BPGPĮ), skaitmeninio

signalų apdorojimo įrenginiais (SSAĮ), lauku programuojamomis loginėmis matricomis (LPLM) ir iš įvairių įrenginių sudaryti kompiuterių klasteriai. Taip pat yra gaminama SDNT algoritmams įgyvendinti skirta aparatinė įranga – tenzorių procesoriniai įrenginiai (TPĮ), neuronų procesoriniai įrenginiai (NPĮ). Tokia įranga gali tapti kita pakopa (po kompiuterių), kuri užtikrintų skaičiavimų efektyvumą energijos sąnaudų atžvilgiu. Vykdant SDNT algoritmus, aparatinei įrangai valdyti reikalinga operacinė sistema ir tvarkyklės, o SDNT modeliavimui – įvairi specializuota programinė įranga: kompiliatoriai, programų bibliotekos, programavimo aplinkos ir paketai. Šiame skyriuje apžvelgti programiniai SDNT modeliavimo įrankiai ir SDNT algoritmų skaičiavimus vykdanti aparatinė įranga.

1.3.1. Programinė įranga, taikoma dirbtinių neuronų tinklais grįstiems algoritmams kurti

Dirbtinių neuronų tinklams (DNT) modeliuoti skirta aparatinė įranga gali būti naudojama taikant šiai įrangai skirtą programinę įrangą: įvarius DNT modeliavimo programų paketus, bibliotekas ir skirtingas programavimo aplinkas. Šiame skyriuje apžvelgta įvairi programinė įranga vaizdų analizės algoritmų, grįstų sąsūkos dirbtinių neuronų tinklais (SDNT), įgyvendinimui.

Šiuo metu SDNT algoritmams, skirtiems vaizdų analizei, kurti ir vykdyti dažniausiai taikomi šie programų paketai: *Tensorflow*, *PyTorch*, *Matlab*, *Torch 7*, *MXNet*, *Theano*, *Caffe*, DIGITS2, *Weka*. Toliau jie apžvelgti plačiau.

Tensorflow – tai atvirojo kodo platforma sistemoms mokyti, turinti lanksčius įrankius, bibliotekas ir naudotojų bendruomenės sukurtus priedus, kurie leidžia tyrėjams kurti pažangiausius sistemų mokymo algoritmus, tarp jų ir SDNT (Abadi *et al*. 2015). Be to, *Tensorflow* yra viena iš geriausių platformų, SDNT algoritmų perkėlimui į serverius ir įterptines sistemas, palaikanti vykdymą įvairiuose spartinančiuosiuose įrenginiuose.

PyTorch – atvirojo kodo sistemų mokymo platforma, skirta tyrimams ir taikymui (*PyTorch* 2019). Ši platforma sukurta kurti ir vykdyti giliojo mokymo algoritmus naudojant GPĮ ir CPĮ skaičiavimams atlikti. Kaip ir *Tensorflow*, ši platforma turi didelę naudotojų bendruomenę ir kartu su *Tensorflow* yra dažniausiai taikoma SDNT mokymui ir algoritmų vykdymui skirta programinė įranga.

Matlab – viena iš populiariausių aukšto lygio kalbų ir aplinkų, naudojamų visame pasaulyje (*Matlab* 2015). Šios aplinkos įrankių paketai leidžia lengvai įgyvendinti aukštesnio lygio algoritmus, naudojant iš anksto paruoštas žemesniojo lygio funkcijas ir negaištant laiko joms sudaryti. Ši aplinka tinka įvairiems reiškiniais modeliuoti, taip pat ir SDNT algoritmams vykdyti. Tam šioje aplinkoje yra įrankių paketai, skirti sistemoms mokyti (angl. *machine learning*) ir giliajam mokymui (angl. *deep learning*) atlikti, vaizdams analizuoti ir kompiuterinėms regos sistemoms kurti, lygiagretiesiems skaičiavimams ir statistiniams skaičiavimams

vykdyti. Taikant lygiagrečiųjų skaičiavimų įrankių paketą, skaičiavimams spartinti galima naudoti GPĮ. *Matlab* yra mokamas programų paketas, pritaikytas veikti *Windows*, *macOS* ir *Linux* operacinėse sistemose. Yra prieinama atvirojo kodo nemokama alternatyva, veikianti komandinės eilutės režimu, visiškai suderinama su baziniu *Matlab* – *Octave* (*GNU Octave* 2015). *Octave* palaiko standartines *Matlab* funkcijas, tačiau neturi specializuotų įrankių paketų, dėl to *Octave* DNT algoritmus reiktų kurti skaičiavimų su matricomis lygmeniu, tai pareikalautų nemažai laiko.

Torch 7 – tai mokslinių skaičiavimų sistema, kuri siūlo platų sistemų mokymo algoritmų palaikymą (*Torch* 2015; Collobert, Kavukcuoglu, Farabet 2011). Šiuo metu *Torch* taikoma skriptinė programavimo kalba *LuaJIT*. *Torch* turi daugybę programuotojų bendruomenės sukurtų paketų, skirtų sistemų mokymui, kompiuterių regos sistemoms, signalų apdorojimui, lygiagretiesiems skaičiavimams. *Torch* pritaikytas skaičiavimus spartinti naudojant *Nvidia* GPĮ per C/CUDA biblioteką. Tai yra atvirojo kodo projektas, pritaikytas *macOS* ir *Ubuntu 12+* operacinėms sistemoms. Naudoti šį paketą *Windows* operacinėje sistemoje rekomenduojama *Linux* virtualiojoje mašinoje, dėl to prarandama skaičiavimų spartinimo naudojant GPĮ galimybė.

MXNet – tai giliojo mokymo sistema, sukurta lankstumui ir efektyvumui užtikrinti (*Dmlc mxnet...* 2015). Ši sistema leidžia derinti simbolinį programavimą su imperatyviniu pasiekiant didžiausią efektyvumą ir produktyvumą. Sistemos branduolyje veikia dinaminis duomenų srautų tarpusavio priklausomybę nuspėjantis modulis, kuris automatiškai išlygiagretina simbolines ir imperatyvines operacijas. *MXNet* taikomas grafų optimizavimo sluoksnis, kuris paspartina simbolinių skaičiavimų vykdymą ir sumažina naudojamą atminties talpą. Ši biblioteka yra portatyvi, naudojanti nedaug resursų bei pritaikoma kelių GPĮ kompiuteriui ar jų klasteriui. Tai yra atvirojo kodo projektas, suderinamas su *Python*, *R*, *C++* ir *Julia* programavimo kalbomis.

Theano – tai *Python* biblioteka, leidžianti efektyviai aprašyti, optimizuoti ir skaičiuoti matematines išraiškas su daugiamačiais masyvais (*Theano...* 2015). Pagrindinės šios bibliotekos savybės yra glaudi integracija su *NumPy*, GPĮ naudojimas, nereikalaujantis atskiro įgyvendinimo, leidžia pasiekti iki 140 kartų pagreitėjimą lyginant su CPĮ viengubo tikslumo slankiojo kablelio operacijų greičiu, efektyvus simbolinis diferencijavimas, greičio ir stabilumo optimizacijos, dinaminis kodo *C* programavimo kalba generavimas, kodo testavimo ir savi patikros įrankiai. *Theano* naudojamas didelio masto moksliniams skaičiavimams nuo 2007 m. Palaiko *Nvidia* DNT algoritmų biblioteką *cuDNN*.

Caffe – tai giliojo mokymo sistema, sukurta skiriant daug dėmesio SDNT aprašymo aiškumui, algoritmų vykdymo spartai ir lengvai plečiamai programos struktūrai (*Caffe* 2015). Ši sistema sukurta Berklio regos ir mokymosi centro ir entuziastų bendruomenės jėgomis, ir jai taikoma atvirojo kodo licencija. *Caffe*

ypatinga savo ekspresyvia architektūra, išplečiamu kodu, sparta ir didele palaikymo bendruomene. Skaičiavimams galima naudoti tiek CPĮ, tiek GPĮ, tai nustatoma vykdymo metu.

DIGITS *2* – tai *Nvidia* siūloma giliojo mokymo naudojant GPĮ sistema, kuri skirta duomenų mokslininkams ir tyrėjams (Gray 2015). Ši sistema leidžia greitą giliųjų DNT kūrimą duomenims apdoroti naudojant tinklo elgsenos vizualizavimą realiuoju laiku. DIGITS yra sistema, kurioje nereikia rašyti papildomo kodo. Šiuo metu DIGITS platinama nemokamai kompanijos *Nvidia* internetiniame puslapyje. Sistema užtikrina spartos padidėjimą naudojant 2–4 GPĮ lyginant su 1 GPĮ. Sistema leidžia peržiūrėti sukurtų tinklų struktūrą, valdyti kelių DNT lygiagretų mokymą keliuose GPĮ. Sistema yra paprastai paleidžiama ir nustatoma, palaiko daugybę formatų ir duomenų šaltinių, galima stebėti tinklo mokymą realiuoju laiku. Sistema yra atvirojo kodo, todėl esant poreikiui gali būti papildoma.

Weka 3 – tai sistemų mokymo algoritmų kolekcija ir aplinka duomenų apdorojimo užduotims (*Weka 3* 2015). Algoritmai gali būti pritaikomi duomenims tiesiogiai arba iš JAVA kodo. *Weka* suteikia įrankius duomenų pirminiam apdorojimui, klasifikavimui, regresijai, grupavimui, ryšių taisyklėms ir vizualizavimui. Taip pat *Weka* tinka naujoms sistemų mokymo schemoms įgyvendinti. Platinama su GNU GPL licencija. Reikalauja, kad kompiuteryje būtų įdiegta JAVA 1.7 programų vykdymo aplinka.

Šiuo metu plačiausiai naudojamos šios DNT modeliavimo ir vaizdų analizės bibliotekos: *Intel Deep Learning Framework, Vulcan,* CUDA*, OpenCL, PyCUDA, Anaconda Accelerate, cuDNN, OpenVX, OpenCV, BIDMach, Scikit-Learn, BigML.io, Cuda-convnet(2).*

Intel Deep Learning Framework – tai giliojo mokymo sistema, skirta *Intel* kompiuterinėms platformoms (*Intel Deep...* 2015). Sistema užtikrina heterogeniškumą tarp *Intel* CPĮ ir GPĮ bei koprocesorių, našumą, išplėtimo galimybes, lygiagretų klasifikavimą ir mokymą. Sistema labai gerai optimizuota išnaudoti *Intel* produktų skaičiavimų pajėgumą.

Vulcan – tai yra programų biblioteka, sukurta *Khronos* konsorciumo, skirta aukšto efektyvumo programoms įgyvendinti naudojant šiuolaikinių GPĮ grafikos ir skaičiavimų resursus (*Vulkan* 2015). Ši biblioteka programoms suteikia tiesioginį GPĮ valdymą maksimaliam našumui ir nuspėjamumui užtikrinti.

CUDA – tai skaičiavimų platforma, įgyvendinanti lygiagrečius skaičiavimus naudojant *Nvidia* GPĮ (*CUDA Parralel Computing* 2015). Į CUDA įeina įrankių rinkinys CUDA *C/C++* programavimo kalba ir įvairios GPĮ spartinamos funkcijos. Suderinamas su įvairiomis programavimo kalbomis: *C, C++, C#, Fortran,* JAVA*, Python* ir daug kitų. CUDA taikoma įvairių mokslo ir pramonės sričių skaičiavimams spartinti ir procesams modeliuoti naudojant *Nvidia* GPĮ.

OpenCL – tai atviras ir nemokamas standartas, skirtas tarpplatforminiams lygiagretiesiems skaičiavimams skirtingais procesoriais, esančiais asmeniniuose

kompiuteriuose (*OpenCL* 2019). *OpenCL* stipriai pagerina algoritmų vykdymo spartą ir atsako laiką pradedant žaidimais ir pramogomis, baigiant mokslinėmis programomis.

PyCUDA – tai įrankis, skirtas iš *Python* kalbos pasiekti CUDA lygiagrečiųjų skaičiavimų sąsają (Klockner 2015).

Anaconda Accelerate – tai įrankis, skirtas iš *Python* kalbos pasiekti CUDA lygiagrečiųjų skaičiavimų sąsają (*Anaconda...* 2015).

Nvidia cuDNN (Nvidia CUDA Deep Neural Network) – tai GPĮ spartinamų primityvų, skirtų GDNT modeliuoti, biblioteka (*Nvidia CuDNN* 2019). Ši biblioteka turi itin gerai optimizuotas funkcijas, leidžiančias paspartinti DNT mokymą ir taikymą naudojant *Nvidia* GPĮ.

Nvidia TensorRT **–** tai programinė platforma, skirta vykdyti didelio našumo giliojo mokymo algoritmų taikymą. Ši platforma įtraukia specialiai *Nvidia* įrangai SDNT pritaikantį optimizavimo įrankį ir vykdymo aplinką, užtikrinančią žemą vėlinimą ir didelį pralaidumą SDNT grįstiems algoritmams. *TensorRT* veikia CUDA pagrindu, realizuoja *Nvidia* GPĮ skaičiavimus naudojant mišriųjų duomenų tipų kintamuosius. *TensorRT* palaiko kitomis programomis sukurtų ir mokytų SDNT konvertavimą į *Nvidia* GPĮ optimizuotą kodą.

OpenVX – tai atviras, nemokamas standartas, skirtas tarpplatforminiam kompiuterinės regos sistemų aplikacijų spartinimui (*OpenVX* 2019). *OpenVX* leidžia pasiekti našų ir galios požiūriu optimizuotą kompiuterinės regos apdorojimą, o tai yra ypač svarbu įterptinėse ir realiojo laiko sistemose, tokiose kaip veidų, kūno gestų sekimo, išmaniosios vaizdo stebėjimo, pažangiosiose pagalbos vairuotui, objekto ir aplinkos rekonstrukcijos, papildytosios realybės, vizualinės inspekcijos, robotikos ir kitose sistemose.

OpenCV – tai atvira kompiuterių regos ir sistemų mokymo (angl. *machine learning*) biblioteka (*OpenCV* 2019). Ji sudaryta siekiant suteikti bendrą infrastruktūrą kompiuterių regos aplikacijoms ir paspartinti sistemų mokymo komercializavimą. Ši biblioteka pateikiama su atvirojo kodo licencija. Biblioteką sudaro daugiau nei 2500 optimizuotų algoritmų, taip pat klasikinių ir moderniųjų kompiuterinės regos ir sistemų mokymo algoritmų rinkinys. Ši biblioteka turi *C/C++*, *Python*, JAVA ir *Matlab* programines sąsajas, pritaikytas naudoti *Windows*, *Linux*, *Android* ir *MacOS* operacinėse sistemose. *OpenCV* daugiausia taikoma realiojo laiko regos sistemoms. Aktyviai plėtojamos šios bibliotekos sąsajos su GPĮ skaičiavimų bibliotekomis CUDA ir *OpenCL*.

BIDMach – itin greita sistemų mokymo biblioteka, skirta sistemoms mokyti (*BID Data...* 2015). Naudojant bibliotekos branduolio klases, galima dirbti su duomenų šaltiniais, atlikti skaičiavimų ir duomenų paskirstymą tarp CPĮ ir GPĮ procesorių, kurti naujus SDNT modelius, remiantis bibliotekoje pateikiamais pavyzdžiais. Šiuo metu tai vienas iš sparčiausių įrankių sistemų mokymo užduotims

spręsti. *BIDMach* pritaikytas veikti *Windows, Linux* ir *macOS* operacinėse sistemose.

Scikit-Learn – tai *Python* biblioteka, skirta paprastai ir efektyviai apdoroti ir analizuoti duomenis (*scikit-learn...* 2015). Ši biblioteka sukurta *NumPy* ir *SciPy* bei *matplotlib* pagrindu. Platinama su atvirojo kodo licencija.

BigML – tai sistemų mokymo biblioteka, skirta kurti, leisti ir pridėti numatančius modelius į naudotojo projektą (*Single Platform...* 2015). Tinka naudoti mokymui su mokytoju ir be jo. Tai yra įrankis darbui su *BigML* siūloma internetine paslauga, kuri leidžia kurti DNT tiesiog debesyje.

Cuda-convnet(2) – tai sparti DNT įgyvendinimo priemonė pasitelkiant C/CUDA. Leidžia modeliuoti sluoksnių sujungimą ir kurti giliuosius DNT (*cuda-convnet...* 2015). DNT mokymui naudojamas atgalinio skleidimo algoritmas.

Iš aprašytų bibliotekų matoma, kad daugiausia palaikymo SDNT modeliavimui šiuo metu skiria *Nvidia* įmonė, jos GPĮ spartinimą naudoja daugelis bibliotekų ir aplinkų bei sistemų. *Intel* įmonė taip pat siūlo nuosavą programinę įrangą, skirtą SDNT algoritmams vykdyti. AMD GPĮ skaičiavimai vykdomi tik per *OpenCL*, tai stipriai apriboja AMD GPĮ panaudojimą. *Nvidia* ir partneriai bei naudotojų bendruomenė siūlo SDNT tyrimo ir įgyvendinimo įrankius – nuo paties žemiausio iki aukščiausio lygmens.

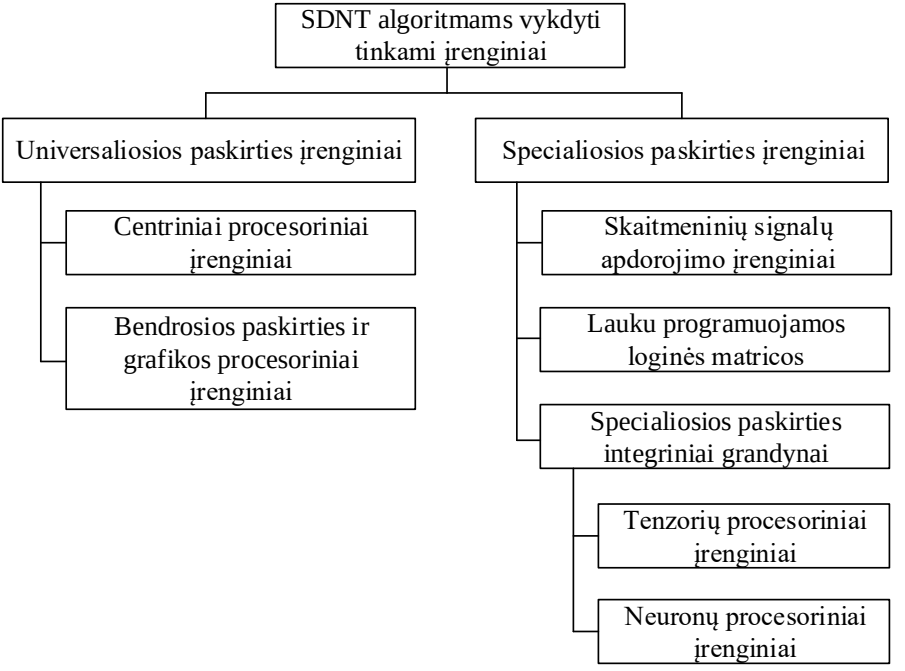
1.3.2. Kompiuterinė technika, tinkama dirbtinių neuronų tinklų algoritmams vykdyti

Lyginant aparatinę sąsūkos dirbtinių neuronų tinklų (SDNT) algoritmų vykdymo įrangą svarbu pabrėžti, kad geriausia įranga nėra pati sparčiausia, naujausia ar brangiausia, o labiausiai tinkanti konkrečiam uždaviniui spręsti. Be to, ne mažiau svarbus sąsūkos dirbtinių neuronų tinklų (SDNT) nesudėtingas įgyvendinimas pasirinktuose spartinančiuosiuose įrenginiuose. Tinkamai įrangai pasirinkti apibrėžiami reikalavimai, pagal kuriuos įranga vertinama atsižvelgiant į jai keliamą užduotį. Pažymėtina, kad skaičiavimo resursus mokymui galima nuomotis, tai gali būti geriau nei pirkti ir išlaikyti nuosavą įrangą, o įgyvendinimui lokaliai įdiegtoje sistemoje gali reikėti specializuotos įrangos.

Įrangai palyginti galima taikyti šiuos kriterijus: kaina, našumas, atminties talpa ir pralaidumas, galios suvartojimas, neuronų tinklų algoritmų įgyvendinimo trukmė ir sudėtingumas, prieinamos nemokamos ar atvirojo kodo bibliotekos ir programinė įranga, galimybė išplėsti skaičiavimų pajėgumą reikiamu momentu, pasinaudojant skaičiavimo resursų nuoma. Įranga, kurią tikimasi naudoti stacionariame ar mobiliame produkte, ir įranga, skirta technologijai tobulinti bei tyrimams atlikti, gali turėti skirtingų tikslų ir reikalavimų. Pažangiųjų SDNT mokymui ir algoritmų vykdymui gali reikėti 11 GB ir didesnės atminties talpos

(Milakov 2014; Dettmers 2018), prieinama GPĮ atminties talpa riboja maksimalų SDNT dydį ir apdorojamų duomenų raišką. Kuo didesni SDNT ir kuo didesni mokymo duomenų rinkiniai ir apdorojamų vaizdų raiška, tuo didesnės talpos ir pralaidumo operatyvios atminties ir GPĮ vaizdo atminties reikia (Dettmers 2018; Dettmers 2019). SDNT algoritmams sparčiai vykdyti svarbi didelė slankiojo kablelio operacijų sparta, o įgyvendinimui produkte – žemi duomenų perdavimo vėlinimai, maža galia ir įrangos kaina.

Sąsūkos dirbtinių neuronų tinklų modeliavimo įrangą galima skirstyti į universaliosios paskirties ir specialiosios paskirties įrangą (1.3 pav.).



1.3 pav. Sąsūkos dirbtinių neuronų tinklų modeliavimo įrangos klasifikavimo schema
Fig. 1.3. Classification scheme of simulation equipment of convolution neural networks

Universaliosios paskirties įranga – tai įranga, kuri atlieka įvairios paskirties skaičiavimus ir turi platų pritaikymo spektrą. Specialiosios paskirties įranga – tai procesoriniai įrenginiai, kurie skirti atlikti tam tikros rūšies skaičiavimus, jiems taikyti reikalingos specialiosios žinios ar kompiliatoriai ir programų konverteriai.

Universaliosios paskirties įrangai priskiriami kompiuteriai su centriniais procesoriniais įrenginiais (CPĮ) ir bendrosios paskirties ir grafikos procesoriniais įrenginiais (BPGPĮ), sutrumpintai vadinamais grafikos procesoriniais įrenginiais (GPĮ).

Centriniai procesoriniai įrenginiai (CPĮ) naudojami pradedant smulkiausiais įrenginiais su mikrovaldikliais ir baigiant didžiuliais superkompiuteriais. Šiuo metu labiausiai paplitusios, žinomiausios ir skaičiavimams atlikti tinkamos architektūros yra *x86_64*, ARM, POWER. Šios architektūros naudojamos 500 naršiausių pasaulio superkompiuterių (*TOP500* 2015). Yra ir kitų architektūrų, tačiau pasirinkus jas gali kilti problemų dėl palaikymo, siauro operacinių sistemų ir kompiliatorių pasirinkimo.

x86_64 architektūros centrinius procesorinius įrenginius (CPĮ) ir koprocesorius siūlo *Intel* ir AMD kompanijos. *Intel* siūlo namų naudotojams skirtus *Intel Core* serijos CPĮ iki 18 branduolių ir serveriams skirtus *Intel Xeon* serijos CPĮ iki 56 branduolių (*Intel Core...* 2019; *Intel Xeon Processor...* 2019). Taip pat *Intel* siūlo *Intel MIC* architektūros *Xeon Phi* koprocesorius, turinčius iki 72 branduolių (*Intel Xeon Phi...* 2019; Shah 2015). AMD siūlo namų naudotojams skirtus A serijos 4 branduolių CPĮ su integruotu bendrosios paskirties grafikos procesoriniu įrenginiu (*AMD A-series...* 2019), iki 8 branduolių FX serijos CPĮ (*AMD FX...* 2019), iki 16 branduolių *Ryzen* serijos procesorius (*AMD Ryzen... 2019*), iki 32 branduolių *Ryzen Threadripper* serijos procesorius (*AMD Ryzen Threadripper...* 2019), serveriams skirtus iki 64 branduolių *Epyc* serijos CPĮ (*AMD Epyc...* 2019).

ARM architektūros centrinius procesorius gamina įvairūs gamintojai. Šie procesoriai orientuoti į mažą galios suvartojimą. Šiuo metu *ARM Cortex-M* naudojami mikrovaldikliuose, *ARM Cortex-R* – realiojo laiko ir saugumo reikalaujančiose sistemose, *ARM Cortex-A* naudojami aplikacijų procesoriams gaminti. Pastarieji naudojami įvairioje elektronikoje: išmaniuosiuose telefonuose, planšetėse, buitinėje ir komunikacijų technikoje, kompiuteriuose ir serveriuose. ARM deda daug pastangų, kad jos procesoriai būtų plačiau taikomi serveriuose. Atliekant DNT modeliavimą ARM architektūros CPĮ naudojami bendrosios paskirties GPĮ valdyti įterptinėse sistemose (*Embedded systems* 2015) arba serveriuose su *Tesla* BPGPĮ (*Machine Learning* 2015). ARM taip pat siūlo ARM ML sistemų mokymo skaičiavimų įrenginį, kuris integruotas sistemose su ARM CPĮ ir GPĮ branduoliais gali paspartinti sistemų mokymo algoritmų vykdymą (*Arm Machine...* 2019).

POWER architektūros procesorius gamina ir savo produktuose naudoja įvairios įmonės, pavyzdžiui: IBM, NXP, *National Instruments*, *Xilinx*, *Sony* (*Power...* 2015). Šie procesoriai sėkmingai naudojami komunikacinėje įrangoje, superkompiuteriuose, tokiuose kaip QSPACE, žaidimų kompiuteriuose *GameCube*, *Wii*, *Wii U*, *Xbox 360*, *Playstation 3*. Kartais žaidimų konsolės naudojamos ir moksliniams skaičiavimams atlikti (Mueler 2007).

Centrinių procesorių universalumas leidžia jų skaičiavimo pajėgumą naudoti SDNT algoritmams vykdyti ir mokyti. Tačiau CPĮ nėra efektyvus SDNT mokymui dėl mažo atminties pralaidumo (iki 100 GB/s) ir nedidelio lygiagrečiųjų skaičiavimų našumo lyginant su BPGPĮ (Rupp 2013). CPĮ gali būti naudojami tik kai

BPGPĮ neturi pakankamai vaizdo atminties, kadangi CPĮ gali turėti daugiau nei 10 kartų didesnę operatyviosios atminties talpą lyginant su didžiausia, rinkoje prieinamų BPGPĮ atminties talpa.

Bendrosios paskirties ir grafikos procesoriniai įrenginiai (BPGPĮ) išsivystė iš grafikos procesorinių įrenginių (GPĮ), kai pikselių ir vektoriniai konvejeriai buvo sujungti į bendrosios paskirties srautinius procesorius. BPGPĮ gali veikti tik kaip koprocesoriai, todėl jie naudojami su CPĮ, kurio spartos turėtų pakakti užduotims ir duomenims į BPGPĮ teikti. Šiuo metu skaičiavimams spartinti naudojami *Nvidia* ir AMD BPGPĮ. Kiti gamintojai gamina tik integruotus grafikos procesorinius įrenginius, skirtus mažai galios vartojančioms sistemoms. Tokie GPĮ, nors nėra spartūs, gali būti naudojami nedideliais SDNT grįstiems algoritmams vykdyti.

GPĮ naudojimas SDNT mokyti ir algoritmams vykdyti yra besiplečianti verslo ir mokslo sritis. Didelę įtaką šios srities plėtimuisi padarė giliojo mokymo SDNT atradimas. BPGPĮ aukštas našumas, didelis atminties pralaidumas ir šiuo metu pasiekiama didelė atminties talpa leidžia naudoti šią įrangą SDNT mokyti ir algoritmams vykdyti (*Machine Learning* 2015; Dettmers 2018).

Naudojant BPGPĮ SDNT skaičiavimams atlikti svarbu pasirinkti tinkamą CPĮ, kadangi jis atlieka šias operacijas (Dettmers 2019): kodo kintamųjų rašymą ir skaitymą, BPGPĮ funkcijų kvietimo inicijavimą, duomenų paketų kūrimą, duomenų perdavimo į ir iš BPGPĮ inicijavimą. Dažniausiai pakanka vieno CPĮ branduolio vienam GPĮ aptarnauti. Tačiau naudotojo sąsajai ir asinchroninėms užduotims vykdyti gali reikėti vienu ar keliais branduoliais daugiau nei sistemoje yra GPĮ. Taip pat svarbu atsižvelgti į procesoriaus *PCI Express* sąsajos linijų skaičių, jei procesorius jungiamas tiesiogiai su GPĮ. Taip pat svarbu, kad *PCI Express* būtų kuo didesnio pralaidumo, todėl ši sąsaja turėtų būti bent 3-iosios kartos, nes daugelis GPĮ vis dar naudoja šios kartos sąsają, arba 4-osios kartos, jei naudojamos GPĮ palaiko šios kartos sąsają. CPĮ taktinis dažnis, naudojant GPĮ SDNT skaičiavimams atlikti, nėra esminis faktorius, nes didžiąją laiko dalį CPĮ praleidžia laukdamas operatyviosios atminties, išorinės atminties ar GPĮ atsako. Didesnis CPĮ branduolių skaičius leistų greičiau paruošti duomenų paketus pateikti į GPĮ, ypač kai būtina atlikti pirminį duomenų apdorojimą.

SDNT modeliavimas keliais GPĮ reikalauja didelio atminties pralaidumo ir talpos. Rekomenduojama, kad operatyviosios atminties talpa kompiuteryje būtų dvigubai didesnė nei suminė GPĮ vaizdo atminties talpa (Dettmers 2018). Geriau, kad kuo daugiau apdorojamų duomenų tilptų į operatyviąją atmintį, nes skaitymas iš išorinės atminties, ypač jei tai standusis diskas, gali stipriai sumažinti apdorojimo spartą. Išoriniai atminčiai taip pat gali būti naudojami puslaidininkiniai duomenų kaupikliai (PDK), jei dirbama su duomenų kiekiais, netelpančiais vienu metu į operatyviąją atmintį (Dettmers 2018). Vėlinimą dėl duomenų perkėlimo į

GPĮ sumažinti galima naudojant asinchroninį duomenų paketų persiuntimą iš operatyviosios atminties į GPĮ vaizdų atmintį, lygiagrečiai duomenų apdorojimui.

Nvidia siūlo tiek GPĮ, tiek ir įterptines sistemas, kuriose kartu su ARM CPĮ yra integruotas *Nvidia* GPĮ. *Nvidia* siūlo mobiliesiems įrenginiams skirtas *Tegra* vieno lusto sistemas (*Tegra* 2015), namų naudotojams skirtus *GeForce* serijos GPĮ (*Geforce*... 2019), profesionaliam naudojimui skirtus *Quadro* GPĮ (*Quadro* 2019), skaičiavimams skirtas *Tesla* BPGPĮ (*Tesla* 2015) bei skaičiavimams įterptinėse sistemose skirtus *Jetson Nano/TX2/AGX* modulius (*Embedded Systems* 2019). Taip pat *Nvidia* siūlo *Grid* technologiją bendrosios paskirties ir grafikos skaičiavimų spartinimui virtualizuoti (*Nvidia GRID* 2019). AMD siūlo namų naudotojams skirtas grafikos procesorių posistemes *Radeon* (*AMD Desktop G*... 2015) ir profesionalams skirtas *FirePro* (*AMD Professional G*... 2015), AMD *x86_64* architektūros procesorių ir grafikos posistemių hibridines sistemas (*AMD A*... 2015), taip pat AMD G serijos integruotas vieno lusto sistemas (*AMD Embedded*... 2015).

Nvidia ir AMD įmonių siūlomų GPĮ, skirtų namų naudotojams ir profesionalams, charakteristikų palyginimas pateiktas 1.1 lentelėje (Martin 2012; *List of Nvidia*... 2015; *List of AMD*... 2015). 1.1 lentelėje surašytos orientacinės kainos, skirtos palyginti įrenginius tarpusavyje, surinktos 2015 metais. Palyginimui praplėsti pateiktos 4 branduolių *Intel i7-4770K* CPĮ, veikiančio 4,7 GHz taktiniu dažniu, ir koprocesoriaus *Intel MIC Xeon Phi 7120X* charakteristikos.

GPĮ lenkia CPĮ viengubo tikslumo operacijų per sekundę skaičiumi, o CPĮ dvigubo tikslumo slankiojo kablelio operacijų skaičius yra vienos eilės su namų naudotojams skirtų *GeForce* ir *Radeon* GPĮ bei profesionalams skirtų *Quadro* (1.1 lentelė). Taip pat *Intel MIC Xeon Phi* koprocesorius turi gerą dvigubo tikslumo slankiojo kablelio operacijų našumą, siekiantį pusę skaičiavimams skirtų GPĮ našumo. *Nvidia* GPĮ yra sukurta daugiau programinės įrangos lyginant su AMD GPĮ, ir didesnė sistemų mokymo specialistų naudoja *Nvidia* GPĮ. Internete prieinama daug aprašymų ir pamokų, susijusių su *Nvidia* GPĮ naudojimu, prie jų kūrimo daug prisideda pati įmonė (Milakov 2014). Aktyvus kompanijos *Nvidia* dalyvavimas nulėmė dažnesnį jos gamintų GPĮ taikymą ir geresnę integraciją su SDNT kūrime taikoma programine įranga.

GPĮ gali būti naudojami bendrosios paskirties skaičiavimams išnaudojant šių skaičiavimo įrenginių lygiagretumo savybę, kurią lemia branduolių skaičius, didesnis nei 1000, ir didelis atminties pralaidumas tarp skaičiavimo įrenginio ir jo atminties, siekiantis 300–1000 GB/s. GPĮ yra pritaikytas vaizdams apdoroti ir yra labai efektyvus vaizdų analizės srityje dėl galimybės lygiagrečiai atlikti operacijas su daugybe duomenų.

1.1 lentelė. Centrinio procesorinio įrenginio ir bendrosios paskirties ir grafikos procesorinių įrenginių parametrų palyginimas
Table 1.1. Specifications comparison of central computing units and general-purpose graphics processing units

Įrenginys / Charakteristika	*GeForce GTX 980 Ti*	*GeForce GTX Titan X*	*Quadro M6000*	*Tesla K80*	*Radeon R9 390X*	*Radeon R9 Fury X*	*FirePro W9100*	*Intel i7-4770K*	*Intel Xeon Phi 7120X*
Viengubo tikslumo slankiojo kablelio operacijų sparta, GFLOPS	5632	6144	6070	8740	5913	8601	5237	580	-
Dvigubo tikslumo slankiojo kablelio operacijų sparta, GFLOPS	176	192	190	2330	739	537	2618	300	1208
Atminties pralaidumas, GB/s	336	336	317	480	384	512	320	~34	325
Maksimali atminties talpa, GB	6	12	12	24	8	4	16	32	16
Nominalioji galia, W	250	250	250	300	275	275	275	~110	300
Kaina, Eur	~750	~1000	~4000	~4000	~500	~750	~3000	~400	~4000

Pastaba. Pateiktos orientacinės kainos, tikrintos 2015 metais.

Atliekant SDNT mokymą, nėra svarbu, ar GPĮ skirta namų, ar profesionaliam naudojimui. Iš aprašytų BPGPĮ DNT mokymui tinkamiausi yra tie įrenginiai, kurie užtikrina didelį atminties pralaidumą ir talpą, didelę viengubo tikslumo slankiojo kablelio arba sveikųjų skaičių operacijų spartą. Akivaizdaus spartos skirtumo modeliuojant SDNT profesionaliosios serijos *Quadro* ir naudotojams skirtos serijos *Geforce* GPĮ nėra, o kaina yra apie 4 kartus mažesnė (1.1 lentelė), tačiau *Geforce* GPĮ vaizdo atminties talpa yra mažesnė nei *Quadro* GPĮ. Tokiu atveju reiktų rinktis tą GPĮ, kuri spartesnė ir pigesnė, tačiau užtikrina reikiamą užduočiai vykdyti vaizdų atminties talpą.

Tesla ir *FirePro* BPGPĮ pasirinkimas SDNT priklauso nuo to, ar bus naudojamos dvigubo tikslumo slankiojo kablelio operacijos ir ar reikalingas atminties su klaidų taisymo kodu palaikymas. Skaičiavimams skirtos BPGPĮ turi nuo kelių iki keliolikos kartų didesnį dvigubo tikslumo slankiojo kablelio našumą (1.1 lentelė), tačiau vykdant SDNT naudojamas „viengubo tikslumo“ 32, 16 skilčių slankiojo kablelio arba 8 skilčių sveikųjų skaičių kintamieji. Namų naudotojams skirtos *Radeon* užtikrina didesnę dvigubo tikslumo slankiojo kablelio operacijų spartą nei *GeForce*. Šis skirtumas atsirado dėl skirtingos GPĮ architektūros. *Maxwell* kartos *GeForce* GPĮ buvo sukurti specialiai žaidimams, kuriuose daugiausia naudojamos viengubo tikslumo slankiojo kablelio operacijos, tuo metu *Radeon* turėjo architektūrą, gerai pritaikytą skaičiavimams atlikti.

Atsižvelgiant į skirsnyje išdėstytas mintis, galima sukonfigūruoti darbo stotį, skirtą mokslininkui ar tyrėjui, užsiimančiam SDNT tyrimais. Gera SDNT mokymo ir algoritmų vykdymo darbo stotis turėtų turėti daugelio branduolių CPĮ su didelės talpos operatyviąja atmintimi ir keliomis GPĮ. Procesorius turėtų palaikyti didelės talpos operatyviosios atminties modulius ir turėti didelį skaičių *PCI Express* sąsajos linijų GPĮ prijungti. Procesoriaus, atitinkančio šiuos reikalavimus pavyzdžiai yra *Intel i7, i9* ir *Xeon* serijų procesoriai. Taip pat itin svarbu užtikrinti maksimalų darbo stoties pagrindinės plokštės magistralių tarp CPĮ ir atminties, bei CPĮ ir GPĮ pralaidumą, todėl galėtų būti pasirinktos tokios dalys:

- *Intel i9-9900X* 8 branduolių CPĮ, palaikantis daugiau nei 40 *PCI Express* sąsajos linijų;
- pagrindinė plokštė, veikianti *Intel X299* mikroschemų rinkinio pagrindu, palaikanti pasirinktą procesoriaus modelį ir 8 atminties modulius;
- aštuoni *DDR4* tipo 16 GB talpos operatyviosios atminties moduliai (iš viso 128 GB);
- 2 *Nvidia RTX Titan* GPĮ su NV-LINK jungtimi, kuri naudojama duomenims tarp GPĮ perduoti;
- 1 TB talpos *nVME* tipo PDK operacinei sistemai ir mokymo duomenims laikyti;
- du 10 TB ar didesnės talpos standžiųjų diskų duomenų rinkiniams saugoti, užtikrinant duomenų pasiekiamumą RAID 1 tipo masyvu;

- 1000 W *80 PLUS Titanium* efektyvumo sertifikatą turintis maitinimo blokas, užtikrinantis maitinimo sistemos efektyvumą;
- didelis korpusas su aušintuvais ir radiatoriais kokybiškam nuolat veikiančios sistemos aušinimui;
- nepertraukiamo maitinimo šaltinis, skirtas apsaugoti nuo netikėto sistemos išsijungimo elektros tiekimo sutrikimų metu.

Toks kompiuteris neskaitant programinės įrangos atsieitų apie 8000 Eur, pusę šios kainos sudaro dviejų GPĮ kaina. Tokio kompiuterio maksimalios galios sąnaudos galėtų siekti 1000 W. Kelių GPĮ darbo stotys ne visada yra maksimaliai efektyvios, kadangi tam tikras DNT algoritmų dalis sunku išlygiagretinti tarp skirtingų GPĮ, tačiau turint kelias GPĮ visada galima vienu metu modeliuoti kelis DNT, taip sutaupant laiko.

Mobilios sistemos SDNT algoritmams vykdyti (Caulfield 2015; Hill 2015) pavyzdžiu galima laikyti *Nvidia* siūlomą *Jetson TX2* modulį ir jam skirtą plėtros rinkinį (*Embedded systems* 2019). Šį rinkinį sudaro sisteminė plokštė ir modulis su aušinimo sistema. Modulio specifikacijos: keturi 64 skilčių *ARM Cortex-A57* branduolių ir du 64 skilčių *Nvidia Denver 2* branduolių procesorius su 256 branduolių *Nvidia Pascal* architektūros GPĮ, 8 GB LPDDR4 operatyviosios atminties, galimybė prijungti iki 6 kamerų, prijungti ekranus, belaidis IEEE 802.11ac standarto *Wi-Fi* ryšys, 1 Gbit/s *Ethernet* sąsają, vienos ir keturių linijų *PCI Express* 2.0 sąsajas, 32 GB puslaidininkinis duomenų kaupiklis (PDK) prijungtas *eMMC* sąsaja, SATA ir SD sąsajas, integruotų sistemų sąsajas: 3 UART, 3 SPI, 4 I2C, 4 I2S ir bendrosios paskirties išvadus / įvadus. Gamintojas pateikia operacinę sistemą *Linux* pagrindu su visomis reikiamomis tvarkyklėmis ir programinės įrangos plėtros rinkiniu. *Nvidia Jetson TX2* kaina siekia apie ~500 Eur, tačiau akademiniams naudotojams numatyta ~300 Eur kaina. Šio modulio galios sąnaudos yra 5–10 W. Dėl nedidelės kainos ir suvartojamos galios bei nedidelių matmenų šis įrenginys tinka diegti taikymo vietoje.

Specialiosios paskirties įrangai priskiriami skaitmeninių signalų apdorojimo įrenginiai (SSAĮ), lauku programuojamos loginės matricos (LPLM) ir specialiosios paskirties integriniai grandynai (SPIG), tokie kaip tenzorių procesoriniai įrenginiai (TPĮ) ir neuronų procesoriniai įrenginiai (NPĮ).

Skaitmeninių signalų apdorojimo įrenginiai (SSAĮ) yra specialiosios paskirties mikroprocesorius, kurio architektūra pritaikyta signalams apdoroti (*Digital signal...* 2015). SSAĮ tikslas – realaus pasaulio tolydinių analoginių signalų matavimas, filtravimas ir suspaudimas. Nors panašias funkcijas gali atlikti ir bendrosios paskirties mikroprocesoriai, SSAĮ pasižymi didesniu energijos sąnaudų efektyvumu, be to, jų atminties struktūra sukurta vykdyti ilgas instrukcijas ir dirbti iš karto su tam tikro dydžio duomenų paketais.

SSAĮ instrukcijų rinkinyje yra daugybos ir sudėties operacijos, sąsūkos, daugybos, polinomo radimo operacijos su matricomis. Tarp standartinių palaikomų

algoritmų yra baigtinio impulso atsako filtrai ir greitoji Furjė transformacija (angl. *Fast Fourier Transform* – FFT). Taip pat yra naudojami vienos instrukcijos, daugelio duomenų (angl. *Single Instruction, Multiple Data* – SIMD*)*, labai ilgo instrukcijos žodžio (angl. *Very Long Instruction Word* – VLIW*)* instrukcijų rinkiniai ir superskaliarinė architektūra lygiagretinimui įgyvendinti. SSAĮ įprastai jungiamas su ARM ar kitos architektūros mikroprocesoriumi, kuris teiktų užduotis. Dėl SSAĮ struktūros, pritaikytos lygiagrečiai vykdyti daugybę sudėties ir daugybos operacijų, šis įrenginys tinka DNT skaičiavimams atlikti (Lane, Georgiev 2015; Hijazi, Kumar, Rowen 2015).

Operacijų vykdymo spartai padidinti ir laukimo trukmei sumažinti SSAĮ gali būti kombinuojamas su lauku programuojama logine matrica (LPLM), kurią taikant duomenys gali būti greitai pateikiami, o rezultatai paimami iš SSAĮ konvejerių bei be užlaikymo grąžinami į SSAĮ tolesniam apdorojimui (Gupta *et al*. 2015). Dirbant su SSAĮ, kyla problemų dėl ribotos atminties talpos. Be to, dėl kompiliavimo skirtumų tarp skirtingų platformų, kodas, kuris gerai veikė naudojant CPĮ ar GPĮ, gali prastai veikti naudojant SSAĮ (Chai 2014). Dažnai SSAĮ taikomi kalbos atpažinimui atlikti, bet rečiau naudojami vaizdui apdoroti dėl spartos ir maksimalaus lygiagrečiai vykdomų operacijų skaičiaus apribojimų.

Lauku programuojama logikos matrica (LPLM) yra integrinis grandynas, kurio funkciją ir struktūrą galima konfigūruoti jau pagaminus lustą (*Field Programmable*... 2015). LPLM konfigūracija ir programa dažniausiai kuriama aparatinės įrangos aprašymo kalba (angl. *Hardware Description Language* – HDL). LPLM sudaro programuojamos logikos blokų masyvai ir konfigūruojami hierarchiniai ryšiai tarp jų. LPLM loginių operacijų blokai gali būti sujungiami sudėtingoms operacijoms atlikti. Daugelyje LPLM taip pat yra registrai, dinaminės atminties ląstelės arba atminties blokai. LPLM yra lanksčiausias iš nagrinėjamų spartinančiųjų įrenginių, kurį galima taikyti daugelyje sričių, kadangi jo vidinę struktūrą gali keisti vykdoma programa. Patikrintos LPLM konfigūracijos gali būti perkeliamos į SPIG, taip atpiginant galutinio produkto gamybą.

Įprastai LPLM programuojamos taikant HDL, tačiau yra kompiliatoriai, keičiantys kitų programavimo kalbų, tokių kaip *C* ir *Matlab* kodą į LPLM konfigūraciją (Abdu-Aljabar 2012). Pagrindiniai LPLM gamintojai yra *Xilinx* ir *Intel*. Abi įmonės gamina analogiškus produktus ir teikia programavimo bei plėtros aplinkas. LPLM dėl savo ypatingo lankstumo taip pat naudojamas DNT algoritmams vykdyti (Sahin, Beceriki, Yazici 2006; Muthuramalingam, Himavathi, Srinivasan 2007; Granado *et al*. 2006). Pastaruoju metu atlikti tyrimai rodo, kad LPLM gali prilygti kai kurioms GPĮ pagal našumą ir naudoja 10–20 kartų mažiau galios bei gerai tinka nedideliems SDNT vykdyti (Ovtcharov *et al*. 2015; Kayaer, Tavsanoglu 2008; Zhang *et al*. 2015; Farabet *et al*. 2011).

Taip pat pastaruoju metu paplito LPLM ir ARM procesorių hibridiniai lustai, tokio LPLM pavyzdys *Xilinx Zynq-7000* (*Zynq*... 2019). Šiame derinyje ARM CPĮ

teikia komunikacijų sąsajas, gali operatyviai dirbti su atmintimi, vykdo operacinę sistemą, o LPLM užtikrina vaizdų ir kitų signalų apdorojimą, nestandartinės periferijos ir atminties prijungimą ir realizavimą. Naudojant tokį derinį, sutaupoma laiko realizuojant DRAM, SRAM, PDK, USB, CAN, I2C, SD, UART sąsajas, kurias įprastai turi įrenginio ARM CPĮ.

Programų įgyvendinimas LPLM reikalauja daugiau laiko resursų lyginant su kitais įrenginiais. Tačiau gaunama didelė sparta su nedideliu galios suvartojimu, o tai teigiamai vertinama projektuojant galutinį produktą.

Specialiosios paskirties integriniai grandynai (SPIG) – tai integriniai grandynai, skirti konkrečiai užduočiai atlikti, nagrinėjamu atveju tai sąsūkos neuronų tinklų skaičiavimams. Šiuo metu rinkoje pateikiami neuronų procesoriniai įrenginiai (NPĮ) ir tenzorių procesoriniai įrenginiai (TPĮ). Tokio tipo įrenginių pavyzdžiai: *Intel Nervana* (*Intel Nervana...* 2019), *Google Coral Edge TPU* (*Coral beta* 2019).

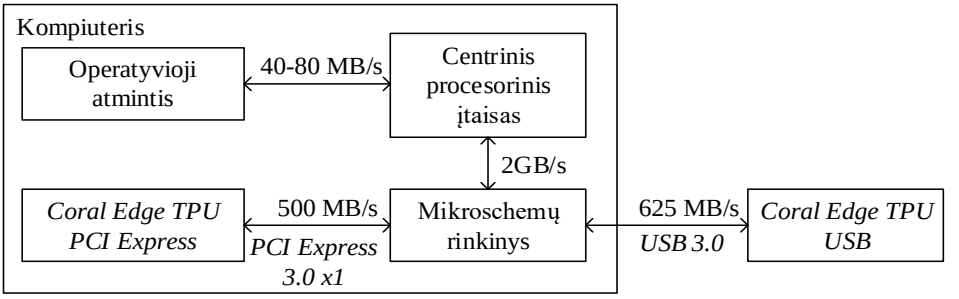
Specialiai giliojo mokymo tinklams modeliuoti SPIG kuria *Intel* kartu su *Nervana Systems* (*Intel Nervana...* 2019). Siūlomas įrenginys, kuris sujungia didelio pralaidumo atminties (angl. *High Bandwidth Memory*), *Intel FinFet* 14 nm technologijas ir specialiai giliesiems dirbtinių neuronų tinklams modeliuoti skirtą integrinį grandyną. Šios įmonės siūlomi įrenginiai gali būti sujungiami tarpusavyje, taip pasiekiamas didesnis bendrasis našumas.

2019 metais *Google* pradėjo tiekti *Google Coral Edge TPU* SPIG atliekantį *Tensorflow Lite* sukurtų, mokytų ir specialiu kompiliatoriumi apdorotų SDNT algoritmų kodą (*Coral beta* 2019). Šis įrenginys turi itin ribotą spartinančiąją atmintį ir veikia kaip koprocesorius, galintis naudoti sisteminę atmintį esant poreikiui. Visi nepalaikomi ar įrenginio spartinančiojoje atmintyje netelpantys veiksmai turi būti atliekami sistemos, prie kurios jis prijungtas, CPĮ (1.4 pav.). Gamintojas specifikacijoje nurodo, kad įrenginys atlieka iki 4 trilijonų skaičiavimo operacijų per sekundę (angl. *Tera Operations Per Second* – TOPS), naudodamas 0,5 W/TOPS, o maksimali galia yra 2 W. Šis TPĮ vykdo tik *Tensorflow Lit*e sukurtus tinklus, kurių algoritmų kodas sukompiliuotas specialiai TPĮ vykdyti. Be to, šis TPĮ turi ribotą vykdomų funkcijų sąrašą.

Yra trys *Google Coral Edge TPU* įrenginio tipai: USB ir *PCI Express* sąsajomis prijungiami spartinantieji įrenginiai ir įterptinė sistema modulyje, kuri sujungia *Google Coral Edge TPU* su ARM mikroprocesoriumi. *PCI Express* spartinantysis įrenginys prijungiamas viena *PCI Express 3.0* sąsajos linija (1.4 pav.) ir veikia kaip kompiuterio vidinis įrenginys. USB spartinantysis įrenginys naudojamas jį prijungus prie kompiuterio per USB 3.0 sąsają ir veikia kaip išorinis įrenginys (1.4 pav.).

Duomenų perdavimo greičius tarp TPĮ ir operatyviosios atminties riboja gamintojo pasirinktos sąsajos. Šie 500–625 MB/s greičiai yra pakankami nedideliems vaizdams perduoti į TPĮ. Apdorojimo metu yra svarbesni duomenų

perdavimo sąsajomis užlaikymai. Užlaikymai tampa ypač aktualūs, kai dalis skaičiavimų turi būti atliekama CPĮ arba TPĮ naudoja sistemos operatyviąją atmintį ir turi perduoti duomenis tarp vidinės spartinančiosios atminties ir kompiuterio operatyviosios atminties.



1.4 pav. Kompiuterio ir tenzorių procesorinio įrenginio sąsajų schema
Fig. 1.4. Scheme of computer and tensor processing unit interfaces

Apibendrinant šiame skirsnyje pateiktą medžiagą, galima išskirti šiuos pagrindinius SDNT mokymui ir įgyvendinimui naudojamus įrenginius:

- Centriniai procesoriniai įrenginiai (CPĮ) tinka įvairiausioms DNT užduotims atlikti. Užtikrina pakankamą spartą įprastiems DNT algoritmams vykdyti, tačiau gali būti per lėtas SDNT mokyti;
- Grafikos procesoriniai įrenginiai (GPĮ) tinka įvairiausioms DNT užduotims atlikti, užtikrina didelę SDNT mokymo ir algoritmų vykdymo spartą. GPĮ veikia kaip koprocesorius, todėl reikalingas centrinis procesorius, kuris teiktų užduotis ir duomenis;
- Lauku programuojamos loginės matricos (LPLM) teoriškai turi neribotas taikymo galimybes, tačiau reikalauja daug laiko įgyvendinant DNT modelius itin logikos lygmeniu;
- Skaitmeninių signalų apdorojimo įrenginiai (SSAĮ) greitai atlieka daugelį su DNT modeliavimu susijusių skaičiavimų, tačiau užduotims formuoti ir duomenims pateikti naudojamas centrinis procesorius arba LPLM. Šis variantas labiau tinka garsui apdoroti ir mažiau tinka vaizdams apdoroti. Užtikrina gerokai mažesnes galios sąnaudas nei CPĮ ar GPĮ;
- Specialiosios paskirties integriniai grandynai (SPIG). Tai įrenginiai, kuriais galima įgyvendinti LPLM suprojektuotus DNT, įgyvendinant integriniu grandynu arba realizuoti labai siauros paskirties tik SDNT skaičiavimams skirtus įrenginius. Tokių siauros paskirties procesorinių įrenginių pavyzdžiai yra tenzorių ir neuronų procesoriniai įrenginiai. Jie

yra itin efektyvūs galios našumo santykiu ir yra pakankamai spartūs DNT algoritmų skaičiavimams vykdyti.

Remiantis atlikta aparatūros apžvalga sudaryta 1.2 lentelė. Tinkamiausi SDNT tyrimams atlikti yra GPĮ dėl savo universalumo ir itin aukštos DNT algoritmų vykdymo spartos. CPĮ gali būti naudojami dirbant su naujo tipo tinklais, kurių elementai dar neįgyvendinti programinėje įrangoje spartinimui naudojant GPĮ arba reikalauja daugiau atminties talpos nei prieinama GPĮ.

1.2 lentelė. Aparatinės įrangos tinkamumas dirbtinių neuronų tinklų užduotims
Table 1.2. Hardware usability for tasks related to artificial neural networks

Įrenginys	Tinka tyrimams	Algoritmų vykdymo sparta	Našumo ir galios santykis	Tinka mokymui	Tinka taikymui	DNT dydis	Ne DNT operacijos
Kompiuterių klasteriai	Taip	Aukšta	Blogas	Taip	Taip	Teoriškai neribotas	Taip
CPĮ	Taip	Žema	Prastas	Taip	Taip	Labai didelis	Taip
GPĮ	Taip	Itin aukšta	Vidutinis	Taip	Taip	Didelis	Taip
LPLM	Ribotai	Aukšta	Geras	Ribotai	Taip	Ribotas	Taip
SSAĮ	Ribotai	Vidutinė	Vidutinis	Ribotai	Taip	Ribotas	Ribotos
TPĮ	Ribotai	Aukšta	Itin geras	Ribotai	Taip	Ribotas	Ne
NPĮ	Ribotai	Aukšta	Itin geras	Ribotai	Taip	Ribotas	Ne

Pastaba. Pilki langeliai iš kairės į dešinę žymi geriausiai SDNT mokymui ir vykdymui tinkančius spartinančiuosius įrenginius.

LPLM, SSAĮ, TPĮ ir NPĮ paprastai naudojami tik sąsūkos dirbtinių neuronų tinklų algoritmams vykdyti. LPLM turi aukščiausią lankstumą, bet ir reikalauja daugiausia darbo programuojant, o kaina yra didesnė ir energijos efektyvumas žemesnis nei SSAĮ, TPĮ ir NPĮ. SSAĮ turi ribotus skaičiavimo resursus, todėl dažniausiai taikomas tik vienmačius signalus apdorojantiems SDNT algoritmams vykdyti.

TPĮ ir NPĮ turi itin gerą našumo ir galios santykį, todėl geriausiai tinka SDNT algoritmams įterptinėse sistemose vykdyti, bet būtina atsižvelgti į jų trūkumus. Šie spartinantieji įrenginiai vykdo tik numatytų SDNT struktūros elementų ir matematinių funkcijų skaičiavimus. Nepalaikomos operacijos turi būti atliekamos CPĮ arba GPĮ. Šie įrenginiai, pritrūkus vidinės spartinančiosios atminties, naudoja sistemos operatyviąją atmintį, kai duomenų mainų sąsajos greitis riboja našumą.

Kompiuterių klasteriai išsiskiria galimybe modeliuoti itin didelius tinklus, tačiau dėl jungčių tarp klasterio mazgų atsirandantys vėlinimai stipriai sumažina skaičiavimų spartą. Vienos GPĮ SDNT algoritmų vykdymo sparta prilygsta kelių ar keliolikos kompiuterių klasterio spartai, o suvartojama energija mažesnė. Dėl šios priežasties klasteriai naudojami tik tiems DNT mokyti, kurių negalima sutalpinti GPĮ atmintyje.

Galima teigti, kad SDNT algoritmams vykdyti gerai tinka specializuotos sistemos su GPĮ dėl itin aukštos DNT algoritmų vykdymo spartos ir įterptinės sistemos su TPĮ ar NPĮ dėl aukšto energetinio efektyvumo. Šių įrenginių išskirtinės savybės 1.2 lentelėje išskirtos pilku fonu.

Apžvelgus paplitusius programinius įrankius, pastebėta, kad:

- daugiausia palaikymo sulaukia CPĮ ir GPĮ skaičiavimo įrenginiai;
- *Nvidia* ir AMD įmonės remia savo gamybos GPĮ palaikymą įvairioje SDNT modeliavimo programinėje įrangoje, tačiau *Nvidia* papildomai teikia savo bibliotekas ir programas bei palaikymą spartinant SDNT skaičiavimus jų gamybos GPĮ. Dėl šios priežasties *Nvidia* GPĮ dažniau pasirenkama SDNT tyrėjų ir taikytojų bendruomenėje;
- daugelis paketų pritaikyti dirbti *Windows, Linux* ir *macOS* operacinėse sistemose, tačiau didelė dalis paketų geriausiai veikia *Linux* ir *macOS*. Iš šių dviejų geriausia rinktis *Linux,* nes yra jos atvirojo kodo versijų, o taip pat ji naudojama SDNT algoritmus vykdančiuose serveriuose;
- spartinantieji įrenginiai, tokie kaip NPĮ ir TPĮ, turi ribotą vykdomų funkcijų sąrašą, todėl įgyvendinant SDNT spartinančiuosiuose įtaisuose būtina atsirinkti ir naudoti tik palaikomus SDNT elementus. Palaikomų elementų sąrašai skirtingiems įrenginiams skiriasi;
- projektuojant SDNT įgyvendinti spartinančiajame įrenginyje, būtina atsižvelgti į bendrą skaičiavimo operacijų apimtį ir naudojamos atminties talpą.

1.4. Pirmojo skyriaus išvados ir disertacijos uždavinių formulavimas

1. Dirbtinių neuronų tinklai ir sąsūkos dirbtinių neuronų tinklai plačiai taikomi vaizdų analizės srityje ir tinka atlikti daugelį operacijų, tačiau kiekvienai užduočiai būtina projektuoti naujas SDNT struktūras arba pritaikyti esamas. Būtų naudinga sukurti SDNT projektavimo metodiką pasirinktam vaizdų analizės uždaviniui spręsti.
2. Dvi aktualios vaizdų analizės užduotys tinkamos išbandyti SDNT projektavimo metodiką. Pirmoji yra akies tinklainės vaizdams tapdinti

naudojamas lokalusis požymių aprašymas, antroji – kelio dangos tipo ir būklės nustatymas iš automobilio priekyje sumontuotos vaizdo kameros vaizdų.

3. Grafinis procesorinis įrenginys yra geriausiai tinkantis SDNT mokyti spartinantysis įrenginys, o algoritmams vykdyti įterptinėse sistemose geriausiai tinka tenzorių ir neuronų procesoriniai įtaisai.

4. Spartinantieji įrenginiai turi ribotą atminties talpą ir skaičiavimo spartą, palaiko ne visas SDNT operacijas, todėl projektuojant SDNT būtina atsižvelgti į spartinančiojo įrenginio charakteristikas.

5. Naudojant *Nvidia* spartinančiuosius įtaisus, sumažinama SDNT įgyvendinimo įterptinėje sistemoje darbo apimtis ir trukmė, nes įmonė pateikia visą reikiamą programinę įrangą ir bendradarbiauja su naudotojų bendruomene.

Remiantis atlikta analitine apžvalga ir pateiktomis išvadomis, iškelta hipotezė, kad galima sudaryti sąsūkos dirbtinių neuronų tinklų elementų sąrašą ir sukurti projektavimo metodiką, kuriuos naudojant galima kurti specializuotus sąsūkos dirbtinių neuronų tinklus įgyvendinti spartinančiuosiuose įtaisuose vaizdams apdoroti realiuoju laiku.

Siekiant patikrinti pateiktą hipotezę, iškelti šie uždaviniai:

1. Ištirti sąsūkos dirbtinių neuronų tinklų (SDNT) struktūros elementų ir parametrų įtaką algoritmų tikslumui ir jų mokymo bei vykdymo spartinančiuosiuose įrenginiuose spartai.

2. Sudaryti SDNT elementų sąrašą, iš kurio galima parinkti elementus projektuojant specializuotą SDNT, skirtą įgyvendinti spartinančiuosiuose įrenginiuose.

3. Sukurti SDNT projektavimo metodiką pasirinktai vaizdų analizės užduočiai spręsti, kai žinomi minimalūs reikalavimai greitaveikai ir algoritmą vykdančio spartinančiojo įrenginio techninės charakteristikos.

4. Taikant sudarytą metodiką, sukurti ir spartinančiuosiuose įrenginiuose įgyvendinti SDNT, skirtus šiems vaizdų analizės uždaviniams spręsti:
   - akies tinklainės vaizdų požymiams aprašyti;
   - kelio dangos tipui ir būklei nustatyti.

2

Sąsūkos dirbtinių neuronų tinklų įgyvendinimo tyrimai

Šiame skyriuje aprašytas sąsūkos dirbtinių neuronų tinklų (SDNT) elementų sąrašas, SDNT veikimo, mokymo ir algoritmų vykdymo spartinančiuosiuose įrenginiuose tyrimai, reikalingi SDNT projektavimo metodikai sudaryti. Sukurta SDNT projektavimo metodika algoritmams, įgyvendinamiems spartinančiuosiuose įtaisuose vaizdams analizuoti realiuoju laiku, pateikta skyriaus priešpaskutiniame poskyryje.

Į sudarytą SDNT elementų sąrašą įtraukti elementai, kurie gali būti įgyvendinami spartinančiuosiuose įrenginiuose. Kiekvienam sąrašo elementui pateikta: veikimo aprašymas, skaičiavimo formulė, skaičiavimo operacijų skaičiaus apskaičiavimo formulė. Sąraše pateikiami SDNT taikomi dirbtinių neuronų, sutelkimo, reguliarizavimo ir normalizavimo sluoksniai.

Skyriuje pateikti SDNT veikimo, mokymo ir įgyvendinimo spartinančiuosiuose įrenginiuose tyrimai, reikalingi SDNT struktūros pritaikymo metodikai sudaryti. Tyrimų metu vizualizuotas SDNT sąsūkos sluoksnių veikimas, palyginti įvairios sandaros SDNT, nustatyta SDNT struktūros ir parametrų įtaka mokymo spartai ir tikslumui klasifikuojant ranka rašytus skaičius, matuota vieno vaizdų analizės SDNT grįstu algoritmu, įgyvendintu tenzorių procesoriniame įrenginyje, trukmė.

Šeštajame poskyryje pateikta, remiantis tyrimų metu sukauptu įdirbiu, sudaryta SDNT struktūros ir parametrų pritaikymo metodika. Šios metodikos tikslas – pritaikyti SDNT struktūrą spartinančiuosiuose įrenginiuose įgyvendinti, atsižvelgiant į užduoties specifiką ir sistemos technines charakteristikas.

Skyriuje pateikti tyrimų rezultatai ir jų taikymas vaizdų analizei pristatytas mokslinėse konferencijose ir aprašytas mokslo publikacijose (Šabanovič, Matuzevičius 2017; Šabanovič, Stankevičius, Matuzevičius 2018; Žuraulis, Surblys, Šabanovič 2019).

2.1. Sąsūkos dirbtinių neuronų tinklų elementų sąrašas

Sudarant sąsūkos dirbtinių neuronų tinklų elementų sąrašą. būtina atsižvelgti į dažniausiai naudojamus SDNT struktūros elementus ir jų suderinamumą su programine įranga bei vykdymu spartinančiuosiuose įrenginiuose. Šiame skyriuje pateikiamas elementų sąrašas, kuris yra pakankamas tokioms SDNT struktūroms kaip *ZFNet*, *VGGNet*, *Inception*, *ResNet* ir *DenseNet* sudaryti. Šiems SDNT sudaryti naudojami visiškai sujungtųjų (VS) ir sąsūkos dirbtinių neuronų, sutelkimo (angl. *pooling*) ir atrinkimo (angl. *subsampling*), aktyvavimo, išmetimo (angl. *dropout*), sujungimo (angl. *concatenation*) sluoksniai, paketinio normalizavimo (PN), lokaliojo atsako normalizavimo (LAN) ir *softmax* sluoksniai. Sąraše pateikiamos sandaugos, sumos, šaknies traukimo, palyginimo skaičiavimo operacijų apimties išraiškos yra suformuotos, taikant prielaidą, kad indeksų didinimo ir lango slinkties operacijos neskaičiuojamos.

Sąsūkos sluoksniai – tai pagrindinis SDNT struktūros elementas. Sąsūkos sluoksnis turi pasirinktą kiekį, pasirinkto dydžio sąsūkos filtrų. Sąsūkos filtru apdorojant įėjimo duomenis slenkančiu pasirinktu žingsniu langu, gaunami požymių žemėlapiai. Požymių žemėlapiai parodo, kuriose įėjime pateikto vaizdo vietose kokios lango ir sąsūkos filtro verčių sandaugų sumos vertės gautos. Kuo gaunama didesnė vertė, tuo lange esančios vertės geriau sutampa su filtro vertėmis. Sąsūkos operaciją galima užrašyti tokia matematine išraiška:

$$X_{n,i,j} = \sum_{n=1}^{\mathrm{N}} \left(\sum_{h=1}^{\mathrm{H}} \sum_{w=1}^{\mathrm{W}} \sum_{c=1}^{\mathrm{C}} W_{n,w,h} \cdot A_{si+w,sj+h,c} \right) + B_n, \tag{2.1}$$

čia $X_{n,i,j}$ – sąsūkos sluoksnio išėjime gaunamo požymių žemėlapio masyvo vertė (prieš aktyvavimo funkciją), su n,i,j indeksu, čia n – sąsūkos filtro indeksas; i – požymių žemėlapio masyvo eilutės indeksas; j – požymių masyvo stulpelio

indeksas; $W_{n,h,w}$ - sąsūkos filtrų masyvo vertė, su n,h,w indeksu, čia n – sąsūkos filtro indeksas; h – sąsūkos filtro eilutės indeksas; w – sąsūkos filtro stulpelio indeksas; $A_{si+w,sj+h,c}$ – įėjimo masyvo vertė, su indeksu $si+w, sj+h, c$, čia s – filtro lango žingsnis; c – įėjimo masyvo sluoksnių skaičius; B_n – sąsūkos dirbtinio neurono, kurio indeksas n, poslinkio vertė; N – sąsūkos neuronų skaičius sluoksnyje; H – sąsūkos filtro eilučių skaičius; W – sąsūkos filtro stulpelių skaičius; C – sąsūkos filtro sluoksnių skaičius, atitinkantis įėjimo masyvo sluoksnių skaičių.

Įėjimo masyvas gali turėti daugiau nei vieną dimensiją. Įėjime pateikus spalvotą vaizdą, jo sluoksnių skaičius C būtų 3. Išėjimo sluoksnių skaičius N priklauso nuo sąsūkos operacijai naudojamų filtrų skaičiaus. Patys filtrai yra sudaryti iš mokymo metu derinamų kintamųjų, vadinamų ryšio svoriais, kurie yra naudojami įėjimo masyvą apdorojant slenkančiu langu. Lyginant su visiškai sujungtaisiais sluoksniais šiame sluoksnyje yra gerokai mažiau ryšio svorių.

Pagrindiniai sąsūkos sluoksnio parametrai yra: filtrų dydis, skaičius ir žingsnis, papildymas. Papildymo dydis nurodo, kiek pikselių turi būti pridėta už vaizdo krašto prieš atliekant sąsūką. Naujai pridėti pikseliai užpildomi nuliais, maksimalia reikšme arba gretima vaizdo krašto reikšme. Naudojant užpildymą nuliais, po sąsūkos operacijos kraštai atrodo tamsesni, naudojant maksimalią reikšmę – šviesesni, o naudojant vidutinę – panašūs į krašto reikšmes. Įprastai sąsūkos operacija atliekama per visus įėjimo vaizdo sluoksnius, o gaunamo požymių žemėlapio sluoksnių skaičius lygus sąsūkai naudotų filtrų skaičiui. Kuo daugiau naudojama sąsūkos sluoksnių, tuo sudėtingesnius požymių rinkinius SDNT gali išrinkti.

Sąsūkos sluoksnio sandaugos ir sudėties operacijų skaičius N_{SA} apskaičiuojamas šia matematine išraiška:

$$N_{\mathrm{SA}} = \frac{2\mathrm{NIJHWC}}{\mathrm{s}^2}, \tag{2.2}$$

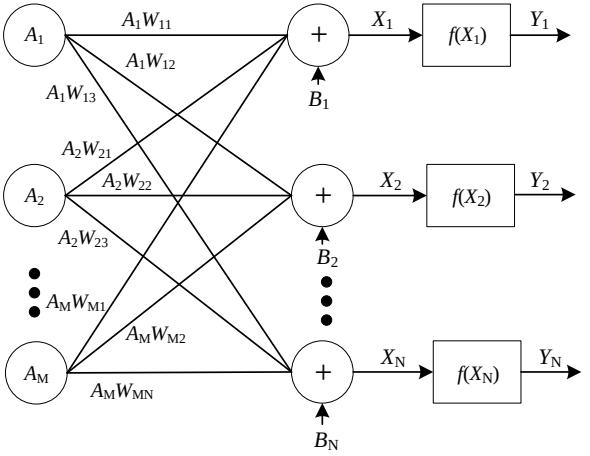
čia N – sąsūkos neuronų skaičius sluoksnyje; H – sąsūkos filtro eilučių skaičius; W – sąsūkos filtro stulpelių skaičius; C – sąsūkos filtro sluoksnių skaičius, atitinkantis įėjimo masyvo sluoksnių skaičių; I – įėjimo masyvo eilučių skaičius; J – įėjimo masyvo stulpelių skaičius, s – sąsūkos filtro lango žingsnis.

Mokymo metu derinamų kintamųjų skaičiaus sąsūkos sluoksnyje M_{SA} matematinė išraiška:

$$M_{\mathrm{SA}} = \mathrm{N(HWC+1)}, \tag{2.3}$$

čia N – sąsūkos neuronų skaičius sluoksnyje; H – sąsūkos filtro eilučių skaičius; W – sąsūkos filtro stulpelių skaičius; C – sąsūkos filtro sluoksnių skaičius, atitinkantis įėjimo masyvo sluoksnių skaičių.

Visiškai sujungtųjų (VS) dirbtinių neuronų sluoksnis (2.1 pav.) – tai dirbtinių neuronų sluoksnis, kuriame visi dirbtiniai neuronai yra sujunti su visais įėjimo taškais. 2.1 paveiksle A_m – įėjimo vektorius, kurio ilgis M, vertė, kurios indeksas m; W_{mn} – dirbtinių neuronų ryšio svorių matricos vertė su indeksu mn, nurodančiu ryšį tarp m-osios įėjimo vertės ir n-ojo neurono; $\mathbf{W}$ matricos dydis yra $\mathrm{M} \times \mathrm{N}$, čia M – įėjimo vektoriaus ilgis, N – neuronų skaičius sluoksnyje. B_m – m-ojo neurono poslinkio reikšmė; X_n – n-ojo neurono išėjimo reikšmė prieš aktyvavimo funkciją; Y_n – n-ojo neurono išėjimo reikšmė po aktyvavimo funkcijos.



2.1 pav. Visiškai sujungtųjų dirbtinių neuronų sluoksnio schema
Fig. 2.1. Scheme of fully connected layer of artificial neurons

VS sluoksnio veikimą aprašo ši matematinė išraiška:

$$\mathbf{Y} = f(\mathbf{X}) = f(\mathbf{A} \times \mathbf{W} + \mathbf{B}), \tag{2.4}$$

čia $\mathbf{Y}$ – VS dirbtinių neuronų sluoksnio rezultatas; $\mathbf{X}$ – dirbtinių neuronų išėjimo verčių vektorius prieš aktyvavimo funkciją, $f(\cdot)$ – aktyvavimo funkcija; $\mathbf{A}$ – įėjimo vektorius, kurio verčių skaičius yra M; $\mathbf{W}$ – visų sluoksnio dirbtinių neuronų ryšio svorių matrica; $\mathbf{B}$ – visų sluoksnio dirbtinių neuronų poslinkio verčių vektorius.

VS sluoksniai naudojami SDNT-VS struktūros tinkluose, jų pavyzdžiai *LeNet-5* ir *Alexnet*. Norint panaudoti šiuos sluoksnius po SDNT būtina vektorizuoti įėjimo masyvą į požymių vektorių. Šie neuronų sluoksniai gali naudoti aktyvavimo funkcijas. Pagrindiniai VS sluoksnių parametrai yra neuronų skaičius N, naudojama aktyvavimo funkcija $f(\cdot)$, ryšio svorių matricos ir poslinkio inicializavimo tipas – pasirenkama atsitiktinių ryšio svorių generatoriaus pasiskirstymo funkcija arba generavimo funkcija.

Atsižvelgiant į schemą, pateiktą 2.1 paveiksle, galima užrašyti VS sluoksnio sandaugos ir sudėties skaičiavimo operacijų skaičiaus N_{VS} išraišką:

$$N_{\mathrm{VS}} = 2\mathrm{MN}, \tag{2.5}$$

čia M – įėjimo vektoriaus ilgis; N – dirbtinių neuronų skaičius sluoksnyje.

Mokymo metu VS sluoksnyje derinamų kintamųjų skaičiaus M_{VS} išraiška:

$$M_{VS} = (\mathrm{M}+1)\mathrm{N}, \tag{2.6}$$

čia M – įėjimo vektoriaus ilgis; N – dirbtinių neuronų skaičius sluoksnyje.

Sutelkimo sluoksniai (angl. *pooling*) veikia panašiai kaip ir sąsūkos sluoksniai, tačiau naudojamos fiksuotos filtro vertės arba pasirinktos matematinės funkcijos. Sutelkimo sluoksnio išėjimas skaičiuojamas atliekant pasirinktą funkciją tarp pikselių, patenkančių į pasirinkto dydžio langą, slenkant šį langą pasirinktu dydžiu. Įprastai sutelkimo parametrai nėra keičiami mokymosi metu. Dažniausiai sutelkimui pasirenkama maksimali reikšmė tarp lange esančių reikšmių (angl. *max pooling*), vidutinė lango reikšmė (angl. *average pooling*), minimali reikšmė (angl. *min pooling*). Kartais apskaičiuojama Euklido norma tarp lango reikšmių (angl. *L2 norm pooling*). Šie sluoksniai dažniausiai naudojami tarp sąsūkos sluoksnių ir pagerina tinklo atsparumą vaizdo poslinkiams, pasukimams ir mastelio keitimui.

Skaičiavimo operacijų skaičius sutelkimui pagal vidurkį:

$$N_{\mathrm{SPV}} = \frac{2\,\mathrm{IJHWC}}{\mathrm{s}^2}, \tag{2.7}$$

čia H – sutelkimo lango eilučių skaičius; W – sutelkimo lango stulpelių skaičius; C – įėjimo masyvo sluoksnių skaičius; I – įėjimo masyvo eilučių skaičius; J – įėjimo masyvo stulpelių skaičius; s – sutelkimo lango žingsnis.

Skaičiavimo operacijų skaičius sutelkimui pagal Euklido atstumą (L2 normą) apskaičiuojamas pagal šią išraišką:

$$N_{\mathrm{SPL2}} = \frac{2\mathrm{IJHWC}}{\mathrm{s}^2}, \tag{2.8}$$

čia H – sutelkimo lango eilučių skaičius; W – sutelkimo lango stulpelių skaičius; C – įėjimo masyvo sluoksnių skaičius; I – įėjimo masyvo eilučių skaičius; J – įėjimo masyvo stulpelių skaičius; s – sutelkimo lango žingsnis.

Maksimalus skaičiavimo operacijų skaičius sutelkimui pagal maksimumą arba minimumą apskaičiuojamas taikant šią išraišką:

$$N_{\mathrm{SPM}} = \frac{2\mathrm{IJHWC}}{\mathrm{s}^2}, \tag{2.9}$$

čia H – sutelkimo lango eilučių skaičius; W – sutelkimo lango stulpelių skaičius; C – įėjimo masyvo sluoksnių skaičius; I – įėjimo masyvo eilučių skaičius; J – įėjimo masyvo stulpelių skaičius; s – sutelkimo lango žingsnis.

Taip pat naudojamas globalusis sutelkimas pagal vidurkį – tai operacija, kuri nustato vidutinį kiekvieno įėjime pateikto požymių žemėlapio sluoksnio vertę, šis sluoksnis neturi mokymo metu derinamų kintamųjų. Globaliojo sutelkimo pagal vidurkį operacijų skaičius N_{GSPV} apskaičiuojamas pagal formulę:

$$N_{\mathrm{GSPV}} = \mathrm{IJC}, \tag{2.10}$$

čia I – įėjimo masyvo eilučių skaičius; J – įėjimo masyvo stulpelių skaičius; C – įėjimo masyvo sluoksnių skaičius.

Aktyvavimo funkcijos naudojamos kaip atskiri sluoksniai po sąsūkos ir VS dirbtinių neuronų sluoksnių, tačiau dažnai laikomos jų dalimi ir nustatomos kaip vienas parametras, nusakantis naudojamos aktyvavimo funkcijos pavadinimą. Šiuo metu dažniausiai naudojamos šios funkcijos: hiperbolinio tangento, sigmoidės, *ReLU, LReLU, PreLU, RReLU, ELU, Maxout, Probout* funkcijos.

ReLU (angl. *Rectified Linear Unit*) yra tiesinė detektorinė aktyvavimo funkcija, jos veikimas primena idealiojo diodo charakteristiką. Praleidžiama tiesiogiai bet kokia teigiama įėjimo vertė, o vietoj neigiamos grąžinamas 0. Jos trūkumas, kad ryšio svoriui tapus 0, per šią aktyvavimo funkciją nevyksta atgalinis skleidimas. Ji gana greitai vykdoma ir užrašoma šia išraiška:

$$f_{\mathrm{ReLU}}(\mathbf{X}) = \max(\mathbf{X}, 0), \tag{2.11}$$

čia $\mathbf{X}$ – įėjimo duomenų masyvas.

ReLU aktyvavimo funkcijos skaičiavimo operacijų skaičių galima rasti taikant šią išraišką:

$$N_{\text{ReLU}} = \text{IJC}, \tag{2.12}$$

čia I – įėjimo masyvo eilučių skaičius; J – įėjimo masyvo stulpelių skaičius; C – įėjimo masyvo sluoksnių skaičius.

LReLU (angl. *Leaky Rectified Linear Unit)* yra tiesinė detektorinė aktyvavimo funkcijos versija, kurios veikimas panašus į ReLU, tačiau neigiamos įėjimo vertės pateikiamos į išėjimą su pasirinktu slopinimo koeficientu $a < 1$ ir aprašoma tokia išraiška:

$$f_{\text{LReLU}}(\mathbf{X}) = \max(\mathbf{X}, \text{a} \cdot \mathbf{X}), \tag{2.13}$$

čia $\mathbf{X}$ – įėjimo duomenų masyvas; a – neigiamos vertės koeficientas.

ReLU aktyvavimo funkcijos skaičiavimo operacijų skaičių galima rasti taikant šią išraišką:

$$N_{\text{LReLU}} = 2\text{IJC}, \tag{2.14}$$

čia I – įėjimo masyvo eilučių skaičius; J – įėjimo masyvo stulpelių skaičius; C – įėjimo masyvo sluoksnių skaičius.

Yra dar kelios šio tipo aktyvavimo funkcijos modifikacijos. *PReLU* (angl. *Parameteric Rectified Linear Unit*) panašus į *LReLU*, tačiau šios aktyvavimo funkcijos neigiamos vertės koeficientas a išmokstamas, o *RReLU* (angl. *Randomized Rectified Linear Unit*) aktyvavimo funkcijoje neigiamos vertės koeficientas a pasirenkamas atsitiktinai, kiekvieno mokymosi žingsnio metu ir fiksuojamas taikymo metu. *LReLU*, *PReLU*, *RReLU* aktyvavimo funkcijos turi nenulinį išėjimą esant neigiamam įėjimui, tai leidžia išvengti atskyrų neuronų atsijungimo dėl nulinio ar neigiamo įėjimo ryšio koeficiento ir pagerinti tinklo charakteristikas lyginant su *ReLU*, tačiau gali reikalauti didesnių skaičiavimo resursų.

ELU (angl. *Exponential Linear Unit*) turi neigiamą išėjimo charakteristikos dalį, kuri padeda greitesniam mokymui ir yra atsparesnė triukšmui. Jos formulė:

$$f_{\text{ELU}}(\mathbf{X}) = \max(\mathbf{X}, \text{a}(e^{\mathbf{X}} - 1)), \tag{2.15}$$

čia $\mathbf{X}$ – įėjimo duomenų masyvas; a – neigiamos vertės koeficientas.

Maxout yra alternatyvi netiesinė aktyvavimo funkcija, kuri paima maksimalų atsaką per kelis kanalus įvairiose įėjimo masyvo pozicijose, kitaip tai specialusis *ReLU* atvejis, kuris gerai pritaikytas veikti su išmetimo sluoksniais (angl. *dropout*). *Probout* – tai tikimybinis *maxout* variantas, kuriame maksimumo operacija pakeičiama kvantavimo procedūra. *Probout* gali pasiekti balansą tarp

maxout savybių ir tinklo invariantiškumo savybių, tačiau reikalauja daugiau skaičiavimo resursų tikimybėms skaičiuoti.

Išmetimo (angl. *dropout*) sluoksnis yra reguliarizavimo metodas, sumažinantis persimokymo tikimybę (Srivastava *et al.* 2014). Išmetimas veikia sudauginant įėjimo masyvą su išmetimo binariniu masyvu, kurio reikšmės atsitiktinai sugeneruojamos pagal Bernulio pasiskirstymą. Sudauginus tarpusavyje šiuos masyvus, gaunamas išėjimo masyvas, kiekvieną kartą išmetamos vis kitos reikšmės, taip mažinama SDNT išėjimų priklausomybė nuo pavienės įėjimo reikšmės. Šis sluoksnis turi vienintelį parametrą, tai išmetimo tikimybė, kuri pasirenkama intervale nuo 0 iki 1. SDNT algoritmo vykdymo metu išmetimas atjungiamas, o grąžinamos reikšmės dauginamos iš prieš tai naudoto tikimybės koeficiento tam, kad bendra išėjimo masyvo verčių suma išliktų tame pačiame lygmenyje.

Kitos naudojamos SDNT operacijos yra matricos formos keitimas į vektorių, lokaliojo atsako normalizavimas, paketinis normalizavimas, *softmax*.

Sudarytas SDNT sąrašas yra pakankamas įprastiems SDNT projektuoti, tačiau nėra baigtinis ir gali būti papildomas atsižvelgiant į papildomus spartinančiųjų įrenginių palaikomų elementų sąrašus.

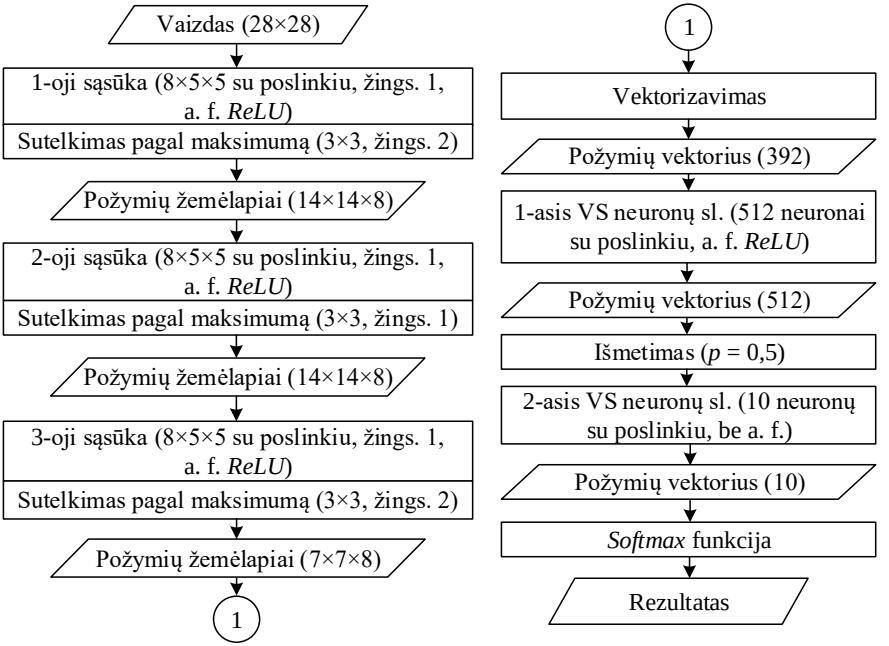
2.2. Sąsūkos sluoksnių veikimo vizualizavimas

Sąsūkos sluoksnių veikimui tirti sudaryta sąsūkos dirbtinių neuronų tinklo (SDNT) struktūra, kurioje specialiai sumažintas filtrų skaičius ir dydis, siekiant vaizdžiau parodyti vaizdo apdorojimą ir požymių išskyrimą SDNT sąsūkos sluoksniuose. Tinklo struktūra pateikta 2.2 paveiksle. Šis tinklas turi 3 blokus, sudarytus iš sąsūkos ir sutelkimo pagal maksimumą sluoksnių, ir du visiškai sujungtųjų (VS) neuronų sluoksnius. Iš viso tinkle yra apie 400 tūkst. mokymo metu derinamų dirbtinių neuronų ryšio svorių. Rezultatui apskaičiuoti naudojama *Softmax* funkcija.

Tiriamas SDNT atlieka 28×28 pikselių raiškos ranka rašyto skaičiaus iš MNIST duomenų rinkinio nespalvotų vaizdų analizę ir grąžina jų priskyrimo klasėms nuo 0 iki 9 tikimybes. 2.2 paveiksle pateikiami SDNT algoritmo sluoksniai, jų parametrai ir kiekvieno bloko išėjime gaunamų požymių žemėlapių masyvų matmenys.

Atliekant sąsūkos ar sutelkimo sluoksnius, kai naudojamas didesnis nei vienetinis lango žingsnis, gaunamo požymio žemėlapio masyvo sluoksnių matmenys sumažėja tiek kartų, kokio dydžio žingsnis naudotas. Kai pasirinktam žingsniui atlikti trūksta vaizdo taškų, yra pridedamos papildomos nulinės vertės. Po trijų sąsūkos ir sutelkimo blokų gaunamas 7×7 verčių dydžio ir 8 sluoksnių

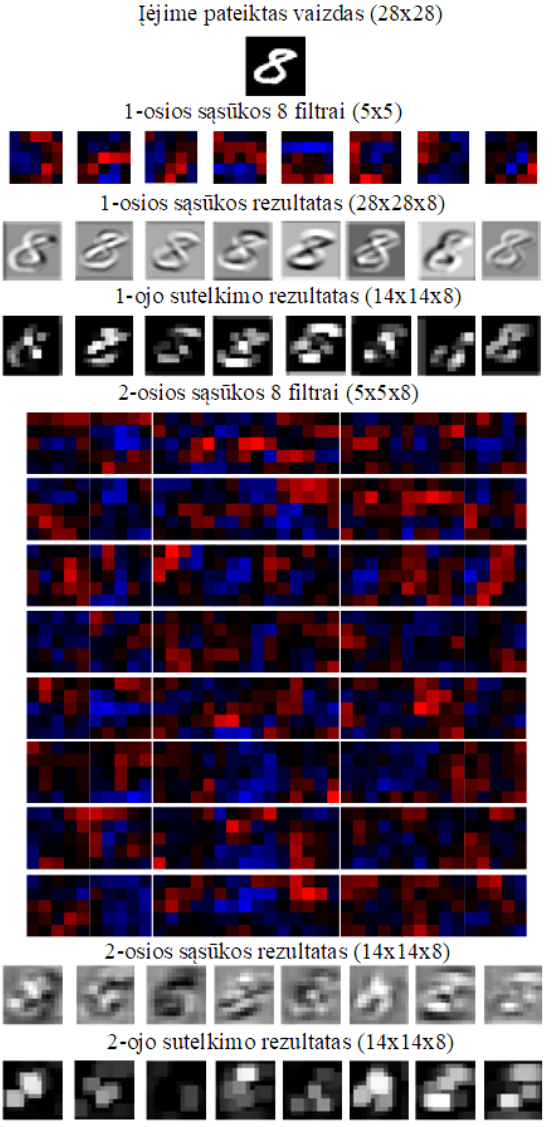
požymių žemėlapių masyvas. Norint šiuos požymius toliau pateikti į visiškai sujungtųjų (VS) neuronų sluoksnį, yra taikomas masyvo verčių vektorizavimas, kurio metu daugiamatis masyvas paverčiamas vektoriumi. Visiškai sujungtieji sluoksniai mokomi atlikti klasifikavimą pagal vaizdo požymius, atrinktus taikant sąsūkos sluoksnius. Išmetimo sluoksnis su 0,5 išmetimo tikimybe naudojamas apsaugoti tinklą nuo persimokymo. Atsitiktinai išmetant pusę iš apdorojamų požymių, SDNT apsaugomas nuo klasifikavimo rezultato priklausomybės nuo vienetinių vaizdo požymių. Po VS sluoksnių gautą 10 dirbtinių neuronų aktyvavimo reikšmių vektorių, apdorojus *Softmax* funkcija, gaunamos analizuoto vaizdo priskyrimo vienai iš dešimties skaičių klasių tikimybės nuo 0 iki 1.



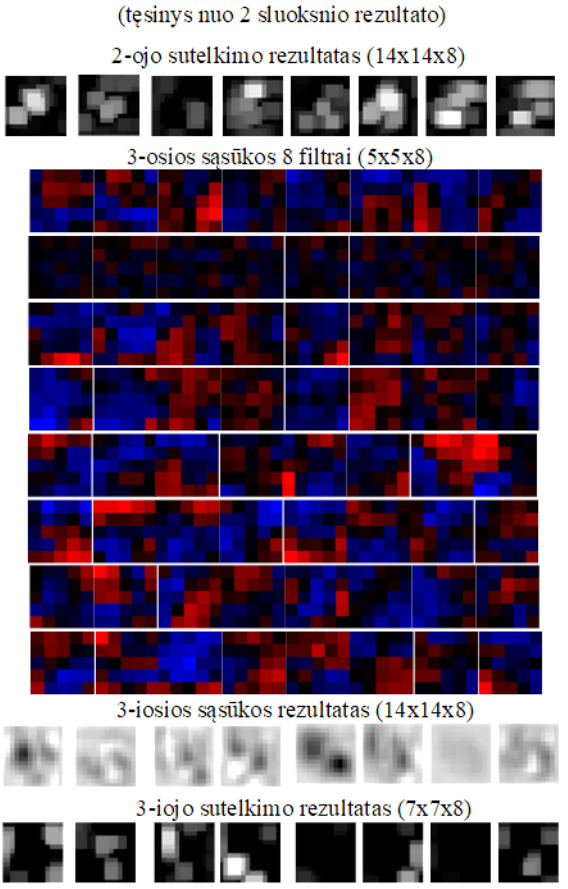
2.2 pav. Sąsūkos dirbtinių neuronų tinklo struktūra, taikyta veikimui vizualizuoti
Fig. 2.2. Structure of convolutional neural network used for performance visualization

SDNT mokymui ir patikrai naudotas grafikos procesorinis įrenginys (GPĮ). Mokymas atliktas 60 kartų panaudojant visus 50 tūkst. ranka rašytų skaičių vaizdus, esančius mokymo rinkinyje. Pasiekta mokymo sparta 37 658 vaizdai/s. Po mokymo atlikus SDNT patikrą nustatyta, kad klasifikacijos tikslumas yra 99,38 %. Patikros metu algoritmas vykdytas 84 599 vaizdų/s sparta. Mokyto SNDT ryšio svoriai išsaugoti, o juos naudojant atliktas tinklo veikimo vizualizavimas.

Atliekant tinklo veikimo vizualizavimą įėjime pateiktas ranka rašyto skaičiaus „8“ vaizdas. Sąsūkos sluoksnių filtrų išėjimo matricos ir sutelkimo sluoksnių rezultatai pateikti 2.3 ir 2.4 paveiksluose.



2.3 pav. Sąsūkos dirbtinių neuronų tinklo 1-ojo ir 2-ojo sąsūkos ir sutelkimo sluoksnių sąsūkos filtrų ir rezultatų vizualizavimas
Fig. 2.3. Visualization of convolutional neural network 1st and 2nd layers' convolution and pooling kernels and results



2.4 pav. Sąsūkos dirbtinių neuronų tinklo 3-iojo sąsūkos ir sutelkimo sluoksnių sąsūkos filtrų ir rezultatų vizualizavimas
Fig. 2.4. Visualization of convolutional neural network 3rd layers' convolution and pooling kernels and results

Pirmojo sąsūkos sluoksnio rezultatas yra apdorojamas taikant aktyvavimo funkciją ir sutelkimo sluoksnį. Apdorojus įėjime pateiktą vaizdą, kiekvienas filtras sugeneruoja požymių žemėlapio vieną sluoksnį. Sujungus visų filtrų sugeneruotus rezultatus, gaunamas filtrų skaičių atitinkančio sluoksnių skaičiaus rezultatas. Šis rezultatas pateikiamas į antrąjį sąsūkos sluoksnį, tačiau dėl to, kad jau yra 8 sluoksniai, tai ir filtrai turi 8 sluoksnius. 2.3 ir 2.4 paveiksluose sluoksniai pavaizduoti išdėstant juos iš kairės į dešinę. Atskiriems įėjimo sluoksniams apdoroti naudojami atitinkami filtro sluoksniai. Rezultatas turi tokį patį sluoksnių skaičių, kiek buvo panaudota filtrų. Po apdorojimo, taikant aktyvavimo funkciją ir sutelkimo sluoksnį, gautas požymių žemėlapis pateikiamas į trečiąjį sąsūkos

sluoksnį, kur viskas kartojama dar kartą. Filtrų svorių teigiamos vertės vaizduojamos raudonai, o neigiamos mėlynai, kuo didesnė absoliutinė vertė, tuo ryškesnė spalva. Visų filtrų ryšio svoriai normalizuoti, kad tilptų į atvaizduojamų intensyvumų intervalą nuo 0 iki 255 prieš paverčiant matricas spalvotais paveikslais.

Panagrinėjus tinklo veikimo vizualizaciją, pateiktą 2.3 ir 2.4 paveiksluose, galima įsitikinti, kad pirmasis sąsūkos sluoksnis leidžia išrinkti paprasčiausius vaizdo požymius, tokius kaip horizontalios, vertikalios, skersinės linijos, kampai, arkos ir susikirtimai, visiškai baltas ar juodas vaizdas. Antrasis sąsūkos sluoksnis išrenka sudėtingesnius vaizdo požymius, kurie sudaryti iš pirmajame sluoksnyje atrinktų paprastesnių požymių kombinacijų, pavyzdžiui, apskritimai su tamsiu centru, apskritimai su šviesiu centru, ilgesnės arkos, linijos ir kraštai. Trečiasis sąsūkos sluoksnis kombinuoja antrajame sluoksnyje išrinktus vaizdo požymius.

Nagrinėjant mokyto tinklo sąsūkos filtrų vaizdus, galima spręsti, ar po SDNT mokymo nėra neaktyvių neuronų. Jei filtrų vizualizacijoje visas filtras yra tamsus, tai jo reikšmės artimos nuliui ir jis mažai dalyvauja priimant sprendimus, tokio sluoksnio pavyzdys yra 2.4 pav. 3-iosios sąsūkos filtrų antrojoje eilutėje. Jie galėjo atsirasti, nes filtro masyvo reikšmei tapus neigiamai, dėl naudojamos *ReLU* aktyvavimo funkcijos sluoksnio išėjime visada gaunamas nulis, ir mokymo metu gradientas visada yra 0, todėl tas filtras nebedalyvauja mokymo procese. Šią problemą leidžia išspręsti *ReLU* modifikacijos *LReLU, RReLU, PReLU* ir ELU, kurios neigiamo įėjimo atveju grąžina pakeistą neigiamą reikšmę.

Atliekant SDNT veikimo tyrimą, vizualizuoti išmokti sąsūkos filtrai, sąsūkos, sutelkimo sluoksniuose apskaičiuoti požymių žemėlapiai. Gauti požymių žemėlapiai parodo, kokius vaizdo požymius ir jų rinkinius SDNT išmoko aptikti. Sąsūkos sluoksnių rezultatai parodo, kaip SDNT apibendrina vaizdo požymius, o sutelkimo – kaip SDNT atrenka svarbiausius vaizdo požymius. Po sąsūkos sluoksnių gaunami požymių žemėlapiai rodo ne tik informaciją apie požymius, bet ir apie jų vietą vaizde. Kelių sąsūkos sluoksnių SDNT gali susieti paprastų vaizdo požymių, tokių kaip linijos, gradientai, šviesių, tamsių taškų vietą vaizde, ir jų tarpusavio padėtis taip, kad būtų galima nustatyti atskirus požymius, klasifikacijos metu nulemiančius vaizdo priskyrimą vienai iš klasių.

2.3. Sąsūkos dirbtinių neuronų tinklo struktūros ir parametrų įtakos spartai ir tikslumui tyrimas

Tyrimui naudota aparatūra yra kompiuteris su *Intel i7-6900K* CPĮ, veikiančiu 3,8 GHz taktiniu dažniu, 64 GB talpos DDR4 tipo operatyviosios atminties, veikiančios 1500 MHz taktiniu dažniu ir 15 ciklų vėlinimu, *Nvidia Geforce GTX*

1080 Ti Founders Edition GPĮ. Sąsūkos dirbtinių neuronų tinklai aprašomi, mokomi ir vykdomi naudojant *Tensorflow r1.0.0* (Abadi *et al.* 2015) programų paketą. Visi testavimo scenarijai buvo parašyti *Python 3.5* skriptai, kurie buvo vykdomi *Ubuntu 16.04 LTS* operacinėje sistemoje su instaliuota CUDA 8.0.44 ir *cuDNN 5.1* programų bibliotekomis.

Eksperimentų metu buvo naudojami šie fiksuoti parametrai:

- modelio įvesties duomenys buvo 28×28 pikselių raiškos vaizdai;
- vaizdo taškų intensyvumas konvertuotas į *TensorFlow* numatytąsias 32 bitų slankiojo kablelio formato 4 dimensijų matricas, kur yra tokios dimensijos: paketo vaizdų skaičius, vaizdo stulpelio vertės, vaizdo eilutės vertės, vaizdo kanalai. Šiuo atveju tai buvo 64, 28, 28, 1;
- klasifikuojama į 10 klasių, todėl išėjimas yra dvejetainis masyvas, kuriame vienetas žymi teisingą klasę, o nuliai – klaidingą;
- ranka rašytų skaičių vaizdų rinkinio mokymo dalyje yra 60 tūkst. vaizdų, o validavimo – 10 tūkst. vaizdų. Yra papildomas testavimo rinkinys, sudarytas iš 5 tūkst. pavyzdžių;
- kiekvienam modelio mokymui naudojama 30 epochų. Viena epocha laikoma visų mokymo pavyzdžių kiekio perėjimas vieną kartą, šiuo atveju 50 tūkst.;
- naudota atsitiktinės atrankos funkcija su normaliuoju pasiskirstymu tinklo sluoksnių ryšio svorių matricų reikšmėms inicijuoti. Atsitiktinės atrankos standartinis nuokrypis nustatytas 0,1 ir vidurkis lygus 0;
- naudotas 64 pavyzdžių paketo dydis;
- vaizdo priskyrimo klasėms tikimybės gaunamos taikant *softmax* funkciją;
- tikslo funkcija buvo apskaičiuota taikant *softmax* kryžminės entropijos funkciją;
- naudojamas L2 reguliarizavimas visiškai prijungtų sluoksnių svoriams ir paklaidoms. Mokymo metu nustatytas reguliarizavimo pagal L2 koeficientas $5 \cdot 10^{-4}$;
- mokymui pasirinktas stochastinio gradientinio nuolydžio su momentu algoritmas, mokymo koeficientas nustatytas 0,01, o momento koeficientas lygus 0,9. Mokymo koeficientas mažintas eksponentiškai po kiekvienos epochos, taikant 0,95 koeficientą.

Atskirų eksperimentų metu buvo keičiami atskiri sąsūkos dirbtinių neuronų tinklo (SDNT) parametrai: sluoksnių skaičius, sąsūkos filtrų dydis ir žingsnis, sutelkimo lango dydis ir žingsnis. Šių bandymų rezultatai parodo SDNT parametrų įtaką spartai ir tikslumui. Gauti mokymo ir vykdymo spartos rezultatai skiriasi kelis kartus dėl to, kad mokymo metu vyksta ne tik tiesioginis apdorojamų duomenų sklidimas, bet ir atgalinis klaidos sklidimas, ryšio svoriams atnaujinti.

Paprastai mokymo žingsnis trunka iki kelių kartų ilgiau nei vykdymo dėl sudėtingesnės matematikos – gradientų ir išvestinių skaičiavimo. Matavimai atlikti ne mažiau kaip 5 kartus, atmestos mažiausia ir didžiausia vertės ir pateiktas likusių matavimų vidurkis. Toks skaičiavimo būdas taikomas, nes tikslumas kinta dėl SDNT ryšio svorių inicijavimo atsitiktinėmis reikšmėmis, o sparta kinta dėl kompiuterio apkrovos ir temperatūros bandymo metu.

Šių eksperimentų metu klasifikavimo klaidingumas E_K skaičiuotas pagal formulę:

$$E_{\mathrm{K}} = \frac{N_{\mathrm{F}}}{N_{\mathrm{V}}} \cdot 100\% = \frac{N_{\mathrm{FP}} + N_{\mathrm{FN}}}{N_{\mathrm{V}}} \cdot 100\%, \tag{2.16}$$

čia N_{F} – neteisingai klasifikuotų vaizdų skaičius; N_{V} – visų klasifikuotų vaizdų skaičius; N_{FP} – neteisingai priskirtų klasėms vaizdų skaičius; N_{FN} – neteisingai nepriskirtų klasėms vaizdų skaičius.

Tikslumas A_{K} – tai atvirkščias dydis klasifikavimo klaidingumui, skaičiuojamas pagal formulę:

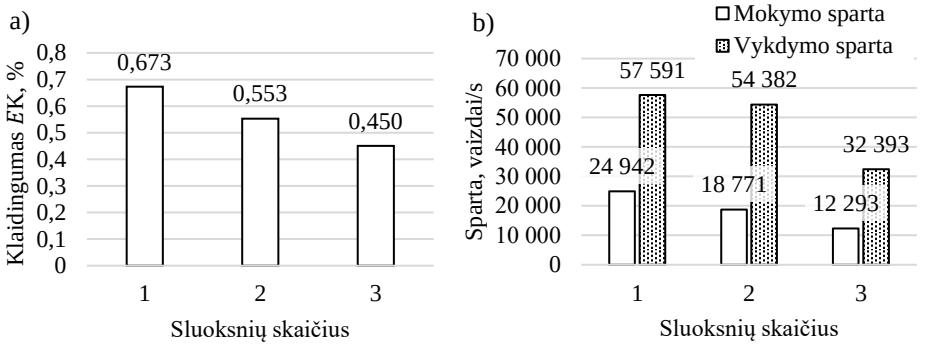
$$A_{\mathrm{K}} = \frac{N_{\mathrm{T}}}{N_{\mathrm{V}}} \cdot 100\% = \frac{N_{\mathrm{TP}} + N_{\mathrm{TN}}}{N_{\mathrm{V}}} \cdot 100\%, \tag{2.17}$$

čia N_{T} – teisingai klasifikuotų vaizdų skaičius; N_{T} – visų klasifikuotų vaizdų skaičius; N_{TP} – teisingai priskirtų klasėms vaizdų skaičius; N_{TN} – teisingai nepriskirtų klasėms vaizdų skaičius.

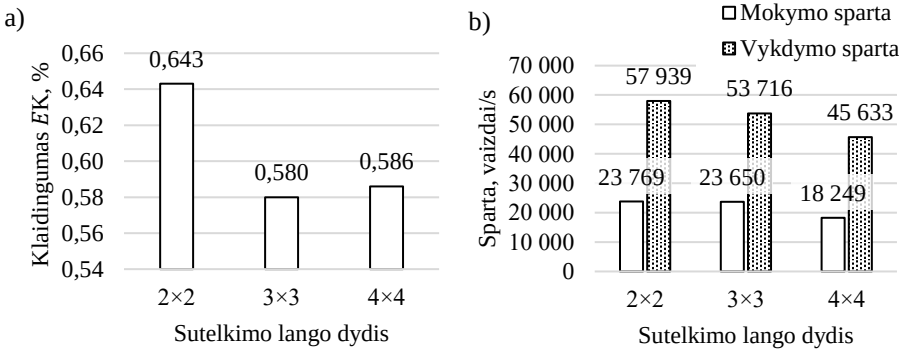
Bandymų metu apskaičiuotos tikslumo reikšmės yra panašios, todėl jas atvaizdavus grafiškai, būtų sunku matyti skirtumus. Dėl to 2.5–2.9 paveiksluose vietoje tikslumo parodomas atvirkščias dydis – klaidingumas.

Skaičiavimo ir matavimo rezultatai (2.5 pav.) rodo, kad MNIST rinkinio skaičių klasifikavimo tikslumas auga didinant sąsūkos sluoksnių skaičių. Skaičiavimo sparta naudojant vieną ir du sluoksnius keičiasi nedaug, kadangi naudojant vieną sluoksnį nebuvo iki galo panaudojamas GPĮ lygiagrečių skaičiavimų potencialas. Pridėjus trečiąjį sluoksnį, sparta sumažėjo nuo 58 tūkst. vaizdų/s iki 36 tūkst. vaizdų/s (2.5 pav.). Taigi, didesnis sluoksnių skaičius pagerina tikslumą, tačiau sumažina spartą.

Konkrečiam uždaviniui spręsti 2×2 pikselių dydžio sutelkimo lango dydis yra per mažas, 3×3 pikselių dydžio yra gana didelis, o 4×4 pikselių dydžio yra jau per didelis ir tikslumas mažėja, nes prarandama per daug informacijos (2.6 pav.). Taip pat matoma, kad didesni lango dydžiai sumažina veikimo spartą, nes padidėja sutelkimo operacijų apimtis.



2.5 pav. Sąsūkos dirbtinių neuronų tinklo sluoksnių skaičiaus įtaka:
a) skaičiavimo rezultatai; b) matavimų rezultatai
Fig. 2.5. Impact of convolutional neural network's layers count:
a) calculation result, b) measurement results

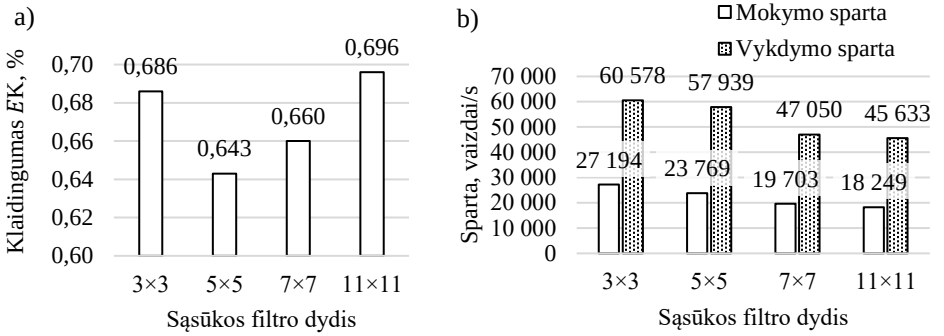


2.6 pav. Sąsūkos dirbtinių neuronų tinklo sutelkimo sluoksnių filtro dydžio įtaka:
a) skaičiavimo rezultatai; b) matavimų rezultatai
Fig. 2.6. Impact of convolutional neural network's pooling layers kernel size:
a) calculation result; b) measurement results

Atlikto sąsūkos filtrų dydžio įtakos skaičiavimo ir matavimo rezultatai pateikti 2.7 paveiksle. 5×5 pikselių dydžio sąsūkos filtrai užtikrino aukščiausią tikslumo lygį. Spartos skirtumas tarp 3×3 pikselių dydžio bei 5×5 pikselių dydžio filtrų dydžio nėra žymus, tačiau 7×7 ir 11×11 jau pastebimai sulėtina tiek mokymo, tiek patikros spartą.

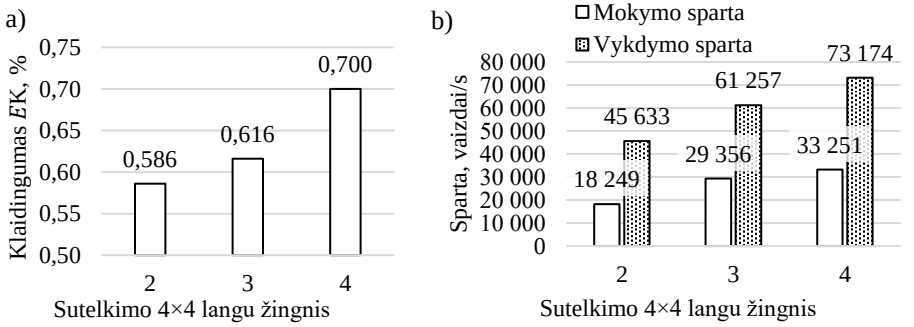
Atlikto sutelkimo sluoksnio lango dydžio įtakos skaičiavimo ir matavimo rezultatai pateikti 2.8 paveiksle. Naudojant fiksuotą 4×4 pikselių dydžio sutelkimo langą, naudojant mažiausią žingsnį 2 buvo gaunamas didžiausias

tikslumas, bet ir mažiausia sparta. Tikslumas buvo didesnis dėl to, kad gana didelis persidengimas tarp sutelkimo langų (50 % lango dydžio) suteikė didesnį invariantiškumą poslinkiui ir posūkiui.



2.7 pav. Sąsūkos dirbtinių neuronų tinklo sąsūkos sluoksnių filtro dydžio įtaka: a) skaičiavimo rezultatai; b) matavimų rezultatai

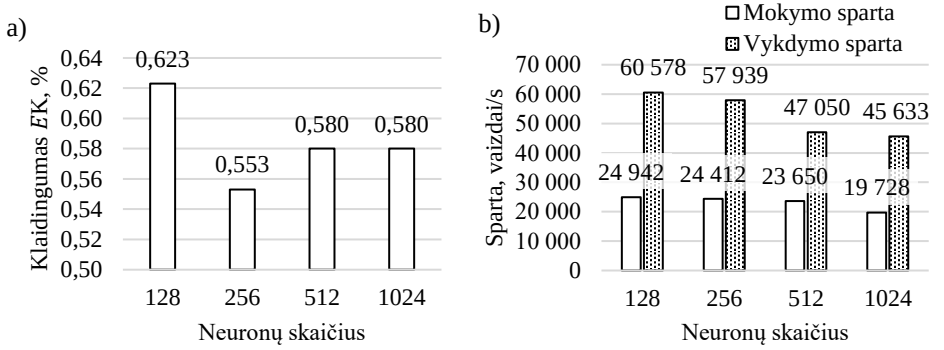
Fig. 2.7. Impact of convolutional neural network's convolution layer kernel size: a) calculation result; b) measurement results



2.8 pav. Sąsūkos dirbtinių neuronų tinklo sutelkimo sluoksnių žingsnio dydžio įtaka: a) skaičiavimo rezultatai; b) matavimų rezultatai

Fig. 2.8. Impact of convolutional neural network's pooling layers stride: a) calculation result; b) measurement results

Visiškai sujungtųjų sluoksnių neuronų skaičių keičiant nuo 128 iki 1024, geriausiai pasirodė 256 (2.9 pav.). Šis neuronų skaičius užtikrino mažiausią klaidą. Esant didesniam skaičiui tikslumas mažėjo, kadangi buvo nepakankamai unikalių požymių mokyti papildomus neuronus, greičiausiai tinklas pradėjo persimokyti.



2.9 pav. Sąsūkos dirbtinių neuronų tinklo visiškai sujungtųjų neuronų sluoksnio neuronų skaičiaus įtaka: a) skaičiavimo rezultatai; b) matavimų rezultatai
Fig. 2.9. Impact of convolutional neural network's fully connected layer's neuron count: a) calculation result; b) measurement results

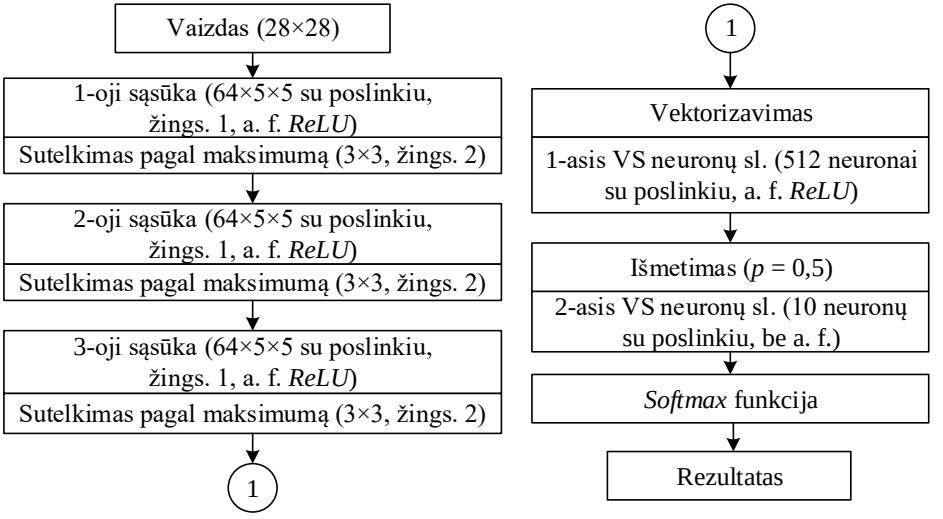
Iš gautų rezultatų galima daryti išvadą, kad konkrečiai užduočiai parinkti tinklo parametrus galima atlikus kelis bandymus naudojant skirtingus parametrus, keičiant vieną parametrą vienu metu. Tai būtina, nes ne visada tikslumas kinta tolygiai, didinant SDNT parametrų reikšmes. Atliekant parametrų parinkimą, galima naudoti dalį duomenų rinkinio siekiant paspartinti bandymus. Kaip pradinę struktūrą galima rinktis vieną iš geriausiai panašią užduotį atliekančių SDNT struktūrų, tokių kaip *LeNet*-5, *Alexnet* ir *VGGNet*, o tada keisti jų parametrus.

2.4. Sąsūkos dirbtinių neuronų tinklų mokymo spartos tyrimas

Programinės įrangos įrankių sąsūkos dirbtinių neuronų tinklų (SDNT) mokymo trukmės, vykdant skaičiavimus centriniu procesoriniu įrenginiu (CPĮ) ir dviem grafikos procesoriniais įrenginiais (GPĮ), palyginimui naudotas tas pats kompiuteris kaip ir 2.3 poskyryje. Bandymams naudojama viena SDNT struktūra (2.10 pav.), kurią panašiai galima įgyvendinti tiriamoje programinėje įrangoje. Šio sąsūkos dirbtinio neuronų tinklo užduotis – klasifikuoti ranka rašytų skaičių nespalvotų 28×28 pikselių raiškos vaizdus į 10 klasių nuo 0 iki 9. Atsižvelgiant į klasifikavimo užduotį, pasirinkta tinklo išėjime taikyti *Softmax* funkciją, kuri pateikia apdoroto vaizdo priskyrimo kiekvienai iš 10 klasių tikimybę.

2.1 lentelėje pateiktos išmatuotos spartos ir apskaičiuotos tikslumo vertės. Pateikiamoms vertėms gauti atlikta po 5 mokymo bandymus. Iš penkių bandymų mokymo trukmės ir mokyto tinklo pasiekiamo klasifikavimo tikslumo reikšmių atmetama didžiausia bei mažiausia vertė ir pateikiamas likusių verčių vidurkis.

Didžiausia ir mažiausia reikšmės atimamos, siekiant išvengti atsitiktinai pasitaikančių matavimų, kurie stipriai skiriasi nuo vidutinių reikšmių. Tikslumas skaičiuotas taikant 2.17 formulę, o sparta apskaičiuota dalijant visų apdorotų mokymo metu vaizdų skaičių iš mokymo trukmės.



2.10 pav. Sąsūkos dirbtinių neuronų tinklo struktūra, taikyta tyrimo metu
Fig. 2.10. Structure of convolutional neural network used for experiment

Mokymo metu ryšio svoriai inicializuoti atsitiktiniais skaičiais, atitinkančiais apribotą normalųjį pasiskirstymą su 0,05 standartine deviacija. Mokymas vykdomas naudojant 64 vaizdų paketus per mokymo iteraciją, jo metu 50 tūkst. vaizdų mokymo rinkinys panaudotas 30 kartų. Mokymui taikytas ryšio svorių reguliarizavimas pagal Euklido atstumą tarp neuronų ryšio svorių. Ryšio svoriams atnaujinti taikytas *Adadelta* mokymo algoritmas, su baziniu mokymo koeficientu 1, mažėjančiu 0,95 karto po kiekvieno perėjimo per visą mokymo rinkinį.

Bandymo metu įranga nenaudota pašalinėms užduotims atlikti. Kadangi vykdymo trukmė priklauso nuo bendros įrangos apkrovos ir GPĮ naudojimo laiko bei temperatūros.

2.1 lentelėje trūksta SDNT mokymo taikant DIGITS aplinką su *caffe* programine įranga bandymo rezultatų. Bandymas nebuvo atliktas dėl to, kad su DIGITS pateikta *caffe* versija kompiliuota darbui su *Nvidia* GPĮ naudojant *cuDNN* biblioteką ir nepasileidžia CPĮ. Taip pat galima pastebėti, kad įprasta *caffe* gerai veikia ir GPĮ, ir CPĮ. Ši versija yra kompiliuota darbui su CPĮ, o naudojant autorių sukurtą biblioteką vietoje *cuDNN* gali vykdyti skaičiavus GPĮ.

2.1 lentelė. Tinklo mokymo taikant įvairius programinės įrangos paketus rezultatai
Table 2.1. Results of model training using different software frameworks

Programų paketas	Skaičiavimo įrenginys	Mokymo sparta, vaizdai/s	Mokymo trukmė, s	Tikslumas, %
Caffe	*GTX 1080 Ti* (GPĮ)	15 125	255	99,52
	GTX 960 (GPĮ)	4192	910	99,55
	i7-6900K (CPĮ)	294	13 263	99,53
DIGITS (caffe)	*GTX 1080 Ti* (GPĮ)	14 187	274	99,48
	GTX 960 (GPĮ)	4202	920	99,5
Tensorflow	*GTX 1080 Ti* (GPĮ)	11 586	322	99,51
	GTX 960 (GPĮ)	3258	1136	99,51
	i7-6900K (CPĮ)	939	3939	99,53
Keras (Tensorflow)	*GTX1080 Ti* (GPĮ)	8593	421	99,57
	GTX 960 (GPĮ)	2749	1313	99,58
	i7-6900K (CPĮ)	905	4009	99,56
Neon	*GTX 1080 Ti* (GPĮ)	15 836	230	99,52
	GTX 960 (GPĮ)	6000	604	99,49
	i7-6900K (CPĮ)	58	~62 379	~99,50
Torch (bazinis)	*GTX 1080 Ti* (GPĮ)	10 972	335	99,31
	GTX 960 (GPĮ)	2751	1317	99,26
	i7-6900K (CPĮ)	795	4589	99,31
Torch (*dp*)	*GTX 1080 Ti* (GPĮ)	10 094	361	99,40
	GTX 960 (GPĮ)	2751	1236	99,40
	i7-6900K (CPĮ)	760	4660	99,37
Theano	*GTX 1080 Ti* (GPĮ)	11 236	320	99,42
	GTX 960 (GPĮ)	2862	1268	99,42
	i7-6900K (CPĮ)	237	14 759	99,44

Daugelis populiariausių SDNT modeliavimui naudojamų programų paketų yra pritaikyti skaičiavimams naudojant GPĮ. Naudojant *Nvidia GTX 960* GPĮ skaičiavimai atliekami mažiausiai 3 kartus sparčiau nei naudojant *Intel i7-6900K* CPĮ, o naudojant *Nvidia GTX 1080 Ti* GPĮ – mažiausiai 9 kartus sparčiau (2.1 lentelė). Kadangi vaizdai yra nedideli, šis testas parodo skirtingos

programinės įrangos įrankių našumą vidutinio sudėtingumo SDNT, turinčio mažą įvesties duomenų raišką. Gauti mokymosi spartos rezultatai parodo, kad skaičiavimai CPĮ nėra efektyvus vidutinio dydžio SDNT mokymui. Mokant nagrinėtą SDNT modelį *Intel i7-6900K* CPĮ mokymui mažiausiai prireikė apie 1 valandos ir 6 minučių, *Nvidia Geforce GTX 960* GPĮ – 10 minučių, o *Nvidia Geforce GTX 1080 Ti* GPĮ – mažiau nei 4 minučių.

2.5. Sąsūkos dirbtinių neuronų tinklų algoritmų įgyvendinimo specializuota įterptine sistema tyrimas

Tyrimo metu tirta, mobiliosioms ir įterptinėms sistemoms skirtų sąsūkos dirbtinių neuronų tinklų (SDNT) algoritmų vykdymo trukmė ir naudojama atminties talpa. Bandymai atlikti su *Inception V1*/*V2*/*V3*/*V4*, *MobileNet V1*/*V2* SDNT struktūromis, skirtomis *ImageNet* 1000 klasių vaizdų rinkiniui klasifikuoti. Bandymo rezultatai pateikti 2.2 lentelėje.

2.2 lentelė. Sąsūkos dirbtinių neuronų tinklų algoritmų įgyvendinimo tenzorių procesoriniame įtaise bandymų rezultatai
Table 2.2. Experiment results of convolutional neural networks' implementation using tensor processing unit

SDNT struktūra	Įėjime pateikiamo vaizdo dydis, taškais	Vidutinė vykdymo trukmė, ms	Naudojama atminties talpa, MB	
			Spartinančioji	Sisteminė
Inception V1	224×224	17,9	6,53	0,182
Inception V2	224×224	28	6,55	4,97
Inception V3	299×299	55,6	5,62	17,74
Inception V4	299×299	98,5	5,91	36,25
MobileNet V1	244×244	11,6	7,14	0
	128×128	2,5	0,511	0
MobileNet V2	244×244	32,5	6,91	0

Pastaba. Pilki langeliai nurodo tinklo architektūras, kurios telpa tenzorių procesorinio įtaiso spartinančiojoje atmintyje.

Visi 2.2 lentelėje pateiktų tinklų elementai buvo vykdomi koprocesoriuje, tačiau didesni *Inception* architektūros tinklai naudojo sisteminę atmintį. Iš gautų

rezultatų matoma, kad tinklai, telpantys į spartinančiąją atmintį, vykdomi greičiau, sisteminės atminties naudojimas ilgina apdorojimo laiką. Įdomu, kad sumažinus įėjimo vaizdų kraštinės dydį mažiau nei dvigubai, sumažėjus *MobileNet* atliekamų operacijų skaičiui ir naudojamos atminties apimčiai, algoritmo vykdymo sparta padidėjo apie 4 kartus.

SDNT algoritmai buvo vykdomi darbo stotyje su *Google Coral Edge TPU* koprocesoriumi, prijungiamu per USB sąsają. Šis tenzorių procesorinis įrenginys (TPĮ) spartina palaikomų SDNT elementų vykdymą. Kai kurie tinklai nedidelę dalį veiksmų vykdė naudojantis darbo stoties, turinčios *Intel i7-8700K* CPĮ, resursais. Visais atvejais tinklai apdorojo iki tinklo įėjimo matmenų sumažintus spalvotus vaizdus, o išėjimų skaičius priklausė nuo klasifikuojamų kategorijų skaičiaus. TPĮ reikalauja naudoti *Tensorflow Lite* sudarytus SDNT, kurie po mokymo naudojant GPĮ yra kvantuojami. Kvantavimo metu svorio koeficientai ir vidiniai kintamieji verčiami iš slankiojo kablelio į 8 bitų sveikojo skaičiaus formatą. Kvantuoti modeliai yra kompiliuojami į binarinį vykdomąjį failą, kuris paleidžiamas vykdyti koprocesoriuje.

2.6. Sąsūkos dirbtinių neuronų tinklo projektavimo metodika

Sąsūkos dirbtinių neuronų tinklų (SDNT) struktūros projektavimo metodiką galima aprašyti taip:

1. Pasirenkamas bazinis SDNT. Šis tinklas turi būti vykdomas pasirinktoje įterptinėje sistemoje ir užtikrinti panašią į reikiamą vykdymo spartą ir rezultatų tikslumą:
   - pasirenkami keli baziniai SDNT;
   - apskaičiuojama sandaugos ir sudėties operacijų apimtis;
   - apskaičiuojama naudojamos atminties talpa;
   - atrenkami tinklai, kurių skaičiavimo operacijų ir naudojamos atminties talpa neviršija arba nežymiai viršija pasirinkto spartinančiojo įrenginio skaičiavimų spartos ir atminties talpos apribojimus;
   - atliekama SDNT tikslumo ir vykdymo trukmės analizė, naudojant mokytus bazinių SDNT tinklų pavyzdžius, atliekant perkeliamąjį mokymą arba naudojant ribotą mokymo žingsnių skaičių.
   - atrenkamas geriausias bazinis SDNT.
2. Keičiama bazinio SDNT struktūra ir parametrai parenkant įėjimo sluoksnio ir tolesnių sluoksnių raišką, sąsūkos filtrų skaičių ir dydį:

- tinklo sluoksnių parametrai parenkami pagal mokymo rinkinio dydį taip, kad nevyktų SDNT permokymas ir nebūtų viršijami spartinančiojo įrenginio skaičiavimų ir atminties resursai;
- pasirenkamas kompromisas tarp veikimo tikslumo ir spartos mažinant ar didinant tinklo gylį, raišką ar filtrų skaičių, tam atliekami bandymai, vienu metu keičiant po vieną struktūros parametrą ir stebint jo įtaką tikslumui.

3. Pasirinktų parametrų tinklai mokomi ir išbandomi:
   - SDNT mokomi pradedant nuo atsitiktinių ryšio svorių;
   - SDNT papildomai mokomi naudojant atrinktas sunkiai išmokstamas vaizdų ir rezultato pavyzdžių poras.
4. Atrenkamas SDNT, geriausiai užsibrėžtas charakteristikas atitinkantis SDNT atrenkamas. Jei jo charakteristikos vis dar netenkina užduoties reikalavimų, sudaromos modifikacijos kartojant 2 ir 3 žingsnius.

Taikant šią metodiką, SDNT tinklo struktūra ir jos parametrai galėtų būti parenkami pusiau automatiškai. Taikant aprašytą metodiką, suprojektuoti ir spartinančiuosiuose įrenginiuose įgyvendinti algoritmai aprašyti 3 skyriuje.

2.7. Antrojo skyriaus išvados

1. Sudaryto sąsūkos dirbtinių neuronų tinklo (SDNT) struktūros elementų sąrašo skaičiavimo apimties formulės leidžia priimti sprendimus dėl SDNT struktūros remiantis sandaugos-sudėties operacijų apimtimi bei apytikriai įvertinti vykdymo spartą.
2. Konkrečiai vaizdų analizės užduočiai parinkti SDNT struktūros parametrus galima atliekant mokymą ir patikrą, keičiant vieną parametrą vienu metu.
3. SDNT mokymas lyginant su *Intel i7-6900K* centriniu procesoriniu įrenginiu taikant grafikos procesorinį įrenginį (GPĮ) *Nvidia Geforce GTX 960* vykdomas mažiausiai tris kartus sparčiau, o *Nvidia Geforce GTX 1080* GPĮ – mažiausiai devynis kartus sparčiau.
4. SDNT algoritmus įgyvendinant spartinančiuosiuose įrenginiuose svarbu užtikrinti, kad algoritmas neviršytų įrenginio spartinančiosios atminties talpos, nes vykdymo sparta gali sumažėti dėl duomenų mainų tarp spartinančiojo įrenginio ir sistemos operatyviosios atminties.

3

Sąsūkos dirbtinių neuronų tinklų taikymas vaizdams analizuoti

Šiame skyriuje aprašomi SDNT, įgyvendinti dviejose srityse. Viena iš jų yra akies tinklainės vaizdų, kurie gaunami fotografuojant akies tinklainę specializuota kamera. Kita – automobilių kelių tipo ir būklės klasifikavimas iš vaizdo kamera gautų vaizdų.

Skyriuje aprašyta: duomenų rinkinių sudarymas, tinklo struktūros parinkimas ir įgyvendinimas. Sukurtas lokaliojo požymių aprašymo algoritmas akies tinklainės vaizdams palyginimas su kitais žinomais vaizdo požymių aprašymo algoritmais. Šiam palyginimui pasiūlyta originali metrika. Sukurtas kelio dangos tipo ir būklės nustatymo algoritmas yra originalus, yra apskaičiuotas jo tikslumas ir išmatuota vykdymo trukmė spartinančiajame įrenginyje. Siekiant pagerinti kelio dangos ir būklės atpažinimo tikslumą ir sumažinti apdorojimo laiką, pasiūlytas originalus vaizdo paruošimo apdorojimui būdas, kuris prieš klasifikavimą iš kelio vaizdo paima reikšmingą kelio tipui nustatyti vaizdo dalį.

Tyrimų rezultatai viešai skelbti mokslinėse konferencijose ir mokslo publikacijose (Šabanovič, Matuzevičius 2015; Šabanovič, Matuzevičius, Vaitkevičius 2016; Šabanovič, Matuzevičius 2017; Šabanovič, Stankevičius, Matuzevičius 2018; Žuraulis, Surblys, Šabanovič 2019).

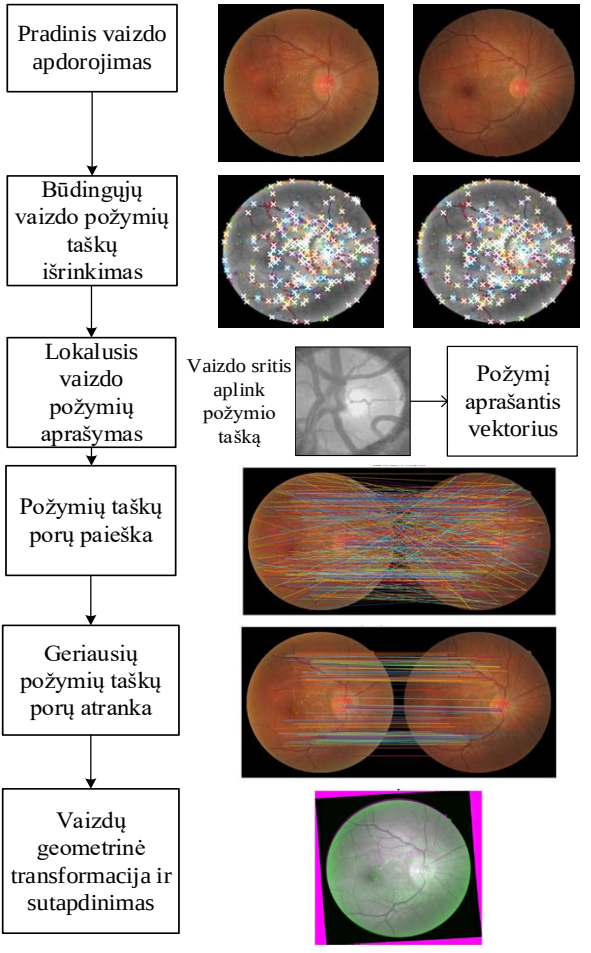
3.1. Akies tinklainės vaizdams sutapdinti tinkamas požymių aprašymo algoritmas

Vienas iš oftalmologijoje naudojamų medicininių tyrimų yra akies tinklainės fotografija. Nustatant akies tinklainės ligas ir stebint jų gydymo eigą, naudingas kelių nuotraukų, surinktų per kelis paciento apsilankymus, palyginimas (Abramoff, Garvin, Sonka 2010; Hernandez-Matas *et al*. 2017). Siekiant pastebėti smulkiausius pakeitimus, pageidautina, kad lyginamos nuotraukos būtų sutapdinamos. Natūralieji tinklainės pakitimai (tiek kliniškai svarbūs, tiek ne) ir vaizdo iškraipymai, atsirandantys dėl fotografavimo procedūros parametrų, sukelia geometrinius ir šviesumo pokyčius tarp skirtingu metu gautų nuotraukų. Šie pokyčiai apsunkina vaizdų sutapdinimą. Vaizdų sutapdinimo metu atliekama vieno iš sutapdinamų vaizdų geometrinė transformacija, po kurios dviejų vaizdų tie patys taškai turėtų atsidurti maksimaliai artimose koordinatėse. Sutapdinti vaizdai gali būti sujungiami į bendrą vaizdą.

Nustatant skirtumus tarp dviejų analizuojamų vaizdų, būtina panaikinti geometrinius iškraipymus, tam atliekamas vaizdų sutapdinimas. Atliekant dviejų vaizdų sutapdinimą, vienas kitą atitinkantys dviejų nuotraukų būdingųjų požymių taškai atsiduria tose pačiose erdvinėse koordinatėse (Hernandez-Matas *et al*. 2017; Hernandez-Matas, Zabulis, Argyros 2017; Noyel *et al*. 2017). Būdingieji vaizdo požymių taškai – tai vaizdo taškai ir juos supantys vaizdo fragmentai, kurie turi išskirtinę struktūrą, yra pastovūs ir randami vaizde nepriklausomai nuo vaizdo geometrinių transformacijų, spalvos, ryškumo, šviesumo ar kontrasto pokyčių. Norint rasti šių požymių taškų poras dviejuose vaizduose, juos reikia aprašyti skaitine išraiška, tam gali būti taikomi lokaliojo požymių aprašymo algoritmai.

Lokaliojo požymių aprašymo algoritmai – tai algoritmai, kurie skaičių vektoriumi aprašo būdinguosius vaizdo požymių taškus. Šie algoritmai turi pasižymėti reikiamu invariantiškumo ir išrankumo santykiu aprašant požymio erdvės sritį vaizde, kad atstumo metrika tarp požymį aprašančių vektoriaus verčių rodytų vaizdo požymių panašumą. Vaizdo būdingųjų požymių taškams aprašyti gali būti naudojami žmogaus atrinktais požymiais grįsti algoritmai, tokie kaip SIFT (Lowe 2004), SURF (Bay *et al*. 2008), BRISK (Leutenegger, Chli, Siegwart 2011), FREAK (Alahi, Ortiz, Vandergheynst 2012), KAZE (Alcantarilla, Artoli, Davison 2012) ir SDNT grįsti algoritmai (Yi *et al*. 2016; Simo-Serra *et al.* 2015; Rosten, Porter, Drummond 2010; Dosovitskiy *et al*. 2016).

Vaizdo požymių taškais grįstą akies tinklainės vaizdų sutapdinimą sudaro keli žingsniai: pradinis vaizdo apdorojimas, būdingųjų vaizdo požymių taškų išrinkimas, lokalusis vaizdo požymių aprašymas, požymių taškų porų paieška, geriausių požymių taškų porų atranka ir vaizdų geometrinė transformacija ir sutapdinimas (3.1 pav.). Būtent lokaliojo vaizdo požymių aprašymo žingsnyje galima pritaikyti šiame poskyryje tiriamą SDNT grįstą algoritmą.



3.1 pav. Dviejų akies tinklainės vaizdų sutapdinimo procesas
Fig. 3.1. Registration process of two eye retina images

Atlikti keli naudingi vaizdo požymių taškų aprašymo algoritmų tyrimai ne tik su akies tinklainės vaizdais, kadangi naudojami panašūs vaizdo analizės algoritmai. Pirmiausia savaime organizuojantys žemėlapių algoritmai ir Gaboro filtrai pritaikyti baltymų dėmių sutapdinimui skirtinguose dvimačiuose elektroforezės gelių vaizduose (Serackis *et al*. 2017). Čia taikyti Gaboro filtrai yra artimi SDNT sąsūkos filtrams, tik sudaromi rankiniu būdu. Vėliau, palyginus tradicinius vaizdų požymių taškų paieškos ir aprašymo algoritmus, nustatyta, kad SURF algoritmas geriausiai tinka požymiams aprašyti monochrominių skystųjų kristalų ekranų vaizdams sutapdinti (Šabanovič, Matuzevičius 2015). Pritaikius SDNT grįstą

požymių aprašymo algoritmą DNN-D (Simo-Serra *et al.* 2015), kuris mokytas naudojant ne akies tinklainės vaizdus, gautas akies tinklainės vaizdų porų sutapdinimo tikslumas buvo geresnis nei BRISK ir FAST algoritmų bei artimas SURF algoritmo tikslumui (Šabanovič, Matuzevičius 2017). Tiriant požymių aprašymo algoritmus, remtasi programų algoritmais, aprašytais Navakausko ir bendraautorių monografijoje (Navakauskas *et al*. 2015). Gauti tyrimų rezultatai leido kelti prielaidą, kad sukurtą SDNT grįstą požymių deskriptorių mokant, naudojant akies tinklainės vaizdus, galima gauti požymių vektorius, leidžiančius tiksliau sutapdinti akies tinklainės vaizdus nei DNN-D.

Šiame poskyryje aprašomas sukurtas SDNT grįstas vaizdo požymių aprašymo algoritmas, skirtas tinklainės vaizdams sutapdinti. Šis algoritmas sukurtas taikant 2015 m. Simo Seros aprašytą procesą (Simo-Serra *et al.* 2015) ir disertacijoje pateiktą sukurtą SDNT projektavimo metodiką. Sprendimas taikyti minėtą procesą grįstas anksčiau atliktu požymių paieškos ir aprašymo algoritmų palyginimo eksperimentu (Šabanovič, Matuzevičius 2017). Aprašomas SDNT tinklo pagrindu sukurtas lokaliojo požymių aprašymo algoritmas ir jo mokymas taikant *Siaminę* architektūrą, mokymui naudojant tinklainės vaizdų iškarpas ir atlikta patikra naudojant sutapdinimo patikrai skirtą duomenų rinkinį.

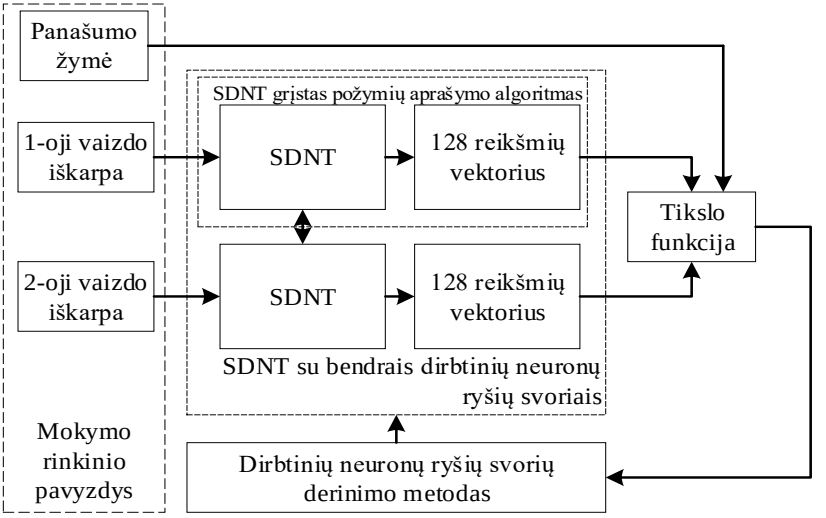
SDNT mokymui taikant *Siaminę* architektūrą, sudarytas specializuotas tinklainės vaizdų duomenų rinkinys. Šis rinkinys sudarytas surinkus ir apdorojus nuotraukas iš 9 duomenų bazių, prieinamų internete: *Chase DB* (*Retinal Images Database... 2018*), *Diaret DB* (Kauppi *et al*. 2006; Kauppi *et al*. 2007, DTSET1 (Golabbakhsh, Rabbani 2013), DTSET2 (Mahmudi, Kafieh, Rabbani 2014), *HRF base* (Köhler *et al*. 2013), tris *Messidor* duomenų rinkiniai (Decencière *et al*. 2014) ir RODREP (Adal *et al*. 2015).

Lyginamasis požymių aprašymo algoritmų įvertinimas atliktas naudojant FIRE duomenų rinkinį (Hernandez-Matas *et al*. 2017). FIRE duomenų rinkinyje yra 134 akies tinklainės nuotraukų poros. Šias poras sudaro tos pačios akies tinklainės nuotraukos, kurios gautos su įvairiu poslinkiu. Kiekvienai vaizdų porai yra pateikiama po 10 ekspertų atrinktų būdingųjų vaizdo požymių taškų porų. Šie taškai leidžia nustatyti, kaip tiksliai sutapdinamos nuotraukos iš skirtingų vizitų. Taip pat taškų poras galima naudoti atliekant lokaliojo požymių aprašymo algoritmų vertinimą ir palyginimą (Hernandez-Matas *et al*. 2017).

3.1.1. Požymių aprašymo algoritmo, grįsto sąsūkos dirbtinių neuronų tinklu, mokymas

Duomenų rinkinį SDNT mokymui sudaro pavyzdžiai iš 3 elementų: dviejų vaizdo iškarpų ir jų panašumo žymės. Mokymo metu SDNT grįstas požymių aprašymo algoritmas veikia kaip dalis *Siaminio* tinklo. Čia tas pats SDNT grįstas požymių aprašymo algoritmas naudojamas lygiagrečiai atlikti dviejų vaizdų analizę, kurios

metu apskaičiuojami požymių vektoriai ir tikslo funkcijos vertė (3.2 pav.). Vaizdo fragmentai yra iškirptos vaizdo dalys, kurių centro koordinatės akies tinklainės vaizde sutampa su būdingojo vaizdo taško koordinatėmis. Vaizdo iškarpų panašumo žymuo nurodo, ar jos priklauso tam pačiam požymio taškui, ar ne.



3.2 pav. Mokymui naudota *Siaminio* tinklo struktūra
Fig. 3.2. Outline of Siamese network structure used for training

Siaminiai tinklai, naudojami SDNT mokyti, yra sudaryti iš dviejų identiškų SDNT, kurie naudoja bendrus mokymo metu derinamus dirbtinių neuronų ryšio svorių koeficientus, tačiau kiekvienas apdoroja skirtingą vaizdo iškarpą (3.2 pav.). Kiekvieno SDNT išėjime gaunamas 128 verčių vektorius, aprašantis vaizdo sritį ties požymio tašku. Tarp gautų dviem iškarpoms vektorių apskaičiuojamas Euklido atstumas d. Šis atstumas panašiems vaizdams turėtų artėti prie 0, o nepanašiems vaizdams būti didesnis nei nustatyta ribinė vertė *m*. Tinklui mokyti taikoma tikslo funkcija užrašoma taip:

$$l = 0,5\left[t \cdot d^2 + (1-t)(\mathrm{m} - d)^2 \right], \tag{3.1}$$

čia l – tikslo funkcijos vertė; t – tikslas, kuris lygus 1, kai vaizdo iškarpos panašios ir 0 – kai nepanašios; d – Euklido atstumas; m – pasirinkta siektino atstumo tarp nepanašių vaizdo iškarpų konstanta.

Tikslo funkcijoje (3.1) m konstanta nustato, koks yra siektinas Euklido atstumas tarp nepanašių vaizdo iškarpų požymius aprašančių vektorių. Atliekant

bandymus pasirinkta konstantos vertė 5, nes SDNT mokymui pasirenkant didesnes ar mažesnes reikšmes, buvo gaunama didesnė tikslo funkcijos vertė. Didesnė tikslo funkcijos vertė, reiškia, kad SDNT veikė prasčiau.

Mokymui naudotas ADAM stochastinis mokymo algoritmas. Nustatyti šie algoritmo parametrai: pirmosios eilės momento koeficientas $\beta_1 = 0{,}9$, antrosios eilės momento koeficientas $\beta_2 = 0{,}999$ ir mažosios pastoviosios vertės koeficientas $\varepsilon = 10^{-6}$. Mokymo metu buvo naudojami skirtingi mokymo žingsnio koeficientai.

Sudarytas duomenų rinkinys požymių aprašymo algoritmui mokyti, taikant *Siaminę* tinklo struktūrą. Atliekant tokį mokymą, pageidautina turėti tos pačios akies tinklainės nuotraukų poras ar sekas, kuriose sužymėti sutampantys būdingųjų vaizdo požymių taškai. Toks yra FIRE duomenų rinkinys, skirtas akies tinklainės nuotraukų sutapdinimo algoritmų veikimui įvertinti. Tačiau jo naudoti sudarant SDNT mokymo rinkinį nebuvo galima, nes jis naudotas atliekant SDNT grįsto lokaliojo požymių aprašymo algoritmo vertinimą ir palyginimą su tradiciniais lokaliojo požymių aprašymo algoritmais. Dėl to duomenų rinkinys SDNT mokymui sudarytas iš kitose oftalmologinių nuotraukų duomenų bazėse prieinamų akies tinklainės vaizdų. Siekiant pagerinti vaizdo požymių aprašymo algoritmo invariantiškumą geometriniams ir šviesumo iškraipymams vaizdai buvo papildomai pakeičiami. Visą duomenų rinkinio kūrimo procesą galima suskaidyti į kelis žingsnius.

Akies tinklainės nuotraukos surinktos iš 9 duomenų rinkinių, prieinamų internete. Pasirinktų duomenų rinkinių sąvadas, įtraukiantis nuotraukų skaičių, raiškas ir nuotraukoms taikytus dydžio mastelius, pateiktas 3.1 lentelėje.

3.1 lentelė. Akies tinklainės vaizdų rinkiniai ir naudoti masteliai
Table 3.1. Eye Retina image datasets and scales used

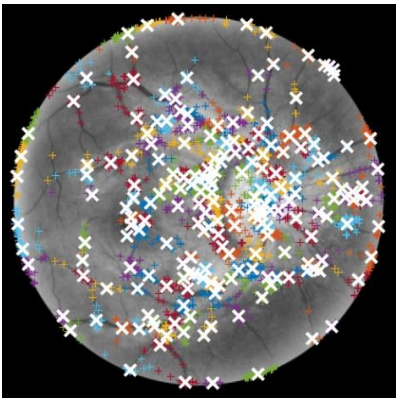
Duomenų rinkinys	Vaizdų kaičius	Raiška, pikseliai	Naudoti masteliai
Chase DB	28	999×960	1,0; 0,75; 0,5
Diaret DB	219	1500×1152	1,0; 0,75; 0,5; 0,25
DTSET1	88	1200×1143	1,0; 0,75; 0,5
DTSET2	100	1612×1536	1,0; 0,75; 0,5
HRF-base	141	3504×2336	1; 0,8; 0,6; 0,4; 0,2
Messidor1	400	1440×960	1; 0,8; 0,6; 0,4
Messidor2	400	2240×1488	1; 0,8; 0,6; 0,4
Messidor3	400	1440×960	1; 0,8; 0,6; 0,4
RODREP	1112	2000×1312	1; 0,8; 0,6; 0,4
FIRE	265	2912×2912	0,25
Iš viso vaizdų	3153		

Iš viso surinkta 3153 vaizdų prieš atliekant dirbtinį papildymą. Duomenų rinkiniai buvo atrinkti atsižvelgiant į juose pateikiamų akies tinklainės vaizdų kokybę ir raišką. Šie duomenų rinkiniai pateikiami skirtingų medicinos institucijų bei surinkti naudojant įvairius akies tinklainės fotografavimo prietaisus ir jų parametrus. Dėl to atsirandantys vaizdų skirtumai turėtų pagerinti kuriamo požymių aprašymo algoritmo invariantiškumą jiems. Daugelis atrinktų vaizdų yra aukštos raiškos, todėl jiems buvo taikomi mastelio mažinimo koeficientai, taip gaunant ir daugiau pavyzdžių ir papildant sudaromą duomenų rinkinį įvairesniais tinklainės nuotraukų masteliais. Atlikus nuotraukų mastelių mažinimą, gauta 11 279 nuotraukos.

Mokant lokaliojo požymių aprašymo algoritmą, grįstą SDNT, naudojamos tinklainės vaizdų iškarpos, esančios aplink požymių taškus. Požymių taškai, jų ir aplink juos esančios vaizdo dalies iškarpoms gauti, atrinkti taikant požymių detektorius. Siekiant išvengti požymių priklausymo nuo vieno iš detektorių, būdingųjų požymių taškai atrinkti taikant detektorių rinkinį, sudarytą iš: FAST (Rosten, Porter, Drummond 2010), *Harris* (Harris, Stephens 1988), *MinEigen* (Shi, Tomasi 1994), *Hessian* (Mikolajczik, Schmid 2002), BRISK (Leutenegger, Chli, Siegwart 2011), SURF (Bay *et al*. 2008), SIFT (Lowe 2004), KAZE (Alcantarilla, Bartoli, Davison 2012). Naudojant kiekvieną iš jų, atrinkta po 500 taškų. Tie taškai sugrupuoti į 200 klasterių taikant k-vidurkių klasterizavimo (angl. *k-means clusterization*) algoritmą. Kiekvieno klasterio vidurkio taškas naudotas kaip požymio srities centras (3.3 pav.), aplink kurį esanti pasirinkto dydžio vaizdo sritis iškerpama. Pasirinktas pradinis iškarpų dydis 101×101 pikselių, kad liktų vietos atlikti geometrines transformacijas prieš iškerpant galutines iškarpas. Tokiu būdu gauta apie 100 iškarpų kiekvienai iš 11 279 nuotraukų.

Pradinių iškarpų kiekis padidintas dar 4 kartus atlikus vertikaliojo, horizontaliojo ir kryžminio veidrodinio atspindžio transformacijas. Šių iškarpų kiekis padidintas dar 4 kartus atlikus pasukimą 90, 180, 270 laipsnių kampu. Taip gauta apie 1600 unikalių iškarpų kiekvienai iš 11 279 nuotraukų.

Iškarpos gautos iš nesuporuotų vaizdų, todėl sudarant mokymo poras taikytas vaizdų papildymas. Panašių iškarpų poros sintezuotos iš tos pačios iškarpos naudojant vaizdo iškraipymus ir papildymus, kurių parametrai parenkami atsitiktinai nustatytose ribose. Vaizdo papildymai ir iškraipymai bei jų ribos parinktos taip, kad gerėtų lokaliojo požymių aprašymo algoritmo, grįsto SDNT, invariantiškumas geometriniams iškraipymams. Naudotas iškarpos pasukimas iki 10 laipsnių prieš ir pagal laikrodžio rodyklę, projekcijos iškraipymai, tokie kaip pasvirimas, mastelio keitimas, perspektyvos transformacijos. Galiausiai, iš papildytų iškarpų iškirpta centrinė 64×64 pikselių dydžio vaizdo sritis. Šio proceso metu sukurta 4 kiekvieno pradinio vaizdo versijų, kurios naudotos formuojant panašių ir nepanašių paveikslų poras.



3.3 pav. Aptiktų vaizdo požymių taškų k-vidurkių klasterizavimo pavyzdys
Fig. 3.3. Example of k-means clustering of detected keypoints

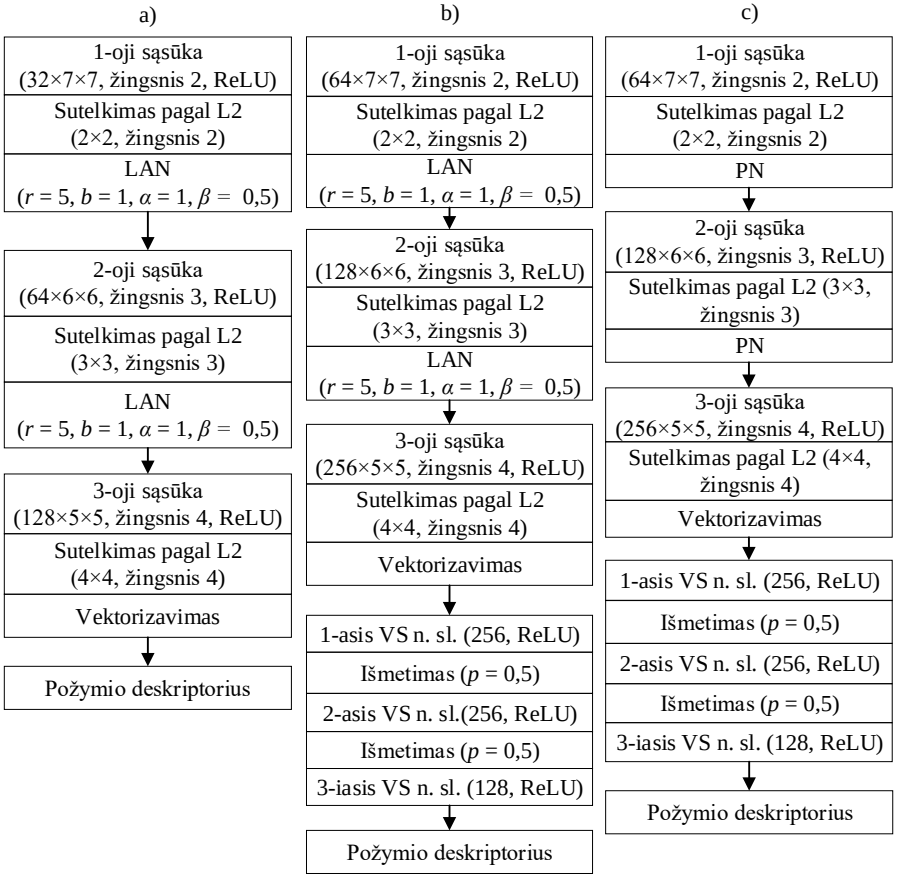
Suformavus iš iškarpų poras, gauta apie 17 milijonų pavyzdžių. Kiekvieną tokį pavyzdį sudaro tinklainės nuotraukos iškarpų pora ir panašumo žymuo. Nepanašių iškarpų poros žyma lygi „0“, o panašių – „1“. Kiekvienas tipas sudaro pusę rinkinio.

Duomenų rinkinys mokymo metu dalijamas į dvi dalis: mokymo ir validavimo. Mokymui skiriama penkios šeštosios duomenų rinkinio pavyzdžių, o validavimui – viena šeštoji dalis pavyzdžių. Validavimas vykdomas kas nustatytą mokymo žingsniu periodą ir leidžia stebėti SDNT veikimo tikslo funkcijos raidą apdorojant nematytus duomenis. Jei mokymo ir validavimo metu tikslo funkcijos reikšmių gradientai yra priešingo ženklo, tai reiškia, kad SDNT persimokė. SDNT persimoko, kai jo aproksimuota funkcija tampa per sudėtinga ir per daug pritaikyta mokymo rinkiniui, nutolstant nuo apibendrintos duomenų tendencijos. Vertinant validavimo tikslo funkcijos reikšmes, galima atrinkti geriau ar blogiau veikiančius SDNT.

Mokymas atliktas dviem būdais: naudojant tik mokymo rinkinį, naudojant mokymo rinkinį ir atrinktus sunkiai išmokstamus paveikslus. Mokymas pasitelkiant atrinktus sunkiau išmokstamus pavyzdžius pagerino lokaliojo požymių aprašymo algoritmo, grįsto SDNT, veikimo kokybę. Iš viso buvo atrinkta 1 milijonas sunkiai išmokstamų pavyzdžių po mokymo naudojant tik mokymo rinkinį. Sunkiai išmokstami pavyzdžiai buvo atrenkami remiantis tikslo funkcijos rezultatu. Buvo atrenkamos tos panašių iškarpų poros, kurių apskaičiuotų požymių vektorių Euklido atstumas buvo didesnis už pasirinktą ribą, ir nepanašios poros, kurių atstumas buvo mažesnis už pasirinktą ribą. Tokiu būdu mokymo metu buvo praleidžiami pavyzdžiai, kuriuos DNT buvo lengva atskirti, dėl to padidėjo mokymo sparta.

3.1.2. Sąsūkos dirbtinių neuronų tinklų struktūros vaizdo požymiams aprašyti

Pasirinkta mokyti ir validuoti kelias sąsūkos dirbtinių neuronų tinklų (SDNT) struktūras (3.4 pav.). Naudoti dviejų tipų SDNT: visiškai sąsūkos dirbtiniai neuronų tinklai (VSDNT) ir SDNT su visiškai sujungtųjų (VS) dirbtinių neuronų sluoksniais (SDNT-VS). Pagrindinis skirtumas tarp jų yra sąsūkos filtrų dydžiai



3.4 pav. Naudotos sąsūkos dirbtinių neuronų tinklų struktūros: a) visiškai sąsūkos dirbtiniai neuronų tinklai, b) sąsūkos dirbtiniai neuronų tinklai su visiškai sujungtųjų neuronų sluoksniais ir lokaliojo atsako normalizavimu, c) sąsūkos dirbtiniai neuronų tinklai su visiškai sujungtųjų neuronų sluoksniais ir paketiniu normalizavimu

Fig. 3.4. Structures of used convolutional neural networks: a) fully convolutional network, b) convolutional neural network with fully connected layers and local response normalization, c) convolutional neural network with fully connected layers and batch normalization

ir žingsnis, taip pat lokaliojo atsako normalizavimo (LAN) ar paketinio normalizavimo (PN) sluoksniai. Šiame skirsnyje aprašytos taikytos SDNT struktūros.

Ištirtos 2 tipų SDNT struktūros. Pirmojo tipo struktūra yra pateikta 3.4 pav. a). Šią SDNT struktūrą sudaro 3 blokai. Kiekviename bloke yra sąsūkos, sutelkimo ir LAN sluoksniai. Šioje struktūroje naudojami vien sąsūkos neuronų sluoksniai, todėl ji vadinama visiškai sąsūkos dirbtinių neuronų tinklu (VSDNT). Antrojo tipo struktūra yra pateikta 3.4 pav. b) ir c). Šią SDNT struktūrą sudaro 3 blokai su sąsūkos neuronų, sutelkimo ir normalizavimo sluoksniais ir vienu bloku iš 3 VS neuronų sluoksnių, tarp kurių yra išmetimo sluoksniai. VS sluoksniai leidžia tinklui išmokti sudėtingesnius sąryšius tarp vaizdų požymių. Išmetimo (angl. *dropout*) sluoksniai, su 0,5 išmetimo tikimybe, taikyti tarp VS sluoksnių, siekiant sumažinti permokymo tikimybę. Dėl to, kad po sąsūkos neuronų sluoksnių naudojami VS sluoksniai, šios struktūros tinklai pavadinti sąsūkos dirbtinių neuronų tinklais su visiškai sujungtųjų neuronų sluoksniais (SDNT-VS). 3.4 pav. b) ir 3.4 pav. c) pateiktos SDNT-VS struktūros skiriasi taikomais normalizavimo sluoksniais. Pirmojoje naudojami LAN sluoksniai, o antrojoje – PN sluoksniai.

3.1.3. Būdingųjų požymių taškų sutapdinimo taikant sudarytus ir tradicinius požymių aprašymo algoritmus palyginimas

Vertinant SDNT grįstų lokaliojo požymių aprašymo algoritmų veikimą, jie palyginti su žmogaus sukurtais aprašymo algoritmais. Vertinimui buvo naudojamos FIRE duomenų rinkinio (Hernandez-Matas *et al*. 2017) būdingųjų vaizdo požymių taškų poros. Požymius aprašantys algoritmai palyginami pagal teisingai surastų porų dalį procentais, lyginant su visomis esamomis poromis, kurių yra 1340. Ši požymių aprašymo metodų veikimo metrika, pavadinta *Rank-1*, pateikiama 3.2 lentelėje.

Iš mokytų ir išbandytų įvairių SDNT struktūrų atrinkti 5 variantai nuo DNN-D-IR(1) iki DNN-D-RI(5). DNN-D-RI(1) yra VSDNT mokytas tik mokymo duomenų rinkinyje, jo struktūra pateikta 3.4 pav. a). DNN-D-IR(2) yra VSDNT papildomai mokytas sunkiau išmokstamais pavyzdžiais (3.3 pav. a). DNN-D-IR(3) yra SDNT-VS su lokaliojo atsako normalizavimu (LAN) mokytas naudojant tik mokymo vaizdų rinkinį, jo struktūra pateikta 3.3 pav. b). DNN-D-RI(4) yra SDNT-VS su LAN, tačiau papildomai mokytas sunkiai išmokstamais vaizdais. DNN-D-RI(5) yra SDNT-VS su paketiniu normalizavimu (PN) mokytas naudojant mokymo vaizdų rinkinį ir sunkiai išmokstamais pavyzdžiais.

Iš gautų DNN-D-RI rezultatų galima spręsti, kad 2 ir 4 variantai, mokyti naudojant atrinktus sunkiau išmokstamus pavyzdžius, sutapdina teisingai didesnį skaičių porų lyginant su 1 ir 3 variantais. Taigi, sunkiau išmokstamų pavyzdžių

naudojimas leidžia pagerinti SDNT grįsto lokaliojo požymio veikimo kokybę remiantis *Rank-1* metrika.

3.2 lentelė. Lokaliojo požymių aprašymo algoritmų palyginimas
Table 3.2. Comparison of local feature descriptors

Parinktų požymių aprašymo algoritmai	*Rank-1*, %	Vidutinė apdorojimo trukmė, ms	Išmoktų požymių aprašymo algoritmai	*Rank-1*, %	Vidutinė apdorojimo trukmė, ms
BRISK	40,9	16,2	DNN-D-RI(1)	35,4	1,4
FREAK	22	2,4	DNN-D-RI(2)	65,30	1,3
SIFT	99,3	21	DNN-D-RI(3)	84,9	1,4
SURF	98,8	0,37	DNN-D-RI(4)	89,2	1,4
KAZE	95	3,4	DNN-D-RI(5)	62,5	1,4

Pastaba**.** Iš kairės į dešinę pilku fonu pažymėti tiksliausias iš parinktų požymių aprašymo algoritmų, greičiausias požymių aprašymo algoritmas, tiksliausias iš išmoktų požymių aprašymo algoritmų.

3 ir 4 variantai, kurie gauti naudojant sąsūkos ir visiškai sujungtuosius neuronų (VS) sluoksnius, yra tikslesni nei 1 ir 2 variantai, kurie gauti naudojant tik sąsūkos sluoksnius. Iš to galima spręsti, kad VS sluoksniai išmoksta išgauti daugiau naudingos informacijos iš vaizdo požymių, kuriuos išskyrė sąsūkos neuronų sluoksniai, todėl juos patartina naudoti SDNT grįsto lokaliojo požymio veikimo kokybei, vertinamai pagal *Rank-1* metriką, pagerinti.

5 variantas, naudojant PN sluoksnius, tapdino taškus prasčiau, lyginant su 4 variantu, kuriame taikyti LAN sluoksniai. Iš to galima spręsti, kad LAN veikia geriau SDNT naudojant vaizdo požymiams aprašyti.

Iš aprašytų SDNT grįstų lokaliojo požymių aprašymo algoritmų variantų daugiausia taškų porų teisingai sutapdinta naudojant DNN-D-IR(4) variantą.

Išmoktų požymių aprašymo algoritmai DNN-D-IR palyginti su žmogaus parinktų požymių aprašymo algoritmais: BRISK, FREAK, SURF, KAZE (*Matlab* priemonių rinkinio įgyvendinimu) ir SIFT (*VLFeat* įgyvendinimu) (Vedaldi, Fulkerson 2010). 3.2 lentelėje pateikiami vaizdo lokaliojo požymių aprašymo algoritmų kokybės įverčiai pagal *Rank-1* metriką ir vidutinio vykdymo trukmės matavimo rezultatai.

Taikant binarinius požymių aprašymo algoritmus BRISK ir FREAK požymių taškų poros sutapdintos teisingai mažiau nei pusėje atvejų. Taikant SIFT, SURF ir KAZE mažiausiai 95 % iš 1340 požymių taškų porų sutapdintos teisingai.

Poskyryje aprašyti išmoktųjų požymių SDNT grįsti požymių aprašymo algoritmai teisingai sutapdina iki 89,18 % visų porų.

Žmogaus parinktųjų požymių aprašymo algoritmai BRISK, FREAK, KAZE įgyvendinti *Matlab* ir SIFT įgyvendintas *VLfeat* įrankių programiniais paketais ir vykdomi CPĮ apdoroja vieną vaizdo požymio sritį lėčiau nei išmoktasis požymių aprašymo algoritmas DNN-D-IR, sukurtas ir įgyvendintas taikant *Tensorflow* programinę įrangą. Tyrimo metu palyginimui atlikti *Tensorflow* vykdomas DNN-D-IR algoritmas su *Matlab* sujungtas naudojant TCP/IP sąsajos prievadus. *Matlab* vienu metu gali pateikti paketais po 10 vaizdo fragmentų DNN-D-IR apdoroti, dešimties vaizdų paketais požymius bandymų metu apdorojo visi palyginti lokaliojo požymių aprašymo algoritmai. SURF įgyvendintas *Matlab* pateikė požymių aprašymo vektorius greičiausiai, kadangi jis nuo pat pradžios ir buvo kurtas maksimaliai greitam požymių aprašymui atlikti.

3.2. Sąsūkos dirbtinių neuronų tinklas kelio dangos tipui ir būklei nustatyti

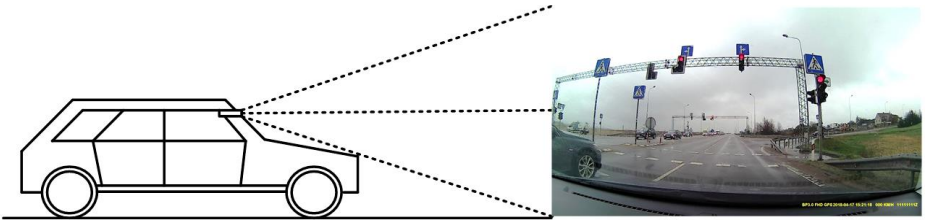
Vaizdų analizė, skirta keliui priešais transporto priemonę suvokti, yra sparčiai tobulėjanti sritis. Dažniausiai analizuojami vaizdai iš monokuliarinių ir binokuliarinių (stereoskopinių) kamerų. Analizės rezultatai naudojami pažangiosiose vairuotojo pagalbinėse sistemose (Krasner, Katz 2016; Chandran *et al*. 2014) arba savivaldėse transporto priemonėse (Yang *et al*. 2018; Mahmud, Arafat, Zuhori 2012; Milz *et al*. 2018). Vaizdų analizė leidžia nustatyti prasto matomumo sąlygas, orų sąlygas ir apšvietimą (Gimonet, Cord, Saint Pierre 2015; Cheng, Zheng, Murase 2018), rasti kelią ir aptikti kliūtis (Nadav ir Katz 2016), atpažinti kelio juostas ir kelkraščius (Yang *et al*. 2018; Hamme, Veelaert, Philips 2013; Zhang, Wu 2009). Binokuliarinių vaizdų analizė gali būti efektyviai naudojama objektams atpažinti ir leidžia pasiekti didesnį lokalizavimo tikslumą lyginant su monokuliarinių vaizdų analize. Ypač binokuliariniai vaizdai reikalingi, kai reikia sukurti gylio žemėlapius ir trimačius taškų debesis iš vaizdo duomenų, pasiekiant panašius ar geresnius rezultatus nei naudojant LIDAR (Smolyanskiy, Kamenev, Birchfield 2018).

Dauguma vaizdų gali būti apdorojami taikant histogramas, slenksčius ir kitus tradicinius vaizdų apdorojimo algoritmus (Oliveira, Lobato Correia 2008), tačiau dažnėja giliųjų dirbtinių neuronų tinklų (GDNT) taikymas aplinkai suvokti, objektams aptikti, lokalizuoti ir sekti ir sprendimams priimti (Smolyanskiy, Kamenev, Birchfield 2018; Meignen, Bernaded, Briand 1997). Įterptinėse sistemose didelė apdorojimo sparta pasiekiama naudojant integruotą GPĮ, tokios įterptinės sistemos pavyzdys yra *Nvidia Jetson TX2* (Smolvanskiy, Kamenev, Birchfield 2018). Lyginamųjų duomenų rinkinių, pavyzdžiui, KITTI (Menze,

Geiger 2015) ir *Udacity* (*Udacity...* 2017) atsiradimas skatina vis geresnių algoritmų kūrimą. Taip pat atsiranda tyrimų naudojant sintezuotus duomenis (Gimonet, Cord, Saint Pierre 2015) ir naujai surinktus duomenis, pavyzdžiui, SAIC (Yang *et al*. 2018). Tyrėjai parodė, kad GDNT yra jautrūs įvairioms fizinio pasaulio atakoms, pvz., suklastotiems objektams, pvz., ženklams ar iškraipymams, kurie gali sukelti netinkamą klasifikavimą ir neteisingą atpažinimą (Eykholt *et al*. 2018). Būtina kurti tokius GDNT, kurie būtų nejautrūs suklastotiems vaizdams bei atsižvelgtų į tinkamą ženklų vietą erdvėje prieš bei gebėtų atskirti tikrus ir galimai suklastotus ženklus, tai yra svarbu funkciniam saugumui užtikrinti (Rao, Frtuniki 2018).

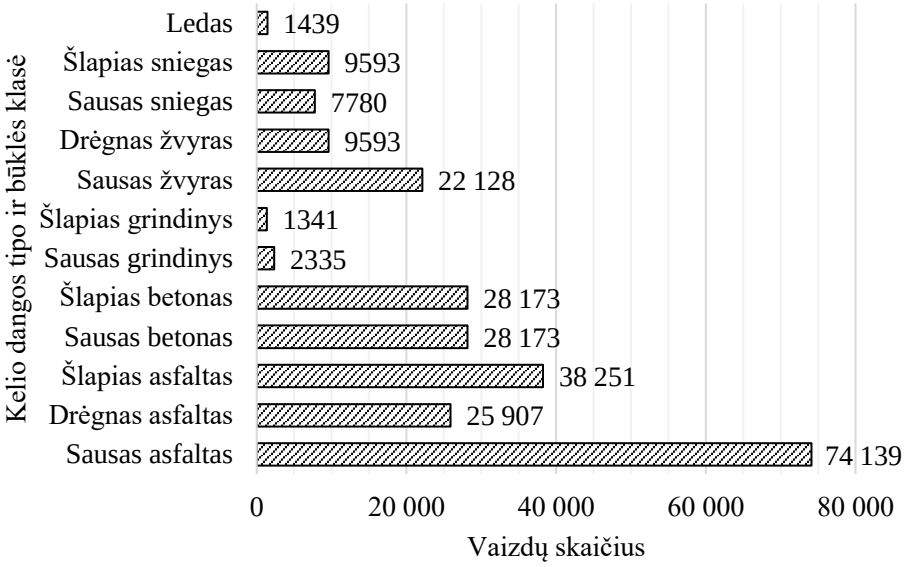
SDNT gali būti naudojamas oro sąlygoms ir apšvietimui nustatyti (Cheng, Zheng, Murase 2018), kelio dangai nustatyti (Naguib, Kim, Lee 2017) iš vaizdų. Taip pat yra tyrimų, kai SDNT naudojami kelio pažaidoms, įtrūkiams ir kitoms kelio pažaidoms nustatyti (Cafiso *et al*. 2017; Shen 2016; Eykholt *et al*. 2018; Meignen, Bernadet, Briand 1997), atliekant statmenai paviršiui nufotografuotų kelio dangos vaizdų analizę.

Šiame poskyryje aprašomas sukurtas SDNT grįstas algoritmas kelio dangos tipui ir būklei nustatyti. Pasirinkti tyrimai šioje srityje, nes dar nėra tirtas SDNT taikymas kelio paviršiaus tipui ir būklei nustatyti iš vaizdų, nufilmuotų automobilio priekyje sumontuota vaizdo kamera (3.5 pav.). Nuspręsta, kad SDNT bus taikomas dvylikai skirtingų kelio dangos tipo ir būklės kombinacijų atpažinti (3.6 pav.).



3.5 pav. Kameros montavimo schema ir kelio vaizdas matymo lauke
Fig. 3.5. Camera mounting scheme and image in field of view

Naudojant kamerą, sumontuotą automobilio priekyje ir nukreiptą į kelią taip, kad kameros matymo lauko vertikalusis centras būtų ties horizontu (3.5 pav.), surinkta daugiau nei 250 tūkst. vaizdo kadrų. Šie vaizdai rankiniu būdu suskirstyti į 12 pasirinktų kategorijų: ledas, šlapias sniegas, sausas sniegas, drėgnas žvyras, sausas žvyras, šlapias grindinys, sausas grindinys, šlapias betonas, sausas betonas, šlapias asfaltas, drėgnas asfaltas ir sausas asfaltas. Surinktų kiekvienos kategorijos pavyzdžių skaičius parodytas 3.6 paveiksle.



3.6 pav. Surinktų vaizdų, suskirstytų į 12 klasių, skaičiai
Fig. 3.6. Numbers of collected data in 12 separate classes

Šios rūšies sistemos išvestis gali būti naudinga pažangiosioms vairuotojo pagalbinėms sistemoms, ratų slydimo greitėjant ir stabdant prevencijos sistemoms, stabilumo sistemoms. Šios sistemos įprastai pasirenka tinkamiausią veikimo algoritmą, jau prasidėjus stabdymui ar slydimui, todėl maksimalus jų efektyvumas pasiekiamas ne iš karto. Veikimo efektyvumą galima pagerinti iš anksto nustatant kelio dangos tipą ir būklę. Turint šią informaciją, galima iš anksto pritaikyti tinkamą valdymo algoritmą tam tikros dangos ir jos būklės nulemiamai sankibai tarp padangos ir kelio dangos. Duomenys, kurie gaunami iš sukurtos sistemos, leistų iš anksto pasirinkti reikiamą algoritmą.

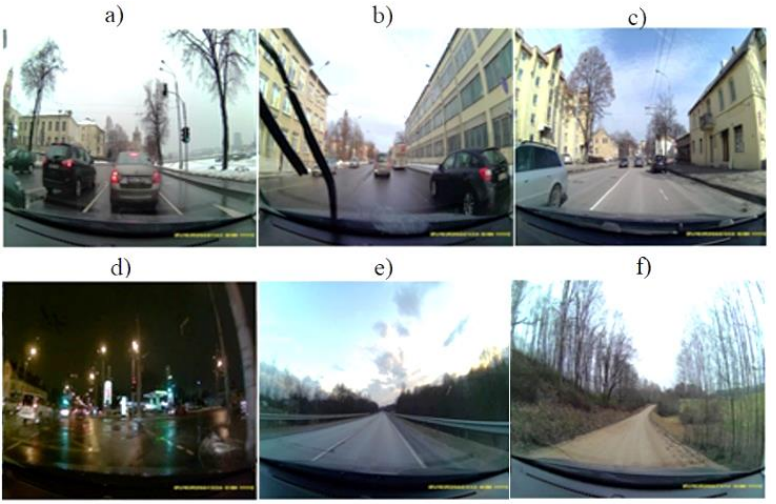
Taip pat sukurtas binokulinės regos skaitinis modelis. Šis modelis aprašo regimosios erdvės atvaizdavimą į erdvės subjektyvų suvokinį (Šabanovič, Matuzevičius, Vaitkevičius 2016). Toks modelis leidžia įvertinti atstumą iki objektų, esančių vaizde, jame aprašytą binokulinio vaizdo analizatoriaus struktūrą pritaikius dirbtiniam sąsūkos neuronų tinklui, galima būtų SDNT taikyti ne tik kelio dangos tipui ir būklei nustatyti, bet ir atstumams iki kelio objektų nustatyti. Tai yra planuojama tolesnių tyrimų kryptis.

Toliau aprašytas kelio dangos ir būklės vaizdų rinkimas ir pirminis apdorojimas, tinklo struktūros parinkimas, jos mokymas ir įgyvendinimas įterptinėje sistemoje.

3.2.1. Duomenų rinkinio sudarymas kelio dangos tipui ir būklei nustatyti

Sąsūkos dirbtinių neuronų tinklų (SDNT) mokymas reikalauja didelių duomenų rinkinių. Surinktas 250 tūkst. vaizdų rinkinys iš vaizdų, gautų iš vaizdo įrašų, išsaugotų kelionių po miestą ir Lietuvos kelius. Vaizdui įrašyti naudota automobilinė vaizdo kamera, įrašanti vaizdą 1920×1080 raiška. Įrašyti vaizdo įrašai išskaidyti į atskirus vaizdo kadrus, iš kurių atrinkti vaizdai priskiriami vienai iš 12 klasių.

Duomenų rinkinys rinktas įvairiais metų laikais ir važiuojant įvairiomis kelio dangomis, kadangi būtina turėti vaizdus esant skirtingiems metų laikams ir numatytoms oro sąlygoms bei įvairioms kelio dangoms. Keletas vaizdų pavyzdžių parodyti 3.7 paveiksle. Siekiant pritaikyti kuriamą algoritmą veikti esant įvairioms apšvietimo sąlygoms, duomenys rinkti tiek šviesiu paros laiku, tiek ir tamsiu paros laiku (3.7 pav. d).



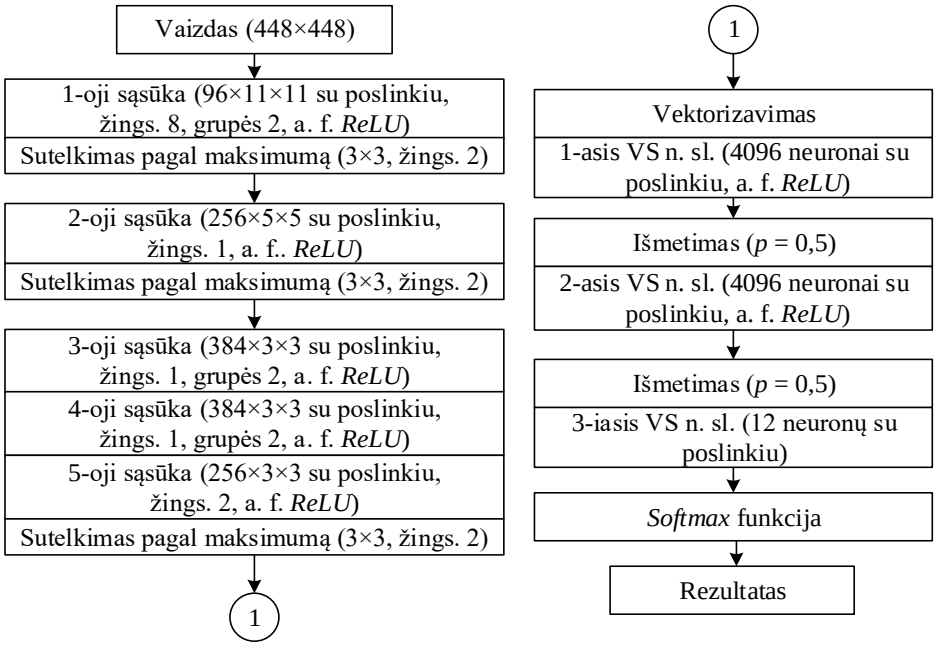
3.7 pav. Kelio dangos tipo ir būklės vaizdų pavyzdžiai: a) šlapias asfaltas; b) drėgnas asfaltas; c) sausas asfaltas; d) šlapias asfaltas naktį; c) sausas betonas; d) šlapias žvyras
Fig. 3.7. Examples of images of road type and conditions: a) wet asphalt; b) moist asphalt; c) dry asphalt; d) wet asphalt at night; e) dry concrete; f) wet gravel

SDNT mokyti vaizdų rinkinys formuojamas, atrenkant apylygį skaičių kiekvienos klasės vaizdų mokymo rinkinius ir vienodus kiekius kiekvienos klasės vaizdų validavimo ir testavimo rinkiniams. Nors mokymo metu gali vyrauti viena klasė, tačiau validavimas ir testavimas leis pastebėti persimokymą atpažįstant vieną ar kitą kelio dangos tipo ir būklės klasę.

Mokymo, validavimo ir testavimo rinkinių vaizdai yra atrenkami iš kadrų, gautų iš skirtingų vaizdo įrašo dalių taip, kad nebūtų atrenkami gretimi vaizdo kadrai. Atsitiktinio išrinkimo atsisakoma, nes yra tikimybė, kad mokymui, validavimui ir testavimui bus panaudoti gretimi laike vaizdo kadrai, kur didžioji vaizdo dalis sutampa. Naudojant panašius vaizdus validavimui ir testavimui, kokie buvo naudoti mokymui, būtų gaunami tikslumo rezultatai, kurie neatitinka SDNT algoritmo tikslumo naujų duomenų ir neleidžia aptikti permokymo.

3.2.2. Sąsūkos dirbtinių neuronų tinklo struktūros pasirinkimas

Naudotas klasifikavimui pritaikytas sąsūkos dirbtinių neuronų tinklas (SDNT), kurio pradinė struktūra atitinka *Alexnet*. Buvo pakeistas įvesties vaizdo dydis ir sluoksnių filtrų dydžiai ir žingsniai, visa tinklo struktūra pateikta 3.8 paveiksle. Vaizdai prieš pateikiant į SDNT sumažinami iki 448×448 pikselių raiškos, kadangi atliktų bandymų rezultatai parodė, kad mažesnės raiškos, pavyzdžiui, 256×256 pikselių vaizdai turi nepakankamai informacijos apie kelio

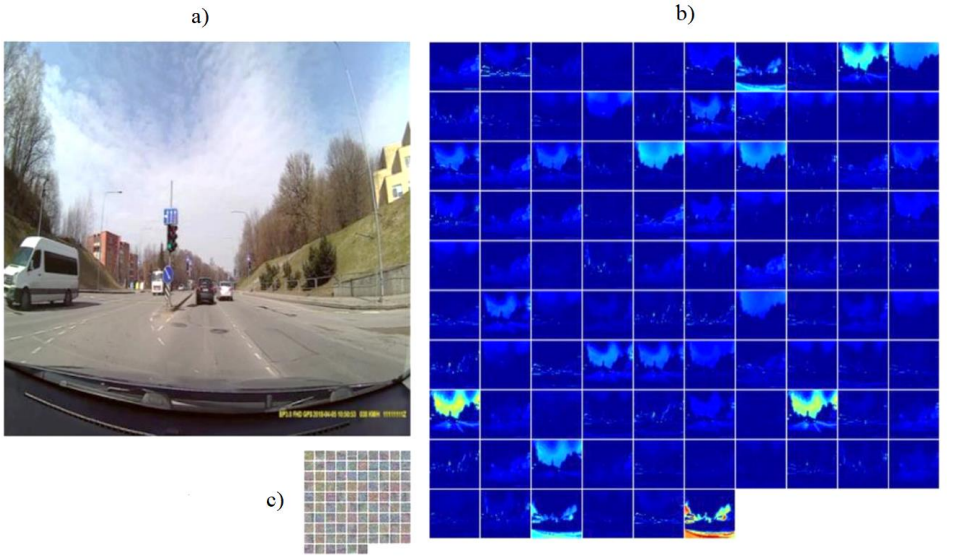


3.8 pav. Sąsūkos dirbtinių neuronų tinklo struktūra kelio dangos tipui ir būklei nustatyti
Fig. 3.8. Structure of convolutional neural network for road pavement type and conditions evaluation

dangą. Atpažįstant kelio dangą, būtina, kad vaizde būtų aiškiai matoma tekstūra, o stipriai sumažinus vaizdą, pavyzdžiui, žvyro ir asfalto tekstūros supanašėja.

Kelio dangos tipui ir būklei nustatyti pritaikytas SDNT sudarytas iš 5 sąsūkos sluoksnių ir 3 visiškai sujungtųjų (VS) dirbtinių neuronų sluoksnių. Visiems dirbtinių neuronų sluoksniams naudojama *ReLU* aktyvavimo funkcija. Siekiant sumažinti klasifikatoriaus priklausomybę nuo atskirų taškų, naudoti išmetimo sluoksniai tarp VS neuronų sluoksnių. Pasirinktas išmetimo koeficientas 0,5 identiškas naudotam *Alexnet*. 1, 3, 4 sąsūkos sluoksnių filtrų operacijos yra išskaidytos į dvi grupes, panašiai kaip tai atlikta originaliame *Alexnet*. Tai leidžia šiek tiek sumažinti sluoksnių skaičiavimo operacijų skaičių. Sudarytas tinklas pakankamas kelio vaizdų požymiams išskirti, tai išsiaiškinta išanalizavus po pirmojo sąsūkos sluoksnio gautus požymių žemėlapius.

3.9 paveiksle matyti, kad pateikus iš kameros nuskaitytą kadrą į apdorojantį tinklą gaunami po pirmosios sąsūkos požymių žemėlapiai rodo skirtingą sluoksnio neuronų reakciją į tamsias, šviesias vaizdo vietas, skirtingas spalvas ir kraštus. Turimi ryšio svoriai pirmajame sluoksnyje išmoksta paprastus požymius, kurie gali būti pasiskirstę įėjimo kadro trijuose spalvų sluoksniuose, kurie atitinka skirtingas paveikslą sudarančias spalvų dedamąsias: raudoną, žalią ir mėlyną.



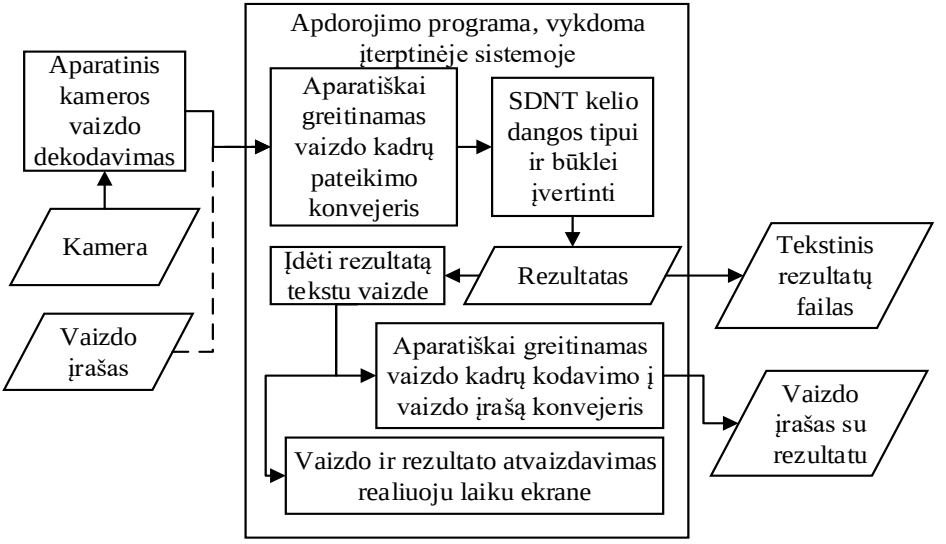
3.9 pav. Pirmojo sąsūkos sluoksnio veikimas: a) pateiktas įėjime vaizdas; b) gauti požymių žemėlapiai; c) naudoti sąsūkos filtrai

Fig. 3.9. Operation of the first convolutional layer: a) input image; b) calculated feature maps; c) used convolutional kernels

Pirmajame sluoksnyje išskiriami baziniai vaizdo požymiai parodo, kad SDNT mokymo procedūra atlikta sėkmingai. Mokymo metu yra derinamos sąsūkos filtrų masyvų reikšmės, kurios naudojamos atliekant sąsūkos operaciją. Atlikus sąsūkos operaciją pirmajame sluoksnyje, turėtų būti išskiriami tokie vaizdo požymiai, kaip tam tikros spalvos, šviesios ar tamsios vaizdo sritys, objektų kraštai, smulki tekstūra. Pirmajame sluoksnyje išskirti vaizdo požymiai naudojami antrajame sluoksnyje, išskiriant iš jų susidedančius požymius. Rastų požymių sritis 3.9 paveiksle parodytuose požymių žemėlapiuose išsiskiria savo intensyvia žalia ar raudona spalvomis, tos vietos, kuriose požymių nerasta, yra tamsiai mėlynos spalvos.

3.2.3. Sąsūkos dirbtinių neuronų tinklo įgyvendinimas įterptinėje sistemoje

Įterptinė sistema *Nvidia Jetson TX2* pasižymi DNT apdorojimo sparta iki 1 TFLOP. Įgyvendinus 3.10 paveiksle pateiktą vaizdų analizės sistemą, kadrai iš vaizdo įrašų arba kameros apdorojami per 20 ms, taikant aparatinį vaizdo perkodavimo įrenginį ir kompiuteriu mokytą modelį konvertavus į taikymui optimizuotą programinę įrangą, naudojant *Nvidia TensorRT*.



3.10 pav. Vaizdo gavimo, apdorojimo ir rezultato pateikimo įterptine sistema schema
Fig. 3.10. Scheme of image capture, processing and result indication using embedded system

Sukurtas SDNT buvo mokytas GPĮ darbo stotimi, naudojant *caffe*, o tada buvo konvertuojamas į *TensorRT* ir išbandytas *Nvidia Jetson TX2* įterptinėje sistemoje, kurios vartojama galia iki 15 W ir suteikiama giliųjų neuronų tinklo skaičiavimų sparta iki 1 TFLOP. Algoritmas apdorojo realiuoju laiku pateikiamą 30 kadrų/s vaizdo įrašo kadrų vaizdus, kadro apdorojimas truko 20 ms. Per šį laiką automobilis, važiuojantis 100 km/h greičiu, nuvažiuotų apie pusę metro. Kelių tipai buvo klasifikuojami iki 84 % tikslumu, kuris yra pakankamas atsižvelgiant į tai, kad klaidos yra atsitiktinio pobūdžio ir pasiskirto laike.

Identifikuojant kelio būklę, dažniausiai kyla sunkumų dėl nevienodo apšvietimo ir šešėlių. Kameros vaizdo jutiklis turi ribotą dinaminį diapazoną, o jo reguliavimas užtrunka ir yra kompromisas tarp tamsių ir šviesių vaizdo vietų, kintant kameros jautrumo šviesai parametrams, sudėtingėja ir atpažinimo užduotis, kadangi sudarytas algoritmas turi užtikrinti atsparumą šviesumo pokyčiams. Taip pat sunkumų sukelia šešėliai nuo viadukų ir tiltų bei tunelių, tamsesnės vietos gali būti palaikomos drėgna danga, tam ateityje reiktų taikyti šešėlių kompensavimo metodus.

Kitas sudėtingas atvejis yra blogas matomumas rūko, stipraus lietaus, krušos ar pūgos atveju – šiais atvejais matomojo šviesos spektro kameros negali garantuoti gero kelio dangos atpažinimo, kaip ir žmogaus rega, tačiau galima nustatyti pačių orų tipą.

Iš vaizdo iškirpus apatinę vidurinę dalį taip, kad 80 % vaizdo iškarpos užpildytų kelias, esantis priešais automobilį, pasiekiamas 90 % klasifikavimo tikslumas. Taip yra dėl to, kad kelio dangos vaizdas užima didesnę vaizdo dalį ir yra daugiau naudingos informacijos apie kelio dangą. Iškerpant vaizdo dalį, joje vis tiek matomi vaizdo glaudinimo metu atsiradę iškraipymai, todėl tikėtina, kad klasifikavimo tikslumas dar padidėtų pagerinus vaizdo kokybę.

3.3. Trečiojo skyriaus išvados

1. Sukurtas lokaliojo požymių aprašymo algoritmas, grįstas sąsūkos neuronų tinklu (SDNT), yra tinkamas akies tinklainės vaizdams sutapdinti, nes patikros metu taikant šį algoritmą gauti požymių vektoriai leido teisingai sutapdinti 89 % vaizdo požymių taškų porų, pateikiamų FIRE akies tinklainės vaizdų rinkinyje.
2. Sintezuotų mokymo vaizdų naudojimas leido mokyti pasirinkto dydžio SDNT, tačiau galėjo apriboti pasiektą sutapdinimo tikslumą, nes nevisiškai atitiko akies tinklainės vaizdų geometrinių iškraipymų įvairovę.

3. Sukurtas kelio dangos tipo ir būklės nustatymo algoritmas, grįstas SDNT, tinkamas kelio dangos tipui ir būklei realiuoju laiku nustatyti, nes pasiekia iki 90 % kelio dangos ir būklės nustatymo tikslumą, o įgyvendintas įterptinėje sistemoje *Nvidia Jetson TX2* pasiekia 20 ms kadro apdorojimo laiką.
4. Vaizdo dalies, kurioje yra kelio paviršius, iškirpimas ir pateikimas tinklui padidina kelio dangos tipo ir būklės klasifikavimo tikslumą nuo 84 % iki 90 %.

Bendrosios išvados

1. Taikant sudarytą sąsūkos dirbtinių neuronų tinklų (SDNT) elementų sąrašą ir pasiūlytą projektavimo metodiką, SDNT sėkmingai įgyvendinti spartinančiuosiuose įrenginiuose dviem vaizdų analizės realiuoju laiku uždaviniams spręsti.
2. Taikant sukurtą metodiką, įgyvendintas vaizdų lokaliojo požymių aprašymo algoritmas, grįstas SDNT, leidžia teisingai sutapdinti 89 % iš FIRE duomenų rinkinio požymių porų, šis rodiklis artimas tradicinių požymių aprašymo algoritmų teisingai sutapdintų porų procentui.
3. Taikant sukurtą metodiką, įgyvendintas algoritmas iš pavienių patikros rinkinio vaizdo kadrų nustato kelio dangos tipą ir būklę 90 % tikslumu.
4. Taikant sukurtą metodiką įterptinėje sistemoje su grafikos posisteme *Nvidia Jetson TX2* įgyvendintas algoritmas kelio dangos tipui ir būklei nustatyti vieną kadrą apdoroja per 20 ms.

Remiantis bendrosiomis išvadomis galima teigti, kad disertacijoje iškelta hipotezė patvirtinta.

Literatūra ir šaltiniai

Abdu-Aljabar, R. D. 2012. Design and Implementation of Neural Network in FPGA, *Journal of Engineering and Development* 16(3): 73–90. ISSN 1813-7822.

Abramoff, M. D.; Garvin, M. K.; Sonka, M. 2010. Retinal Imaging and Image Analysis, *IEEE Reviews in Biomedical Engineering* 3: 169–208. ISSN: 1937-3333.

Adal, K. M.; van Etten, P. G.; Martinez, J. P.; van Vliet, L. J.; Vermeer, K. A. 2015. Accuracy Assessment of Intra and Inter-Visit Fundus Image Registration for Diabetic Retinopathy Screening, *Investigative Ophthalmology & Visual Science* 56(3): 1805–1812. ISSN: 1552-5783.

Abadi, M.; Agarwal, A.; Barham P.; Brevdo, E.; Chen, Z.; Citro, C.; *et al*. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems, *CoRR*. *arXiv:1603.04467v2 [cs.DC]*.

Afifi, A. J.; Hellwick, O. 2016. Object Depth Estimation from a Single Image Using Fully Convolutional Neural Network, in *2016 International Conference on Digital Image Computing: Techniques and Applications (DICTA)*, Gold Coast, Australia: 1–7.

Alahi, A.; Ortiz, R.; Vandergheynst, P. 2012. FREAK: Fast Retina Keypoint, in *2012 IEEE Conference on Computer Vision and Pattern Recognition*: 510–517.

Alcantarilla, P. F.; Bartoli A.; Davison, A. J. 2012. KAZE Features, in *Computer Vision (ECCV 2012)*: 214–227.

AMD A-Series Desktop APUs [interaktyvus]. 2019. AMD [žiūrėta 2019 m. rugsėjo 19 d.]. Prieiga per internetą: <http://www.amd.com/en-us/products/processors/desktop/a-series-apu#>.

AMD Desktop Graphics [interaktyvus]. 2015. AMD. [žiūrėta 2015 m. lapkričio 22 d.]. Prieiga per internetą: <http://www.amd.com/en-us/products/graphics/desktop>.

AMD EPYC products [interaktyvus]. 2019. AMD. [žiūrėta 2019 m. rugsėjo 19 d.]. Prieiga per internetą: <https://www.amd.com/en/products/epyc>.

AMD FX Processors [interaktyvus]. 2019. AMD [žiūrėta 2019 m. rugsėjo 19 d.]. Prieiga per internetą: <http://www.amd.com/en-us/products/processors/desktop/fx#>.

AMD Instinct Accelerators [interaktyvus]. 2019. AMD [žiūrėta 2019 m. rugsėjo 19 d.]. Prieiga per internetą: <https://www.amd.com/en/graphics/servers-radeon-instinct-mi>.

AMD Ryzen Desktop Processors [interaktyvus]. 2019. AMD [žiūrėta 2019 m. rugsėjo 19 d.]. Prieiga per internetą: <https://www.amd.com/en/ryzen>.

AMD Ryzen Threadripper Processors [interaktyvus].]. 2019. AMD [žiūrėta 2019 m. rugsėjo 19 d.]. Prieiga per internetą: <https://www.amd.com/en/products/ryzen-threadripper>.

AMD Embedded G-Series Family of Processors [interaktyvus]. 2015. AMD. [žiūrėta 2015 m. lapkričio 22 d.]. Prieiga per internetą: <http://www.amd.com/en-us/products/embedded processors/g-series#>.

Anaconda Accelerate [interaktyvus]. Nvidia [žiūrėta 2015 m. lapkričio 23 d.]. Prieiga per internetą: <https://developer.Nvidia.com/anaconda-accelerate>.

ARM Machine Learning Processor [interaktyvus]. ARM [žiūrėta 2019 m. rugsėjo 19 d.]. Prieiga per internetą: <https://developer.arm.com/ip-products/processors/machine-learning/arm-ml-processor>.

Artificial neural network [interaktyvus]. 2019. Wikipedia [žiūrėta 2019 m. birželio 19 d.]. Prieiga per internetą: <https://en.wikipedia.org/wiki/Artificial_neural_network#History>.

Bay, H.; Ess, A.; Tuytelaars, T.; Gool, L. V. 2008. Speeded-Up Robust Features (SURF), *Computer Vision and Image Understanding* 110(3): 346–359. ISSN: 1077-3142.

Biederman, I. 1987. Recognition by Components: A Theory of Human Image Understanding. *Psychol. Review* 94(2): 115–147.

BID Data Project. [interaktyvus]. 2015. [žiūrėta 2015 m. lapkričio 23 d.]. Prieiga per internetą: <http://bid2.berkeley.edu/bid-data-project/>.

Bigorgne, E.; Achard, C.; Devars, J. 2000. Local Zernike Moments Vector for Content-Based Queries in Image Data Base, in *IAPR Workshop on Machine Vision Applications*: 327–330.

Bottou, L. 2010. Large-Scale Machine Learning with Stochastic Gradient Descent. *Proc. of COMPSTAT*: 177–186. DOI: 10.1007/978-3-7908-2604-3

Caffe [interaktyvus]. 2015. Berkeley Vision [žiūrėta 2015 m. lapkričio 23 d.]. Prieiga per internetą: <http://caffe.berkeleyvision.org/>.

Cafiso, S.; D'Agostino, C.; Delfino, E.; Montella, A. 2017. From manual to automatic pave-ment distress detection and classification, in *2017 5th IEEE International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS)*: 433–438.

Canessa, A; Gibaldi, A.; Chessa, M.; Sabatini, S. P., Solari, F. 2012. The Perspective Geometry of the Eye: Toward Image-Based Eye-Tracking, chapter in *Human-Centric Machine Vision* edited by Solari, F.; Chessa, M.; Sabatini, S. P. IntechOpen.

Caulfield, B. 2015. *Kespry Shows How Jetson TX1-Powered Drones Can Keep an Eye on Construction Sites* [interaktyvus]. Nvidia [žiūrėta 2015 m. lapkričio 23 d.]. Prieiga per internetą: <http://blogs.nvidia.com/blog/2015/11/19/kespry-jetson-tx1/>.

Chai, E. 2014. *Implementation of Deep Convolutional Neural Net on a Digital Signal Processor* [interaktyvus] [žiūrėta 2015 m. lapkričio 23 d.]. Prieiga per internetą: <http://cs229.stanford.edu/proj2014/Elaina%20Chai,Implementation%20of%20Deep%20Convolutional%20NeuralNet%20on%20a%20DSP.pdf>.

Chandran, P.; John, M.; Santosh Kumar S.; Mithilesh N S R. 2014. Road Tracking Using Particle Filters for Advanced Driver Assistance Systems," in *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*: 1408–1414. ISSN: 2153-0009.

Cheng, G.; Zheng, J. Y.; Murase, H. 2018. Sparse Coding of Weather and Illuminations for ADAS and Autonomous Driving, in *2018 IEEE Intelligent Vehicles Symposium (IV)*: 2030–2035. ISSN: 1931–0587.

Chollet, F. 2016. Xception: Deep Learning with Depthwise Separable Convolutions, *CoRR. arXiv: 1610.02357*.

Coates, A,; Huval, B.; Wang, T.; Wu,. D. J.; Ng, A. Y.; Catanzaro, B. 2013. Deep learning with COTS HPC systems, in *30th International Conference on Machine Learning (ICML 2013)*: 2374–2382.

Collobert R.; Kavukcuoglu K.; Farabet, C. 2011. Torch7: A Matlab-like Environment for Machine Learning, in *BigLearn, NIPS Workshop*: 1–6.

Coral beta [interaktyvus]. Google. [žiūrėta 2019 m. rugsėjo 19 d.]. Prieiga per internetą: <https://coral.withgoogle.com/>.

CUDA Parallel Computing Platform [interaktyvus]. Nvidia [žiūrėta 2015 m. lapkričio 23 d.]. Prieiga per internetą: <http://www.nvidia.com/object/cuda_home_new.html>.

cuda-convnet: High-performance C++/CUDA implementation of convolutional neural networks [interaktyvus]. 2015. [žiūrėta 2015 m. lapkričio 23 d.]. Prieiga per internetą: <https://code.google.com/p/cuda-convnet/>.

Decencière, E.; Zhang, X.; Cazuguel, G.; Lay, B.; Cochener, B.; Trone, C.; Gain, P.; Ordonez-Varela, J.-R.; Massin, P.; Erginay, A.; Charton, B.; Klein, J.-C. 2014. Feedback on a publicly distributed database: the Messidor database, *Image Analysis & Stereology* 33(3): 231–234. ISSN: 1580-3239.

Dettmers, T. 2018. *A Full Hardware Guide to Deep Learning* [interaktyvus]. [Žiūrėta 2019 m. birželio 19 d.]. Prieiga per internetą: <https://timdettmers.com/2018/12/16/deep-learning-hardware-guide/>.

Dettmers, T. 2019. *Which GPU(s) to Get for Deep Learning: My Experience and Advice for Using GPUs in Deep Learning* [interaktyvus]. [Žiūrėta 2019 m. birželio 19 d.]. Prieiga per internetą: <https://timdettmers.com/2019/04/03/which-gpu-for-deep-learning/>.

Digital signal processor. 2015. Wikipedia [žiūrėta 2015 m. lapkričio 23 d.]. Prieiga per internetą: <https://en.wikipedia.org/wiki/Digital_signal_processor>.

Dmlc mxnet for Deep Learning [interaktyvus]. 2015. [žiūrėta 2015 m. lapkričio 23 d.]. Prieiga per internetą: <https://github.com/dmlc/mxnet>.

Dosovitskiy, A.; Fisher, P.; Springenberg, J. T.; Riedmiller, M.; Brox, T. 2016. Discriminative Unsupervised Feature Learning with Exemplar Convolutional Neural Networks, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 38(9): 1734–1747. ISSN: 0162-8828.

Dou, P.; Shah, S. K.; Kakadiaris, I. A. 2017. End-to-End 3D Face Reconstruction with Deep Neural Networks, in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, USA: 1503–1512.

Duchi, J., Hazan, E., Singer, Y. 2011. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization, *Journal of Machine Learning Research*, 12: 2121–2159.

Duchi, J., Singer, Y. 2009. Efficient online and batch learning using forward backward splitting, *Journal of Machine Learning Research* 10: 2873–2908.

Eitel, A.; Springenberg, J. T.; Spinello, L.; Riedmiller, M.; Burgard, W. 2015. Multimodal deep learning for robust RGB-D object recognition, in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Hamburgas, Vokietija: 681–687.

Eykholt, K.; Evtimov, I.; Fernandes, E.; Li, B.; Rahmati, A.; Xiao, C.; Prakash, A.; Kohno, T.; Song, D. 2018. Robust Physical-World Attacks on Deep Learning Visual Classification, in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*: 1625–1634. ISSN: 2575-7075.

Embedded systems for Next-Generation Autonomous Machines [interaktyvus]. 2019. Nvidia. [žiūrėta 2019 m. rugsėjo 19 d.]. Prieiga per internetą: <https://www.nvidia.com/en-gb/autonomous-machines/embedded-systems/>.

Farabet, C.; LeCun, Y.; Kavukcuoglu, K.; Culurcielo, E.; Martini, B.; Akselrod, P.; Talay, S. 2011. Large-Scale FPGA-based Convolutional Networks, chapter in *Machine Learning on Very Large Data Sets*, edited by Bekkerman, R.; Bilenko, M. and Langford, J. Cambridge University Press.

Field-programmable gate array. 2015. Wikipedia [žiūrėta 2015 m. lapkričio 23 d.]. Prieiga per internetą: <https://en.wikipedia.org/wiki/Field-programmable_gate_array>.

Fiser, J.; Subramaniam, S.; Biederman, I. 2001. Size Tuning in the absence of spatial frequency tuning in object recognition, *Vision Research* 41(15): 1931–1950.

Fukushima, K. 1980. Neurocognitron: A self-organizing neural network model for a machanism of pattern recognition unaffected by shift in position, *Biological cybernetics* 36(4): 193–202.

GeForce RTX 20 Series [interaktyvus]. 2015. Nvidia. [žiūrėta 2019 m. rugsėjo 19 d.]. Prieiga per internetą: <https://www.nvidia.com/en-gb/geforce/20-series/>.

Gimonet, N.; Cord, A.; Saint Pierre G. 2015. How to predict real road state from vehicle embedded camera?, in *2015 IEEE Intelligent Vehicles Symposium (IV)*: 593–598. ISSN: 1931-0587.

GNU Octave [interaktyvus]. 2015. Mathworks [žiūrėta 2015 m. lapkričio 23 d.]. Prieiga per internetą: <https://www.gnu.org/software/octave/>.

Golabbakhsh M.; Rabbani, H. 2013. Vessel-based registration of fundus and optical coherence tomography projection images of retina using a quadratic registration model, *IET Image Processing* 7(8): 768–776. ISSN: 1751-9659.

Gray, A. 2015. *Easy Multi-GPU Deep Learning with DIGITS 2* [interaktyvus]. Nvidia [žiūrėta 2015 m. lapkričio 23 d.]. Prieiga per internetą: <http://devblogs.Nvidia.com/parallelforall/easy-multi-gpu-deep-learning-digits-2/>.

Granado, J. M.; Vega, M. A.; Perez, R.; Sanchez, J. M.; Gomez, J. A. 2006. Using FPGAs to Implement Artificial Neural Networks, in *13th IEEE International Conference on Electronics, Circuits and Systems (ICECS '06)*: 934–937.

Gu, J.; Wang, Z.; Kuen, J.; Ma, L.; Shahroudy, A.; Shuai, B.; Liu, T.; Wang, X.; Wang, G.; Cai, J.; Chen, T. 2018. Recent Advances in Convolutional Neural Networks, *Pattern Recognition* 77: 354–377. ISSN: 0031-3203.

Gupta, S.; Agrawal, A.; Gopalakrishnan, K.; Narayanan, P. 2015. Deep Learning with Limited Numerical Precision, in *32nd International Conference on Machine Learning (PLMR) 37*: 1737–1746.

Harris, C. G.; Stephens, M. J. 1988. A Combined Corner and Edge Detector, in *Proceedings 4th Alvey Vision Conference*: 147–151.

He, K.; Zhang, X.; Ren, S. Sun.; J. 2016. Deep residual learning for image recognition, in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*: 770–778. ISSN: 1063-6919.

Hernandez-Matas, C.; Zabulis, X.; Argyros, A. A. 2017. An Experimental Evaluation of the Accuracy of Keypoints-based Retinal Image Registration, in *2017 39th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*: 377–381. ISSN: 1558-4615.

Hernandez-Matas, C.; Zabulis, X.; Triantafyllou, A.; Anyfanti, P.; Douma, S.; Argyros, A. A. 2017. FIRE: Fundus Image Registration Dataset, *Journal for Modeling in Ophthalmology* 1(4): 16–28. ISSN: 2468-3922.

Hertel, L.; Barth, E.; Kaster, T.; Martinetz, T. 2015. Deep Convolutional Neural Networks as Generic Feature Extractors, in *2015 International Joint Conference on Neural Networks (IJCNN)*: 1–4.

Hijazi, S.; Kumar, R.; Rowen, C. 2015. *Using Convolutional Neural Networks for Image Recognition* [interaktyvus]. Cadence [žiūrėta 2015 m. lapkričio 23 d.]. Prieiga per internetą: <http://ip.cadence.com/uploads/901/cnn_wp-pdf>.

Hill, B. 2015. *NVIDIA Jetson TX1 Brings Supercomputer Performance To Autonomous Drones, Vehicles. Sites* [interaktyvus]. Hot Hardware [žiūrėta 2015 m. lapkričio 23 d.]. Prieiga per internetą: <http://hothardware.com/news/Nvidia-jetson-tx1-brings-supercomputer-performance-to-autonomous-drones-vehicles>.

Hsu, G.-S. J. 2011. Stereo Correspondence with Local Descriptors for Object recognition, chapter in *Advances in Theory and Applications of Stereo Vision* edited by Bhatti, A.. IntechOpen.

Huang, G., Liu, Z., Maaten., L., Weinberger, K. Q. 2016. Densely Connected Convolutional Networks, *CoRR. arXiv: 1608.06993.*

Hubel, D. H.; Wiesel, T. N. 1968. Receptive fields and functional architecture of monkey strate cortex, *The Journal of Physiology* 195(1): 215–243.

ImageNet [interaktyvus]. 2016. Standford Vision Lab [žiūrėta 2019 m. rugsėjo 19 d.]. Prieiga per internetą: <http://www.image-net.org/>.

Intel Core processor family [interaktyvus]. 2019. Intel [žiūrėta 2019 m. rugsėjo 19 d.]. Prieiga per internetą: <https://www.intel.com/content/www/us/en/products/processors/core.html>.

Intel Deep Learning Framework [interaktyvus]. 2015. Intel [žiūrėta 2015 m. lapkričio 23 d.]. Prieiga per internetą: <https://01.org/intel-deep-learning-framework>.

Intel Nervana Neural Network Processors [interaktyvus]. 2019. [žiūrėta 2019 m. rugsėjo 19 d.] Prieiga per internetą: <https://www.intel.ai/nervana-nnp/#gs.43kflb>.

Intel Xeon Phi Processors [interaktyvus]. 2019. Intel [žiūrėta 2019 m. rugsėjo 19 d.]. Prieiga per internetą: <https://www.intel.com/content/www/us/en/support/products/75557/processors/intel-xeon-phi-processors.html>.

Intel Xeon Processors [interaktyvus]. 2019. Intel [žiūrėta 2019 m. rugsėjo 19 d.]. Prieiga per internetą: <https://www.intel.com/content/www/us/en/products/processors/xeon.html>.

Yadan, O.; Adams, K.; Taigman, Y.; Ranzato M. 2013. Multi-GPU Training of ConvNets. *CoRR. arXiv:1312.5853v4 [cs.LG]*.

Yang, Z.; Zhang, Y.; Yu, J.; Cai, J.; Luo, J. 2018. End-to-end Multi-Modal Multi-Task Vehicle Control for Self-Driving Cars with Visual Perceptions, in *2018 24th International Conference on Pattern Recognition (ICPR)*: 2289–2294. ISSN: 1051-4651.

Yi, K. M.; Trulls, E.; Lepetit, V.; Fua, P. 2016. LIFT: Learned Invariant Feature Transform, in *European Conference on Computer Vision (ECCV)*: 467–483. ISSN: 0302-9743.

Jones, N. 2014. The learning machines, *Nature* 505: 146–148.

Kayaer, K.; Tavsanoglu, V. 2008. A new approach to emulate CNN on FPGAs for real time video processing, in *11th International Workshop on Cellular Neural Networks and Their Applications (CNNA 2008)*: 23–28.

Kauppi, T.; Kalesnykiene, V.; Kamarainen, J.-K.; Lensu, L.; Sorri, I., Uusitalo, H.; Kälviäinen, H.; Pietilä, J. 2006. DIARETDB0: Evaluation Database and Methodology for Diabetic Retinopathy Algorithms, *Finland, Tech. Rep*.

Kauppi, T.; Kalesnykiene, V.; Kamarainen, J.-K.; Lensu, L.; Sorri, I.; Uusitalo, H.; Kälviäinen, H.; Pietilä, J. 2007. DIARETDB1: Diabetic Retinopathy Database and Evaluation Protocol, *Finland, Tech. Rep*.

Kingma, D.; Ba, J. 2017. Adam: A Method for Stochastic Optimization. CoRR. *arXiv:1412.6980 [cs.LG]*.

Klockner, A. 2015. *PyCUDA* [interaktyvus]. [žiūrėta 2015 m. lapkričio 23 d.]. Prieiga per internetą: <http://mathema.tician.de/software/pycuda/>.

Kogler, J; Sulzbachner, C; Humenberg, Martin; Eibensteiner, F.. 2011. Stereo Correspondence with Local Descriptors for Object recognition, chapter in *Advances in Theory and Applications of Stereo Vision* edited by Bhatti, A.. IntechOpen. ISBN 978-953-307-516-7.

Köhler, T.; Budai, A.; Kraus, M.; Odstrčilik, J.; Michelson G.; Hornegger, J. 2013. Automatic No-Reference Quality Assessment for Retinal Fundus Images Using Vessel Segmentation, in *Proceedings of the 26th IEEE International Symposium on Computer-Based Medical Systems*: 95–100. ISSN: 1063-7125.

Krizhevsky, A.; Sutskever, I.; Hinton, G. E. 2012. ImageNet Classification with Deep Convolutional Neural Networks in Advances, in *Neural Information Processing Systems* 25: 1097–1105.

Kweon, I. S; Kim, S. 2007. An Effective 3D Target Recognition Imitating Robust Methods of Human Visual Systems. Vision Systems: Applications. I-TECH Education and Publishing.

Krasner, G.; Katz, E. 2016. Automatic Parking Identification and Vehicle Guidance with Road Awareness," in *2016 ISCEE International Conference on the Science of Electrical Engineering*: 1–5.

Lane, N. D.; Georgiev, P. 2015. Can Deep Learning Revolutionize Mobile Sensing?, in *16th International Workshop on Mobile Computing Systems and Applications HotMobile'15*: 117–122.

LeCun, Y.; Denker, J. S.; Henderson, D.; Howard, R. E.; Hubbard, W.; Jackel, D. 1990. Handwriten Digit Recognition with a Back-Propagation Network, in *Advances in Neural Information Processing Systems 2 (NIPS 1989)*: 396–404.

LeCun, Y.; Bottou, Y.; Bengio, Y.; Haffner, P. 1998. Gradient-Based Learning Applied to Document Recognition, in *Proceedings of the IEEE* 86(11): 2278–2324.

LeCun, Y.; Cortes, C., Burges, C. J. C. 2019. The MNIST Database of handwritten digits [interaktyvus] [žiūrėta 2019 m. rugsėjo 19 d.]. Prieiga per internetą: <http://yann.lecun.com/exdb/mnist/>

Leutenegger, S.; Chli, M.; Siegwart, R. Y. 2011. BRISK: Binary Robust Invariant Scalable Keypoints, in *2011 International Conference on Computer Vision*: 2548–2555. ISSN: 2380-7504.

List of Nvidia graphics processing units [interaktyvus]. 2015. Wikipedia [žiūrėta 2015 m. lapkričio 22 d.]. Prieiga per internetą: <https://en.wikipedia.org/wiki/List_of_Nvidia_graphics_processing_units#Quadro_Mxxx_Series>

List of AMD graphics processing units [interaktyvus]. 2015. Wikipedia [žiūrėta 2015 m. lapkričio 22 d.]. Prieiga per internetą: <https://en.wikipedia.org/wiki/List_of_AMD_graphics_processing_units>.

Lowe, D. G. 2004. Distinctive Image Features from Scale-Invariant Keypoints, *International Journal of Computer Vision* 60(2): 91–110. ISSN: 0920-5691.

Machine Learning [interaktyvus]. 2015. Nvidia [žiūrėta 2015 m. lapkričio 23 d.]. Prieiga per internetą: <http://www.nvidia.co.uk/object/tesla-gpu-machine-learning-uk.html#tp>.

Machine learning [interaktyvus]. 2019. Wikipedia [žiūrėta 2019 m. birželio 19 d.]. Prieiga per internetą: <https://en.wikipedia.org/wiki/Machine_learning>.

McMahan, H. B. 2011. Follow-the-Regularized-Leader and Mirror Descent: Equivalence Theorems and L1 Regularization, in *Proc. of 14th Internation Conference on Artificial Intelligence and Statistics* 15, Fort Lauderdale, USA.

McMahan, H. B. 2017. A Survey of Algorithms and Analysis for Adaptive Online Learning, *Journal of Machine Learning Research* 18: 1–50.

Mahmud, F.; Arafat, A.; Zuhori, S. T. 2012. Intelligent Autonomous Vehicle Nav-igated by using Artificial Neural Network, in *2012 7th International Conference on Electrical and Computer Engineering*: 105–108.

Mahmudi, T.; Kafieh, R.; Rabbani, H. 2014. Comparison of macular OCTs in right andleft eyes of normal people, in *Progress in Biomedical Optics and Imaging - Proceedings of SPIE* 9038: 15–20.

Martin, R. 2012. *Intel's New Xeon Phi Co-Processors include A 61 Core 7120X With 8 GB Onboard Cache* [interaktyvus]. 2015. eTeknix [žiūrėta 2015 m. lapkričio 23 d.]. Prieiga per internetą: <http://www.eteknix.com/intels-new-xeon-phi-co-processors-include-a-61-core-7120x-with-8gb-onboard-cache/>.

Matas, J.; Chum, O; Urban, M.; Pajdla, T. 2002. Robust wide baseline stereo from maximally stable extremal regions, in *Proc. of British Machine Vision Conference*: 384–396.

Matlab: The Language of Technica Computing [interaktyvus]. 2015. Mathworks [žiūrėta 2015 m. lapkričio 23 d.]. Prieiga per internetą: <http://se.mathworks.com/products/matlab/>

Menze, M.; Geiger, A. 2015. Object Scene Flow for Autonomous Vehicles, in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*: 3061–3070. ISSN: 1063-6919.

Meignen, D.; Bernadet, M.; Briand, H. 1997. One Application of Neural Networks for Detection of Defects Using Video Data Bases: Identification of Road Distress, in *Database and Expert Systems Applications. 8th International Conference (DEXA '97) Proceedings*: 459–464.

Mikolajczyk, K.; Schmid, C. 2002. An affine invariant interest point detector, in *Proc. European Conf. on Computer Vision, LNCS 2350* 1: 128–142.

Milakov, M. 2014. Deep learning with GPUs [interaktyvus]. Nvidia [žiūrėta 2019 m. birželio 19 d.]. Prieiga per internetą: <http://www.Nvidia.co.uk/docs/IO/147844/Deep-Learning-With-GPUs-MaximMilakov-NVIDIA.pdf>.

Milz, S.; Arbeiter, G.; Witt, C.; Abdallah, B.; Yogamani, S. 2018. Visual SLAM for Automated Driving: Exploring the Applications of Deep Learning, in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*: 360–370.

Minsky, M.; Papert, S. 1969. Perceptrons: An Introduction to Computational Geometry. MIT Press. ISBN 978-0-262-63022-1.

Morgan, T. P. 2014. Netflix Speeds Machine Learning With Amazon GPUs [interaktyvus]. EnterpriseTech [žiūrėta 2019 m. birželio 19 d.]. Prieiga per internetą: <http://www.enterprisetech.com/2014/02/11/netflix-speeds-machine-learning-amazon-gpus/>.

Mueler, F. 2007. NC State Engineer Creates First Academic Playstation 3 Computing Cluster [interaktyvus]. NC SATE University [žiūrėta 2015 m. lapkričio 22 d.]. Prieiga per internetą: <http://www.engr.ncsu.edu/news/news_articles/ps3.html>.

Muthuramalingam, A.; Himavathi, S.; Srinivasan, E. 2007. Neural Network Implementation Using FPGA: Issues and Application, *International Journal of Information Technology* 4(2): 86–92.

Nadav, I.; Katz, E. 2016. Off-road Path and Obstacle Detection using Monocular Camera, in *2016 ISCEE International Conference on the Science of Electrical Engineering*: 1–5.

Naguib, A. M.; Kim, J.; Lee, S. 2017. 3D Environmental Modeling and Drivable Road Identification for a Long-Range Rover, in *2017 14th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*: 697–700.

Navakauskas, D; Serackis, A.; Matuzevicius, D.; Laptik, R. 2015. Specializuotos elektroninės intelektualiosios sistemos garsams ir vaizdams apdoroti: *teorija ir taikymai: monografija*. Technika. 574 p.

Noh, H.; Seo, P. H.; Han, B. 2015. Image Question Answering using Convolutional Neural Network with Dynamic Parameter Prediction, *CoRR. arXiv: 1511.05756v1 [cs.CV]*: 1–9.

Noyel, G.; Thomas, R.; Bhakta, G.; Crowder, A.; Owens, D.; Boyle P. 2017. Superimposition of eye fundus images for longitudinal analysis from large public health databases, *Biomedical Physics and Engineering Express* 3(4): 045015.

Ng, J. Y.-H.; Yang, F.; Davis, L. S. 2015. Exploiting Local Features from Deep Networks for Image Retrieval, in *IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*: 53–61.

Nugraha, A. A.; Arifianto, A.; Suyanto. 2019. Generating Image Description on Indonesian Language using Convolutional Neural Network and Gated Recurrent Unit, in *7th International Conference on Information and Communication Technology*, Kualalumpuras, Malaizija: 1–6.

NVIDIA cuDNN [interaktyvus]. Nvidia [žiūrėta 2019 m. rugsėjo 19 d.]. Prieiga per internetą: <https://developer.nvidia.com/cudnn>.

Nvidia GRID [interaktyvus]. 2019. Nvidia. [žiūrėta 2019 m. rugsėjo 19 d.]. Prieiga per internetą: <https://www.nvidia.com/en-us/data-center/virtual-pc-apps/>.

Ogawa, T. 2012. Transformations of Image Filters for Machine Vision Using Complex-Valued Neural Networks, chapter in *Human-Centric Machine Vision* edited by Solari, F.; Chessa, M.; Sabatini, S. P. IntechOpen.

Oliveira, H.; Lobato Correia, P. 2008. Identifying and retrieving distress images from road pavement surveys, in *2008 15th IEEE International Conference on Image Processing*: 57–60. ISSN: 1522-4880.

OpenCL. [interaktyvus]. 2019. The Khronos group inc. [žiūrėta 2019 m. rugsėjo 19 d.]. Prieiga per internetą: <https://www.khronos.org/opencl>.

OpenCV. [interaktyvus]. 2019. OpenCV team. [žiūrėta 2019 m. rugsėjo 19 d.]. Prieiga per internetą: <https://www.opencv.org>.

OpenVX. [interaktyvus]. 2019. Khronos group [žiūrėta 2019 m. rugsėjo 19 d.]. Prieiga per internetą: <https://www.khronos.org/openvx>.

Ovtcharov, K.; Ruwase, O.; Kim, J.-Y.; Fowers, J.; Strauss, K.; Chung, E. S. 2015. Accelerating Deep Convolutional Neural Networks Using Specialized Hardware [interaktyvus]. Microsoft Research. [žiūrėta 2015 m. lapkričio 23 d.]. Prieiga per internetą: <http://research.microsoft.com/pubs/240715/CNN%20Whitepaper.pdf>.

PyTorch. [interaktyvus]. 2019. [žiūrėta 2019 m. rugsėjo 19 d.]. Prieiga per internetą: <https://pytorch.org/>.

Power Architecture [interaktyvus]. 2015. Wikipedia [žiūrėta 2015 m. lapkričio 22 d.]. Prieiga per internetą: <https://en.wikipedia.org/wiki/Power_Architecture>.

Quadro [interaktyvus]. 2019. Nvidia. [žiūrėta 2019 m. rugsėjo 19 d.]. Prieiga per internetą: <https://www.nvidia.com/en-us/design-visualization/quadro/>.

Quan, N. 1999. On the momemtum term in gradient descent learning algorithms, *Neural Networks: the official journal of the International Neural Network Society*, 12(1): 145–151.

Rao, Q.; Frtunikj, J. 2018. Deep Learning for Self-Driving Cars: Chances and Chal-lenges, in *2018 IEEE/ACM 1st International Workshop on Software Engineering for AI in Autonomous Systems (SEFAIAS)*: 35–38.

Retinal Image Database [interaktyvus]. 2018. Kingston University London. [žiūrėta 2019 m. birželio 20 d.] Prieiga per internetą: <https://blogs.kingston.ac.uk/retinal/chasedb1/>.

Rojas, R. 1996. *Neural Networks*: A Systematic Introduction. Berlin: Springer-Verlag. 502 p.

Rosenblatt, F. 1957. The Perceptron – a perceiving and recognizing automaton. Ataskaita 85-460-1, Kornelio Aeronautikos laboratorija. Prieiga per internetą: <https://blogs.umass.edu/brain-wars/files/2016/03/rosenblatt-1957.pdf>.

Rosten, E.; Porter, R.; Drummond, T. 2010. Faster and better: A machine learning approach to corner detection, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32(1): 105–119. ISSN: 0162-8828.

Rupp, K. 2013. *CPU, GPU and MIC Hardware Characteristics over Time* [interaktyvus]. [žiūrėta 2019 m. rugsėjo 19 d.]. Prieiga per internetą: <https://www.karlrupp.net/2013/06/cpu-gpu-and-mic-hardware-characteristics-over-time/>.

Rumelhart, D.; McClelland, J. L. 1986. Parallel Distributed Processing: explorations in the microstructure of cognition. PDP Research Group. MIT Press, Cambridge.

Sahin, S.; Beceriki, Y.; Yazici, S. 2006. Neural Network Implementation in Hardware Using FPGAs, in *13th International Conference of Neural Information Processing (ICONIP 2006)*: 1105–1112.

Sathya, R.; Abraham, A. 2013. Comparison of Supervised and Unsupervised Learning Algorithms for Pattern Classification, *International Journal of Advanced Research in Artificial Intelligence*, 2(2): 34–38.

scikit-learn: Machine Learning in Python [interaktyvus]. 2015. [žiūrėta 2015 m. lapkričio 23 d.]. Prieiga per internetą: <http://scikit-learn.org/stable/>.

Sermanet, P.; Eigen, D.; Zhang, X.; Mathieu, M.; Fergus, R.; LeCun, Y. 2013. Overfeat: Integrated recognition, localization and detection using convolutional networks, in *International Conference on Learning Representations (ICLR). arXiv:1312.6229v4 [cs.CV]*

Shah, A. 2015. Intel's 72-core processor jumps from supercomputers to workstations [interaktyvus]. IDG News Service [žiūrėta 2019 m. birželio 19 d.]. Prieiga per internetą: <http://www.pcworld.com/article/3005414/computers/intel-plugs-72-core-supercomputing-chip-into-workstation.html>.

Shelhamer, E.; Long, J.; Darrell, T. 2017. Fully Convolutional Networks for Semantic Segmentation, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(4): 640–651.

Shi, J.; Tomasi, C. 1994. Good Features to Track, in *1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition:* 593–600. ISSN: 1063-6919.

Shen, G. 2016. Road Crack Detection Based on Video Image Processing, in *2016 3rd International Conference on Systems and Informatics ICSAI*: 912–917.

Simo-Serra, E.; Trulls, E.; Ferraz, L.; Kokkinos, I.; Fua, P.; Moreno-Noguer, F. 2015. Discriminative Learning of Deep Convolutional Feature Point Descriptors, in *2015 IEEE International Conference on Computer Vision (ICCV)*: 118–126. ISSN: 2380-7504.

Simonyan, K.; Zisserman, A. 2015. Very deep convolutional networks for large-scale image recognition, in *International Conference on Learning Representations (ICLR 2015)*. CoRR. *arXiv:1409.1556v6 [cs.CV]*

Single Platform for All Predictive Use Cases: Machine Learning built for Developers [interaktyvus]. 2015. BigML [žiūrėta 2015 m. lapkričio 23 d.]. Prieiga per internetą: <https://bigml.com/>

Smolyanskiy, N.; Kamenev, A.; Smith, J.; Birtchfield, S. 2017. Toward low-flying autonomous MAV trail navigation using deep neural networks for environmental awareness, in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Vankuveris, Kanada: 4241–4247.

Smolyanskiy, N.; Kamenev, A.; Birchfield, S. 2018. On the Importance of Stereo for Accurate Depth Estimation: An Efficient Semi-Supervised Deep Neural Network Approach," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*: 1120–1128. ISSN: 2160–7516.

Song, D.-H.; Cho, J.-H.; Kim, K.-B.; Je, S.-K. 2007. Image Magnification based on the Human Visual Processing. Vision Systems: Applications. I-TECH Education and Publishing.

Sutskever, O.; Martens, J.; Dahl, G.; Hinton, G. 2013. On the importance of initialization and momentum in deep learning, in *Proceedings of the 30th International Conference on Machine Learning (PMLR)*, 28(3): 1139–1147.

Szegedy, C.; Liu, W.; Jis, Y.; Sermanet, P.; Reed S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; Rabinovich, A. 2015. Going deeper with convolutions, in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*: 1–9. ISSN: 1063-6919.

Tan, M., Le, Q. 2019. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks: *CoRR*. Prieiga per internetą: <https://arxiv.org/abs/1905.11946>.

Tieleman, T.; Hinton, G. 2012. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning [žiūrėta 2019 m. rugsėjo 19 d.]. Prieiga per internetą: <http://www.cs.toronto.edu /~tijmen/csc321/slides/lecture_slides_lec6.pdf>

Tegra [interaktyvus]. 2015. Nvidia. [žiūrėta 2015 m. lapkričio 22 d.]. Prieiga per internetą: <http://www.nvidia.co.uk/object/tegra-uk.html>.

Tesla [interaktyvus]. 2019. Nvidia. [žiūrėta 2019 m. rugsėjo 19 d.]. Prieiga per internetą: <https://www.nvidia.com/en-us/data-center/tesla/>.

Theano 0.7 documentation [interaktyvus]. 2015. LISA lab [žiūrėta 2015 m. lapkričio 23 d.]. Prieiga per internetą: <http://deeplearning.net/software/theano/>.

Torch: A Scientific computing framework for LuaJIT [interaktyvus]. 2015. [žiūrėta 2015 m. lapkričio 23 d.]. Prieiga per internetą: <http://torch.ch/>.

TOP500 [interaktyvus]. 2019. Wikipedia [žiūrėta 2019 m. birželio 19 d.]. Prieiga per internetą: <https://en.wikipedia.org/wiki/TOP500>.

Treisman, A. 1998. Feature binding, attention and object perception, *Philosophical Transactions: Biological Sciences* 29, 353(1373): 1295–1306.

VanRullen, R. 2003. Visual saliency and spike timing in the ventral visual pathway, *Journal of Physiology* 97: 365–377.

Van Hamme, D.; Veelaert, P.; Philips, W. 2013. Lane Identification based on Robust Visual Odometry, *in 16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*: 1173–1183. ISSN: 2153-0009.

Vedaldi, A.; Fulkerson, B. 2010. Vlfeat: An open and portable library of computer vision algorithms, in *Proceedings of the 18th ACM international conference on Multimedia*: 1469–1472.

Vulkan. [interaktyvus]. 2015. Khronos group [žiūrėta 2015 m. lapkričio 23 d.]. Prieiga per internetą: <https://www.khronos.org/vulkan>.

Weka 3: Data mining Software in Java [interaktyvus]. 2015. The University of Waikato [žiūrėta 2015 m. lapkričio 23 d.]. Prieiga per internetą: <http://www.cs.waikato.ac.nz/ml/weka/>.

Werbos, P. J. 1974. Beyond regression: new tools for prediction and analysis in the behavioral sciences. Disertacija. Hardvardo universitetas: 454 p.

Xie, S., Girshick, R., Dollar, P., Tu., Z., He, K. 2016. Aggregated Residual Transformations for Deep Neural Networks, *CoRR. arXiv: 1611.05431*.

Yao, L.; Torabi, A.; Cho, K.; Ballas, N.; Pal, C.; Larochelle, H.; Courville, A. 2015. Describing Videos by Exploiting Temporal Structure, *CoRR. arXiv: 1502.08029v5 [stat.ML]*: 1–23.

Yuan, Y.; Mou, L.; Lu, X. 2015. Scene Recognition by Manifold Regularized Deep Learning Architecture, *IEEE Transactions on Neural Networks and Learning Systems*, 26(10): 2222–2233.

Zeiler, M, D. 2012. ADADELTA: An Adaptive Learning Rate Method, *CoRR. arXiv: 1212.5701*: 1–6.

Zeiler, M. D., Fergus, R. 2014. Visualizing and understanding convolutional networks. *Computer Vision (ECCV 2014). LNCS* 8689.

Zhang, C.; Li, P.; Sun, G.; Guan, Y.; Xiao, B.; Cong, J. 2015. Optimizing FPGA-based Accelerator Design for Deep Convolutional Neural Networks, in *23rd International Symposium on Field-Programmable Gate Arrays (FPGA2015*): 161–170.

Zhang, L.; Wu, E. 2009. A Road Segmentation and Road Type Identification Approach Based on New-Type Histogram Calculation, in *2009 2nd International Congress on Image and Signal Processing*: 1–5.

Zhang, W.; Itoh, K.; Tanida, J.; Ichioka, Y. 1990. Parralel distributed processing model with local space-invariant interconnections and its optical architecture, *Applied Optics* 29(32): 4790–4797.

Zhou, C.; Du, Y.; Tang, Y. 2011. Evolutionary Approach to Epipolar Geometru Estimation, chapter in *Advances in Theory and Applications of Stereo Vision* edited by Bhatti, A. IntechOpen.

Zynq-7000 SoC Product Advantages [interaktyvus]. 2019. Xilinx [žiūrėta 2019 m. rugsėjo 19 d.]. Prieiga per internetą: <https://www.xilinx.com/products/silicon-devices/soc/zynq-7000.html>.

Autoriaus mokslinių publikacijų disertacijos tema sąrašas

Straipsniai recenzuojamuose mokslo žurnaluose

Žuraulis, V.; Surblys, V.; Šabanovič, E. Technological measures of forefront road indentification for vehicle comfort and safety improvement, *Transport* 34(3): 363–372. DOI: 10.3846/transport.2019.10372 (citavimo rodiklis IF = 1,524).

Serackis, A.; Matuzevičius, D.; Navakauskas, D.; Šabanovič, E.; Katkevičius, A.; Plonis, D. 2017. A robust identification of the proteins standard bands in two-dimensional electrophoresis gel images, *Electrical, control and communication engineering* 13(1): 63–68. DOI: 10.1515/ecce-2017-0009

Šabanovič, E.; Matuzevičius, D.; Vaitkevičius, P. H. 2016. Binokulinės regos skaitinis modelis horizontalių erdvės koordinačių kodavimui = Numerical model of binocular vision for horizontal space coordinate coding, *Inžinerinės ir edukacinės technologijos = Engineering and educational technologies*, 2(8): 184–189.

Straipsniai kituose leidiniuose

Šabanovič, E.; Stankevičius, G.; Matuzevičius, D. 2018. Deep neural network-based feature descriptor for retinal image registration, in *6th Workshop on Advances in Information, Electronic and Electrical Engineering (AIEEE)*: 1–4. DOI: 10.1109/AIEEE.2018.8592033

Šabanovič, E.; Matuzevičius, D. 2017. Experimental Investigation of Feature Descriptors for Retinal Image Registration, in *5th Workshop on Advances in Information, Electronic and Electrical Engineering (AIEEE)*: 1–4. DOI: 10.1109/AIEEE.2017.8270537

Šabanovič, E.; Matuzevičius, D.; Serackis, A. 2016. Evaluation of the background extraction influence to quantitative analysis of two-dimensional electrophoresis gel images. *Biomedical engineering 2016: proceedings of 20th international conference*: 113–116.

Šabanovič, E.; Matuzevičius, D. 2015. Application of human visual system models to LCD image analysis, in 3*rd Workshop on Advances in Information, Electronic and Electrical Engineering (AIEEE)*: 1–5. DOI: 10.1109/AIEEE.2015.7367313

Summary in English

Introduction

Problem Formulation

Implementation of convolutional neural networks (CNN) in accelerating units for real-time image analysis requires many decisions. Accelerating unit for CNN algorithms, CNN structure, training algorithm are selected and dataset for training, validation, and testing is prepared. The problem to be solved by CNN defines the maximum execution time and minimal accuracy of the algorithm. Accelerating units have limited computing speed and memory capacity and support limited list of operations. It is necessary to ensure the possibility of CNN structure implementation in the selected accelerating unit, do not exceed the computation duration set by the task, while achieving the required performance of the algorithm. Usually, the structure and parameters of CNN are selected manually which inevitably lead to errors, which reduce the efficiency of the algorithm. The more efficient the algorithm is, the lower performance accelerators are required for implementation and less power consumed. This is an especially important embedded system that processes visual information using CNN-based algorithms. Methodologies that define the process of CNN creation are being developed, currently. However, there is no design methodology for CNN implementation in accelerating units.

In this thesis, the problem of designing convolutional neural networks for implementation in accelerating units for real-time image analysis is being investigated. In order to solve this problem, such a main hypothesis have been raised and proven: it is possible to create a list of elements and develop a design methodology for convolutional neural

networks that could be used to create specialized convolutional neural networks for implementation in accelerating units for real-time image processing.

Relevance of the Thesis

There is an increasing demand for the convolutional neural network (CNN) based image analysis implementation in embedded systems. Accelerating units are created for real-time large-scale CNN calculations. Most of the CNN research is focused on the improvement of accuracy, a lot less on efficiency in terms of power consumption, and the characteristics and limits of accelerating units are being ignored. This often results in inefficient use of computing resources and therefore use of accelerating units with higher performance, price and power consumption. The compiled elements' list of CNN allows estimating the calculations' quantity of the CNN and grants compatibility with accelerating units. It is possible to partially automate the design process of CNN using the developed design methodology for CNN, thus speeding up the design process and avoiding common mistakes. Two CNN-based image analysis algorithms have been developed to test the proposed methodology. The first is a specialized algorithm for local feature description of eye retinal images. This algorithm is relevant for ophthalmologic eye examinations. The second is a specialized algorithm for road pavement type and condition evaluation, which is relevant for the safety and comfort of cars.

Research Object

The object of work research is convolutional neural networks for real-time image analysis. The following main aspects of the research object are investigated in the present thesis: implementation in accelerating units and design methodology.

The Aim of the Thesis

To create and apply the design methodology for the implementation of convolutional neural networks in accelerating units for real-time image analysis.

The Tasks of the Thesis

In order to reach the aim of the thesis the following main tasks are formulated:

1. Investigate the influence of structural elements and parameters of convolutional neural networks (CNN) on the accuracy and speed of their training and execution in accelerating units.
2. Create a list of CNN elements that can be used to select elements when designing a specialized CNN for implementation in accelerating units.
3. Develop a design methodology for CNN that is specialized for the selected task of image analysis, when minimum speed requirement and the technical characteristics of accelerating unit are known.
4. Design and implement CNN in accelerating units, using created methodology, for these tasks of image analysis:
    - description of eye retinal image local features;
    - evaluation of road pavement type and condition.

Research Methodology

Theories of digital image processing, artificial neural networks, deep learning, statistical analysis have been applied. Methods for the image preprocessing and augmentation, training, and inference of convolutional neural networks (CNN), speed and accuracy evaluation of CNN, accuracy evaluation of image feature matching, selection and use of hard-to-learn samples have been implemented and applied.

Specialized CNN training and testing datasets for feature description of eye retinal images and evaluation of road pavement type and condition have been collected and compiled. CNN has been researched using various software, the most used were DIGITS, Tensorflow, and Matlab. SDNT has been implemented using Intel i7-6900K central processing unit and various accelerating units: Nvidia Geforce GTX 960 and GTX 1080 Ti graphics processing units (GPU), Google Coral Edge tensor processing unit and Nvidia Jetson TX2 embedded system with GPU.

Scientific Novelty of the Thesis

1. A list of convolutional neural network (CNN) elements suitable for implementation in accelerating units has been compiled. The list includes formulas for calculation of computational operations for impact evaluation of elements and their parameters on execution speed.
2. A CNN design methodology, which allows partial automatization of CNN structure and parameter selection according to image analysis task requirements and constraints of the accelerators, has been developed.
3. A new CNN-based image analysis algorithm for feature description, which is specialized for retinal image registration, has been developed and implemented in the graphics processing unit (GPU).
4. A new CNN-based image analysis algorithm for real-time road pavement type and condition evaluation has been developed and implemented in Nvidia Jetson TX2 embedded system with GPU.

Practical Value of Research Findings

The proposed list of CNN elements allows the selection of elements and their parameters based on the number of computational operations and estimated execution time in the particular accelerating unit.

The developed algorithm for local feature descriptions can be applied for eye retinal image registration. This is useful for ophthalmologic examinations of retinal images to identify changes between visits of the patient. In addition, the developed algorithm can be adapted for image analysis of other medical images, such as alignment of protein spots in two-dimensional electrophoresis gel images.

The developed algorithm for road pavement type and condition real-time evaluation can be used to improve car safety and comfort through environmental awareness. In the future, this algorithm may be refined to determine the distance to road objects using a binocular vision model.

The results obtained and image datasets collected are planned to use for further research and application of CNN in science projects.

The Defended Statements

1. Convolutional neural networks (CNN) can be implemented in accelerating units for real-time image processing using the compiled list of CNN elements and the proposed design methodology.
2. A CNN-based local feature description algorithm, created using proposed methodology, allows registration of feature point pairs of FIRE dataset with similar accuracy to traditional feature descriptors.
3. A CNN-based algorithm, created using the proposed methodology, allows evaluation of road pavement type and condition from a single image with the accuracy of at least 84 %.
4. A CNN-based road pavement type and condition evaluation algorithm, when implemented in an embedded system Nvidia Jetson TX2, process one image faster than in 30 ms.

Approval of the Research Findings

Seven articles are published on the subject of the dissertation: one in scientific journal included in Clarivate Analytics Web of Science (CA WoS) with 1.524 citation index (Žuraulis, Surblys, Šabanovič 2019), one in scientific journal included in CA WoS (Serackis *et al*. 2017), one in scientific journal included in Index Copernicus (Šabanovič, Matuzevičius, Vaitkevičius 2016), three - in international conference proceedings included in CA WoS Proceedings (Šabanovič, Matuzevičius 2015; Šabanovič, Matuzevičius 2017; Šabanovič, Stankevičius, Matuzevičius 2018), one in conference proceedings included in other database (Šabanovič, Matuzevičius, Serackis 2016).

Dissertation research results have been announced in nine scientific conferences:

- International Conferences "Advances in Information, Electronic and Electrical Engineering" in 2015, 2017 Riga, Latvia and in 2018 Vilnius;
- Young Scientists Conference "*Mokslas – Lietuvos ateitis. Electronics and electrotechnics*" in 2016 Vilnius, Lithuania;
- International Conference "*Biomedical Engineering"* in 2016 Kaunas, Lithuania;
- International Conference "*Data Analysis Methods for Software Systems*" in 2016 Druskininkai, Lithuania;
- International Conference "*Electrical, Electronic and Information Sciences*" in 2018 Vilnius, Lithuania;
- International Conference "*Conference Vision Zero for Sustainable Road Safety in Baltic Sea Region*" in 2018 Vilnius, Lithuania;
- International Conference "*Advanced Microwave Devices and Systems*" in 2018 Vilnius, Lithuania.

Structure of the Dissertation

The dissertation consists of an introduction, three chapters, general conclusions, lists of references, a list of scientific publications by the author on the topic of the dissertation and a summary in English. The total scope of the dissertation is 117 pages, 18 indexed equations, 23 pictures, 6 tables, 182 references have been used.

1. Review of Image Analysis Using Convolutional Neural Networks

Image analysis tasks can be split into six categories: image preprocessing, segmentation, image perception, compression, object recognition, and optimization. Most of these tasks can be solved by computer vision methods, part of them are based on artificial neural networks (ANN) algorithms of machine learning. Deep learning and deep neural networks (DNN) enable the usage of ANNs in new areas of image processing and improve accuracy compared to other machine learning methods.

Convolutional neural network (CNN) is a type of DNN, where isolated neurons are positioned to match the overlapping regions in the field of vision. CNN is inspired by the biological vision system and is a multilayer perceptron variation designed to use less computational resources. CNNs are used in the fields of computer vision, automatic language recognition and natural language processing, audio recognition, and bioinformatics and often reach human-level performance.

In the field of image analysis, there are published articles in three-dimensional image reconstruction, image feature extraction, image segmentation, object recognition and classification in the image, object detection, image perception, image description, image depth map estimation, detection of a change in object orientation and even answering questions about the image. The most common task is image classification of a single object in the image.

The history of CNN evolution lasts about 20 years. During that period a number of innovative CNN structures were presented. Lenet-5 is the first CNN used for recognition of hand-written numbers. Alexnet and ZFNet are bigger CNNs similar to LeNet-5 used for image recognition of ImageNet dataset. Inception presented a new way to implement CNN with a bigger number of layers using Inception blocks. VGGNet is CNN that has a uniform structure that scales well. ResNet presented a new residual block, that allows CNN with a high number of layers. Depth-wise separated convolution has been introduced in Xception. ResNeXt-50 had structure combining residual and Inception blocks. DenseNet introduced Dense blocks, there next layer was connected to all previous layers. EfficientNet marks the new direction of CNN development towards improvement of efficiency. In the beginning, there was a tendency for growth of CNN complexity, but innovative blocks reduced a number of learnable parameters and improved both speed and accuracy of CNN-based algorithms.

Usually, CNNs are composed of convolution, pooling, or subsampling layers, and fully connected neuron layers also called a perceptron layer. Also, self-organizing maps (SOMs), Softmax or another type of output layer.

CNN is being trained using learning algorithms with the teacher and without the teacher. Various deep learning algorithms are applied, including variants of the backpropagation algorithm, such as stochastic gradient descent (SGD), SGD with momentum, Adadelta, Adagrad, AdagradDA, ADAM, FTLR-Proximal, RMSprop. SGD with momentum is implemented in most software for CNN implementation. Adadelta, Adagrad, and ADAM are proved to be useful when learning rate is hard to select.

Currently, these application packages for ANN modeling and image processing are available: Tensorflow, PyTorch, Matlab, Torch, MXNet, Theano, Caffe, DIGITS2, Weka. Also, there are various image processing and low-level libraries for CNN implementation:

Intel Deep Learning Framework, Vulcan, CUDA, TensorRT, OpenCL, PyCUDA, Anaconda Accelerate, CuDNN, OpenVX, OpenCV, BIDMach, Scikit-Learn, BigML.io, Cuda-convnet (2). The most popular software for CNN implementation is Tensorflow and PyTorch.

Central processing units (CPU) and coprocessors, general-purpose and graphics processing units (GPGPU), digital signal processing devices (DSP), field programmable gate arrays (FPGA) and distributed computing systems are used for CNN implementation. Tensor processing units (TPU) and neuron processing units (NPU) are accelerating units that are specialized for CNN implementation that ensure the efficiency of calculations in terms of energy consumption. There are systems that combines CPU and GPU, such as Nvidia Jetson TX2. Such systems can be used to implement CNN in car or various specialized equipment. Each type of processing unit that can be used for CNN implementation has various advantages and disadvantages. The suitability of the equipment for different tasks is given in Table S.1.1.

Table S.1.1. Hardware usability for tasks related to artificial neural networks

Device	Research	Performance	Energy effectiveness	ANN training	ANN inference	ANN size	Other operation
Computer cluster	Yes	High	Very Low	Yes	Yes	Extremely High	Yes
CPU	Yes	Low	Low	Yes	Yes	Very High	Yes
GPU	Yes	Very High	Medium	Yes	Yes	High	Yes
FPGA	Limited	High	High	Limited	Yes	Limited	Yes
DSP	Limited	Medium	Medium	Limited	Yes	Limited	Limited
TPU	Limited	High	Very High	Limited	Yes	Limited	Yes
NPU	Limited	High	Very High	Limited	Yes	Limited	Yes

Note. Grey fields from left to right indicates accelerating units that are: the best for CNN training and the best for inference.

Based on the review of the hardware, it can be concluded that the most appropriate accelerating unit for the implementation of the CNN is GPU. FPGAs, DSPs, TPUs, and NPUs can be used mainly for efficient application in end systems. FPGA has the highest flexibility, but also requires most work and energy efficiency lower than CPU and GPU. TPU and NPU have a limited list of supported operations. Therefore, a selection of supported elements is needed. It is necessary to consider the total amount of computing operations and memory used while designing CNN for implementation in accelerating units. The design methodology for CNN implementation in accelerating units for real-time image tasks is necessary.

2. Research of Convolutional Neural Networks Implementation

In order to create a design methodology for a specialized convolutional neural network, which can be implemented in accelerating units an element list was composed, the impact

of CNN structure on training time, execution time and accuracy, implementation of convolutional neural networks in accelerating units have been researched.

The composed list of CNN elements includes convolutional, fully connected, various pooling, normalization, and output layers. Formulas for calculation of required computational operations are provided.

The operations count of the convolutional layer can be calculated using this formula:

$$N_{\mathrm{SA}} = \frac{2\mathrm{NIJHWC}}{\mathrm{s}^2}, \tag{S.2.1}$$

here N – the number of convolutional filters; H – the number of rows in convolutional filter; W – the number of columns in convolutional filter; C – the number of channels of convolutional filter, that is same as channel number of input data array; I – the number of rows in input array; J – the number of columns in input array; s – step of convolutional filter.

The operations count of the fully connected layer can be calculated using this formula:

$$N_{\mathrm{VS}} = 2\mathrm{MN}, \tag{S.2.2}$$

here M – the length of an input vector; N – the number of artificial neurons in layer.

The operations count of average pooling layer can be calculated using this formula:

$$N_{\mathrm{SPV}} = \frac{2\,\mathrm{IJHWC}}{\mathrm{s}^2}, \tag{S.2.3}$$

here H – the rows number of the pooling window; W – the columns number of the pooling window; C – channels number of input array; I – the number of rows in input array; J – the number of columns in input array; s – the step size of pooling window.

The operations count of L2 pooling layer can be calculated using this formula:

$$N_{\mathrm{SPL2}} = \frac{2\mathrm{IJHWC}}{\mathrm{s}^2}, \tag{S.2.4}$$

here H – the rows number of the pooling window; W – columns number of the pooling window; C – channels number of input array; I – the number of rows in input array; J – the number of columns in input array; s – the step size of pooling window.

The operations count of max or min pooling layer can be calculated using this formula:

$$N_{\mathrm{SPM}} = \frac{2\mathrm{IJHWC}}{\mathrm{s}^2}, \tag{S.2.5}$$

here H – rows number of the pooling window; W – columns number of the pooling window; C – channels number of input array; I – the number of rows in input array; J – the number of columns in input array; s – the step size of pooling window.

The operations count of global average pooling layer can be calculated using this formula:

$$N_{\text{GSPV}} = \text{IJC}, \tag{S.2.6}$$

here C – the channels number of input array; I – the number of rows in input array; J – the number of columns in the input array.

Activation functions are used on the results of fully connected and convolutional layers of DNN. Activation functions are an obligatory part of building any ANN because they provide the network with means to learn some complex information. The most recent activation functions are ReLU, ELU, PReLU, RReLU, ELU, Maxout, Probout, but there are also some classic functions such as hyperbolic tangent and sigmoid used.

The operations count of ReLU activation function can be calculated using this formula:

$$N_{\text{ReLU}} = \text{IJC}, \tag{S.2.7}$$

here C – channels number of input array; I – the number of rows in input array; J – the number of columns in the input array.

ReLU is the Rectified Linear unit. It is as simple as selecting the maximum between the input value and zero. There is also leaky ReLU, where the function is changed in such a way that it additionally adds the negative input value multiplied by some factor a. Another version of that activation function is PReLU, where the static factor is changed to learnable parameter a. Also, there is Randomised ReLU (RReLU), where parameter a is randomized for each example of training data and is fixed for testing data. LReLU, PReLU, and RReLU non zero slopes for the negative part in ReLU consistently improve the performance. Exponential Linear Unit (ELU) uses exponential saturation function with static weight factor a.

Dropout is an efficient method for reducing overfitting presented by Srivastava, Hinton, Krizhevsky, Sutskever, and Salakhutdinov in 2014. The dropout begins from generating a random binary array (same size as an input array) with Bernoulli distribution. After that input is element-wise multiplied by this binary array. This method can prevent the network from becoming too depended on any particular combination of neurons and can force the network to generalize. There is usually only one parameter for that layer, it is a probability factor of dropout. The higher the rate the more values dropped out.

The flattening layer is used to make a one-dimensional vector from a multi-dimensional input array while maintaining the batch size. Otherwise, reshape can also be used, but the resulting output array shape should be defined as a parameter.

Local response normalization and batch normalization can be used as normalization layers to improve data values distribution during learning and execution.

Softmax function is usually used as an output layer in classification tasks.

CNN-FC models with 1, 2, 3 convolutional layers compared. The more convolutional layers are in the model, the better is accuracy, but speed may be reduced. CNN-FC model with 3 layers has the lowest error rate but is slowest. Training speed decreases linearly while testing speed saturates in case of using one convolutional layer.

This may be due to too low number of calculation operations that is not able to saturate graphical processing unit or due to limited PCI Express bandwidth.

CNN-FC models with a pooling area size of 3×3 and 4×4 performed better in comparison to models with 2×2. 2 layers CNN-FC model best pooling area size is 3×3. The speed depends on the pooling area.

There is a noticeable speed drop after going from 3×3 to 4×4 pooling area. It is known, that a bigger pooling area can provide better invariance, but it should not be too big, as it can introduce too much loss of detail. Error rate began to rise after switching from 3×3 to 4×4 area, it also caused bigger speed reduction compared to switching from 2×2 to 3×3.

The pooling layer output size directly depends on the pooling layer stride parameter. Pooling layer with stride bigger than 1 reduces image size by times of stride value. Bigger stride can be used for reducing the number of input features after convolutional layers. To preserve features, but improve shift-invariance, the stride of 1 can be used. The same structure models have been compared with the different stride of pooling layers. Higher stride value reduces the number of features before fully connected layers. That causes a decrease in performance and an increase in the speed of training and testing.

One of the main parameters for the CNN layer is kernel size. Bigger kernels can be more exact and smaller ones are more abstract. DNN models best kernel size can depend on image set and image size, it is needed to test before selecting the size. CNN-FC models with two CNN layers with kernel sizes of 3×3, 5×5, 7×7 and 11×11 where compared by error rate and speed. The best accuracy in the analyzed network is achieved using a 5×5 kernel size. Bigger kernels introduce more calculations and therefore reduces speed. Notice, that switching 3×3 (9 weights) to 5×5 (25 weights) and 7×7 (49 weights) to 11×11 (121 weights) caused less speed drop compared to switching from 5×5 to 7×7. This can be related to GPU optimizations for smaller kernels as 3×3 and 5×5 are more common compared to 7×7 and 11×11.

The purpose of fully connected layers after CNN layers is to make a classification based on features that come from CNN layers, and there the count of neurons needs to be selected. Four tests with 128, 256, 512, 1024 neurons in the hidden layer of the FC part of CNN-FC have been made.

The testing of Intel i7-6900K CPU, Nvidia Geforce GTX 960 GPU and Nvidia Geforce 1080 Ti GPU provided the speed of training in samples per second. The achieved speed of GTX 960 GPU was at least 3 times faster than CPU, and GTX 1080 Ti was at least 9 times faster than the CPU. This showed, that GPU has very high performance while energy consumption is at a similar level as CPU.

The error rate is not directly related to neuron numbers in the hidden layer. There is some value, which grants the best combination of both error rate and speed for a defined structure network. Bigger neuron count will make DNN speed slower with diminishing returns in error rate.

During the research, a few CNN has been implemented in Google Coral Edge TPU. The test showed that it is necessary to fit the algorithm inside the cache memory of the unit. Otherwise, there is a lot of communication happening with system memory, and performance can be reduced.

Based on the research presented in this chapter the CNN design methodology was created. This methodology consists of the following steps:

1. CNN base structure is selected. This CNN should be implemented in selected accelerating unit and provide similar speed and accuracy as required by the task:
   - a few base structures are selected;
   - operations count is calculated using provided formulas;
   - the used memory amount is checked;
   - CNN, which fit in resources provided by selected accelerating unit, is selected;
   - analysis of execution time and accuracy of selected CNNs is performed using trained networks or transfer learning;
   - the best base CNN structure is selected.
2. The structure of base CNN is being changed to achieve target performance:
   - the parameters of CNN layers are selected based on the size of the training dataset, to prevent overfitting.
   - the trade-off between accuracy and execution speed is achieved by changing such parameters as network depth, resolution and number of filters, one at a time and monitoring its impact.
3. The CNNs with selected parameters are tested:
   - CNNs are trained beginning with randomized weight arrays;
   - CNNs are additionally trained using hard-to-learn samples.
4. The best CNN is selected. In case, the selected CNN does not satisfy the requirements of tasks, the modification of these networks should be made and the process steps 2 and 3 repeated.

The proposed CNN design methodology can be used for partial automatization of CNN structure and parameter selection. The algorithms, which were designed and implemented using the described methodology, are presented in Chapter 3.

3. Application of Convolutional Neural Networks for Image Analysis

This section describes the application of the proposed design methodology for CNN implementation for two image processing tasks. One of them is a feature description for eye retinal images registration. Another is the classification of the road pavement type and condition from the forefront road image received from the camera.

One of the important tests performed during the ophthalmological examination is retinal imaging. A set of eye fundus images may need to be investigated and compared in order to perform an assessment of retinal diseases. The doctor may need to perform a comparison of images taken during separate visits; therefore, images should be overlaid. Natural changes in the retina (clinically relevant or irrelevant) and deviations of imaging procedure parameters cause geometric and intensity distortions of retinal images that complicate image registration.

Feature-based image registration consists of several steps: keypoint detection, keypoint description, matching of detected and descripted keypoints, and image warping. Keypoint descriptors should represent a local image area in an invariant and discriminative

manner so the distance between descriptors reflects image similarity. There are well-known traditional, hand-crafted descriptors such as SIFT, SURF, BRISK, FREAK, KAZE and learned descriptors based on Deep Neural Networks (DNN).

The aim of this experiment is creating a learnable, DNN-based keypoint descriptor, optimized for retinal image registration. Such a decision is based on our earlier comparative experiments on keypoint detector and descriptor performances for registration of eye fundus images.

A specialized dataset of retinal image patch pairs was created for training using the Siamese CNN structure. This dataset is based on images from 9 datasets: Chase DB, Diaret DB, DTSET1, DTSET2, HRF base, three Messidor datasets, RODREP. A comparative evaluation of feature descriptors has been performed on the FIRE dataset. There were 3153 initial images in total. Most of the datasets had high-resolution images, therefore we used various downscaling factors to generate more images to increase the scale-richness of the dataset. Introduced downscaling led to a dataset of 11 279 retinal images.

A set of retinal image patches is needed to train the DNN-based feature descriptor. Points of interest for image patch cropping were selected based on the results of popular keypoint detectors. To make the training of descriptors less dependent on single feature detector performance, a set of detectors have been employed: FAST, Harris, MinEigen, Hessian, BRISK, SURF, SIFT, KAZE. Each of these feature detectors have been used to select up to 500 keypoints and grouped them into 200 clusters using the k-means algorithm. Each cluster's mean point was used as a center coordinate of the patch. This way keypoints have been prevented from being feature detector specific. The size of the cropped image patch was 101×101 pixels. A set of original (unaugmented) patches was increased by introducing vertical, horizontal and cross flipping of original patches.

Image patches were collected from unpaired fundus images. Synthetic pairs of collected patches were generated by several image augmentations. Image augmentations were selected such that would lead to improvement of the DNN feature detector's invariance to geometric distortions. The rotation was limited to 10 degrees in both directions; projective distortions included skew, scale and perspective transformations. processed image patches were reduced to 64×64 pixels size patches by cropping, for DNN descriptor training. 4 new image patches from each of the initial patch have been generated using augmentation.

Finally, a set of possible combinations of image patch pairs were created. There are two kinds of image pairs included: matching and non-matching pairs. Matched pairs were made of augmented image patches that originated from the same initial patch. Pairs that do not match include patches from the same image but different locations to minimize the amount of false negative pairs. It is likely to find more similar patches from distinct fundus images than in a set of patches from one image.

As a result, a training set of 17 million samples was created. Each training sample has a pair of image patches and a similarity label. "0" has been used to denote non-similar pairs and "1" to denote similar pairs. Half of the samples are image pairs that are similar and another half non-similar. 1/6-th of the final dataset was used for validation purposes during training.

Siamese network (which is used only during the descriptor training phase) is composed of two identical DNN models that share learnable weight coefficients but process

different images from the training set. Each model outputs a 128-dimensional floating-point feature descriptor (Fig. S.3.1).

Euclidean distance is computed to evaluate the similarity of the descriptors, and this distance is used in the contrastive loss as a cost function for DNN learning:

$$l = 0.5\left(t \cdot d^2 + (1-t) \cdot (\mathrm{m} - d)^2\right), \qquad \text{(S.3.1)}$$

here l – loss; t – target; d – Euclidean distance; m – margin.

Selected margin value was m was 5, because lower and higher margins resulted in worse learned feature descriptor performance and a higher probability of underfitting during training. ADAM stochastic optimization method has been used with various learning rates, and constant parameters for first-order momentum $\beta_1 = 0.9$, second-order momentum $\beta_2 = 0.999$ and small fixed value $\varepsilon = 10^{-6}$.

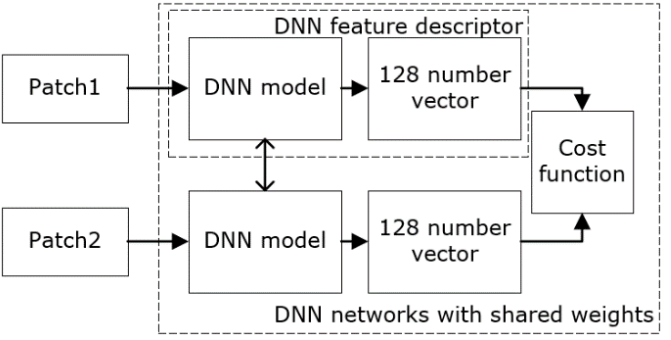


Fig. S.3.1. Outline of Siamese network structure used for training

In this experiment, two basic architectures of convolutional neural networks were used. The first one is a fully connected network (FCN). The second one is a convolutional neural network with fully connected layers (CNN-FC). There were two modifications of CNN-FC: one with local response normalization (LRN) and other with batch normalization (BN) layers. CNN-FC is a larger network compared to FCN and it has more kernels in convolutional layers to bring more information to FC layers. Dropout layers were used between FC layers to prevent overfitting.

During the training, two approaches were tested: learning on training set only; learning on the training set and hard to learn examples. Training on hard to learn samples improved learning speed. Nearly 1 million hard to learn samples were selected after the first training stage. The selection was based on a loss function result.

The performance of learned feature descriptors has been evaluated and compared to the performance of a collection of well-known hand-crafted ones. For evaluation, the FIRE dataset was selected, as it contains paired images with provided ground truth matching vectors. The matching performance of descriptors is presented as histograms of appearances of ground truth image patch pairs in ranked results of feature vector matches. Rank-1 values (the proportion of matches in which the correct match appeared in the top position of the candidate match list, i.e. the first bin of the histogram). Candidate match

list of the ground truth pairs of feature points is acquired after computing Euclidean distances between all descriptors that were extracted at these points and sorting them in ascending order. The results are presented in Table S.3.1.

Table S.3.1. Comparison of Local Feature Descriptors

Hand-crafted descriptor	Rank-1, %	Average processing duration, ms	Learned descriptor	Rank-1, %	Average processing duration, ms
BRISK	40.9	16.2	DNN-D-RI (1)	35.4	1.4
FREAK	22	2.4	DNN-D-RI (2)	65.30	1.3
SIFT	99.3	21	DNN-D-RI (3)	84.9	1.4
SURF	98.8	0.37	DNN-D-RI (4)	89.2	1.4
KAZE	95	3.4	DNN-D-RI (5)	62.5	1.4

Note. The grey fields from left to right indicate: the most accurate hand-crafted feature descriptor, the fastest feature descriptor, the most accurate learned feature descriptor.

The results of descriptor comparisons are summarized in Table S.3.1. Binary feature descriptors BRISK and FREAK provide the least accuracy, floating-point feature descriptors SIFT, SURF and KAZE give good performance finding at least 95% of all (1340 in total) ground truth point pairs have been matched correctly. Variants of our learned DNN-based feature descriptors perform between binary and floating-point handcrafted feature descriptors. Additional insights are that DNN-D-RI variants 2 and 3 trained on hard to learn samples perform better compared to variants 1 and 3. Variants 3 and 4 (CNN-FC) are better compared to variants 1 and 2 (FCN), so FC layers learned to extract more useful information from the image patches. Variant 5 that included BN layers performed worse compared to variant 4 that uses LRN and is trained on hard to learn samples.

DNN-based attribute descriptors trained using eye retinal clippings have made it possible to achieve up to 89% correct-set pairs of Euclidean true points. This is a better result than the BRISK (40%) and FREAK (22%) attribute descriptors but can be improved compared to SIFT, SURF and KAZE.

The fast-developing area of forefront road identification is based on image processing from monocular and binocular (stereo) cameras for using extracted data in ADAS or self-driving vehicle systems. Monocular vision is usually used for determining whether conditions, illumination, detection of the path, obstacles, road lines, and edges. The proposed design method was used for CNN-based algorithm for classification of 12 different combinations of road pavement types and conditions: ice, snow wet, snow dry, gravel wet, gravel dry, cobble wet, cobble dry, concrete wet, concrete dry, asphalt wet, asphalt moist, asphalt dry.

Deep neural networks require big datasets to learn from. Therefore, more than 250 thousand of labeled images were collected and prepared as training, validation and testing dataset. First, a dash car camera was used to collect 1920×1080 resolution video. Second, the video was separated into image frames. At last, images of different road pavement

types and conditions were selected and processed preprocessed by squeezing into 512×512 pixel size format as input. This was made because images of 256×256 pixel size had not enough information about road pavement.

A convolutional neural network (CNN) similar to AlexNet, but with some changes, has been used for road pavement type and state estimation. The structure of this CNN is provided in Fig. S.3.2. A deep neural network was made of 5 convolutional layers and 3 fully connected layers, all with Rectified Linear Units as the activation function. In order to reduce classifier dependence on separate pixels, Dropout layers have been used.

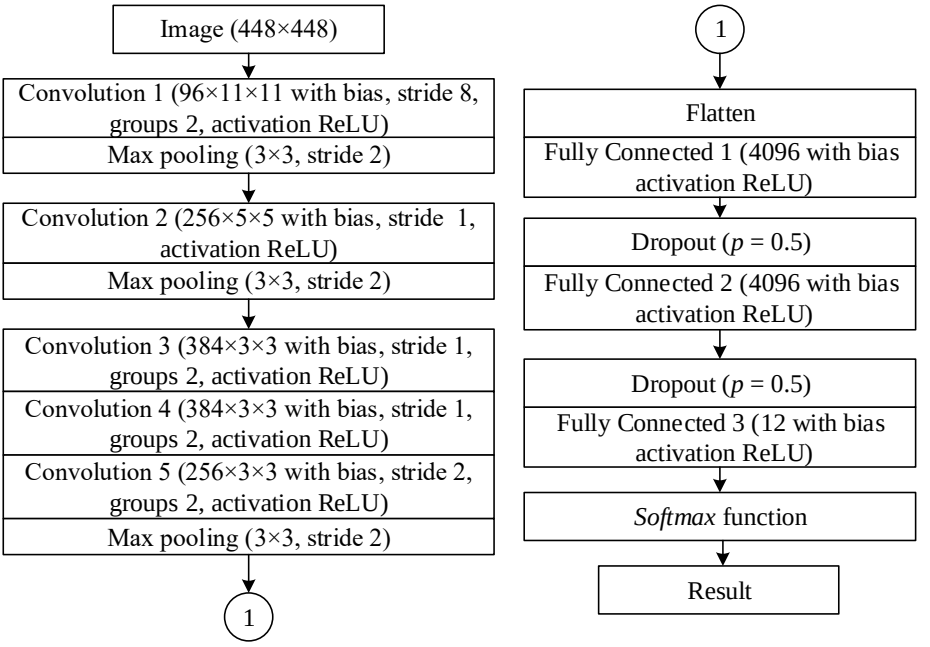


Fig. S.3.2. Convolutional Neural Network with Fully Connected Layers

Fig. S.3.3 presents an input image, kernel weighs and extracted feature maps from the first convolution layer. The feature maps show what information layer learned to react to. These are various parts of the video frame, such as dashboard, sky, road, and other objects road infrastructure and traffic.

CNN was trained on GPU workstation using DIGITS with caffe backend then converted to TensorRT and tested on Nvidia Jetson TX2 embedded system. This embedded system consumes only up to 15 W of power and provides up to 1 TFLOP of deep neural network calculations. The structure of the created application executed by the embedded system is shown in Fig S.3.4.

Nvidia Jetson TX2 provides hardware acceleration for most parts that are required for real-time image analysis. This embedded system includes camera signal processing, video decoding and encoding hardware pipelines that saves resources of central processing

unit and graphics processing unit for CNN based image classification algorithm implementation. Created application, can be used to process real-time video images from a camera or video record, and provides the result as an overlay on a displayed image or a line of text in console or results file. The actual duration of image processing by CNN-based algorithm was measured between the moment of image input to CNN and the output of the result.

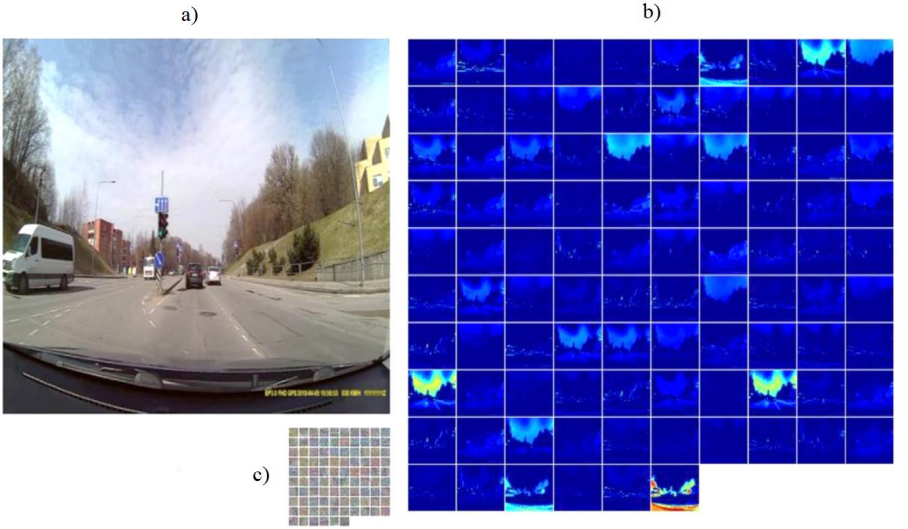


Fig. S.3.3. Operation of the first convolutional layer: a) input image; b) calculated feature maps; c) used convolutional kernels

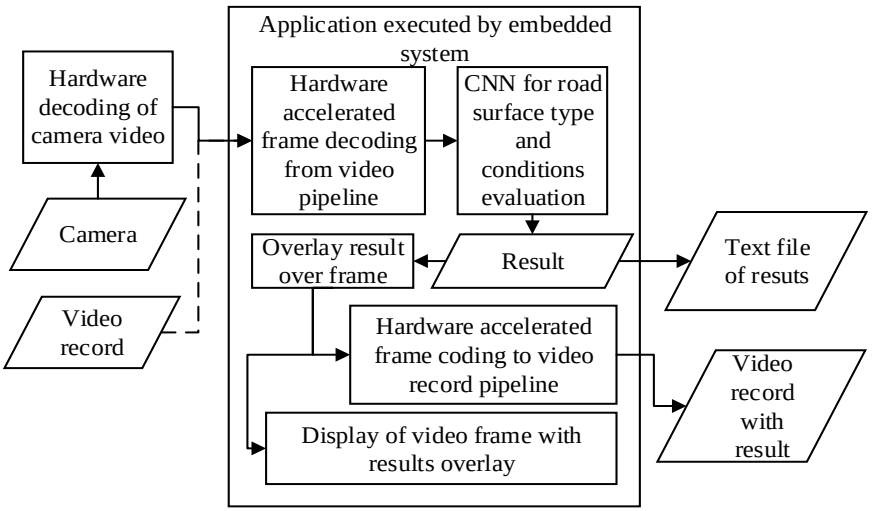


Fig. S.3.4. Scheme of image capture, processing and result indication in the embedded system

The output of the system can be useful for the improvement of car safety and comfort. The road pavement type and conditions of the forefront road can be used for braking and suspension system parameters selection in advance. Usage of such a system can potentially reduce braking distance on various road pavement types and also allow a higher comfort level when the semi-active suspension is being used.

Identifying the road pavement type and conditions usually are a difficulty due to uneven lighting, shadows, sudden changes in image brightness, bad visibility due to fog, heavy rain, hail or snow. In these cases, the visible light spectrum cameras cannot guarantee the high accuracy of road pavement conditions.

The CNN-based road pavement type and condition evaluation algorithm are implemented in the embedded system Nvidia Jetson TX2. The implemented algorithm achieves 20 ms frame processing time and up to 90% accuracy. The feeding of the cropped frame part of the road increased accuracy from 84% to 90%.

General Conclusions

1. Convolutional neural networks have been successfully implemented in accelerating units, using a compiled list of CNN elements and the proposed design methodology, to solve two tasks of real-time image analysis.
2. A CNN-based local feature description algorithm that has been created using the proposed methodology, used for registration of feature point pairs of FIRE dataset provides 89% accuracy, which is similar to the accuracy of traditional feature descriptors.
3. A CNN-based algorithm, which has been created using the proposed methodology, evaluates road pavement type and condition from single images of the training set with an accuracy of 90 %.
4. A CNN-based road pavement type and condition evaluation algorithm, when implemented in an embedded system Nvidia Jetson TX2, process one image in 20 ms.

Based on general conclusions, it can be stated that the hypothesis, which has been raised in the dissertation, is confirmed.

Priedai[3]

A priedas. Disertacijos autoriaus sąžiningumo deklaracija
B priedas. Bendraautorių sutikimai teikti publikacijų medžiagą disertacijoje
C priedas. Autoriaus mokslinių publikacijų disertacijos tema sąrašas

[3] Priedai pateikiami pridėtoje kompaktinėje plokštelėje.

Eldar ŠABANOVIČ

SĄSŪKOS DIRBTINIŲ NEURONŲ TINKLŲ ĮGYVENDINIMAS
SPARTINANČIUOSIUOSE ĮRENGINIUOSE
VAIZDAMS ANALIZUOTI REALIUOJU LAIKU

Daktaro disertacija

Technologijos mokslai,
elektros ir elektronikos inžinerija (T 001)

IMPLEMENTATION OF CONVOLUTIONAL NEURAL NETWORKS
IN ACCELERATING UNITS
FOR REAL-TIME IMAGE ANALYSIS

Doctoral Dissertation

Technological Sciences,
Electrical and Electronic Engineering (T 001)

2019 11 05. 10,5 sp. l. Tiražas 20 egz.
Vilniaus Gedimino technikos universiteto
leidykla „Technika“,
Saulėtekio al. 11, 10223 Vilnius,
http://leidykla.vgtu.lt
Spausdino UAB „BMK leidykla“,
A. Mickevičiaus g. 5, LT-08119 Vilnius