VILNIUS TECH
Elektroninių sistemų katedra

Dainius VARNA

ELEKTRONIKOS KOMPONENTŲ ANT KONVEJERIO SKAIČIAVIMO METODŲ TYRIMAS

INVESTIGATION OF METHODS FOR COUNTING ELECTRONIC COMPONENTS ON A CONVEYOR

Magistro baigiamasis darbas

Elektronikos studijų programa, valstybinis kodas 6211EX050
Kompiuterizuotų elektroninių sistemų specializacija
Elektronikos inžinerijos studijų kryptis

Vilnius, 2022

VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS
ELEKTRONIKOS FAKULTETAS
ELEKTRONINIŲ SISTEMŲ KATEDRA

TVIRTINU
Katedros vedėjas

_____
(parašas)
Prof. dr. A. Serackis
2022 m. 05 mėn. 25 d.

Dainius VARNA

ELEKTRONIKOS KOMPONENTŲ ANT KONVEJERIO SKAIČIAVIMO METODŲ TYRIMAS

INVESTIGATION OF METHODS FOR COUNTING ELECTRONIC COMPONENTS ON A CONVEYOR

Magistro baigiamasis darbas

Elektronikos studijų programa, valstybinis kodas 6211EX050
Kompiuterizuotų elektroninių sistemų specializacija
Elektronikos inžinerijos studijų kryptis

Vadovas dr. Vytautas Abromavičius 2022-05-24
(*pedag. vardas, moksl. laipsnis, vardas, pavardė*) (*parašas*) (*data*)

Konsultantas - _____
(*pedag. vardas, moksl. laipsnis, vardas, pavardė*) (*parašas*) (*data*)

Lietuvių kalbos konsultantas dr. Aušra Žemienė _____
(pedag. vardas, moksl. laipsnis, vardas, pavardė) (parašas) (data)

Vilnius, 2022

VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS

ELEKTRONIKOS FAKULTETAS

ELEKTRONINIŲ SISTEMŲ KATEDRA

Technologijos mokslų sritis
Elektros ir elektronikos inžinerijos mokslo kryptis
Elektronikos inžinerijos studijų kryptis
Elektronikos inžinerijos studijų programa, valst. kodas 6211EX050
Kompiuterizuotų elektroninių sistemų specializacija

TVIRTINU
Katedros vedėjas
_____
prof. dr. A. Serackis
2022 m. 05 mėn. 23 d.

MAGISTRO BAIGIAMOJO DARBO UŽDUOTIS

Studentui Dainius VARNA

Baigiamojo darbo tema: Elektronikos komponentų ant konvejerio skaičiavimo metodų tyrimas

Investigation Of Methods For Counting Electronic Components On A Conveyor

Patvirtinta 2020 m. lapkričio mėn. 25 d. dekano potvarkiu Nr. 15Pel.

Baigiamojo darbo užbaigimo terminas 2022 m. gegužės mėn. 25 d.

Darbo tikslas – išnagrinėti smulkių objektų atpažinimui ir aptikimui taikomus intelektualiuosius metodus ir pritaikyti pasirinktą metodą elektronikos komponentų skaičiavimo ant konvejerio sistemai.

Darbo uždaviniai

1. Atlikti vaizdo apdorojimo bei klasifikavimo, giliojo mokymosi ir objektų aptikimo metodų analitinę apžvalgą.

2. Sudaryti individualų duomenų rinkinį klasifikavimo modeliui mokyti ir suformuluoti elektronikos komponentų skaičiavimo metodą eksperimentiniams tyrimams atlikti.

3. Ištirti pasirinktų metodų efektyvumą, pateikti pasiūlymus taikant metodus praktikoje ir palyginti gautus rezultatus.

Aiškinamojo rašto turinys

Įvadas

- Darbo aktualumas ir tikslas
- Darbo uždaviniai
- Naudoti tyrimo ir analizės metodai
- Darbo naujumas ir praktinė nauda
- Darbo struktūra

1. Vaizdo apdorojimo ir klasifikavimo metodų analitinė apžvalga
2. Objektams aptikti ir atpažinti taikomų metodų analitinė apžvalga
3. Duomenų rinkinio sudarymas ir paruošimas elektronikos komponentų aptikimo ir skaičiavimo metodų eksperimentiniams tyrimams
4. Elektronikos komponentų aptikimo ir skaičiavimo metodų tyrimų rezultatai

Apibendrinimas. Išvados

Literatūra

Priedai

A priedas. Elektronikos komponentų aptikimo ir skaičiavimo metodų tyrimo rezultatai

B priedas. Pranešimo 25-oje Lietuvos jaunųjų mokslininkų konferencijoje medžiaga

Baigiamojo darbo rengimo konsultantas: -

(pedag. vardas, moksl. laipsnis, vardas, pavardė)

Vadovas (*parašas*) dr. Vytautas Abromavičius

(pedag. vardas, moksl. laipsnis, vardas, pavardė)

Užduotį gavau (*parašas*) 2022 m. gegužės mėn. 25 d.

Vilniaus Gedimino technikos universiteto egzaminų sesijų ir baigiamųjų darbų rengimo bei gynimo organizavimo tvarkos aprašo
2 priedas

(Baigiamojo darbo sąžiningumo deklaracijos forma)

VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS

Dainius Varna, 20200033

(Studento vardas ir pavardė, studento pažymėjimo Nr.)

Elektronikos fakultetas

(Fakultetas)

Elektronikos inžinerija, EKSfm-20

(Studijų programa, akademinė grupė)

BAIGIAMOJO DARBO (PROJEKTO)
SĄŽININGUMO DEKLARACIJA

2022 m. gegužės 25 d.

Patvirtinu, kad mano baigiamasis darbas tema „Elektronikos komponentų ant konvejerio skaičiavimo metodų tyrimas“ patvirtintas 2020 m. lapkričio 25 d. dekano potvarkiu Nr. 159el , yra savarankiškai parašytas. Šiame darbe pateikta medžiaga nėra plagijuota. Tiesiogiai ar netiesiogiai panaudotos kitų šaltinių citatos pažymėtos literatūros nuorodose.

Parenkant ir įvertinant medžiagą bei rengiant baigiamąjį darbą, mane konsultavo mokslininkai ir specialistai: daktaras Vytautas Abromavičius. Mano darbo vadovas daktaras Vytautas Abromavičius.

Kitų asmenų indėlio į parengtą baigiamąjį darbą nėra. Jokių įstatymų nenumatytų piniginių sumų už šį darbą niekam nesu mokėjęs.

D. Varna
(Parašas)

Dainius Varna
(Vardas ir pavardė)

Dainius VARNA

Elektronikos komponentų ant konvejerio skaičiavimo metodų tyrimas. Magistro baigiamasis darbas elektronikos inžinerijos laipsniui. Vilniaus Gedimino technikos universitetas. Vilnius, 2022, 71 p., 27 iliustr., 16 lent., 59 bibl., 2 priedai.

Šio darbo tikslas – išnagrinėti smulkiems objektams atpažinti ir aptikti taikomus intelektualiuosius metodus ir pritaikyti pasirinktą metodą elektronikos komponentų ant konvejerio skaičiavimo sistemai. Tikslui pasiekti buvo atlikta vaizdo apdorojimo bei klasifikavimo, giliojo mokymosi ir objektų aptikimo metodų analitinė apžvalga. Buvo sukurtas individualus duomenų rinkinys klasifikavimo modeliui mokyti ir suformuluotas elektronikos komponentų skaičiavimo metodas eksperimentiniams tyrimams atlikti. Eksperimentinio tyrimo dalyje įgyvendinti ir ištirti mokymui pasirinkti objektų aptikimo modeliai, įvertintas jų efektyvumas (vaizdo apdorojimo sparta, vidutinis aptikimo tikslumas, išvedimo laikas) aptinkant elektronikos komponentus. Įdiegus pasiūlytą skaičiavimo metodą papildomai įvertinti suskaičiuotų elektronikos komponentų kiekiai, metodo poveikis modelių efektyvumui. Sudaryta sistema geba identifikuoti ir skaičiuoti keturių rūšių elektronikos komponentus realiuoju laiku.

Prasminiai žodžiai: intelektualieji metodai, gilusis mokymasis, elektronikos komponentai, objektų aptikimas, objektų klasifikavimas, skaičiavimo metodas.

Dainius VARNA

Investigation of methods for counting electronic components on a conveyor. Masters' graduation thesis for electronics engineering degree, Vilnius Gediminas Technical University. Vilnius, 2022, 71 p., 27 pictures., 16 tables., 59 lit., 2 ex.

The aim of this thesis is to investigate intelligent techniques used to identify and detect tiny objects and apply chosen method for counting electronic components on a conveyor. For achieving the aim, an analytical review of image processing and classification, deep learning and object detection methods was conducted. An individual dataset for training regression-based object detectors to identify electronic components was collected and a method for counting electronic components was formulated. During the experimental research, trained object detection models were implemented and investigated, their image processing speed, mean average precision, inference time while detecting electronic components were evaluated. After the implementation of the proposed calculation method, quantities of the calculated components and the effect of the method on the efficiency of the models were additionally evaluated and compared. The system is capable to identify and count four different types of electronic components.

Keywords: intelligent techniques, deep learning, electronic components, object detection, object classification, calculation method.

TURINYS

Paveikslų ir lentelių sąrašas **10**

Žymenys ir santrumpos **12**

Įvadas **13**

Darbo aktualumas ir tikslas 13

Darbo uždaviniai 13

Naudoti tyrimo ir analizės metodai 14

Darbo naujumas ir praktinė nauda 14

Darbo struktūra 14

1. Vaizdo apdorojimo ir klasifikavimo metodų analitinė apžvalga **15**

1.1. Klasikiniai vaizdo apdorojimo metodai 15

1.2. Šiuolaikiniai vaizdo apdorojimo metodai 16

1.3. Šiuolaikiniai vaizdo klasifikavimo metodai 19

1.4. Duomenų išankstinio apdorojimo metodai 20

1.5. Duomenų rinkinių sudarymas 21

1.6. Skyriaus apibendrinimas 22

2. Objektams aptikti ir atpažinti taikomų metodų analitinė apžvalga **23**

2.1. Mokymo algoritmų apžvalga 23

2.2. Klasifikacija pagrįstų objektų aptikimo algoritmai 27

2.2.1. Regioninis sąsūkos neuronų tinklas 27

2.2.2. Erdvinis piramidžių telkimo tinklas 28

2.2.3. Greitasis regioninis sąsūkos neuronų tinklas 30

2.2.4. Greitesnysis regioninis sąsūkos tinklas 31

2.2.5. Maskuotasis regioninis sąsūkos tinklas 32

2.3. Regresija pagrįsti objektų aptikimo algoritmai 32

2.3.1. YOLO tinklo algoritmas 33

2.3.2. SSD tinklo algoritmas 34

2.3.3. *SqueezeDet* tinklo algoritmas 34

2.4. Tyrimams skirtos sistemos projektavimas ir analizė 37

2.5. Skyriaus apibendrinimas 39

3. Duomenų rinkinio sudarymas ir paruošimas elektronikos komponentų aptikimo ir skaičiavimo metodų eksperimentiniams tyrimams **40**

3.1. Mokymo duomenų paruošimas 40

3.2. Tinklų mokymas 44
3.3. Objektų aptikimas ir skaičiavimas 47
3.4. Skyriaus apibendrinimas 50

4. Elektronikos komponentų aptikimo ir skaičiavimo metodų tyrimų rezultatai 51

4.1. Tinklų objektų aptikimo efektyvumo tyrimo rezultatai 51
4.2. Tinklų objektų skaičiavimo efektyvumo tyrimo rezultatai 56
4.3. Objektų aptikimo ir skaičiavimo algoritmų lyginamosios analizės rezultatai 61
4.4. Skyriaus apibendrinimas 63

Apibendrinimas. Išvados 64

Literatūra 66

PRIEDAI 72

A priedas. Elektronikos komponentų aptikimo ir skaičiavimo metodų tyrimo rezultatai ... 72
B priedas. Pranešimo 25-oje Lietuvos jaunųjų mokslininkų konferencijoje medžiaga 74

PAVEIKSLŲ IR LENTELIŲ SĄRAŠAS

1.1 pav. Pagrindiniai klasikinio vaizdo apdorojimo žingsniai: a) įėjime spalvotas paveikslas; b) SLIC superpikseliai; c) išėjime grupuojamas paveikslas iš sujungtų RAG; d) prisotinimo spalvų kanalas; e) paveikslas su paryškintais kraštais; f) iš superpikselių ir kraštų žemėlapio sukonstruotas RAG [7] 16

1.2 pav. *Halide* programa kurtas kompiliatorius [8] 17

1.3 pav. *InceptionV3* modulio struktūra: a) naudoja 1 – 4-us ir 10 – 11-us *InceptionV3* modulius; b) naudoja 5 – 9-us *InceptionV3* modulius [12] 20

2.1 pav. Mašininis mokymo procesas [16] 23

2.2 pav. Kaupiamoji paskirstymo funkcija, apibūdinanti algoritmų pasiektus greičius: a) Gauso-IC; b) IMAC [20] 24

2.3 pav. Atsitiktinai keičiamo apšviestumo rezultatai [21] 26

2.4 pav. Užfiksuoti medvilnės užteršimo vaizdai: a) sistemos užfiksuotas vaizdas; b) CNN tinklo atpažinti taršos objektai [22] 26

2.5 pav. RCNN tinklo architektūra [26] 28

2.6 pav. Erdvinio piramidžių telkimo tinklo architektūra [26] 29

2.7 pav. Greitojo RCNN tinklo architektūra [26] 30

2.8 pav. Greitesniojo RCNN tinklo architektūra [26] 31

2.9 pav. Maskuotojo regioninio sąsūkos tinklo architektūra [26] 32

2.10 pav. YOLO tinklo algoritmo architektūra [26] 33

2.11 pav. SSD tinklo algoritmo architektūra [26] 34

2.12 pav. *SqueezeDet* tinklo algoritmas [33] 35

2.13 pav. RHF tinklo architektūra [34] 36

2.14 pav. Elektronikos komponentų atpažinimo ir skaičiavimo sistemos sandaros schema 37

3.1 pav. Pasirinkti elektronikos komponentai duomenų rinkiniui: a) kondensatorius; b) rezistorius; c) diodas; d) tranzistorius 40

3.2 pav. *SMD* komponentų duomenų rinkinio nuotraukų pavyzdžiai: a) įpakuoti komponentai; b) išpakuoti pavieniai komponentai 41

3.3 pav. Išankstinio apdorojimo metodai skirti duomenų didinimui: a) atsitiktiniai išpjovimai; b) spalvų reguliavimas; c) atsitiktiniai pasukimai tarp -15° ir +15° kampu 42

3.4 pav. Objektų žymėjimų žemėlapiai: A – pirmas duomenų rinkinys; B – antras duomenų rinkinys; C – trečias duomenų rinkinys 44

3.5 pav. Tinklo mokymo struktūrinė schema 45

3.6 pav. YOLOv4-Scaled ir YOLOv5s modelių nuostolių grafikai 47

3.7 pav. Elektronikos komponentų atpažinimo ir skaičiavimo programos veikimo algoritmas.....49
4.1 pav. Tyrimams naudotų vaizdo įrašų kadrai: a) kondensatoriai; b) rezistoriai; c) diodai; d) visos komponentų klasės (rezistoriai, kondensatoriai, diodai ir tranzistoriai) 52
4.2 pav. Kadras iš kondensatorių atpažinimo tyrimo naudojant YOLOv4 detektorių 55
4.3 pav. Kadras iš rezistorių aptikimo ir skaičiavimo tyrimo su YOLOv4-tiny ir YOLOv4 detektoriais 60

1.1 lentelė. Algoritmų vykdymo laikai (sekundėmis) [9].....18
1.2 lentelė. Tyrimams atlikti naudotų GPU ir CPU specifikacijos [9] 18
2.1 lentelė. Rezultatai su modifikuotu greitesniuoju RCNN ir *SORSI* duomenų rinkiniu [27].....31
2.2 lentelė. RESC ir hibridinio sintezės modulio tyrimas [34].....36
3.1 lentelė. Duomenų rinkinių specifikacija 43
3.2 lentelė. Objektų aptikimo modelių specifikacija 46
4.1 lentelė. Kondensatorių aptikimo tyrimo rezultatai 53
4.2 lentelė. Rezistorių aptikimo tyrimo rezultatai.....53
4.3 lentelė. Diodų aptikimo tyrimo rezultatai 54
4.4 lentelė. Visų komponentų aptikimo tyrimo rezultatai 55
4.5 lentelė. Kondensatorių skaičiavimo tyrimo rezultatai 57
4.6 lentelė. Rezistorių skaičiavimo tyrimo rezultatai 58
4.7 lentelė. Diodų skaičiavimo tyrimo rezultatai.....59
4.8 lentelė. Visų komponentų skaičiavimo tyrimo rezultatai 60
4.9 lentelė. Bendri komponentų aptikimo tyrimo rezultatai 61
4.10 lentelė. Bendri komponentų skaičiavimo tyrimo rezultatai 62

ŽYMENYS IR SANTRUMPOS

AI (angl. *Artificial Intelligence*)	dirbtinis intelektas
AP (angl. *Average Precision*)	vidutinis tikslumas
API (angl. *Application Programming Interface*)	aplikacijų programavimo sąsaja
BB (angl. *Bounding Boxes*)	aprėpties dėžės
CNN (angl. *Convolutional Neural Network*)	sąsūkos neuronų tinklas
CPU (angl. *Central Processing Unit*)	centrinis apdorojimo blokas
CSI (angl. *Camera Serial Interface*)	kameros serijinė sąsaja
CV (angl. *Computer Vision*)	kompiuterinė rega
DL (angl. *Deep Learning*)	gilusis mokymas
DNN (angl. *Deep Neural Network*)	gilusis neuronų tinklas
FPS (angl. *Frames per Second*)	kadrai per sekundę
GPU (angl. *Graphic Processing Unit*)	grafikos apdorojimo blokas
HSV (angl. *Hue Saturation Value*)	atspalvio sodrumo vertė
IoU (angl. *Intersection over Union*)	aibių sankirtos ir sąjungos santykio vertė
IT (angl. *Inference Time*)	išvedimo laikas
mAP (angl. *mean Average Precision*)	vidutinio tikslumo vidurkis
mIT (angl. *mean Inference Time*)	išvedimo laiko vidurkis
RAG (angl. *Region of Adjacency Graph*)	gretinamo regiono grafikas
ROI (angl. *Region of Interest*)	dominantis regionas
RS (angl. *Remote Sensing*)	nuotolinis jutimas
SMD (angl. *Surface Mount Device*)	paviršiuje montuojamas įtaisas
SVM (angl. *Support Vector Machine*)	atraminių vektorių klasifikatorius

ĮVADAS

Per pastaruosius keletą metų gilusis mokymasis (angl. *Deep Learning* (DL)) sparčiai tobulėjo daugelyje sričių, pavyzdžiui, kompiuterinės regos (angl. *Computer Vision* (CV)), balso atpažinimo ir natūralios kalbos apdorojimo. Kadangi nuotolinis jutimas (angl. *Remote Sensing* (RS)) turi daug unikalių iššūkių, susijusių su jutikliais ir programomis, neišvengiamai RS remiasi daugeliu tų pačių teorijų kaip ir CV, pavyzdžiui, statistika, sinteze ir giliuoju mokymusi [1]. Objektų aptikimas yra viena iš pagrindinių ir sudėtingiausių kompiuterinės paieškos natūraliuose vaizduose problemų, siekiant surasti objektus iš gausybės anksčiau nustatytų kategorijų [2]. Spartėjančios ir vis našesnės tampančios įterptinės sistemos suteikia galimybes investuoti ir pritaikyti naujus metodus produkcijos gamyboje. Šiame darbe tiriami klasifikacijos principu veikiantis algoritmai, skirti objektams vaizdo sraute identifikuoti ir jų pozicijai nustatyti. Taip pat siekiama išanalizuoti jų pritaikymo galimybes realiojo laiko sistemoje elektronikos komponentams skaičiuoti.

Darbo aktualumas ir tikslas

Šiuolaikiniai giliojo mokymosi metodai atveria galimybes analizuoti smulkių objektų aptikimą realiuoju laiku. Pramonėje ir gamyboje kasdien yra naudojamos įvairios staklės, užtikrinančios akivaizdų gamybos proceso greičio padidėjimą. Pasitelkus intelektualiuosius metodus galima šį procesą dar labiau patobulinti. Kadangi gaminami produktai gali turėti trūkumų (pvz., skylės, įdubimai, įbrėžimai ir kiti defektai), norint juos greičiau pastebėti gali būti pasitelktas objektų atpažinimas [3]. Objektų aptikimo sistemos skiriasi viena nuo kitos savo metodika, kurią sudaro duomenų rinkinių sudarymas, mokymas ir atpažinimas [4]. Pritaikant vaizduose esančių objektų aptikimo ir atpažinimo realiuoju laiku algoritmus ir užtikrinant didelį tikslumą, atsiranda galimybė įterpti – kokybės kontrolės, gamybos, produkcijos skaičiavimo – sistemas į pramonę ir patobulinti naujus staklių automatizavimo metodus.

Šio darbo tikslas – išnagrinėti smulkiems objektams atpažinti ir aptikti taikomus intelektualiuosius metodus ir pritaikyti pasirinktą metodą elektronikos komponentų ant konvejerio skaičiavimo sistemai.

Darbo uždaviniai

- Atlikti vaizdo apdorojimo bei klasifikavimo, giliojo mokymosi ir objektų aptikimo metodų analitinę apžvalgą.
- Sudaryti individualų duomenų rinkinį klasifikavimo modeliui mokyti ir suformuluoti elektronikos komponentų skaičiavimo metodą eksperimentiniams tyrimams atlikti.

- Ištirti pasirinkto metodo efektyvumą, pateikti pasiūlymus taikant metodą praktikoje ir palyginti gautus rezultatus.

Naudoti tyrimo ir analizės metodai

Šiame baigiamajame darbe tyrimo metu atlikta vaizdo apdorojimo, klasifikavimo ir objektų aptikimo metodų mokslinės literatūros analizė, kurios metu buvo gilinamos žinios apie dirbtinio intelekto metodus ir jų taikymą. Duomenų rinkiniui sudaryti naudota *Data Capture Control* aplikacija, duomenų kiekiui didinti pasitelkta *Matlab* aplinkoje rašyta programa. Pasirinkti algoritmai praktiškai pritaikyti ir eksperimentiškai ištirti. Tyrimams atlikti naudota *Google Colab* debesijos paslauga su *Tesla P100-16GB* GPU ir mikrokompiuteris *Nvidia Jetson Nano 4GB*. Eksperimentinių tyrimų programinis kodas rašytas *Python* kalba, naudotos *PyTorch*, *Keras*, *CV2* ir *Numpy* bibliotekos.

Darbo naujumas ir praktinė nauda

Rašto darbe atliekant mokslinės literatūros apie klasifikacija ir regresija grįstų neuroninių tinklų algoritmus analizę, pastebėta, kad algoritmai dažniausiai yra mokomi pasitelkiant įvairius duomenų rinkinius (pvz., *VOC2007, VOC2012, COCO*). Su tokiais duomenų rinkiniais objektų aptikimo algoritmai yra pritaikomi įprastų objektų identifikavimo ir apdorojimo uždaviniams spręsti. Dėl to buvo nuspręsta pritaikyti dirbtinį intelektą spausdintinių plokščių gamybos pramonei. Baigiamajame darbe pasiūlytas ir įgyvendintas elektronikos komponentų ant konvejerio skaičiavimo algoritmas, kuris buvo įdiegtas į objektų aptikimo programą. Elektronikos SMD komponentų identifikavimo problemai spręsti buvo sudarytas individualus duomenų rinkinys neuroniniams tinklams mokyti. Šios problemos sprendimas atveria plačias pritaikymo galimybes industrijoje – nuo skaičiavimo, rūšiavimo iki kokybės kontrolės.

Darbo struktūra

Darbą sudaro keturi skyriai, išvados, literatūros sąrašas ir priedai. Pirmajame skyriuje aprašyti klasikiniai ir šiuolaikiniai vaizdo parengiamojo apdorojimo, duomenų klasifikavimo, duomenų rinkinio sudarymo metodai. Antrajame skyriuje apžvelgti neuroninių tinklų mokymo algoritmai, klasifikacija ir regresija pagrįstų objektų aptikimo algoritmų architektūros, jų pritaikymas praktikoje bei atliktas tyrimams skirtos sistemos projektavimas ir analizė. Trečiajame skyriuje pateiktas mokymo duomenų rinkinio sudarymas ir paruošimas, jo panaudojimas tinklų modeliams mokyti. Turint apmokytus tinklus, pasiūlytas ir sudarytas objektų aptikimo ir skaičiavimo algoritmas. Ketvirtajame skyriuje pateikti ir paaiškinti atliktų eksperimentinių tyrimų rezultatai. Atlikti tinklų objektų aptikimo efektyvumo ir tinklų objektų skaičiavimo tyrimai, rezultatai palyginti tarpusavyje.

1. VAIZDO APDOROJIMO IR KLASIFIKAVIMO METODŲ ANALITINĖ APŽVALGA

Šiame darbe elektronikos komponentų skaičiavimo metodui pasirinkta pritaikyti ir susieti dirbtinio intelekto objektų aptikimą, apdorojimą ir analizavimą. Gilusis mokymasis šiuolaikiniam mašininiam mokymuisi dažniausiai yra naudojamas atliekant stebėjimo užduotis – semantinį segmentavimą, vaizdų antraščių priskyrimą, objektų aptikimą, atpažinimą ir vaizdų klasifikavimą [5]. Norint pasiekti vaizdų atpažinimą, reikia apdoroti vaizdus, kad būtų galima juos atskirti. Šiame skyriuje apžvelgiami DL metodai, naudojami vaizdams apdoroti ir klasifikuoti.

1.1. Klasikiniai vaizdo apdorojimo metodai

Vaizdų apdorojimo metodai egzistuoja jau daugelį metų, bet nauji proveržiai dirbtinio intelekto (AI) srityje leidžia iš esmės pakeisti vaizduose esančių duomenų aptikimą. Taip pat AI leidžia tobulinti aptikimą mokantis iš gautų duomenų [6].

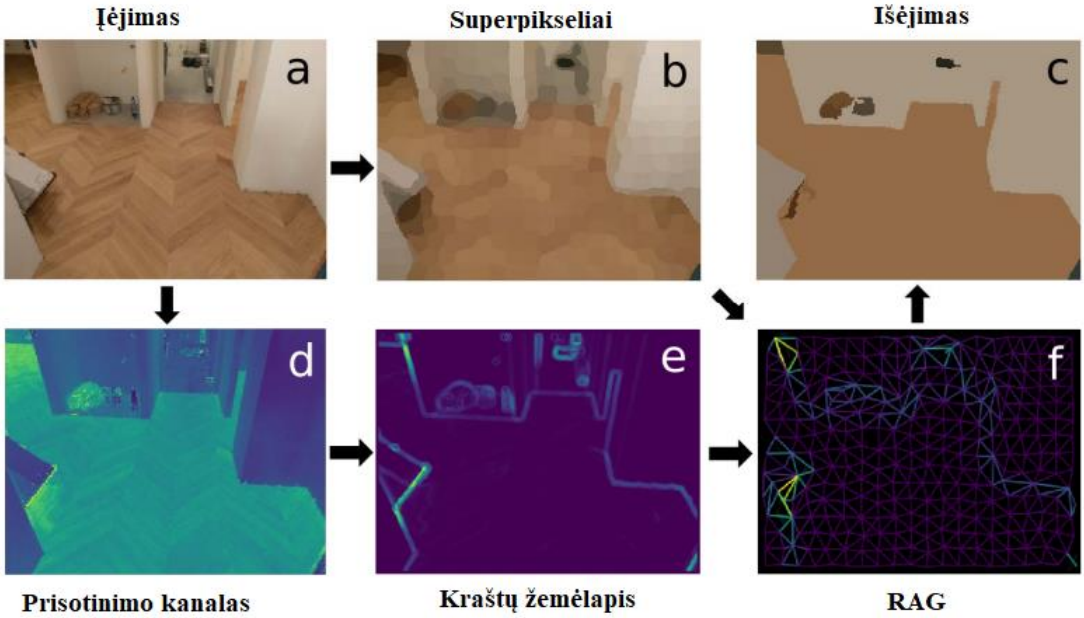
Superpikseliavimo metodas leidžia sugrupuoti vaizdo pikselius, turinčius bendrų savybių, į bendras zonas [7]. Tokiu būdu pasitelkiant superpikseliavimą suformuojamas kompaktiškas vaizdas.

Šis metodas turi skirtingas superpikseliavimo algoritmų variacijas. Vienas iš plačiausiai naudojamų metodų yra paprastasis tiesinis kartotinis grupavimas (angl. *simple linear iterative clustering*), kuris pritaiko *k* reikšmės grupavimą [7]. Naudojant metodą yra grupuojami pikseliai pagal jų spalvų panašumą ir artumą vaizdo plokštumoje. Atstumas tarp dviejų taškų sujungtose penkiose dimensijose yra aprašomas 1.1 formule [7].

$$\begin{aligned} d_{abc} &= \sqrt{(a_k - a_i)^2 + (b_k - b_i)^2 + (c_k - c_i)^2}; \\ d_{de} &= \sqrt{(d_k - d_i)^2 + (e_k - e)^2}; \\ D_s &= d_{abc} + \frac{m}{S} d_{de}. \end{aligned} \tag{1.1}$$

čia S – tinklelio intervalas; m – kompaktiškumo parametras.

Superpikselių grupavimo pavyzdys yra pateiktas 1.1 pav. (b). Paprastai yra sureguliuojami du svarbūs parametrai: superpikselių skaičius (nustatomas 300) ir kompaktiškumo parametras (nustatomas 7). Pirmasis atitinka maksimalų superpikselių kiekį, kurį reikia gauti iš vaizdo, o antrasis – dėl artumo ir spalvų panašumo. Iš 1.1 paveikslo matyti, kad kompaktiškumas kontroliuoja superpikselių ribų formą ir lygumą. Esant didesniam kompaktiškumui ribos tampa lygesnės, o superpikseliai – taisyklingesni [7].



1.1 pav. Pagrindiniai klasikinio vaizdo apdorojimo žingsniai: a) įėjime spalvotas paveikslas; b) SLIC superpikseliai; c) išėjime grupuojamas paveikslas iš sujungtų RAG; d) prisotinimo spalvų kanalas; e) paveikslas su paryškintais kraštais; f) iš superpikselių ir kraštų žemėlapio sukonstruotas RAG [7]

Superpikseliavimo problema yra ta, kad paprastai superpikseliai nesudaro idealios segmentų juostos [7]. Tad norint išvengti šios problemos, buvo pridėtas papildomas įėjimas vaizdams apdoroti. Pirmiausia spalvoto paveikslo pikseliai (1.1 pav. a) sugrupuojami į superpikselius (1.1 pav. b). Paraleliškai, yra transformuojamas RGB (angl. *Red, green, blue*) paveikslas į HSV (angl. *Hue saturation value*) spalvų erdvę ir apdorojamas tik su prisotinimo kanalu (1.1 pav. d), nes tai daugiausiai išryškina pasikeitimus tarp pateikto kambario paviršių [7]. Toliau iš S-kanalo paveikslo išgaunami žemėlapio kraštai (1.1. pav. e). Iš superpikselių vaizdo ir krašto žemėlapio derinio sukonstruojamos gretimumo grafiko zonos (RAG), kurios turi tikslesnius kraštų įverčius (1.1 pav. f). Galiausiai naudojant grafikų hierarchinį sujungimo algoritmą superpikseliai suskirstomi į grupes (1.1 pav. c).

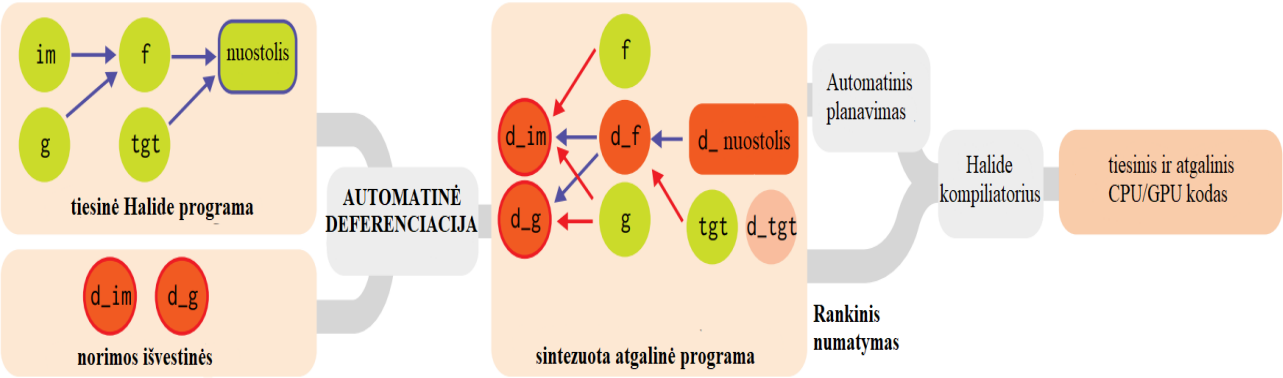
1.2. Šiuolaikiniai vaizdo apdorojimo metodai

Šiuolaikinių kompiuterių procesorių ir vaizdo plokščių galingumui sparčiai didėjant vaizdo apdorojimas tampa vis labiau prieinamas ir puikus įrankis atliekant su grafika susijusius procesus. Svarbu atsižvelgti į kompiuterio resursų naudojimą, dėl to reikia suprojektuoti optimalų algoritmą ir optimizuoti programos kodą.

Viena iš programavimo kalbų, skirtų vaizdų apdorojimui, yra *Halide* programavimo kalba [8]. Ši kalba yra suprojektuota taip, kad būtų lengva rašyti didelio našumo vaizdo ir masyvo apdorojimo

kodą. Pagrindė *Halide* idėja yra programos atskyrimas nuo algoritmo, kuriame nurodoma, kas yra skaičiuojama, ir skaičiavimo bei saugojimo eiliškumo tvarkaraštis [8]. Algoritmas išreikštas kaip grynas funkcinis aritmetinių operacijų daugiamačių tinklelių srautas. Tvarkaraštyje aprašomas taškų suskirstymas, vektorizavimas, lygiagretinimas, susiejimas su GPU (angl. *Graphical Processing Unit*). Ši kalba garantuoja, kad programos išvestis priklausytų tik nuo algoritmo, o ne nuo tvarkaraščio.

Remiantis [8] straipsniu, norint panaudoti šią sistemą, pirmiausia reikia programuoti tiesinį *Halide* algoritmą. Tada aprašoma išvestis iš aktyvavimo funkcijos skaliarinių nuostolių, susijusių su *Halide* kompiliavimo funkcija, vaizdo buferiu ar parametrų eilute. Automatinė diferenciacijos sistema susiejama su funkcijų grafiku, kuris apibūdina pirmyn perduodamą algoritmą ir sintezuoja naujas *Halide* funkcijas, įgyvendinančias gradiento skaičiavimą. Programoje galima pažymėti laiką šioms funkcijoms rankiniu būdu arba panaudoti automatinį nustatymą. Darbo eigos kompiliatorius pavaizduotas 1.2 paveiksle.



1.2 pav. *Halide* programa kurtas kompiliatorius [8]

Kitas vaizdo apdorojimo būdas yra *Google Tensorflow* platforma, leidžianti kurti modelius mašininiam mokymui. Ši platforma yra naudojama neuronų tinklams kurti ir yra plačiai pritaikyta vaizdų apdorojimui atlikti. Toliau remiantis [9] straipsniu, apibūdinimas *Tensorflow* veikimas. Pagrindinis *Tensorflow* vienetas yra skaičiavimo grafikas, kurį sudaro taškai, esančios operacijos ir briaunos, vaizduojančios tensorius. Tensoriai yra savavališkos matmenų matricos, t. y. aukšto laipsnio matricos arba vektoriaus apibendrinimas. Kiekvienas taškas gali turėti kelis įėjimus ir kelis išėjimus, o tensoriai gali būti perduoti iš vieno taško į kitą. *Tensorflow* naudoja dviejų tipų tensorius: vietos rezervavimo ženklus ir kintamuosius. Kraštai kontroliuoja skaičiavimo srautą. *Tensorflow* grafikų skaičiavimams naudoja sesijas. Skaičiavimo grafikas pradžioje yra tuščias ir neturi kintamųjų. Sesijos metu paskelbiamos vartotojo komandos, kad būtų nustatyti kintamieji ir sudarytas skaičiavimo grafikas.

Vaizdo apdorojime yra tokie etapai: sąsūkos operacija, kampų aptikimas, Gauso filtravimas, matricų dauginimas, vaizdo dydžio keitimas, vaizdo segmentavimas, vaizdo išblukinimas, vaizdo apsukimas, dalinė diferencialinė lygtis [9].

Vaizdų apdorojime sąsūkos naudojamos objektų kraštams aptikti. Tai galima padaryti naudojant branduolį arba kaukę, kuri paprastai yra kvadratinė nelyginio dydžio matrica [9]. Kraštų aptikimas dažnai yra pirmasis vaizdo apdorojimo etapas prieš atpažįstant vaizdą. Vaizdo gradientas yra vienas iš paprastesnių kraštų aptikimo metodų, jis atspindi krypties pokyčius paveikslėlyje ir gali būti apskaičiuotas kaip sąsūka su branduolio dydžiu. Kitas patikimas kraštų aptikimo metodas yra Kenio (angl. *Canny*) algoritmas, kuris yra daugiapakopis krašto detektorius, naudojamas struktūrinei informacijai gauti [9]. Gauso švelninimas yra naudojamas slopinti vaizdų iškraipymą, dėl kurio gaunamos sumažintos vaizdo detalės. Matricos yra labai svarbios pateikiant vaizdą. Paprastas vaizdo transformacijas galima traktuoti kaip matricos operacijas. Vaizdo interpoliacijos metu pasitelkiant žinomas kaimyno intensyvumo vertes, numatomos intensyvumo reikšmės trūkstamoje vietoje [9]. Norint atkurti vaizdą yra aktyvinamos sąsūkos matricos ir vykdomas išblukinimas [9].

1.1 lentelė. Algoritmų vykdymo laikai (sekundėmis) [9]

Algoritmas	**GPU**		**CPU**	
	1	**2**	**1**	**2**
Kenio filtras	0,81	**1,03**	0,38	5,65
Vaizdo išblukinimas	**0,77**	**8,2**	2,75	33,68
Gauso filtras	0,46	1,94	0,1	1,53
Gradiento filtras	0,47	180	0,1	1,33
K-mean segmentacija	0,52	**3,56**	0,29	53,35
Matricų daugyba	**0,46**	188	0,7	193
Dvikampio vaizdo dydžio keitimas	**0,6**	0,4	0,88	0,12
Abipusio atvaizdo dydžio keitimas	0,38	1,71	0,07	1,09
Vaizdo apsukimas	0,37	**0,42**	0,03	2,29
2-D vaizdo sąsūka	0,03	20,88	0,02	20,63
3-D vaizdo sąsūka	0,98	19,66	0,67	18,84
PDE - šilumos lygtis	0,77	**0,66**	0,28	4,5

1.2 lentelė. Tyrimams atlikti naudotų GPU ir CPU specifikacijos [9]

Procesorius	**CPU**	**GPU**
Branduolių skaičius	2	2 x 2496
Maksimalus taktinis dažnis	3 GHz	875 MHz
Bendras atminties dydis	46,08 MB	2 x 12 GB
Galios santykis	95 W	300 W

Vaizdo apdorojimo algoritmai buvo įvertinti skirtinguose lygiagrečiuose procesoriuose, naudojant *Tensorflow* platformą. Beveik visi išbandyti algoritmai gali gauti gerą greitį naudodami lygiagretinimą GPU. 1.1 lentelėje pateikti rezultatai tyrimų su 1.2 lentelėje nurodyta įranga. Matyti, kad pateiktuose algoritmuose GPU nuo 3,6 iki 15 kartų greitesnis nei CPU.

Norint išnaudoti optimizavimo galimybes su *Tensorflow* buvo pasitelkta aukšto lygio neuroninių tinklų biblioteka *Keras* [48], galinti veikti naudojant *Tensorflow* arba *Theano*. Šis API buvo sukurtas norint užtikrinti spartų eksperimentų atlikimą. Pagrindinės *Keras* duomenų struktūros yra sluoksniai ir modeliai. Paprasčiausias modelio tipas yra nuoseklus modelis, linijinis sluoksnių paketas. Norint sukurti sudėtingesnes architektūras, galima naudoti *Keras* funkcinę API, leidžiančią sudaryti sutartinius sluoksnių grafikus arba rašyti modelius visiškai nuo nulio naudojant poklasius. Toliau apžvelgiami [48] straipsnyje atlikto išvedimo greičio tyrimo pritaikius *Keras* API rezultatai. Buvo naudojami *TITAN XP* ir *TITAN X* GPU aptikti pėsčiuosius su YOLOv3, YOLOv3-tiny modeliais, įdiegtais su *Keras* ir *Darknet* API. Rezultatai parodė, kad gauti YOLOv3 detektorių rezultatai buvo per pusę mažesni su *Keras* nei su *Darknet* API. O YOLOv3-tiny detektoriaus su *Keras* gautas FPS rezultatas skyrėsi nuo gauto su *Darknet* daugiau nei 4 kartus.

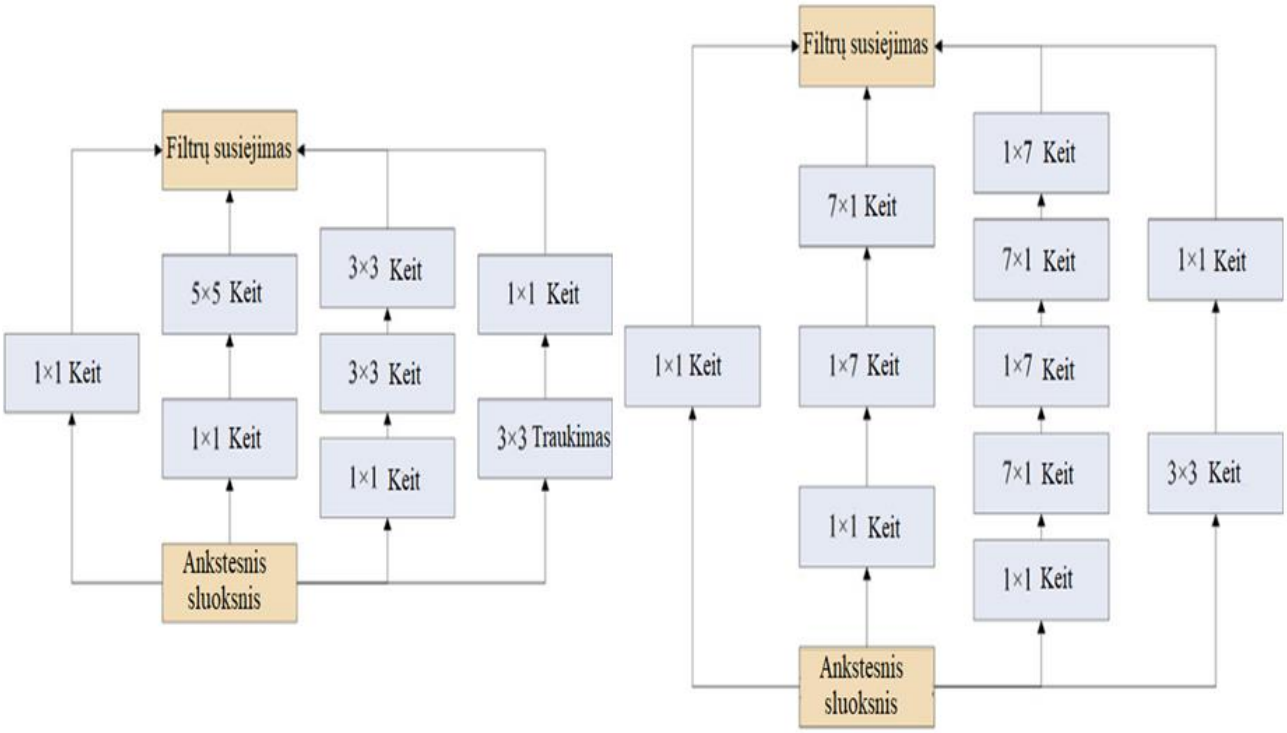
Kitas metodas norint optimizuotai analizuoti neuroninių tinklų parametrus yra *PyTorch* biblioteka [10]. Skirtingai nuo daugelio kitų populiarių giliojo mokymosi sistemų (pvz., *Tensorflow*), kurios naudoja statinius skaičiavimo grafikus, *PyTorch* naudoja dinaminį skaičiavimą, suteikiantį daugiau lankstumo kuriant sudėtingas architektūras. Kadangi *PyTorch* yra skirta *Python* aplikacijoms, tai reiškia, kad ši biblioteka naudoja *Python* koncepciją (pvz., klases, struktūras, sąlygų ciklus), kuri yra daugiau žinoma ir lengviau suprantama naudotojams. Tai daro šią API daug paprastesnę nei kitos, (pvz., *Tensorflow*), kurios turi savo programavimo stilių [10].

1.3. Šiuolaikiniai vaizdo klasifikavimo metodai

Mašininio mokymo modeliams optimizavimo pasirinkimas tebėra svarbi problema, kurią norint gauti gerus rezultatus reikia išspręsti. Įrodyta, kad funkcijų ir klasifikatorių pasirinkimas gali reikšmingai paveikti modelių našumą [10]. Naujausiuose tyrimuose padarytos išvados, kad nėra universaliai visiems duomenų rinkiniams tinkamo klasifikatoriaus [10].

Kompozicinėse programose, pavyzdžiui, OTB (angl. *Orfeo Toolbox*), yra įdiegti įvairūs algoritmai [11]. Tyrimas parodė, kad DL tinklo parametrai gali būti pritaikyti įvesties algoritmams, pavyzdžiui, klasifikavime [11]. OTB yra savo klasifikavimo programoje įgyvendinusi daug algoritmų, pavyzdžiui: SVM (angl. *Support vector machine*), atsitiktiniai miškai (angl. *Random forest*), postūmio klasifikatorius, sprendimų medžių klasifikatorius (angl. *decision tree*), padidinto gradiento medžių klasifikatorius ir įprastas Bajeso (angl. *Bayes*) klasifikatorius [11].

InceptionV3, sujungtas su mokymo algoritmais, gali būti naudojamas kaip pagrindas klasifikavimui. 1.3 paveiksle pavaizduota šio modelio struktūra, kurioje skirtingose linijose vienu metu galima nustatyti skirtingų mastelių ypatybes. Dėl šio metodo ne tik sumažinamas parametrų skaičius, bet ir naudojant kelių skalių ypatybes padidinamos tinklo atpažinimo galimybės. *1x1 Keit* bloke atliekant nedidelius skaičiavimus įvykdoma funkcijų transformacija, kuri gali pagerinti tinklo atpažinimo galimybes ir pakeisti išvesties aspektą [12]. Ši sąsūkos struktūra gali valdyti vis gausesnes erdvines ypatybes ir padidinti savybių įvairovę.



1.3 pav. *InceptionV3* modulio struktūra: a) naudoja 1 – 4-us ir 10 – 11-us *InceptionV3* modulius; b) naudoja 5 – 9-us *InceptionV3* modulius [12]

CNN (angl. *Convolution Neural Network*) gali būti naudojamas kaip vaizdo klasifikatorius [13]. CNN klasifikavimo tikslumas gali priklausyti nuo kelių dalykų – duomenų rinkinio, optimizavimo metodo ir neuronų tinklo struktūros. Šiuo klasifikatoriumi galima automatiškai gauti funkcijas. Daugeliu klasifikatorių, pagrįstų CNN, galima nustatyti pateiktų vaizdų minimalias daleles [13].

1.4. Duomenų išankstinio apdorojimo metodai

Duomenų gavyboje apdorojimo procesas yra naudingos informacijos išskyrimas iš nereikalingų, iškraipytų ir nenuoseklių pirminių duomenų. Duomenų gavybos metu išskiriama informacija iš didelio duomenų rinkinio ir suskirstoma į norimą formatą, grupes. Informacijos parametrų atrinkimas yra taip pat šio proceso dalis. Sprendimų medis yra tinkamas algoritmas duomenų apdorojimui, nes

įprastai yra naudojamas modeliui numatyti ir naudingai informacijai gauti didžiulių duomenų klasifikavimo metu [14].

WEKA yra vienas iš įrankių, turinčių mašininio mokymosi algoritmų rinkinį, skirtą duomenų gavybai [14]. Šį įrankį sukūrė *WEKA* komanda. *WEKA* taip pat įgyvendina duomenų apdorojimo, klasifikavimo, regresijos, grupavimo, susiejimo taisyklių algoritmus [14].

Norint pasirinkti tinkamą algoritmą duomenims gauti reikia atsižvelgti į jo optimizaciją. Pagrindiniai kriterijai, į kuriuos atsižvelgiama, yra numatoma galia, atminties naudojimas, atkūrimo laikas, sprendimo laikas, reikalavimas dėl klasių paskirstymo [15]. Numatomos galios skaičiavimas yra akivaizdus kriterijus, matuojamas visose mokymosi sistemose. Atminties suvartojimas yra būtinas kriterijus dėl įrangos apribojimų apdorojant potencialiai neribotą duomenų srautą. Atkūrimo laikas suteikia informaciją, per kiek laiko algoritmas turi prisitaikyti prie naujų duomenų ir atnaujinti struktūrą [15]. Sprendimo laikas yra dar viena laiko resursus tausojanti priemonė, leidžianti apskaičiuoti, kiek laiko tam tikri algoritmai turi numatyti kiekvienai naujai gautai informacijai apdoroti. Reikalavimas priskirti tikras klases gali padėti apdoroti didelio duomenų srauto gavimo algoritmą, pritaikytą realiuoju laiku [15].

1.5. Duomenų rinkinių sudarymas

Mažų objektų klasifikavimas vaizde yra susijęs su įvairiais iššūkiais, įskaitant menkus skirtumus, neatitikimus, dydžių variacijas, formų išsiskyrimus. Nors vaizdų apdorojimo ir mašininio mokymosi derinys buvo pažangus ir davė rezultatų, praktinis tokių priemonių naudojimas buvo ribotas [17]. Viena iš priežasčių – duomenų rinkinių trūkumas. Daugeliu mašininio mokymosi algoritmų nagrinėjami dideli duomenų kiekiai, kad pavyktų giliau ištirti algoritmų paslėptus sluoksnius. Šių algoritmų testavimas gali būti brangus ir reikalaujantis daug laiko [17].

Norint paspartinti atliekamų bandymų procesą, ypač paslankaus kūrimo proceso metu, kai bandymai dažnai atliekami su skirtingomis neuronų tinklų architektūromis, vienas iš būdų yra duomenų rinkinio žymėjimų (angl. *annotation*) konvertavimas ir automatizuotas duomenų kaupimas [18]. Automatizuoto duomenų kaupimo būdų būna paremtų tikimybių ir ne tikimybių metodais, mokymu ir paieška pagrįstais metodais, prižiūrimų, pusiau prižiūrimų ir neprižiūrimų [18]. Šiuos metodus galima suskirstyti į penkias kategorijas:

- generatyviniu (angl. *generative*) modeliu pagrįsti metodai [18], skirti maksimaliai padidinti vaizdo ypatybių ir etikečių generavimo tikimybę;
- artimiausio kaimyno (angl. *nearest neighbor*) modeliu pagrįsti metodai [18], paremti prielaida, kad panašių savybių vaizdai turi didelę tikimybę dalintis panašiomis etiketėmis;
- diskriminaciniu (angl. *discriminative*) modeliu pagrįsti metodai [18], kuriuos pasitelkiant žymėjimo užduotis vertinama kaip kelių etikečių klasifikavimo problema;

- paskutinės žymėtos etiketės (angl. *tag completion*) principu pagrįsti metodai [18], kurie dėl automatinio trūkstamų eilučių užpildymo leidžia nuspėti etiketes ir pataisyti netinkamai pateiktas vaizdų žymes;
- giliojo mokymosi metodai [18], kuriuose naudojami giliojo mokymosi algoritmai, kad būtų gauti patikimi vaizdo parametrai, ypač didelio duomenų kiekio automatizuotam duomenų kaupimui.

Generatyvinio, diskriminacinio ir giliojo mokymosi metodai pagrįsti mokymosi procesu. Pasitelkiant generatyviniu modeliu pagrįstus metodus vaizdai žymimi pagal sąlyginę tikimybę po gauto vaizdo ir etiketės palyginimo. Pasitelkiant diskriminaciniu modeliu pagrįstus metodus kelių žymų etikečių vaizduose traktuojamos kaip klasifikavimo problema. Taigi koreliacija tarp klasių yra labai svarbi, bet negali būti tiesiogiai išspręsta dvejetainiu klasifikatoriumi. Giliojo mokymosi metodai paprastai pagrįsti CNN, kad būtų gautos patikimos vaizdinės funkcijos, arba naudoja skirtingas tinklo sistemas, pavyzdžiui, RNN, kad būtų išnaudota šalutinė informacija, t. y. etiketės koreliacija, skirta automatizuotam žymėjimui. Palyginimui, pasitelkiant artimiausio kaimyno modeliu pagrįstus automatizavimo metodus vaizdams komentuoti naudojama dviejų pakopų sistema. Paskutine žymės etikete pagrįstiems metodams nereikalingas mokymo procesas ir esant netikslumų etiketės gali būti ištaisytos automatiškai [18].

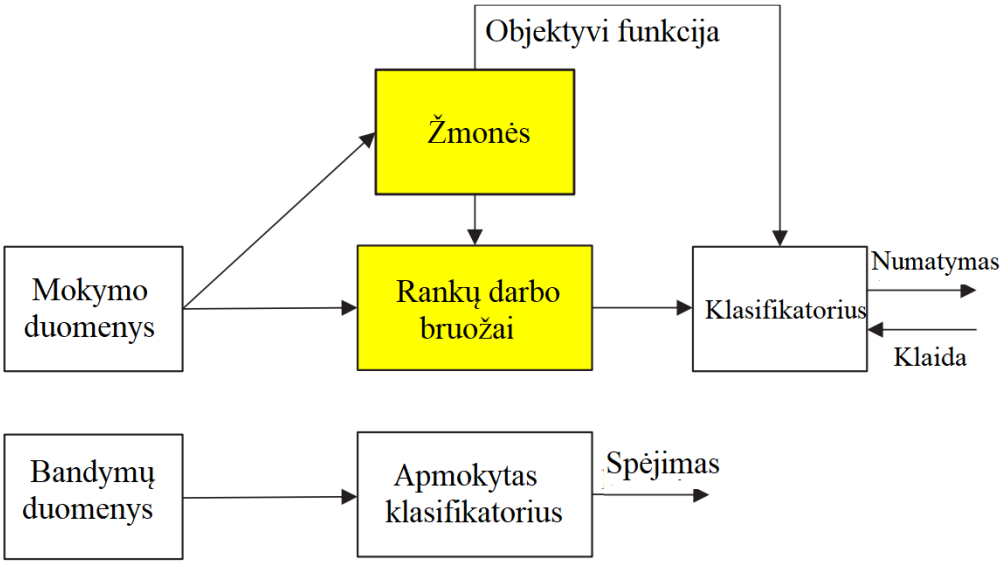
1.6. Skyriaus apibendrinimas

Šiame skyriuje buvo apžvelgti vaizdams apdoroti ir klasifikuoti taikomi metodai. Pirmiausia analizuotas klasikinis vaizdo apdorojimo metodas – superpikseliavimas, buvo apžvelgtas praktinis panaudojimas apdorojant gauto vaizdo parametrus. Toliau išnagrinėti šiuolaikiniai vaizdo apdorojimo metodai – *Halide* programavimo kalba, *Tensorflow* API ir su ja gauti rezultatai atliekant įvairias vaizdo apdorojimo funkcijas naudojant GPU ir CPU, *Keras* ir *PyTorch* giliojo mokymosi bibliotekas. Atlikta *Tensorflow*, *Keras* ir *PyTorch* metodų lyginamoji analizė. Kitame etape analizuoti šiuolaikiniai vaizdo klasifikavimo metodai, kurie gali būti taikomi kaip duomenų kaupimo įrankiai – OTB įrankis, kuriame yra daug algoritmų, skirtų klasifikacijai (pvz., SVM, atsitiktiniai miškai, postūmio klasifikatorius, sprendimų medžių klasifikatorius), *InceptionV3* ir CNN algoritmai. Taip pat buvo apžvelgtas duomenims rūšiuoti skirtas išankstinio duomenų apdorojimo įrankis *WEKA*. Galiausiai buvo išnagrinėti automatizuotam duomenų rinkinių kūrimui naudojami penki metodai – generatyviniu, artimiausio kaimyno, diskriminaciniu, paskutinės žymės etiketės ir giliuoju mokymusi pagrįstos kategorijos.

2. OBJEKTAMS APTIKTI IR ATPAŽINTI TAIKOMŲ METODŲ ANALITINĖ APŽVALGA

Objekto aptikimas vaizduose dar vadinamas objekto kategorijos aptikimu vaizduose arba objekto klasės aptikimu [23]. Pateikiant vaizdą, nustatoma, ar yra objektų iš anksto nustatytų kategorijų atvejų ir, jeigu yra, grąžinama kiekvieno egzemplioriaus erdvinė vieta ir mastas [23]. Taikant gilųjį mokymąsi AI išmoksta savybių ir automatiškai priima sprendimus (klasifikuoja), todėl veikia paprasčiau nei rankiniu būdu žmogaus suprojektuota sistema [24]. Šiame skyriuje apžvelgiami DL algoritmai, naudojami aptinkant objektą vaizde, atliekama tų algoritmų lyginamoji analizė.

Tradicinis statistinis mašininis mokymasis veikia su duomenų lentele ir numatymo tikslu. Eilučių lentelė atitinka nepriklausomus stebėjimus, stulpeliai atitinka rankiniu būdu sukurtus pagrindinių duomenų rinkinius [16]. Tada galima pritaikyti įvairius mašininio mokymosi algoritmus, kad būtų galima mokyti modelį, kuriame kiekviena duomenų eilutė susiejama su prognoze. Dar svarbiau tai, kad apmokytas modelis taip pat leistų prognozuoti bandymų duomenis, kurie gaunami iš panašaus paskirstymo kaip ir mokymo duomenys [16]. 2.1 paveiksle pavaizduotas šis procesas.



2.1 pav. Mašininis mokymo procesas [16]

Šis metodas gerai tinka daugeliui užduočių atlikti, pvz., rekomendacijų sistemoms, objektų atpažinime, kur žmogus–srities ekspertas gali lengvai sukonstruoti gerą funkcijų rinkinį.

2.1. Mokymo algoritmų apžvalga

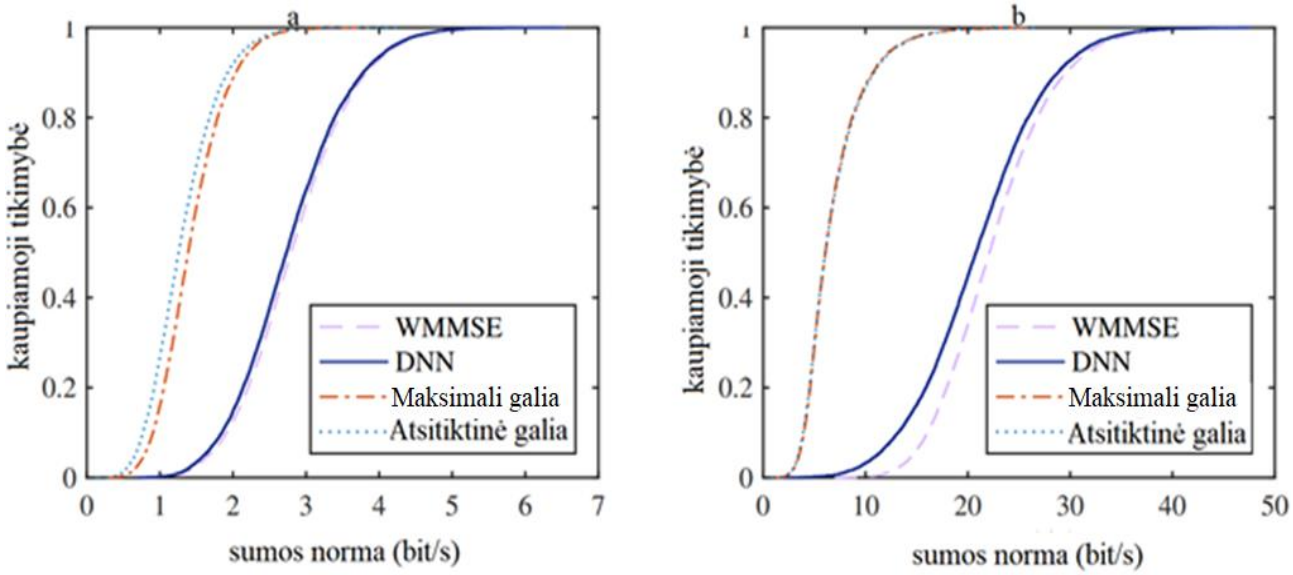
Pagrindinis pranašumas naudojant DNN (angl. *Deep neural network*) iteracinio algoritmo atžvilgiu yra tas, kad DNN skaičiavimo efektyvumas realiuoju laiku yra aukštas, palyginti su tipiniu

iteracinio optimizavimo algoritmu. Šio algoritmo vykdymo trukmės etapas nereiškia skaitmeninio optimizavimo, o tik kai kurias paprastas operacijas, pavyzdžiui, vektoriaus dauginimą, pridėjimą ir paprastas (skaliarines) netiesines transformacijas [20]. Rezultatas toks, kad, jeigu mokymo etape gali būti apytiksliai įvertintas algoritmas, tada gali būti paprasčiausiai paskirstyti ištekliai realiuoju laiku.

Atliktame [20] tyrime panaudotas tinklas su trimis paslėptais sluoksniais, vienu įvesties ir vienu išvesties sluoksniu. Pirmajame paslėptame sluoksnyje yra 200 neuronų, o antrajame ir trečiajame paslėptuose sluoksniuose yra 80 neuronų. Įėjimas į tinklą yra kanalų koeficientų rinkinys, todėl jo dydis priklauso nuo kanalo modelių. Tiksliau, Gauso IC įvesties dydis yra K^2, o IMAC – N×K. Išvestis yra energijos paskirstymo rinkinys, todėl jos dydis yra K tiek Gauso IC, tiek IMAC (angl. *Interfering multiple-access channel*). Norint rasti mokymo algoritmo parametrus, buvo atlikti skirtingų kanalų modelių kryžminiai patvirtinimai [20].

Gauso-IC modelis skaičiuoja paketo dydžio ir mokymosi žingsnio įtaką vidutinei kvadratinei paklaidai, įvertintai pagal patvirtinimo rinkinį, taip pat bendrą mokymo laiką [20].

Praktinis IMAC modelis yra parametrų parinkimo procesas, kuris beveik toks pat kaip ir Gauso-IC, išskyrus tai, kad galima nuspręsti palaipsniui mažinti mokymosi žingsnį, kai tikrinimo klaida nesumažėja [20].



2.2 pav. Kaupiamoji paskirstymo funkcija, apibūdinanti algoritmų pasiektus greičius: a) Gauso-IC; b) IMAC [20]

Atliktas [20] sumos normos įvertinimas naudojant Gauso-IC ir IMAC algoritmus ir lyginant su skirtingais veiksmais pavaizduotas 2.2 paveiksle. Iš gautų rezultatų galima pastebėti, kad DNN efektyvumas yra labai artimas WMMSE, o kitas dvi bazines linijas reikšmingai lenkia. Atkreiptinas dėmesys, kad taikant Gauso-IC yra optimizuotos paskirstymo funkcijos, kurios, gautos WMMSE,

daugeliu atvejų yra dvejetainės [20]. Todėl taip pat diskretizuotas dvejetainių kintamųjų numatymas, siekiant padidinti tikslumą.

Kitas metodas DNN algoritmui mokyti yra su duomenų rinkiniu. Vienas iš jų yra apšvietimo variantų modeliavimas naudojant 2D Gauso funkcijas [21]. Apšvietimo sąlygos yra dažna problema nagrinėjant realaus pasaulio vaizdus. Norint modeliuoti apšvietimo variacijas, galima paprasčiausiai pakeisti pikselių intensyvumą, atsižvelgiant į paprastojo santykio koeficiento variacijas. Vietiniai apšvietimo pokyčiai yra sudėtingesni, o norint gauti tikroviškus sintetinius vaizdus, reikalingi daug laiko atimantys vaizdavimo algoritmai. Tam palengvinti [21] straipsnyje pasirinkta dirbti tik su 3D erdvėje esančiomis plokštumomis, leidžiant modeliuoti šviesą kaip vietinius 2D šviesos šaltinius ir gauti realius rezultatus. Kiekvienam pasirinktam vaizdo keitimui pikseliuose (x, y) taikomas 2D Gauso sumaišymas pagal 2.1 formulę [21]:

$$I_l(x,y) = \sum_{l=1}^{N} I(x,y)\, fl(x,y) \tag{2.1}$$

Kiekvienas 2D Gauso sumaišymas savo ruožu gali būti modeliuotas pagal pateiktą 2.2 formulę [21]:

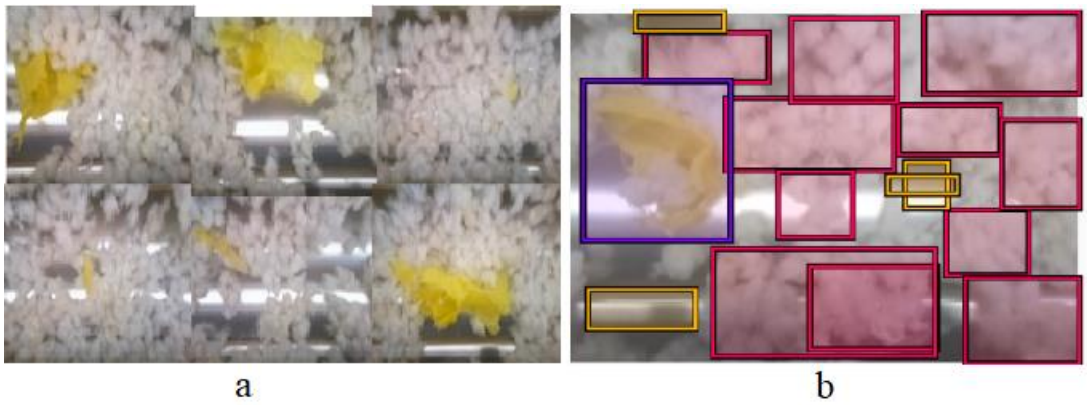
$$f_l(x,y) = Ae^{-(\frac{x-x_0}{2\sigma_x^2}+\frac{(y-y_0)}{2\sigma_y^2})} \tag{2.2}$$

čia (x_0,y_0) – imituota šaltinio šviesos centro projekcijos kryptis; A – šviesos intensyvumas; (σ_x, σ_y) – šviesos plitimas išilgai kiekvienos ašies.

Gautas vaizdo pavyzdys naudojant 2.1 ir 2.2 formules, pateiktas 2.3 paveiksle. Remiantis [21] modeliavimas buvo atliktas spėjimo būdu, nes nebuvo žinomos duomenų ir atspindžio savybės. Iš pateikto paveikslo matyti, jog panaudojus siūlomas formules, galima išryškinti arba užtamsinti vaizde esančius objektus. Tyrėjams, nors apšvietimo pokytį pritaikė spėjimo būdu, pavyko išryškinti vieną ir kitą pavaizduotą žmogų atskirai. Tokio tipo šviesos krypties imitacija būtų tinkama pritaikyti objektų klasifikavime, kai norima išryškinti identifikuotą objektą turint nespalvotas vaizdo stebėjimo kameras.

2.3 pav. Atsitiktinai keičiamo apšviestumo rezultatai [21]

Vienas iš praktinių apmokyto CNN tinklo su paruoštu duomenų rinkiniu taikymo pavyzdžių yra plastiko užterštumo šalinimas iš medvilnės pūkų [22]. Tyrėjai panaudojo duomenų rinkinį, kurį sudarė 401 plastikinės taršos, 4751 sėklinės medvilnės, 2847 aliuminio dėklo atvejai [21]. Atlikto [21] tyrimo metu virš konvejerio buvo pritvirtinta kamera, kuri fiksavo vamzdžio užterštumą. Gauti rezultatai atpažįstant konvejerio juostoje esančius užteršimus pateikti 2.4 paveiksle. Matyti, jog CNN algoritmui pavyko identifikuoti kur yra taršos objektai, algoritmas gebėjo išskirti rezultatus į tris dalis: mėlynos linijos, kai yra didelė taršos objekto aptikimo tikimybė, geltonos, kai vidutinė, ir raudonos, kai nėra aptiktas taršos objektas.



2.4 pav. Užfiksuoti medvilnės užteršimo vaizdai: a) sistemos užfiksuotas vaizdas; b) CNN tinklo atpažinti taršos objektai [22]

Remiantis nagrinėtų neuroninių tinklų algoritmų mokymais, pastebėta, jog norint pasiekti aukštą aptikimo tikslumą turintį tinklo modelį yra svarbu atsižvelgti į mokymo duomenų paruošimą, kiekį ir kokybę. Turint surinktą tvarkingą duomenų rinkinį sumažinama tikimybė, kad atliekant tinklo mokymą gali įsivelti paklaida, dėl kurios modeliui nepaisant mokymo iteracijų skaičiaus gali nepavykti pasiekti aukšto preciziškumo. Taip pat galima pritaikyti tinkamus duomenų rinkinio didinimo metodus, nesumažinant kokybės ir svarbių detalių.

2.2. Klasifikacija pagrįstų objektų aptikimo algoritmai

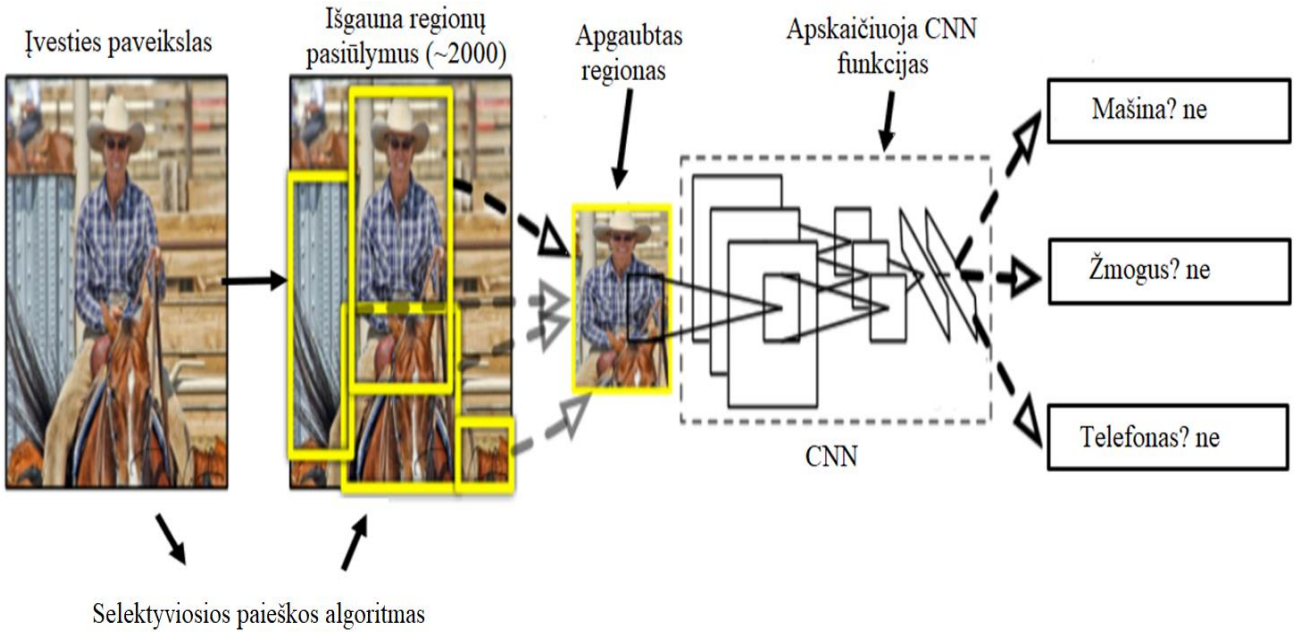
Grafinių objektų atpažinimo problema siejama su simbolių eilutės pavidalo informacijos apdorojimu taip, kad klasifikatorius galėtų ją tinkamai atpažinti ir suprasti [25]. Modeliui atpažinti įvesties vaizdas turėtų būti standartizuoto formato. Tai reiškia, kad prieš klasifikuojant klasifikatoriaus įvesties vaizdus reikia transformuoti į vieną konkretų formatą ir tuos pačius matmenis [25].

Objektų detektorius sukuria sritis, kuriose yra modeliai, priklausantys nuo įvesties failo, todėl juos pradedama normalizuoti taip, kad būtų sumažintos detektoriaus nurodytos sritys ir paverstos taškų bitų žemėlapiu [25]. Standartizuoti vaizdai tampa neuronų tinklo klasifikavimo įvestimi. Šiame poskyryje apžvelgiamos klasifikacija grįstų objektų aptikimo modelių architektūros ir jų praktinio pritaikymo efektyvumo rezultatai.

2.2.1. Regioninis sąsūkos neuronų tinklas

Regioninis sąsūkos neuronų tinklas (RCNN) buvo sudarytas siekiant pašalinti kelių regionų pasirinkimo vienu metu problemą [26]. Šis metodas naudoja selektyviąją paiešką. 2.5 paveiksle pavaizduota RCNN tinklo architektūra, naudojanti selektyvųjį paieškos metodą, kai yra išgaunamas objekto srities pasiūlymų rinkinys [26]. Kiekvienas objekto regiono pasiūlymas transformuojamas į fiksuoto dydžio vaizdą pakeičiant jo mastelį ir tada pritaikomas sąsūkos neuronų tinklo modeliui, kuris yra iš anksto apmokytas naudojant duomenų rinkinius. SVM klasifikatorius numato objekto buvimą kiekvieno regiono pasiūlyme ir atpažįsta objektų klases.

Tyrėjai [26] su RCNN architektūra atliko objektų klasifikavimo bandymą pritaikę *VOC-2007* duomenų rinkinį. Remiantis gautais rezultatais matyti, kad su *VOC-2007* duomenų rinkiniu apmokyto RCNN modelio vidutinio aptikimo tikslumo vidurkio (angl. *mean Average Precision*) įvertinimas pasiekė 58,5 %. *VOC* duomenų rinkinys, kurį sudaro 20 objektų kategorijų, yra plačiausiai naudojamas etaloninis duomenų rinkinys bendram objektų aptikimui [27]. Duomenų rinkinyje objektų paveikslai paprastai yra dideli, jie užima didžiąją vaizdo dalį.



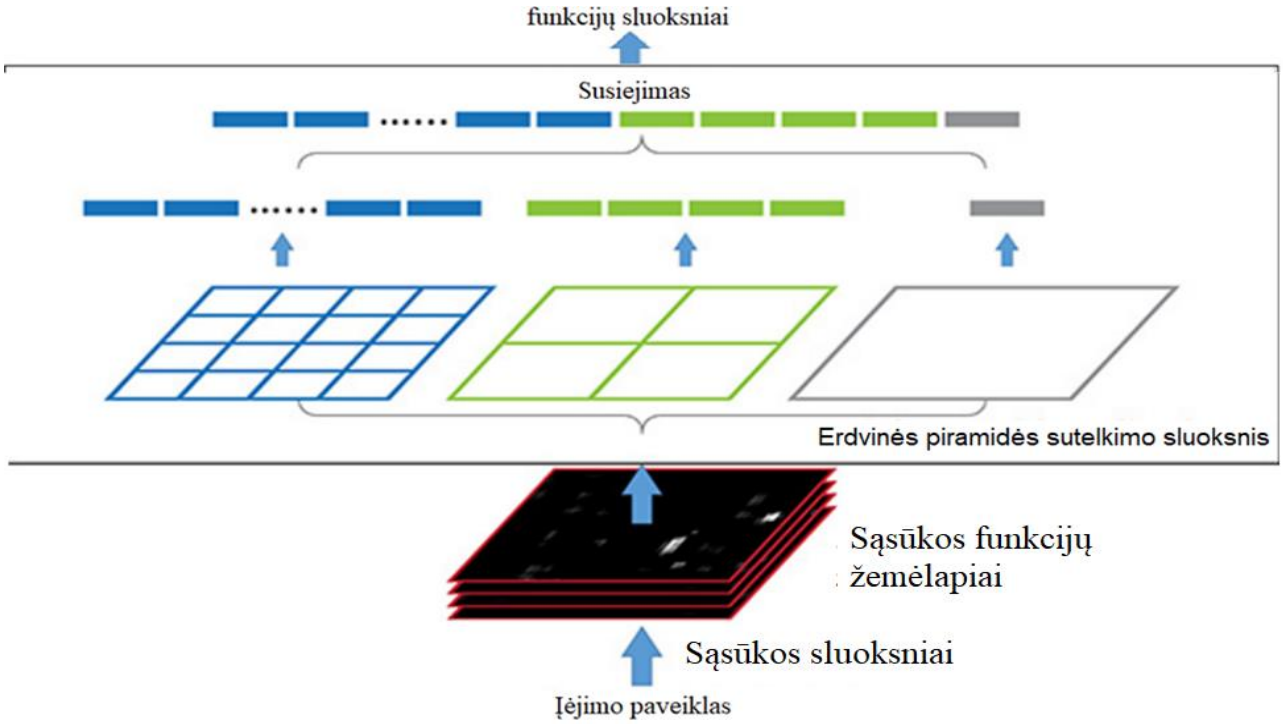
2.5 pav. RCNN tinklo architektūra [26]

Apžvelgus RCNN modelį, pastebėta, kad turi daug trūkumų, jie atskleisti [26] atliktame tyrime. Tinklo mokymas užima daugiau laiko, nes viename vaizde reikia klasifikuoti 2000 objektų regionų pasiūlymų [26]. Architektūros negalima įgyvendinti realiuoju laiku, nes kiekvienam [26] vaizdui apdoroti bandymo metu reikėjo apie 47 sekundžių, kadangi atrankinis paieškos metodas yra fiksuotas algoritmas, tokiu greičiu nesimokoma ir tai lemia blogų objektų regionų pasiūlymų atsiradimą [26]. Norint šias problemas išspręsti buvo sukurtas erdvinis piramidžių telkimo tinklas.

2.2.2. Erdvinis piramidžių telkimo tinklas

Erdvinio piramidžių telkimo tinklo (angl. *Spatial Pyramid Pooling Network*) (SPPNet) architektūra pavaizduota 2.6 paveiksle. Ankstesniems CNN modeliams reikėdavo nustatyto dydžio paveikslų įėjime, pvz., *AlexNet* architektūrai būtina, kad įėjime paveikslai būtų *224 x 224* pikselių dydžio [26]. SPPNet turi vieną SPP sluoksnį, kuris leidžia CNN modeliui sukurti fiksuoto ilgio seką neatsižvelgiant į dominančio regiono dydį.

Atliekant objektų aptikimą su SPPNet, funkcijų žemėlapių skaičiavimas yra atliekamas tik vieną kartą iš viso įvesties vaizdo, o pasirinktiems regionams generuojamos fiksuoto ilgio sekos naudojant apmokytus detektorius [26]. Šie detektoriai dažnai suteikia galimybę išvengti sąsūkos funkcijų skaičiavimo.



2.6 pav. Erdvinio piramidžių telkimo tinklo architektūra [26]

Dažniausiai objektų aptikimo efektyvumo įvertinimui apskaičiuoti yra naudojamas vidutinio tikslumo vidurkis (mAP) [28]. Geresnis rezultatas būna nurodytas kaip didesnis žemėlapio įvertis, atsižvelgiant į pagrindinės tiesės langelius ir nurodytas objektų aptikimo užduoties klases. Vidutinio aptikimo tikslumo vidurkį galima apskaičiuoti pagal pateiktą 2.3 formulę [28]. Vidutiniam tikslumui apskaičiuoti yra įvertinimas plotas pagal tikslumą, kurį sudaro atkūrimo kreivė pagal skaitmeninę integraciją, o mAP pasiekiamas apskaičiuojant visų klasių vidutinio tikslumo vidurkį [28].

$$mAP = \frac{TP}{TP + FP} \tag{2.3}$$

čia TP – tikras teigiamas; FP – netikras neigiamas.

SPPNet remiantis [26] atliktu tyrimu veikia daug greičiau nei RCNN, taip pat naudojant *VOC-2007* duomenų rinkinį aptikimo tikslumas nesumažėjo, o padidėjo iki 59,2 % mAP. Nors pritaikyto RCNN rezultatai pakankamai tikslūs, tačiau jis turi trūkumų. Tinklo mokymas vis dar yra daugiapakopis, tikslumas pasiektas tik tiksliai sukonfigūravus sąsūkos sluoksnius, kurie nepaiso ankstesnių sluoksnių [26].

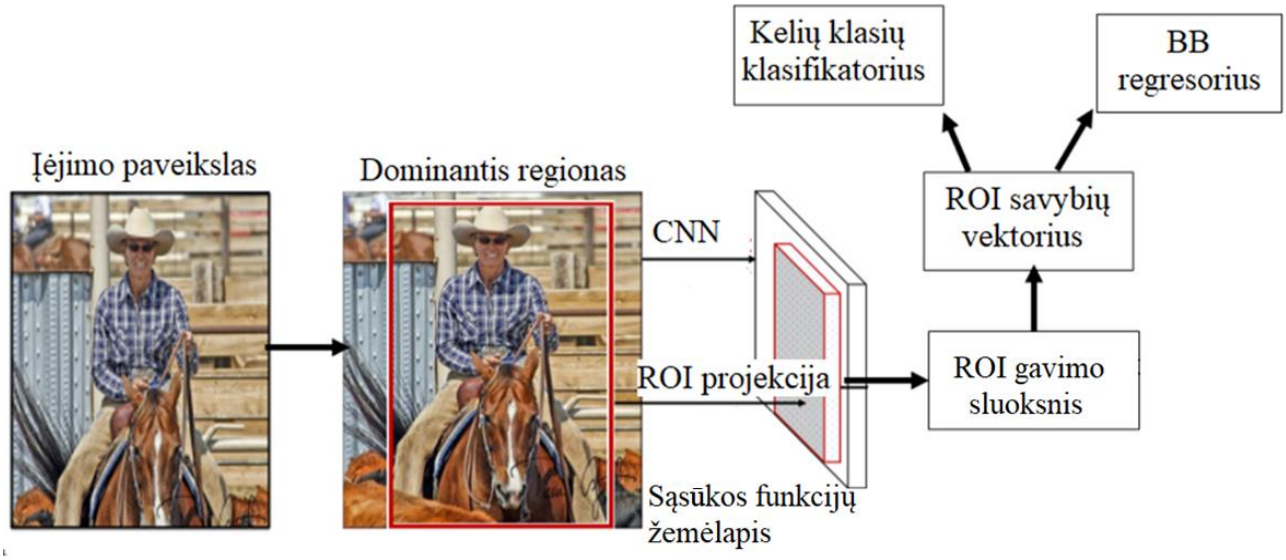
$$M_x = \frac{I_x - K_x}{S_x}, M_y = \frac{I_y - K_y}{S_y} \tag{2.4}$$

čia (M_x, M_y) – žemėlapių dydžiai; (I_x, I_y) – įėjimo dydžiai; (K_x, K_y) – branduolio dydžiai; S_x,– stulpelio žingsnis; S_y – eilutės žingsnis.

Sąsūkos sluoksnis, kuris aprašytas pagal 2.4 formulę [30], leidžia iš įvesties glaustai išskirti jos dominančias ypatybes ir sukurti funkcijų žemėlapį, kuris pritaikomas skirtingose detektorių funkcijose [30]. Norint panaikinti minėtus SPPNet trūkumus, buvo suprojektuotas greitasis RCNN tinklas.

2.2.3. Greitasis regioninis sąsūkos neuronų tinklas

Greitasis regioninis sąsūkos neuronų tinklo (Greitasis RCNN) detektorius buvo įdiegtas Roso Girshicko ir yra SPPNet su RCNN kombinacijos patobulinimas [26]. Greitojo RCNN architektūra pateikta 2.7 paveiksle. Tinklas sudarytas iš dviejų dalių: sąsūkos tinklas funkcijoms gauti ir ROI tinklas su ROI ištraukimo sluoksniu, kuris kartu su keliais visiškai sujungtais sluoksniais leidžia atlikti objektų klasių ir aprėpties dėžių (angl. *Bounding Boxes* (BB)) identifikavimą [29]. Atsižvelgiant į įvesties vaizdą, sąsūkos greitasis RCNN tinklas panaudoja visą vaizdą kaip įvestį ir sukuria sąsūkos žemėlapius išvestyje [29]. Šis tinklas leidžia vienu metu mokyti tiek detektorių, tiek BB regresorių.

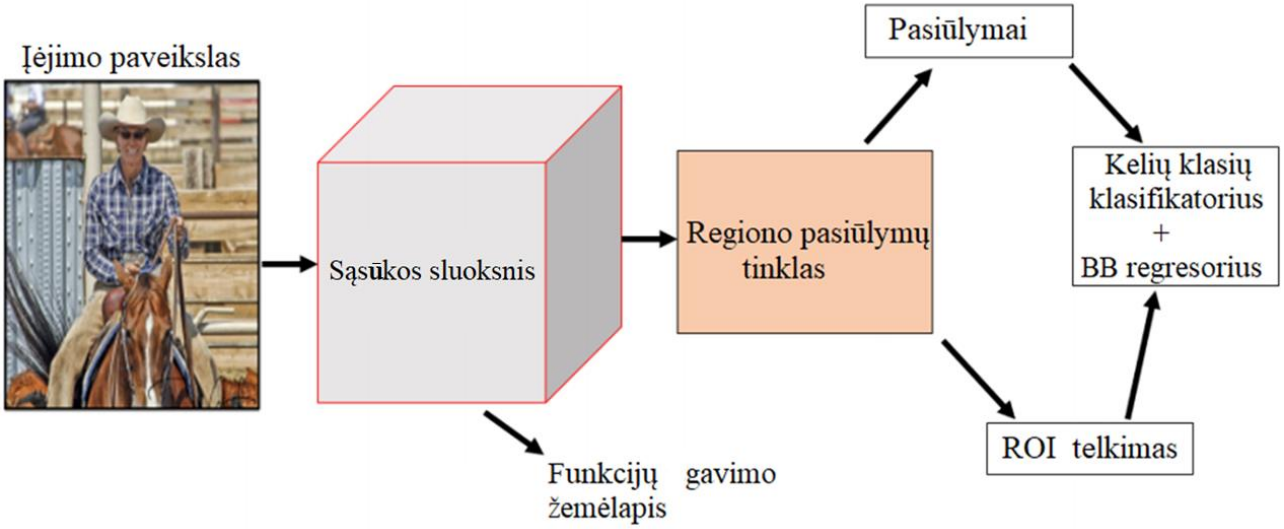


2.7 pav. Greitojo RCNN tinklo architektūra [26]

Naudojant *VOC-2007* duomenų rinkinį atliktame [26] tyrime, gautas mAP padidėjo nuo 58,5 % (RCNN) iki 70 % (Greitasis RCNN) [26]. Visi RCNN ir SPPNet pranašumai sėkmingai integruoti į Greitąjį RCNN tinklą, tačiau aptikimo greitis vis tiek liko ribotas [26]. Šiuos trūkumus pašalina Greitesnysis RCNN.

2.2.4. Greitesnysis regioninis sąsūkos tinklas

Greitesnysis RCNN detektorius buvo įdiegtas netrukus po RCNN suprojektavimo [26]. Siekiant pašalinti Greitojo RCNN tinklo trūkumus, buvo įdiegtas Greitesnysis RCNN, vadinamas regionų pasiūlymų tinklu (RPN), kurio architektūra pavaizduota 2.8 paveiksle. Greitesnysis RCNN atlieka tiek regiono pasiūlymų generavimą, tiek aptikimo užduotis. Išskyrus RPN, Greitesniojo RCNN ir Greitojo RCNN architektūros yra labai panašios [26]. Iš pradžių atliekamas ROI sujungimas, o tada sujungtas plotas yra perduodamas į CNN ir du funkcijų sluoksnius, kad būtų galima klasifikuoti ir panaudoti aprėpties dėžės (BB) regresorių.



2.8 pav. Greitesniojo RCNN tinklo architektūra [26]

Norint aptikti mažus objektus naudojant Greitesnįjį RCNN tinklą, reikia jį modifikuoti, panaudojant *ResNet-50* kaip pagrindą. Atlikto [27] tyrimo, naudojant įvairias strategijas treniruoti modifikuotą Greitesnįjį RCNN su *SORSI* duomenų rinkiniu, rezultatai pateikti 2.1 lentelėje.

2.1 lentelė. Rezultatai su modifikuotu greitesniuoju RCNN ir *SORSI* duomenų rinkiniu [27]

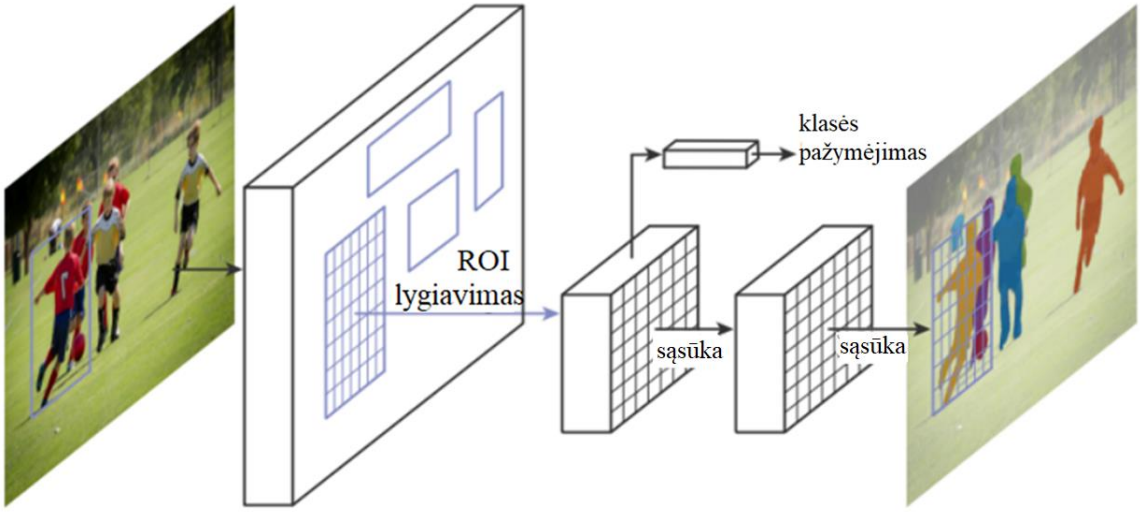
Metodas	**mAP**, %	**AP**, %	
		Laivas	**Lėktuvas**
Modifikuotas Greitesnysis R-CNN (žingsnis = 8)	76,7	69,8	83,6
	76,1	71,4	80,8
	77,1	70,4	83,9
	78,3	72,3	84,3
	78,9	72,9	85

Iš lentelėje pateiktų rezultatų matyti, kad aukščiausias mAP buvo pasiektas 78,9%, kai laivo AP siekė 72,9 % ir lėktuvo AP 85 %. Aptinkant laivus, žemiausias gautas AP įvertis buvo 69,8 % ir 76,7

%, o aptinkant lėktuvus 80,8 % AP. Pagrindinė vaizdų aptikimo problema su modifikuotu greitesniuoju RCNN tinklu buvo objektų neatpažinimas dėl geometrinių figūrų. Problema ypač išryškėjo atpažįstant laivus, kurių geometrija neturėjo specialių išskirtinumų, leidžiančių lengviau juos atpažinti ir pasiekti aukštesnį AP.

2.2.5. Maskuotasis regioninis sąsūkos tinklas

Maskuotasis regioninis sąsūkos tinklas (M-RCNN) yra greitesniojo RCNN tinklo architektūros plėtotė [26]. Pateiktame 2.9 paveiksle parodyta, kad M-RCNN architektūra yra dviejų pakopų linija [26]. Pagrindinis M-RCNN tikslas yra spręsti egzempliorių segmentavimo problemas CV programose, atskirti skirtingus objektus vaizde ar vaizdo įraše. Taip pat kiekviename regione į M-RCNN yra įtraukta kaukės atšaka (angl. *Mask Branch*), norint numatyti, kad objektas veiktų lygiagrečiai su klasės etikečių ir aprėpties dėžių (BB) regresijos identifikavimo funkcijomis [26]. Tai sukuria tris išvestis: klasės etiketę, BB ir objekto kaukę. M-RCNN efektyviai nustato objektus įvesties vaizde arba vaizdo įraše ir tuo pačiu metu sukuria aukštos kokybės segmentavimo kaukę kiekvienam aptiktam objektui [26].



2.9 pav. Maskuotojo regioninio sąsūkos tinklo architektūra [26]

Norint pasiekti didesnį greitį ir tikslumą su M-RCNN tinklo modeliu reikėtų naudoti *ResNet-FPN* kaip pagrindą modelio funkcijoms gauti. Tačiau pagrindinis šio modelio trūkumas – jis prideda tinkle mažas skaičiavimo paklaidas ir veikia beveik 5 kadrų per sekundę greičiu [26].

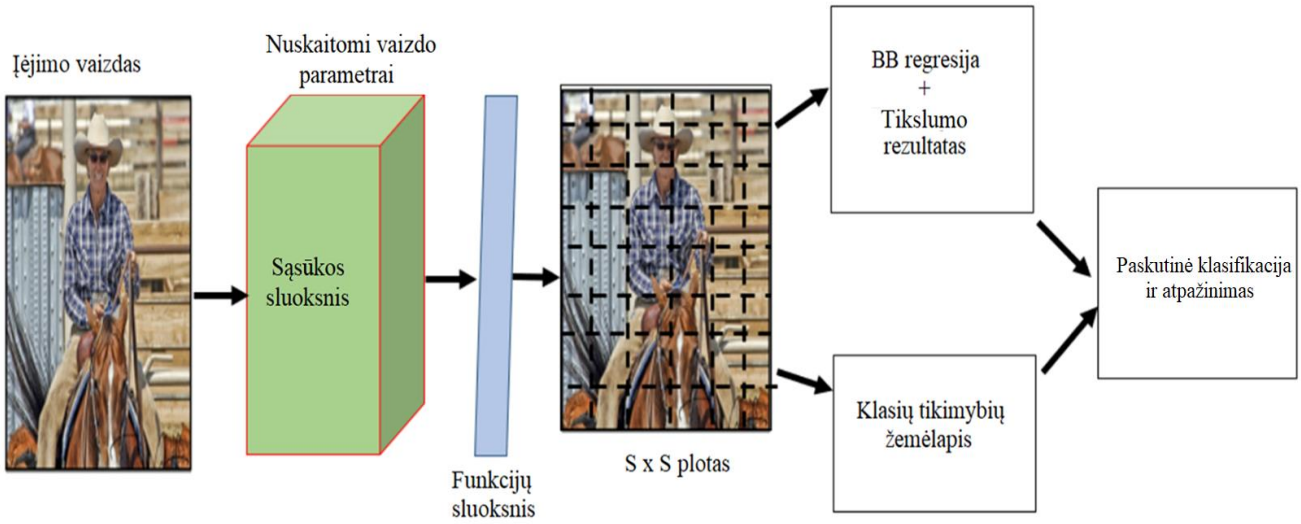
2.3. Regresija pagrįsti objektų aptikimo algoritmai

Šiame poskyryje apžvelgiami regresija pagrįsti objektų aptikimo algoritmai, kurie yra plačiai paplitę ir taikomi realiojo laiko sistemose [31]. Regresija pagrįsti objektų detektoriai, pavyzdžiui,

YOLO (angl. *You Only Look Once*), SSD (angl. *Single Shot Multibox Detector*), *Retina-Net*, taip pat atsižvelgia į funkcijų piramidės struktūrą, kad numatytų aprėpties dėžę ir klasifikaciją pagal skirtingas funkcijų lygio skiriamąsias gebas [31].

2.3.1. YOLO tinklo algoritmas

YOLO tinklo modelis padalija visą vaizdą į regionus, o kiekvienam regionui yra priskiriamos aprėpties dėžė ir klasės tikimybė [26]. YOLO architektūra pavaizduota 2.10 paveiksle. Šis tinklo algoritmas yra vientisas detektorius, kuris išskaido vaizde esantį objektą nuo vaizdo taškų iki erdvėje atskirtų aprėpties langelių ir susijusios klasės tikimybės [26]. Kadangi regiono pasiūlymų generavimo etapas yra visiškai priklausomas nuo įėjimo vaizdo, su YOLO naudojant nedidelį regionų kandidatų rinkinį tiesiogiai numatomas aptikimas [23]. Algoritmu vaizdas padalijamas į $S \times S$ tinklelį, kiekvienam tinkleliui numatant C klasės tikimybę, B aprėpties langelio vietas ir numatomą tikslumo įvertinimą [23].

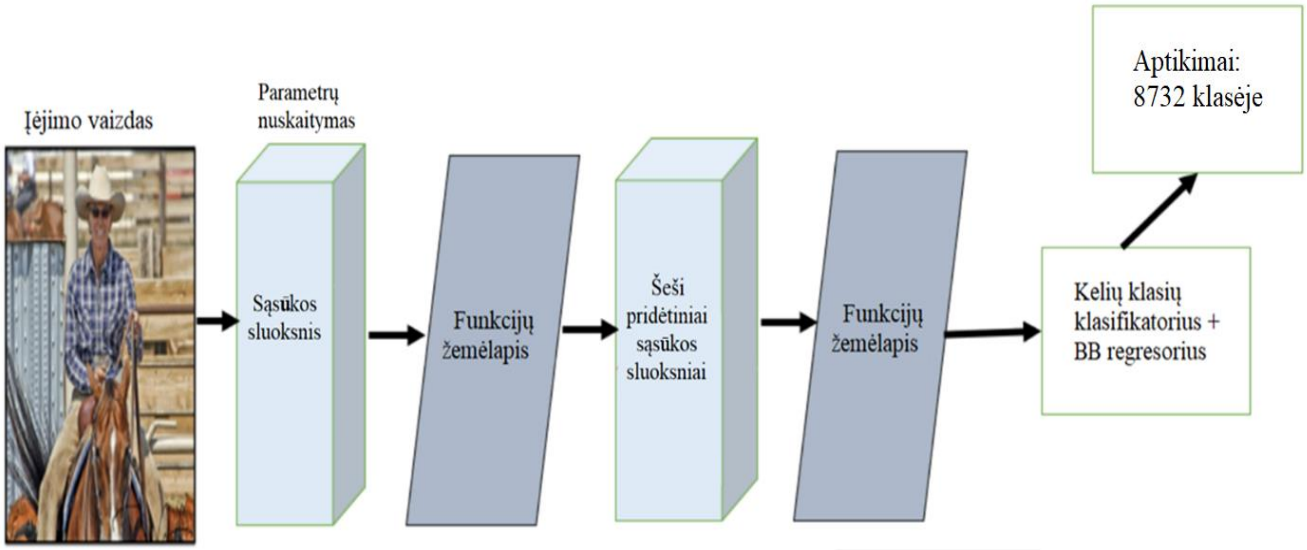


2.10 pav. YOLO tinklo algoritmo architektūra [26]

Atliktame [26] tyrime su YOLO architektūra tyrėjams naudojant *VOC-2007* duomenų rinkinį pavyko pasiekti 155 FPS ir 52,7 % mAP. Pasirinkus naudoti naujesnę YOLO versiją YOLOv2, minėtiems tyrėjams pavyko gauti aukštesnį vidutinį aptikimo tikslumo vidurkį, jis siekė 63,4 %, bet mažesnį apdorojamų kadrų per sekundę greitį – 45 FPS. Pagrindiniai YOLO objektų aptikimo modelių trūkumai yra paveiksle esančių mažų objektų atpažinimas ir lokalizacijos tikslumo sumažėjimas, lyginant su dviejų pakopų detektoriais [26].

2.3.2. SSD tinklo algoritmas

SSD tinklo architektūros modelis yra skirtas objektų aptikimui realiuoju laiku [32]. Modelis naudoja tik vieną kadrą, kad aptiktų vaizde esančius objektus [26]. Siekiant pagerinti SSD aptikimo tikslumą, ypač aptinkant mažus objektus, jam buvo įdiegti kelių sąsajų ir kelių skiriamųjų gebų aptikimo metodai [31]. SSD tinklo architektūra yra pavaizduota 2.11 paveiksle. SSD modelis objektų aptikimo metu virš kelių žemėlapio objektų tiesiogiai klasifikuoja, tankiai uždeda aprėpties dėžes ir bando patobulinti kiekvieną uždėtą BB [34].



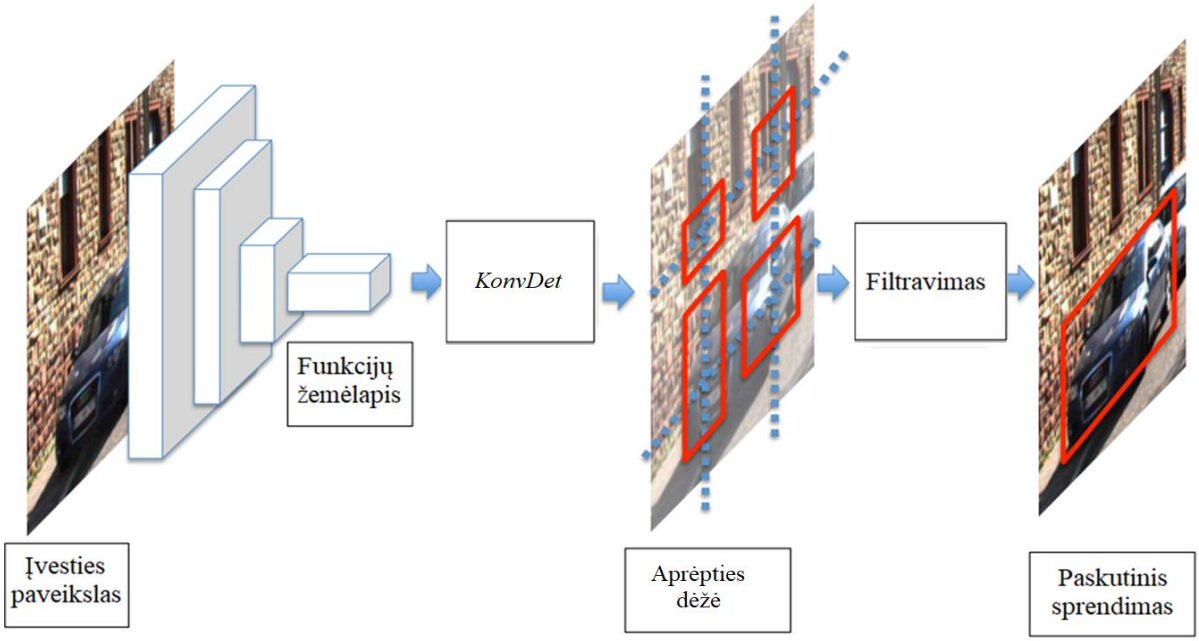
2.11 pav. SSD tinklo algoritmo architektūra [26]

SSD300 modelis naudojant *VOC-2012* duomenų rinkinį atliktame [26] tyrime pasiekė 74,3 % mAP ir 59 FPS spartą. O tyrėjams [26] su SSD 500 modeliu naudojant *VOC-2007* duomenų rinkinį pavyko pasiekti 76,9 % mAP ir 22 FPS spartumą, kurio gauti rezultatai pralenkė RCNN ir YOLOv1 tinklus. Pagrindiniai SSD trūkumai lyginant su R-CNN: dėl didesnio išvedimo greičio pasitaiko daugiau klasifikavimo klaidų, tačiau yra mažiau lokalizavimo klaidų [26].

2.3.3. *SqueezeDet* tinklo algoritmas

SqueezeDet tinklo architektūros aptikimo linija pavaizduota 2.12 paveiksle. Šis tinklas veikia taip, kad sąsūkos neuronų tinklas panaudoja funkcijų žemėlapį iš įvesties vaizdo ir pateikia jį į *KonvDet* sluoksnį [33]. Tada su *KonvDet* sluoksniu apskaičiuojamos aprėpties dėžės (BB), sutelktos aplink *W* ir *H* tolygiai paskirstytus tinklo centrus. Čia *W* ir *H* yra tinklelio centrų skaičius išilgai horizontalios ir vertikalios ašių [33]. Kiekviena aprėpties dėžė susiejama su pirmu aptikimo tikslumo įverčiu ir *C* sąlygine klasės tikimybe. Tuomet tinklas saugo viršutinės *N* aprėpties dėžes su didžiausiu

tikslumu ir naudoja NMS (*ne maksimalų slopinimą*), kad jas nufiltruotų ir būtų baigtas aptikimo procesas.



2.12 pav. *SqueezeDet* tinklo algoritmas [33]

Kiekviena aprėpties dėžė susieta su *C+1* reikšmėmis, kur *C* yra reikiamų atskirti klasių skaičius, o papildomas 1 yra skirtas aptikimo tikslumo įverčiui [33]. Kiti *C* skaliarai rodo sąlyginį klasės tikimybių pasiskirstymą atsižvelgiant į tai, kad objektas egzistuoja aprėpties laukelyje.

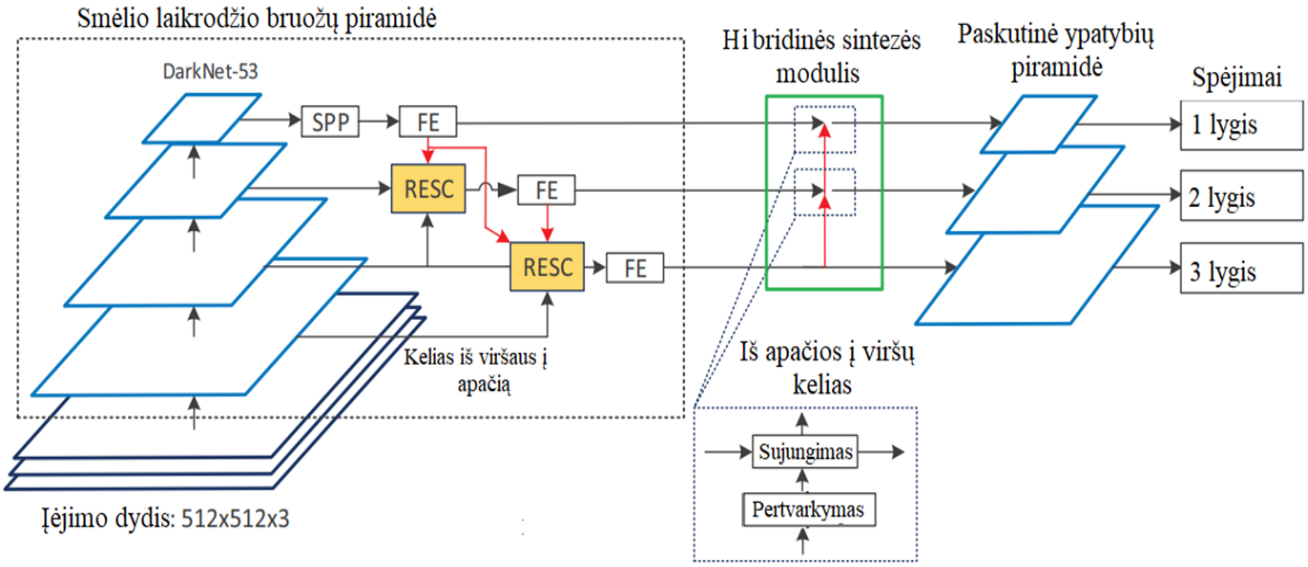
KonvDet sluoksnis veikia kaip stumdomas langas, kuris gali objekto žemėlapyje slinkti kiekviena erdvės kryptimi. Kiekvienoje pozicijoje apskaičiuojamos $K \times (4 + 1 + C)$ reikšmės, kurios koduoja aprėpties dėžės prognozes [33]. Čia *k* nurodo aprėpties dėžių su iš anksto pasirinktomis formomis skaičių. 1 yra patikimumo balas, o 4 – aprėpties dėžių koordinačių reikšmės. Aprėpties dėžių koordinates galima apskaičiuoti pagal pateiktas 2.5 formules [33].

$$\begin{aligned} x_i^p &= \hat{x}_i + \hat{w}_k\, \delta x_{ijk}, \quad y_j^p = \hat{y}_j + \hat{h}_k\, \delta y_{ijk}, \\ w_k^p &= \hat{w}_k \exp(\delta w_{ijk}), \quad h_k^p = \hat{h}_k \exp(\delta h_{ijk}), \end{aligned} \tag{2.5}$$

čia $x_i^p, y_j^p, w_k^p, h_k^p$ yra numatomos aprėpties dėžių koordinatės.

Atliktame [33] tyrime su *SqueezeDet* modeliu naudojant *KITTI* duomenų rinkinį mašinas pavyko aptikti iki 90,2 %, dviratininkus iki 82,9 %, pėsčiuosius 77,1 % aptikimo tikslumu bei pasiekti 57,2 FPS apdorojimo spartą [33]. Tokie rezultatai rodo, kad *SqueezeDet* yra tinkamas realiuoju laiku aptikti ir identifikuoti objektus.

Regresija pagrįsti objektų aptikimo detektoriai suvartoja daug energijos ir patiria sunkumų atpažįstant mažus objektus. Pasitelkiant pridėjimo ir sąsūkos operacijas yra patiriami skaičiavimo našumo nuostoliai, kuriuos galima sumažinti taikant sujungimo operaciją [33]. Atliktame [34] tyrime buvo pasiūlytas RHF (angl. *Recursive Hybrid Fusion*) tinklas: RESC (angl. *Reshaped and Reconstructed*) blokas sujungtas su hibridinio sintezės sujungimo modeliu (angl. *Hybrid bottom-up*) [34]. RHF tinklo architektūra pateikta 2.13 paveiksle.



2.13 pav. RHF tinklo architektūra [34]

RHF tinklas buvo išbandytas [34] straipsnyje naudojant skirtingus pagrindus – *Darknet53, CSPdarknet53, VGG-16*. Bandymų rezultatams, nurodytiems 2.2 lentelėje, gauti buvo panaudotas *UAVDT* duomenų rinkinys [34]. Reikšmingiausias AP padidėjimas taikant RESC ir hibridinę sintezę buvo gautas su *CSPdarknet53* pagrindu, AP pakilo iki 70,42 %. Didžiausias spartos nuostolis pasiektas su *CSPdarknet53* pagrindu. Gauti rezultatai rodo, kad sudarius RHF tinklą pagerėjo AP, bet sumažėjo FPS greitis.

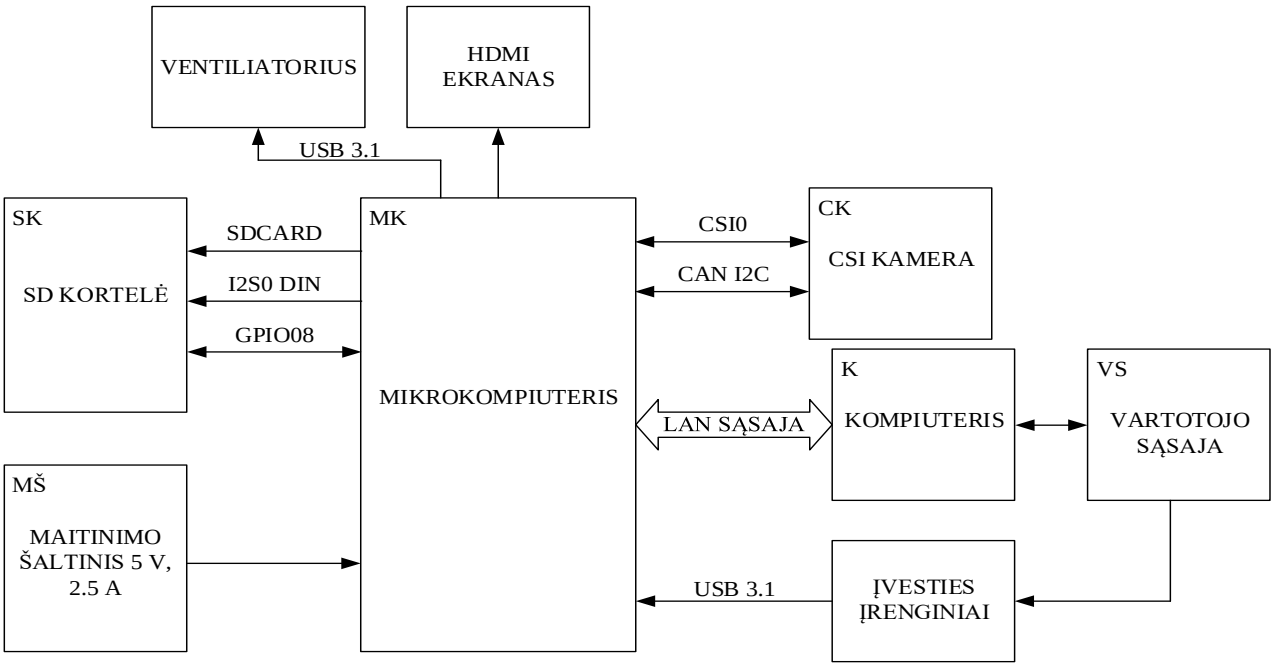
2.2 lentelė. RESC ir hibridinio sintezės modulio tyrimas [34]

Pagrindas	RESC	Hibridinė sintezė	AP, %	FPS
Darknet53		×	62,18	28,5
	×		61,25	28,1
	×	×	**66,18**	**27,8**
CSPdarknet53		×	68,75	36,4
	×		67,61	35,2
	×	×	**70,42**	**34,8**
VGG-16		×	55,18	31,3
	×		56,14	30,8
	×	×	**60,13**	**29,8**

Hibridinio sintezės modulio efektyvumas siekiant pagerinti mažų objektų aptikimo tikslumą yra įrodytas ir gali būti panaudotas su skirtingais pagrindais [34]. Kadangi telkimas reikalauja mažiau resursų nei sąsūkos operacija, jis dažnai pasitelkiamas siekiant lengviau pagerinti pagrindo efektyvumą. Hibridinis „iš apačios į viršų“ sintezės modulis yra geras sprendimas siekiant pagerinti mažų objektų aptikimo tikslumą. Rekursinis RESC modulis pagerina visų mastelio vaizdų (mažų, vidutinių ir didelių) kontekstines savybes. Eksperimentiniai [34] rezultatai įrodo, kad RHF tinklas yra pranašesnis naudojant programas realiuoju laiku, ypač mažų objektų aptikimui.

2.4. Tyrimams skirtos sistemos projektavimas ir analizė

Šiame poskyryje apžvelgiama eksperimentiniams tyrimams atlikti ir rezultatams gauti naudota sistema su aparatine įranga ir jos ypatumai. Norint sudaryti elektronikos komponentų atpažinimo ir skaičiavimo sistemos sandaros schemą nuspręsta, kokius įrenginius reikės valdyti, kaip komunikuoti ir tinkamai bei saugiai valdyti sistemą. Objektams atpažinti ir skaičiuoti pasirinktas mikrokompiuteris, kuris maitinimas iš bendro maitinimo šaltinio, vaizdui gauti panaudota kamera, rezultatams vaizduoti išvestame ekrane, atmintyje laikoma operacinė sistema su programine įranga ir kaupiami tarpiniai rezultatai. Vartotojo įvesčiai pasirinkta naudoti LAN (angl. *Local area network)* sąsają ir įvesties įrenginius. Norint užtikrinti sistemos neperkaitimą įdiegtas ventiliatorius. Elektronikos komponentų atpažinimo ir skaičiavimo sistemos sandaros schema pateikta 2.14 paveiksle.



2.14 pav. Elektronikos komponentų atpažinimo ir skaičiavimo sistemos sandaros schema

Turint sudarytą sistemos sandaros schemą, galima apžvelgti, kokią aparatinę įrangą pasirinkti. Dėl savo našumo, kokybės ir kainos santykio šiuolaikiniams dirbtinio intelekto darbo krūviams buvo pasirinktas *Nvidia Jetson Nano* mikrokompiuteris. Mikrokompiuteris palaiko *Nvidia JetPack,* kuris apima plokštės palaikymo paketą (angl. *board support package*), *Linux* OS (angl. *Operating System*), CUDA (angl. *Compute Unified Device Architecture*), cuDNN (angl. *CUDA Deep Neural Network*) ir *TensorRT* programinės įrangos bibliotekas, skirtas giliajam mokymuisi, objektų atpažinimui [38]. Taip pat įrenginys gali būti maitinimas per *micro USB* ir turi daug Į / I, nuo GPIO (angl. *General-purpose input/output*) iki CSI. Mikrokompiuteris pagal pateiktą [39] specifikaciją turi 128 branduolių *Maxwell* GPU, 4 branduolių *ARM A57* CPU su 1,43 GHz taktiniu dažniu, operatyvioji atmintis yra 4 GB 64bit LPDDR4 (angl. *Low Power Double Data Rate*), duomenų atmintis yra palaikoma *microSD* kortelėje arba prisijungus prie USB galima naudoti išorinį kietąjį diską. Mikrokompiuteris taip pat turi dvi MIPI (angl. *Mobile Industry Processor Interface*) CSI-2 DHPY juostos įvestis, ekrano įvestys yra HDMI (angl. *High Definition Multimedia Interface*) ir DVI (angl. *Digital Visual Interface*) bei turi 4 USB 3.0 įvesties įrenginiams prijungti ir USB 2.0 Micro-B įrenginio maitinimui skirtą prievadą.

Turint CSI (angl. *Camera Serial Interface*) įvestį į mikrokompiuterį pasirinkta naudoti *Raspberry Pi* CSI kameras. Turint omenyje, kad bus atliekamas komponentų ant konvejerio skaičiavimas, galima daryti prielaidą, kad ne visada bus tinkamas apšvietimo lygis. Šiai problemai spręsti pasirinkta *Raspberry Pi NoIR Camera v2* kamera. Šios kameros modulis turi aukštos kokybės *Sony IMX219* vaizdo jutiklį su fiksuoto židinio lęšiu. 8 megapikselių skiriamoji geba leidžia užfiksuoti *3280 x 2464* pikselių dydžio nuotrauką [40]. Pačios kameros modulis mažas ir nesunkus, tad išvengiama galimos problemos konstruojant kamerą virš konvejerio. NoIR (angl. *No InfraRed*) filtras ant lęšio leidžia su šia kamera gauti kokybišką vaizdą esant mažam apšviestumui. *Raspberry Pi NoIR Camera v2* CSI kamera pagal [40] specifikaciją palaiko 1080p30, 720p60 ir 480p60/90 vaizdo režimus. Kameros horizontalus 62,2° ir vertikalus 48,8° regėjimo laukai gali tinkamai aprėpti norimą matymo lauką, stebėti slenkančius elektronikos komponentus, o kameros optikos dydis lengvai įmontuojamas nesudarant pašalinio šešėlio nuo apšvietimo juostos.

Mikrokompiuterio atminčiai pasirinkta *microSD* kortelė su 64 GB dydžio talpa, pakankama operacinei sistemai ir kitiems reikalingiems papildiniams įdiegiant neuroninių tinklų modelius. Šios sistemos maitinimo šaltiniui pasirinktas *Raspberry T6712DV* [41] maitinimo šaltinis dėl 5,1 V išėjimo įtampos, 2,5 A išėjimo srovės ir maitinimo jungties micro-USB išvado, tinkančio pasirinktam mikrokompiuteriui. Bandymams ir rezultatams gauti buvo naudotas ekranas, prijungtas per HDMI, vartotojo sąsajai pasirinktas komunikavimas per LAN naudojant *VNC Server* programinę įrangą, lengvai įvesčiai į mikrokompiuterį buvo naudota klaviatūra ir pelė. Mikrokompiuteriui aušinti pasirinktas stalinis, 2,5 W galią turintis ventiliatorius, kuris maitinamas per USB jungtį iš mikrokompiuterio.

Norint išbandyti sistemos veikimą, buvo atliktas apkrovos bandymas atliekant dirbtinio neuronų tinklo mokymą su dideliu apdorojamų duomenų kiekiu (angl. *batch size*). Atliekant mokymus, buvo pastebėta, kad mikrokompiuteris įkaista iki didesnės nei +75 °C temperatūros, kuri nėra saugi ir gali pažeisti įrenginį. Todėl norint išlaikyti saugias darbines CPU ir GPU temperatūras esant dideliam procesų vykdymo krūviui, nuspręsta naudoti išorinį ventiliatorių, kuris iš išorės vėsintų mikrokompiuterį. Pasirinktas ventiliatorius yra *160 x 150 x 100* mm matmenų dydžio, turintis 5 V ir 0,5 A *USB-A* įėjimą maitinimui, tokio ventiliatoriaus galia siekia 2,5 W. Išbandžius minėtą sistemą su ventiliatoriumi buvo pasiektos saugios CPU ir GPU darbinės temperatūros – apie +35 °C. Apšvietimui pasirinkta reguliuojama LED juosta su 4400 °K spalvos temperatūra, siekia apie 6600 lux apšviestumo.

2.5. Skyriaus apibendrinimas

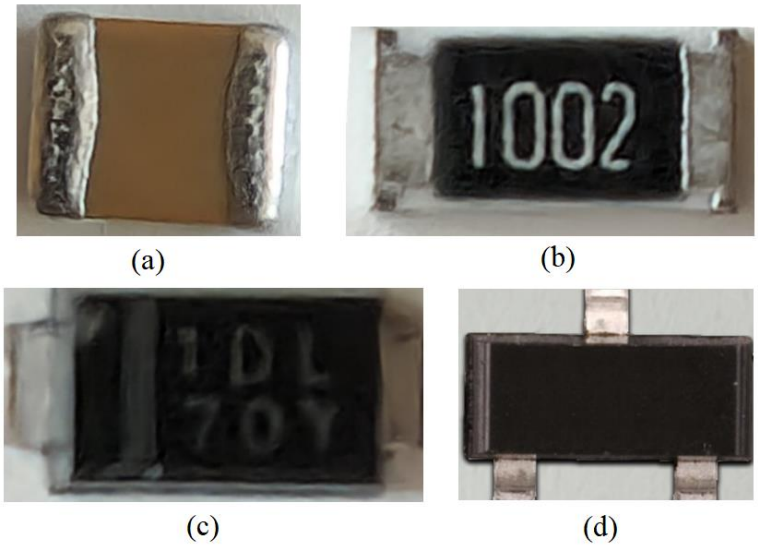
Šiame skyriuje buvo apžvelgti vaizdo aptikimo algoritmai. Pirmiausia nagrinėti klasifikacija pagrįsti objektų aptikimo algoritmai: RCNN, SPPNet, greitasis RCNN, greitesnysis RCNN ir maskuotasis RCNN. Atlikta algoritmų lyginamoji analizė, remiantis gautais rezultatais aptinkant objektus su *VOC2007* duomenų rinkiniu. Toliau išnagrinėti regresija pagrįsti objektų aptikimo detektoriai: YOLO, SSD, *SqueezeDet*. Taip pat buvo nagrinėjamas RHF tinklo algoritmas ir jo pritaikymas naudojant skirtingus pagrindus: *Darknet53*, *CSPdarknet53* ir *VGG-16*. Galiausiai buvo suprojektuota ir apžvelgta sistema eksperimentiniams tyrimams atlikti, ji skirta elektronikos komponentams ant konvejerio skaičiuoti. Minėta sistema yra pritaikyta atlikti realaus laiko vaizdo apdorojimą pritaikant intelektualiuosius metodus, išlaikant puikų kainos ir galios santykį.

3. DUOMENŲ RINKINIO SUDARYMAS IR PARUOŠIMAS ELEKTRONIKOS KOMPONENTŲ APTIKIMO IR SKAIČIAVIMO METODŲ EKSPERIMENTINIAMS TYRIMAMS

Ankstesniuose skyriuose buvo atlikta mokslinės literatūros apie vaizdų apdorojimo ir analizavimo etapus, kai objektas vaizde aptinkamas pritaikant dirbtinį intelektą, apžvalga. Šiame skyriuje aptariamas eksperimentiniams tyrimams pasirinktas duomenų rinkimo metodas, sudarytas duomenų rinkinys, tinklo mokymo ir objektų atpažinimo algoritmai. Pasiūlytas ir paaiškintas objektų atpažinimo ir skaičiavimo algoritmas bei pasirinkta sistemos įvertimo metodika. Šie metodai ir būdai bus naudojami eksperimentiniams tyrimams atlikti.

3.1. Mokymo duomenų paruošimas

Šiame tyrime duomenų rinkiniui formuoti buvo naudotas *Data Capture Control* įrankis, skirtas duomenims iš tiesioginio kameros vaizdo kaupti ir žymėti realiuoju laiku. Šis įrankis leidžia pasirinkti duomenų, reikalingų klasifikuoti arba aptikti, rinkinio tipą ir nurodyti rinkinį, skirtą saugoti duomenims, reikalingiems patikrinimui, mokymui, patvirtinimui. Įrankis taip pat turi ir kitų funkcijų, pavyzdžiui, išsaugoti duomenis, išvalyti žymėjimus panaikinus polangių fiksavimą, sujungti išskaidytus rinkinius į vieną, žymėti / keisti dydį / keisti žymės klasę.



3.1 pav. Pasirinkti elektronikos komponentai duomenų rinkiniui: a) kondensatorius; b) rezistorius; c) diodas; d) tranzistorius

Duomenų rinkiniai buvo sudaryti iš SMD tipo elektroninių komponentų nuotraukų. Šiuos rinkinius sudaro 4-ios klasės: kondensatoriai, rezistoriai, diodai ir tranzistoriai. Pasirinkti elektronikos

komponentai, kurie naudoti duomenų rinkiniui, yra pateikti 3.1 paveiksle. Iš pavaizduotų komponentų labiausiai išsiskiria kondensatoriai, kurie vizualiai gali būti atpažinti, kaip turintys keramikinį, šviesiai rudos spalvos pagrindą centre ir iš abiejų pusių gaubiami litavimui skirtų sidabrinės spalvos sienelių. Likusios trys pasirinktos rūšys: rezistoriai, diodai ir tranzistoriai, vizualiai skiriasi vienas nuo kito mažiau. Pagrindiniai dalykai, leidžiantys šiuos tris komponentus atskirti vienus nuo kitų, yra litavimo kojų dydis ir pozicija prie pagrindo. Rezistoriaus litavimo kojos yra panašios į kondensatoriaus, t. y. plotis sutampa su pagrindu, o pasirinktas diodas turi siauresnes litavimo kojas, kurios yra centruotos. Taip pat pasirinktas tranzistorius turi tris kojas, kurios yra iš apačios ir viršaus. Kiti galimi vizualūs būdai atskirti šiuos komponentus yra viršuje esantys žymėjimai – ant rezistorių gali būti išspausdintas jų varžos nominalas, naudojant skaičius ir skaičius su raidėmis. Diodas taip pat gali turėti spausdintų raidžių, skaičių, bet dažniausiai pasitaiko jų srovės kryptį indikuojanti pilka arba balta juosta, kuri yra nubrėžta nuo viršaus iki apačios per pagrindą. Tačiau ne visada ant nurodytų komponentų būna žymenys, pasitaiko, kad gamintojai nenurodo arba nutrina.

Elektronikos SMD komponentai buvo pasirinkti dėl jų identifikavimo sunkumo ir labai mažo dydžio – tai kelia iššūkį realaus laiko objektų aptikimo algoritmams. Tai pat šie komponentų tipai yra kiekybiškai dažniausiai naudojami gaminamoje produkcijoje ir eksploatavime ir parduodami dideliais kiekiais, dažniausiai supakuoti ritėmis. Buvo fotografuojami elektronikos komponentai, esantys ant skirtingų spalvų paviršių. Elektronikos komponentai buvo įpakuoti atskirai, išpakuoti ir esantys ant spausdintinės plokštės. Duomenų rinkinyje naudojamų nuotraukų pavyzdžiai pateikti 3.2 paveiksle. Komponentų nuotraukose buvo sutalpinta nuo 1 iki 55 vienetų.

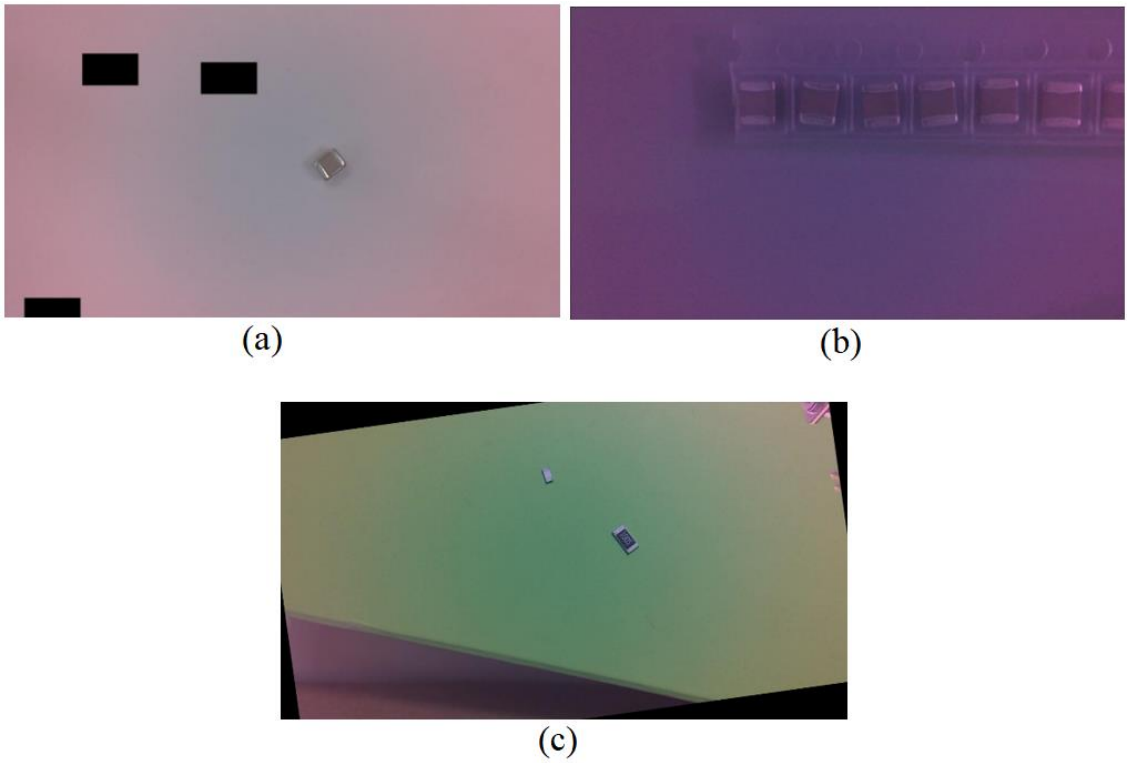
(a) (b)

3.2 pav. *SMD* komponentų duomenų rinkinio nuotraukų pavyzdžiai: a) įpakuoti komponentai; b) išpakuoti pavieniai komponentai

Pradinį duomenų rinkinį sudaro 3005 vnt. nuotraukų. Duomenų rinkinys išskaidytas į 2405 nuotraukas tinklui mokyti (angl. *training*), 300 patvirtinti (angl. *validation*) ir 300 bandymams atlikti (angl. *test*). Rinkinį sudaro 11875 žymų, iš kurių 4478 rezistoriai, 3404 diodai, 3297 kondensatoriai

ir 696 tranzistoriai. Tokio dydžio rinkinys buvo sudarytas norint išbandyti duomenų rinkinio kaupimą, patikrinti, ar tinkamas apšvietimas, kampas ir kokybė. Su šiuo rinkiniu išbandytas pirmasis etapas – komponentų aptikimas.

Kitu etapu padidintas sukurtas pirminis duomenų rinkinys. Atlikus išankstinį apdorojimą su automatinių pikselių duomenų orientavimu (su *EXIF* orientacijos pašalinimu), rinkinys buvo didinamas naudojant duomenų didinimo metodus (angl. *data augmentation*), pavyzdžiui, atsitiktinius iškarpymus, sužymėto ploto ryškumo reguliavimą. Po duomenų rinkinio didinimo procedūros duomenų rinkinys padalintas naudojant *split-folders Python* bibliotekos funkciją *ratio()*. Duomenų rinkinys iš 4061 nuotraukų padalintas į 3249 skirtas tinklo mokymui, 406 patvirtinimui ir 406 bandymams. Duomenų rinkinyje yra 15950 žymės, iš kurių 6007 rezistoriai, 4581 diodas, 4470 kondensatorių ir 892 tranzistoriai.



3.3 pav. Išankstinio apdorojimo metodai skirti duomenų didinimui: a) atsitiktiniai išpjovimai; b) spalvų reguliavimas; c) atsitiktiniai pasukimai tarp -15° ir +15° kampu

Trečiuoju etapu padidintas pirminis duomenų rinkinys. Atlikus išankstinį automatinio pikselių duomenų orientavimo (su *EXIF* orientacijos pašalinimu) apdorojimą pereita prie rinkinio didinimo, kuriam naudoti duomenų didinimo metodai: nuotraukos vertikalūs apvertinimai, iki 90° pasukimas aukštyn ir žemyn, atsitiktinis pasukimas nuo -15° iki +15° kampu. Po duomenų rinkinio didinimo procedūros duomenų rinkinį sudarė 7661 nuotrauka, jos buvo padalintos į 7061 nuotrauką, skirtą

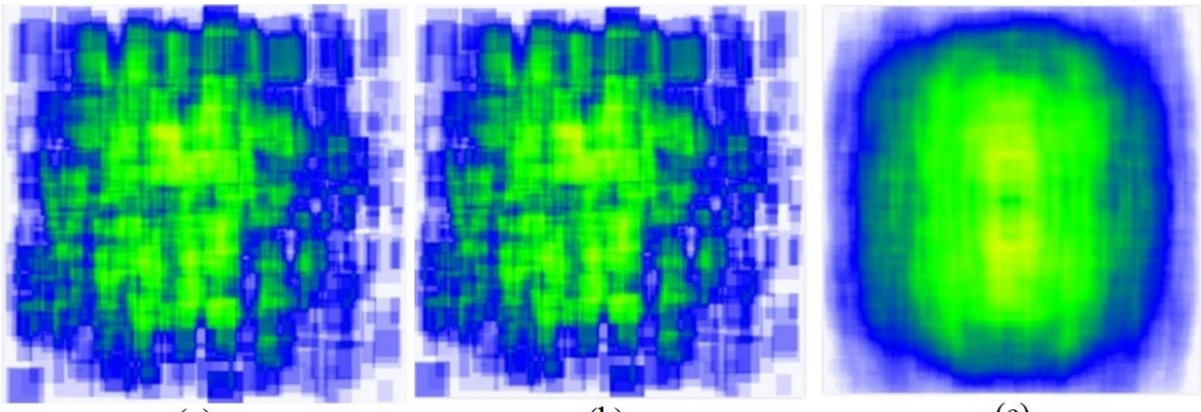
tinklui mokyti, 300 patvirtinimui ir 300 bandymams. Po pirminio duomenų rinkinio didinimo gautos 28536 žymės, iš kurių 10467 rezistoriai, 8543 diodai, 8198 kondensatoriai ir 1328 tranzistoriai.

Išankstinio nuotraukų apdorojimo metodai, pavyzdžiui, atsitiktiniai išpjovimai, spalvų reguliavimas, atsitiktiniai pasukimai, skirti duomenų rinkinio didinimui, pavaizduoti 3.3 paveiksle. Šie metodai buvo pasirinkti naudoti sudarant trečią duomenų rinkinį dėl jų atliekamų apdorojimų, kurie nepakenkia svarbių elektronikos komponentų detalių matomumui. Naudoti kiti nuotraukų apdorojimo metodai, pvz., šviesos ir kontrasto keitimai galiausiai buvo atmesti dėl tikimybės, kad rezistorių, diodų ir tranzistorių litavimo kojos gali išsilieti ir pasislėpti fone. Be svarbių detalių, pavyzdžiui, litavimo kojų, nepavyktų tinkamai ir tiksliai identifikuoti komponentų. O spalvų reguliavimas, atspalvio sodrumo keitimas, pakeisdavo tik nuotraukos atspalvį, todėl nepakenkdavo elektronikos komponentų struktūros matomumui.

3.1 lentelė. Duomenų rinkinių specifikacija

Eil. Nr.	**Nuotraukų kiekis**, vnt.				**Klasifikavimo žymų kiekis**, vnt.				
	Iš viso	**Patvirtinimui**	**Bandymams**	**Mokymui**	**Iš viso**	**Rezistoriai**	**Diodai**	**Kondensatoriai**	**Tranzistoriai**
1.	3005	300	300	2405	11875	4478	3404	3297	696
2.	4061	406	406	3249	15950	6007	4581	4470	892
3.	7661	300	300	7061	28536	10467	8543	8198	1328

Sukurtų duomenų specifikacijos pateiktos 3.1 lentelėje. Kuriant duomenų rinkinius buvo atkreiptas dėmesys į pašalinių objektų, turinčių panašią spalvų paletę kaip žymimų komponentų, įterpimą, taip pat buvo keičiamas apšvietimas, fotografavimo kampas. Šie iškraipymai sugeneruoti, norint sumažinti galimą paklaidą, jeigu, paleidus objektų aptikimo sistemą, atsirastų pašalinių objektų arba dulkių. Klasifikuojamų objektų žymėjimų žemėlapiai pateikti 3.4 paveiksle. Matoma didžiojoje dalyje nuotraukų arčiau centro (geltona, žalia spalva) esanti komponentų pozicija ir duomenų didinimo poveikis bendram komponentų išsidėstymui.



(a) (b) (c)

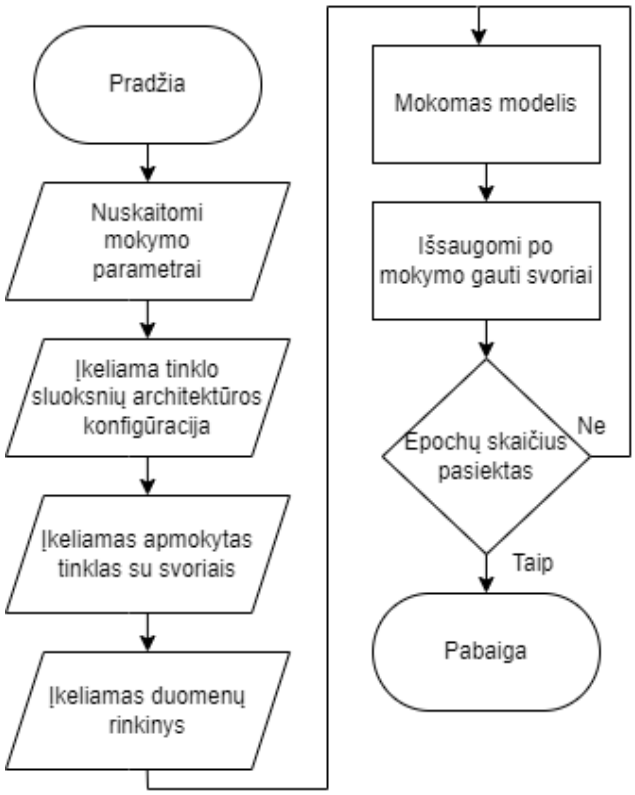
3.4 pav. Objektų žymėjimų žemėlapiai: a) pirmas duomenų rinkinys; b) antras duomenų rinkinys; c) trečias duomenų rinkinys

Duomenų rinkiniai buvo sukurti *Pascal VOC* formatu ir vėliau konvertuoti į papildomus formatus *COCO*, *Darknet*, *YOLO-Keras*, *YOLOv4-PyTorch*, *YOLOv5-PyTorch*. Duomenų rinkinių nuotraukų rezoliucija *1280 x 720*, failo dydis svyruoja tarp 26–40 KB. Skirtingiems duomenų rinkinio formatams eksportuoti naudota *Roboflow* [42] terpė. Skirtingų formatų žymės bus naudojami atitinkamos architektūros modelių mokymo algoritmams.

3.2. Tinklų mokymas

Tiriamajame darbe nuspręsta sistemoje panaudoti skirtingus objektų atpažinimo SSD-Mobilenet-V1, YOLOv3, YOLOv3-tiny, YOLOv4, YOLOv4-tiny, YOLOv4-Scaled, YOLOv5s detektorius. Modeliams optimizuoti pritaikytos *PyTorch*, *Keras* bibliotekos, kad pavyktų pasiekti kuo geresnį rezultatą [58, 59]. Tinklų detektoriams mokyti naudotas kompiuteris su *AMD Ryzen 5 5800H* CPU, *Nvidia Geforce RTX 3060* GPU ir *Google Colab* [43] debesijos paslauga su *Tesla P100-16GB* GPU.

Objektų aptikimo detektorių mokymams pasirinktas vienodas duomenų rinkinys, pateiktas 3.1 lentelėje 3 numeriu. Mokyti pasirinkti modeliai su iš anksto turimais apmokytais svoriais. Mokymo iteracijų, epochų skaičius išlaikytas vienodas visiems modeliams, norint adekvačiai palyginti tarpusavyje. Remiantis [44, 45] straipsniuose pateikta metodika, taip pat bus aptarti baigtų mokyti modelių failų dydžiai. Modelių informacijos palyginimas pateiktas 3.2 lentelėje. Tinklo mokymo supaprastintam procesui pavaizduoti buvo sudaryta tinklo mokymo struktūrinė schema, ji pateikta 3.5 paveiksle.



3.5 pav. Tinklo mokymo struktūrinė schema

Pirmiausia objektams aptikti ir skaičiuoti pasirinktas regresija pagrįstas SSD-Mobilenet-V1 detektorius, kurio pagrindą sudaro *SSD-300* su *Mobilenet* [46]. Šis modelis pasirinktas dėl jo populiarumo realiojo laiko objektų aptikimui išmaniuosiuose ir įterptiniuose įrenginiuose. Norint lengviau optimizuoti aptikimo modelio mokymo spartumą, pasirinkta naudoti *PyTorch* biblioteką ir iš anksto jau apmokytą modelį *mobilenet-v1-ssd-mp-0.675*, kuris iš naujo mokytas su sukurtuoju duomenų rinkiniu [36]. Tinklo mokymai buvo vykdyti 100 epochų. Iš gautų mokymo rezultatų matyti, kad mažiausias gautas patvirtinimo nuostolio koeficientas siekia 2,1466. Bandant toliau mokyti tinklą, nebepavyko sumažinti šio gauto rezultato. Apmokyto tinklo modelio failo dydis yra 28,227 MB.

Toliau šiame darbe mokomi skirtingi YOLO detektoriai: YOLOv3, YOLOv3-tiny, YOLOv4, YOLOv4-tiny, YOLOv4-Scaled, YOLOv5s. Mokant modelius pasirinkta įėjime pateikti *416 x 416* nuotraukų pirminius dydžius. Pagrindiniai šių modelių architektūrų skirtumai yra tokie, kad YOLOv3 naudoja *Darknet53* pagrindą, YOLOv4 *CSPdarknet53* pagrindą, o YOLOv5 naudoja *Focus* struktūrą su *CSPdarknet53* pagrindu [47]. Šie algoritmai pasirinkti dėl jų greičio ir tikslumo apdorojant vaizdą realiuoju laiku [47].

YOLOv3 ir YOLOv3-tiny modeliams naudota *Keras* biblioteka, kad būtų paprasčiau pritaikyti tinklą ir algoritmus [48]. Abu modeliai mokyti 100 epochų taikant sluoksnių fiksavimo (angl. *Layers freeze*) metodą. Panaikinus sluoksnių fiksavimą, modelis buvo mokomas, kol nebepavyko mažinant

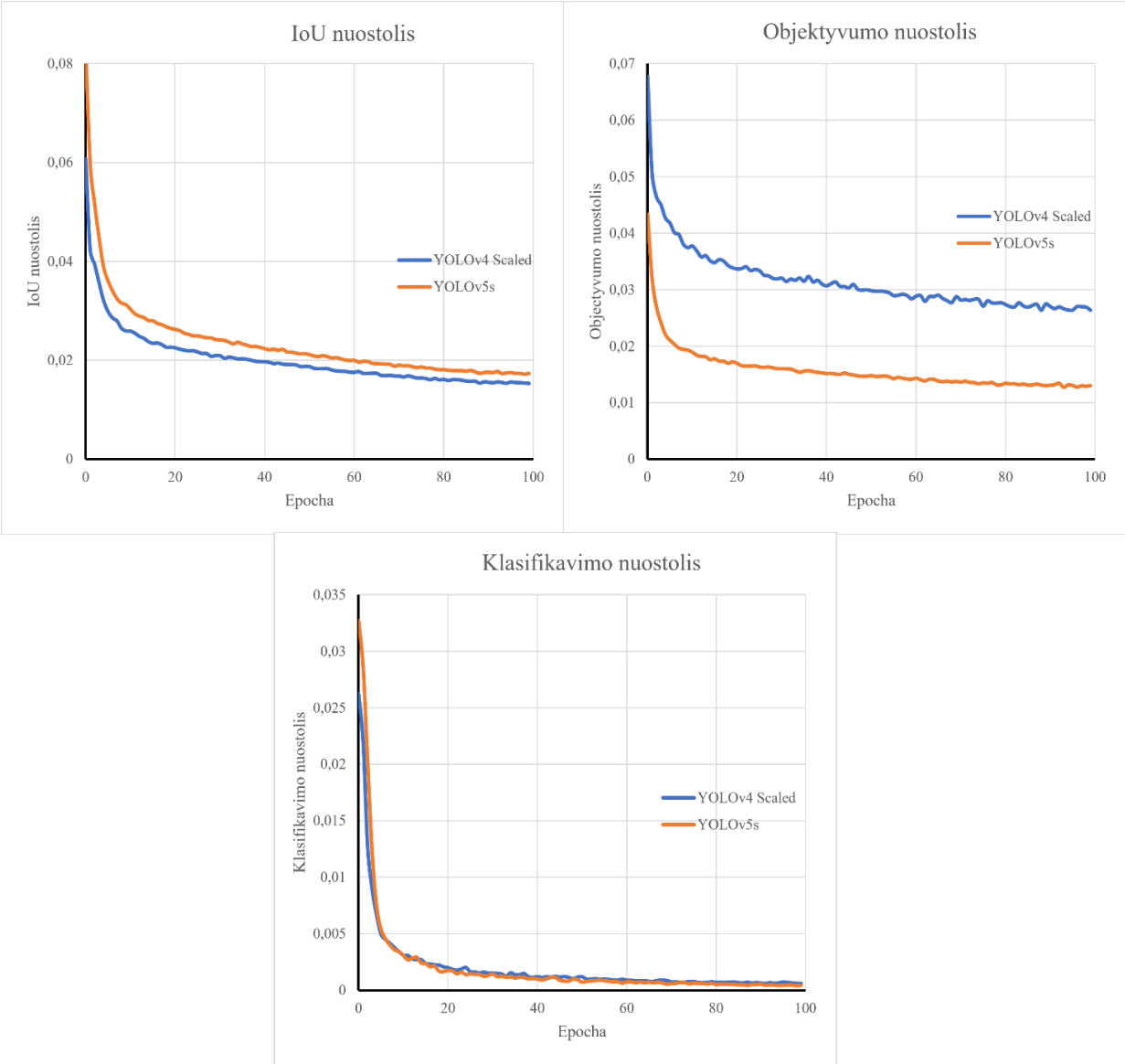
mokymosi žingsnį gauti mažesnio patvirtinimo nuostolio. YOLOv3 modelio patvirtinimo nuostolio koeficientas siekia 12,607, svorių failo dydis 237,2 MB. Atlikus mokymus su YOLOv3-tiny modeliu patvirtinimo nuostolis siekė 11,642, modelio failo dydis – 33,9 MB. YOLOv4 ir YOLOv4-tiny modelių valdymui naudota atvirojo kodo giliojo mokymosi sistema *Darknet* [49]. Naudojant *Darknet* YOLOv4 ir YOLOv4-tiny modeliai mokyti po 8000 iteracijų. YOLOv4 modelio svorių failo dydis 244,2 MB, YOLOv4-tiny – 22,5 MB. Toliau YOLOv4-Scaled ir YOLOv5s modelių mokymams atlikti ir algoritmams pritaikyti naudota *PyTorch* biblioteka. Modeliai buvo mokomi 100 epochų, baigto mokyti modelio failo dydis 401,3 MB.

3.2 lentelė. Objektų aptikimo modelių specifikacija

Modelis	**Biblioteka**	**Modelio dydis**, MB
SSD-MobileNet-v1	PyTorch	36,2
YOLOv3	Keras	237,2
YOLOv3-tiny		33,9
YOLOv4	Darknet	244,2
YOLOv4-tiny		22,5
YOLOv4-Scaled	PyTorch	401,3
YOLOv5s		14,1

Kadangi YOLOv4-Scaled ir YOLOv5s modelių algoritmai naudoja *PyTorch* biblioteką ir mokant visi tarpiniai parametrai buvo saugomi, buvo galima tarpusavyje palyginti gautus mokymo rezultatus. Jie pateikti 3.6 paveiksle. Grafikuose pavaizduotos modelių mokymų įvertinimo kreivės, kuriose pateiktas IoU (angl. *Intersection over Union*) nuostolis, objektyvumo (angl. *Objectness*) nuostolis ir klasifikavimo nuostolis. IoU nuostolis atspindi, kaip gerai detektorius gali nustatyti objekto centrą ir kaip gerai numatomos BB ribos, dengiančios objektą [50].

Remiantis gautais rezultatais, YOLOv4-Scaled detektorius gavo mažesnį nuostolį atlikus 100 epochų mokymą. Objektyvumo nuostolis – tai pateikiamas tikimybės matas, leidžiantis įvertinti, ar pateiktame regione galėtų egzistuoti objektas. Iš objektyvumo nuostolio grafiko matyti, jog YOLOv5s atlikus mokymus išlaikė apie 0,014 mažesnį nuostolio koeficientą, nei YOLOv4-Scaled modelis. Klasifikavimo nuostoliu įvertinama, kaip gerai detektorius gali numatyti teisingą tam tikro objekto klasę. Gauti klasifikavimo nuostolio rezultatai leidžia teigti, kad abu modeliai per 100 epochų išlaikė minimalų nuostolio skirtumą, bet YOLOv5s pavyko gauti šiek tiek mažesnį nuostolį.



3.6 pav. YOLOv4-Scaled ir YOLOv5s modelių nuostolių grafikai

Atlikus visų pasirinktų tinklo modelių mokymus su sudarytu duomenų rinkiniu, toliau vykdomas objektų aptikimas ir skaičiavimas. Šiems uždaviniams atlikti taikomi šiame skyriuje pagal 3.5 pav. struktūrinę schemą mokyti detektoriai, kurie bus įdiegti į norimas sistemas. Modelių svoriai parinkti turintys mažiausią nuostolio rezultatą atlikus visą mokymo ciklą.

3.3. Objektų aptikimas ir skaičiavimas

SMD komponentams aptikti ir skaičiuoti naudojami mokymo svoriai su sukurtu duomenų rinkiniu, turinčiu 4 klases (kondensatoriai, rezistoriai, diodai, tranzistoriai). Pasitelkiami SSD-Mobilenet-V1, YOLOv3, YOLOv3-tiny, YOLOv4, YOLOv4-tiny, YOLOv4-Scaled, YOLOv5s detektoriai. Turint jau apmokytus tinklų modelius galima pereiti prie objektų aptikimo realiuoju laiku sistemos realizavimo.

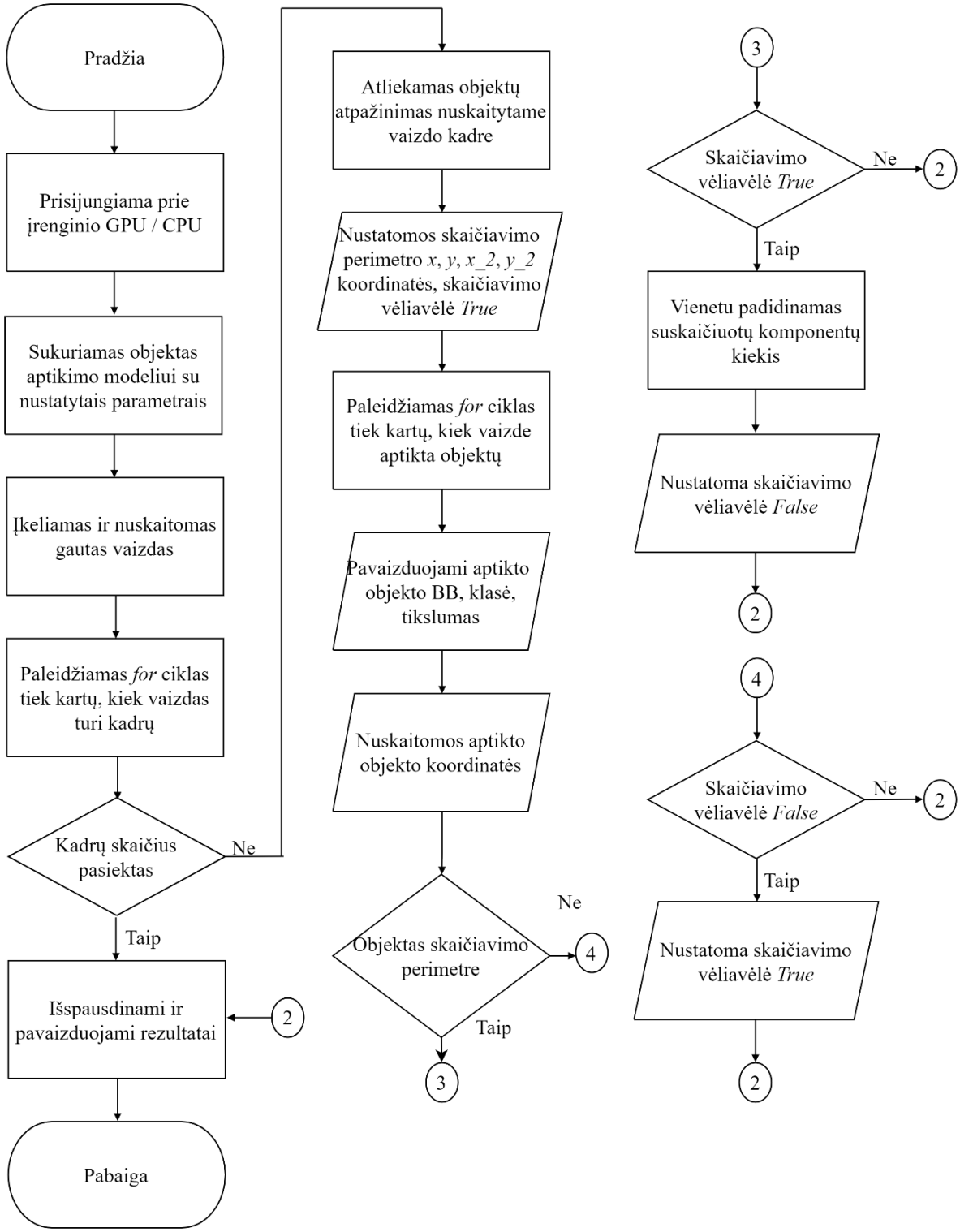
Objektų aptikimui įgyvendinti panaudotos modeliams tinkamos bibliotekos *PyTorch*, *Keras*, *Darknet*. Taikant minėtas specializuotas bibliotekas optimizuojamas detektoriaus veikimas, sutrumpinimas, paspartinamas atliekamas skaičiavimų greitis ir sumažinamas resursų naudojimas. Šios bibliotekos pritaikytos aptiktų objektų parametrams nuskaityti, o *CV2* biblioteka leido pavaizduoti BB, etiketes ir atpažinimo tikslumą. Remiantis [37, 41] aprašoma *detectNet* objektų atpažinimo struktūra, buvo sudarytas elektronikos SMD komponentų atpažinimo ir skaičiavimo programos veikimo algoritmas, kuris pavaizduotas 3.7 paveiksle.

Pirmiausia paleidus programą ir pritaikius mašininio mokymosi bibliotekas yra prisijungiama prie įrenginio GPU, kitu atveju pasirenkama naudoti CPU. Kitame etape su nustatytais parametrais ir apmokytu modeliu yra sukuriamas objektas, kuris bus panaudotas gautiems duomenims apdoroti (objektams aptikti). Toliau yra įkeliamas ir nuskaitomas vaizdas tiesiogiai iš kameros arba vaizdo įrašo *mp4* formatu. Nuskaičius vaizdo medžiagą ir gavus parametrus yra inicializuojamas *for* ciklas, kuris veiks, kol pasibaigs įkelta vaizdo medžiaga (kadrų skaičius). Atlikus patikrinimą, ar nepasibaigė vaizdo medžiaga, yra atpažįstami objektai gautame vaizdo kadre, išsaugomi aptiktų objektų parametrai.

Toliau elektronikos komponentų skaičiavimui paruošiamos koordinatės, nurodančios nustatytą lauką, kuriame bus vykdomi skaičiavimai. Paleidžiamas *for* ciklas tiek kartų, kiek aptikta objektų. Šiame cikle pavaizduojami aptikto objekto BB, klasė, aptikimo tikslumas ir atliekamas komponentų skaičiavimas. Patikrinamos gautos aptikto objekto parametrų koordinatės, ar objekto pozicija yra skaičiavimo perimetre ir, jeigu skaičiavimo vėliavėlės būsena yra *True*, tuomet komponentas pridedamas prie aptiktų komponentų skaičiaus. Norint išvengti nejudančio komponento skaičiavimo su kiekvienu kadro atnaujinimu, buvo pridėtas tikrinimas, ar komponentas yra pajudėjęs iš savo pozicijos, ir, jeigu suskaičiuotas komponentas yra nebe skaičiavimo koordinatėse, skaičiavimo būsena pakeičiama į *True*. Pasibaigus vaizdo medžiagai rezultatai yra išspausdami ir, jeigu buvo įkeltas vaizdo įrašas, o ne tiesioginiai kameros vaizdai, yra išsaugoma vaizdo medžiaga su pavaizduotais rezultatais. Komponentų skaičiavimo logikai įgyvendinti buvo pritaikyti tyrimų metodai iš [51-56].

Norint išbandyti, ar algoritmas tinkamai veikia su judančiu konvejeriu, buvo paleista elektronikos komponentų atpažinimo ir skaičiavimo programa ir išbandytas jos veikimas su keturiomis objektų klasėmis (kondensatoriais, rezistoriais, diodais, tranzistoriais). Norint matyti pasirinktą skaičiavimo plotą, buvo pažymėtos kraštinės. Išbandžius programos veikimą pastebėta, kad komponentų atpažinimas pavyko ir komponentui pasiekus reikiamą koordinatę, jis buvo pridėtas prie bendros sumos. Komponentui pajudėjus iš norimos skaičiavimo pozicijos ir grįžus atgal jis buvo vėl prisumuotas. Siekiant lengvesnio komponentų skaičiavimo ir pozicionavimo, buvo pridėta galima paklaida atpažinto objekto centro koordinatei, kuri kompensuota išplečiant skaičiavimo ploto

koordinates. Padidinus komponentų skaičiavimo plotą, kontrolinė juosta leido panaikinti tikimybę nesuskaičiuoti objektų, jeigu jų vertikali pozicija ant konvejerio pasikeičia.



3.7 pav. Elektronikos komponentų atpažinimo ir skaičiavimo programos veikimo algoritmas

Eksperimentiniams tyrimams atlikti naudotos kelios vertinimo metrikos, jos įgyvendintos taip, kaip aprašyta [51-57]. Kiekvienam SMD komponentui apskaičiuotas vidutinio tikslumo (AP) įvertis,

o vidurkio AP (mAP) įvertis bus naudojamas sistemos našumui įvertinti siekiant aptikti visus keturis komponentus. Tikslumo slenkstinis koeficientas numatytas 0,5 ir IoU [57] slenkstinis koeficientas 0,75. Be to, siekiant įvertinti panaudotų objektų aptikimo įterptose architektūrų sistemose greitį, bus suskaičiuojamas apdorojamų kadrų per sekundę (FPS) greitis ir išvedimo laikas (angl. *Inference Time* (IT)) [35]. Taip pat bus atliekamas SMD komponentų skaičiavimas, naudojant 3.5 pav. pasiūlytą algoritmą su minėtais skirtingais detektoriais. Eksperimentiniams tyrimams atlikti bus naudojama *Google Colab* su *Tesla P100-16GB* ir *Nvidia Jetson Nano 4GB*.

3.4. Skyriaus apibendrinimas

Šiame skyriuje pateiktos baigiamajame darbe mokymui sukurtų duomenų rinkinių specifikacijos, mokymo algoritmų modeliai su parametrais ir naudojamomis bibliotekomis, pasiūlytas objektų aptikimo ir skaičiavimo ant konvejerio algoritmas, pateiktos eksperimentinių tyrimų vertinimo metodikos. Tyrimams atlikti pasirinktas duomenų rinkinys, kurį sudaro 4 klasės (rezistoriai, diodai, kondensatoriai, tranzistoriai), 7661 vnt. SMD komponentų nuotraukų, kuriose yra 28536 vnt. objektų žymės. Šias žymes sudaro 10467 vnt. rezistorių, 8543 vnt. diodų, 8198 vnt. kondensatorių ir 1328 vnt. tranzistorių. Objektams aptikti pasirinkti SSD-Mobilenet-V1, YOLOv3, YOLOv3-tiny, YOLOv4, YOLOv4-tiny, YOLOv4-Scaled, YOLOv5s detektoriai. Šie modeliai buvo mokyti su 3.1 lentelėje pateiktu trečiuoju duomenų rinkiniu ir buvo sudarytas elektronikos komponentų aptikimo ir skaičiavimo algoritmas tyrimams atlikti. Galiausiai pasirinktos eksperimentų vertinimo metrikos: vidutinis tikslumas, vidutinio tikslumo vidurkis, apdorotų kadrų per sekundę greitis ir išvedimo laikas.

4. ELEKTRONIKOS KOMPONENTŲ APTIKIMO IR SKAIČIAVIMO METODŲ TYRIMŲ REZULTATAI

Šiame skyriuje pateikti ir įvertinti elektronikos komponentų aptikimo ir skaičiavimo eksperimentinio tyrimo rezultatai naudojant skirtingus dirbtinio intelekto objektų atpažinimo detektorius. Taip pat palyginti gauti bendri objektų aptikimo ir skaičiavimo rezultatai. Eksperimentiniams tyrimams atlikti pasirinktas planas, kuris vykdomas tokia eiga:

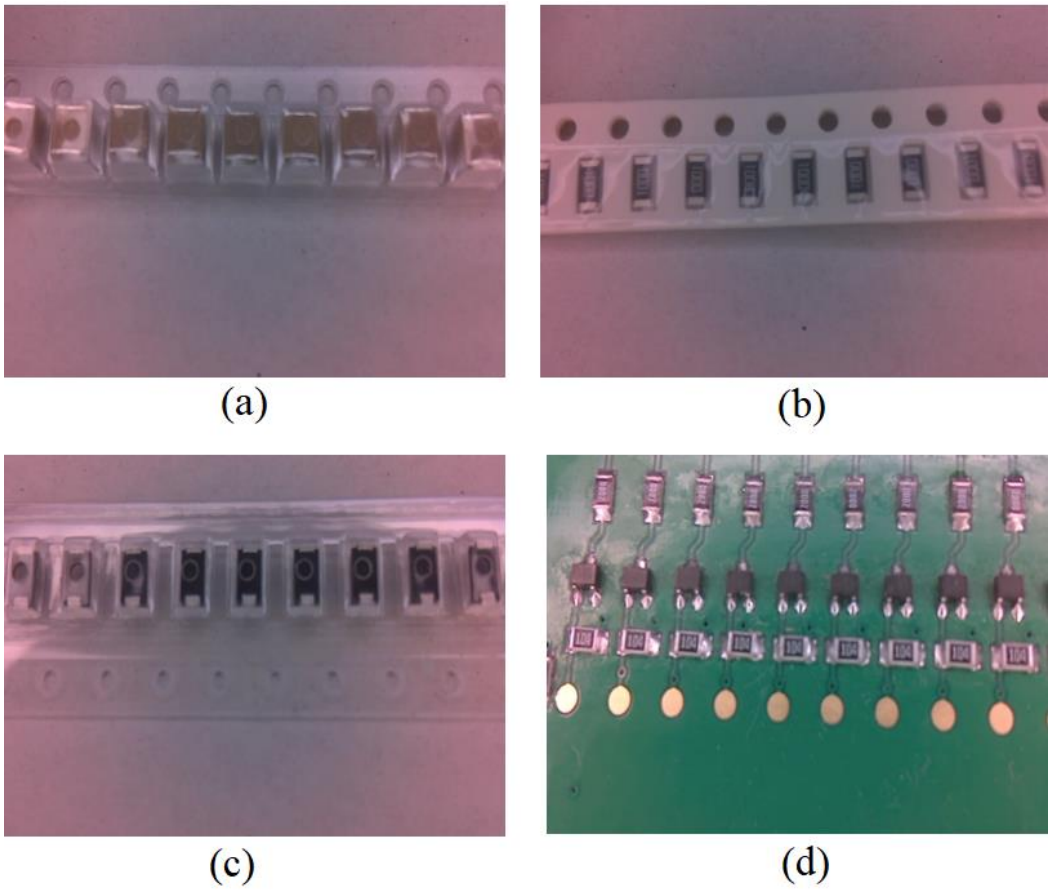
- tinklų objektų aptikimo efektyvumo tyrimas;
- tinklų objektų skaičiavimo efektyvumo tyrimas;
- objektų aptikimo ir skaičiavimo algoritmų lyginamoji analizė.

Toliau pateikti elektronikos komponentų aptikimo ir skaičiavimo metodų tyrimų rezultatai. Atliktų tyrimų eigos rezultatai pateikti atskiruose poskyriuose.

4.1. Tinklų objektų aptikimo efektyvumo tyrimo rezultatai

Šio tyrimo tikslas yra išmatuoti ir įvertinti skirtingų dirbtinio intelekto objektų atpažinimo detektorių efektyvumą, kai yra tik atliekamas objektų atpažinimas be elektronikos komponentų skaičiavimo. Tyrimas atliktas norint išsiaiškinti galimybes aptikti smulkius ir mažai išsiskiriančius elektronikos komponentus, peržvelgti detektorių aptikimo ir pritaikymo galimybes, kai komponentai slenka įvairiu greičiu. Tyrimo metu atlikta detektorių SSD-Mobilenet-V1, YOLOv3, YOLOv3-tiny, YOLOv4, YOLOv4-tiny, YOLOv4-Scaled, YOLOv5s lyginamoji analizė, įvertintas gautas aptikimo tikslumas (AP), vaizdo apdorojimo sparta (FPS) ir išvedimo laikas (IT).

Norint kuo tiksliau eksperimentiškai ištirti ir palyginti modelius tarpusavyje, tyrimams panaudoti keturi vaizdo įrašai, filmuoti su antrame skyriuje apžvelgta *Raspberry Pi NoIR v2* kamera. Vaizdo įrašuose matomi ant konvejerio slenkantys įpakuoti elektronikos SMD komponentai, esant 6300 lux apšvietimui, vaizdo įrašų kadrai pateikti 4.1 paveiksle. Pirmajame vaizdo įraše matomi rezistoriai, antrajame kondensatoriai, trečiajame diodai, o ketvirtajame rezistoriai, kondensatoriai, diodai ir tranzistoriai vienu metu. Pirmuose trijuose vaizdo įrašuose komponentai yra įpakuoti, o paskutiniame išpakuoti ir prilituoti prie spausdintinės plokštės. Paskutiniame vaizdo įraše taip pat yra pašalinių komponentų ir objektų. Vaizdo įrašas buvo filmuotas pasirinkus 720p30 režimą. Filmuotoje medžiagoje telpa nuo 1 iki 10 komponentų eilutėje, esant kelioms juostoms atitinkamai daugėja komponentų. Tyrimams atlikti naudota *Google Colab* debesijos paslauga su *Tesla P100-16GB* GPU ir mikrokompiuteris *Nvidia Jetson Nano 4GB*. Programos algoritmas įgyvendintas su trečiame skyriuje pateiktais giliojo mokymosi algoritmų detektoriais, rezultatams saugoti ir pavaizduoti naudotos *CV2*, *Numpy* bibliotekos.



4.1 pav. Tyrimams naudotų vaizdo įrašų kadrai: a) kondensatoriai; b) rezistoriai; c) diodai; d) visos komponentų klasės (rezistoriai, kondensatoriai, diodai ir tranzistoriai)

Šio eksperimentinio tyrimo metu objektų aptikimo detektorių nustatyti parametrai buvo: objekto pasitikėjimo slenkstinis koeficientas – 0,5, IoU NMS slenkstinis koeficientas – 0,75, įrenginys GPU, jau apmokyti tinklų svoriai panaudoti iš 3.2 lentelės. Pasirinktoje techninėje įrangoje įdiegus giliojo mokymosi detektorius atliktas eksperimentinis tyrimas, kai aptinkami slenkantys kondensatoriai virš konvejerio (pirmas vaizdo įrašas). Rezultatai pateikti 4.1 lentelėje.

Rezultatai rodo, kad geriausias AP įvertis buvo gautas naudojant YOLOv4-tiny detektorių – 97,3 %. Lyginant gautus vidutinio tikslumo rezultatus tarp abiejų aparatinės įrangos sąrankų, SSD-Mobilenet-V1, YOLOv3, YOLOv3-tiny, YOLOv4, YOLOv4-tiny detektorių architektūrų skirtumas buvo mažesnis nei 0,5 %. *Tesla P100* GPU, palyginus su *Jetson Nano*, parodė 1,1 % aukštesnį AP su YOLOv4-Scaled modeliu ir 3,3 % AP su YOLOv5s detektoriumi. Lyginant įrangų spartą apdorojant gautus vaizdus su detektoriais, išvedimo laikas skyrėsi nuo 2 iki 50 ms, priklausomai nuo tiriamo algoritmo. Mažiausi išvedimo laiko (IT) rezultatai – 10,4 ms YOLOv4-tiny detektoriaus naudojant *Tesla P100* GPU ir 23,8 ms SSD-Mobilenet-V1 modelio naudojant *Jetson Nano* mikrokompiuterį. Daugiausiai kadrų per sekundę (FPS) pavyko apdoroti YOLOv4-tiny detektoriui naudojant *Tesla P100* vaizdo plokštę (96 FPS) ir *Jetson Nano* mikrokompiuteryje su SSD-Mobilenet-V1 modeliu (42 FPS).

4.1 lentelė. Kondensatorių aptikimo tyrimo rezultatai

Eil. nr.	Modelis	Tesla P100-16GB			Jetson Nano 4GB		
		AP, %	**FPS**	**IT**, ms	**AP**, %	**FPS**	**IT**, ms
1.	SSD-Mobilenet-V1	66,7	**89**	**11,2**	66,6	**42**	**23,8**
2.	YOLOv3	90,2	14	71,4	90,1	1	1000
3.	YOLOv3-tiny	83,1	19	52,6	82,7	4	250
4.	YOLOv4	94,4	49	20,4	94,6	1	1000
5.	YOLOv4-tiny	**97,3**	**96**	**10,4**	**97,3**	**18**	**55,6**
6.	YOLOv4-Scaled	87,8	44	22,7	86,7	4	250
7.	YOLOv5s	88,6	**89**	**11,2**	85,3	8	125

Kitame etape buvo atliktas eksperimentinis tyrimas su antruoju vaizdo įrašu, aptinkami slenkantys rezistoriai ant konvejerio naudojant minėtas aparatinės įrangos sąrankas. Rezultatai pateikti 4.2 lentelėje. Iš gautų rezultatų matyti, jog aukščiausi vidutinio tikslumo rezultatai buvo gauti 98,6 % su YOLOv4 detektoriumi naudojant *Jetson Nano* mikrokompiuterį ir 98,5 % AP su YOLOv4-tiny modeliu naudojant *Tesla P100* vaizdo plokštę. Palyginus vidutinio tikslumo skirtumą tarp dviejų aparatinės įrangos sąrankų su SSD-Mobilenet-V1, YOLOv3, YOLOv3-tiny, YOLOv4, YOLOv4-tiny ir YOLOv4-Scaled detektoriais AP skirtumas buvo mažesnis nei 1 %. Lyginant gautus rezultatus, YOLOv5s modelis parodė 1,9 % didesnį AP naudojant *Tesla P100* nei *Jetson Nano*. Mažiausi išvedimo laiko (IT) rezultatai siekė 10,9 ms su YOLOv4-tiny detektoriumi naudojant *Tesla P100* GPU ir 24,4 ms su SSD-Mobilenet-V1 modeliu naudojant *Jetson Nano* įrenginį. Didžiausi kadrų per sekundę (FPS) greičiai gauti 92 FPS su YOLOv4-tiny detektoriumi naudojant *Tesla P100* vaizdo plokštę ir 41 FPS su SSD-Mobilenet-V1 detektoriumi naudojant *Jetson Nano* mikrokompiuterį.

4.2 lentelė. Rezistorių aptikimo tyrimo rezultatai

Eil. nr.	Modelis	Tesla P100-16GB			Jetson Nano 4GB		
		AP, %	**FPS**	**IT**, ms	**AP**, %	**FPS**	**IT**, ms
1.	SSD-Mobilenet-V1	75	**89**	**11,2**	75,1	**41**	**24,4**
2.	YOLOv3	92,7	15	66,7	92,7	1	1000
3.	YOLOv3-tiny	76,3	18	55,6	76,1	6	166,7
4.	YOLOv4	97,7	48	20,8	**98,6**	1	1000
5.	YOLOv4-tiny	**98,5**	**92**	**10,9**	98,5	**18**	**55,6**
6.	YOLOv4-Scaled	89,1	44	22,7	89,1	2	500
7.	YOLOv5s	92,9	70	14,3	91	8	125

Siūlomoje aparatinėje įrangoje įdiegtų giliojo mokymosi modelių eksperimentiniai rezultatai, kai aptinkami judantys įpakuoti diodai ant konvejerio juostos, pateikti 4.3 lentelėje. Aukščiausi vidutinio tikslumo (AP) rezultatai buvo gauti su YOLOv3 detektoriumi – 88,6% naudojant *Tesla P100* GPU ir 88,7% naudojant *Jetson Nano* mikrokompiuterį. Tarp dviejų naudotų įrenginių su SSD-Mobilenet-V1, YOLOv3, YOLOv3-tiny, YOLOv4-tiny ir YOLOv4-Scaled modeliais buvo gautas mažesnis nei 1 % AP skirtumas. Lyginant *Tesla P100-16GB* rezultatus su *Jetson Nano*, gautas 19,7% mažesnis AP su YOLOv4 detektoriumi ir 1,6% aukštesnis AP su YOLOv5s modeliu. Priklausomai nuo tiriamo detektoriaus, išvedimo laikas (IT) skyrėsi nuo 2 iki 50 kartų. Mažiausi IT buvo nustatyti – 8,2 ms su YOLOv4-tiny detektoriumi naudojant *Tesla P100-16GB* GPU ir 24,4 ms su SSD-Mobilenet-V1 algoritmu naudojant *Jetson Nano* (AP siekė 67,2 %). Didžiausi greičiai apskaičiavus apdorojamus kadrus per sekundę (FPS) buvo nustatyti – 122 FPS su YOLOv4-tiny detektoriumi naudojant *Tesla P100-16GB* GPU *Google Colab* aplinkoje ir 41 FPS su SSD-Mobilenet-V1 modeliu naudojant *Jetson Nano* mikrokompiuterį.

4.3 lentelė. Diodų aptikimo tyrimo rezultatai

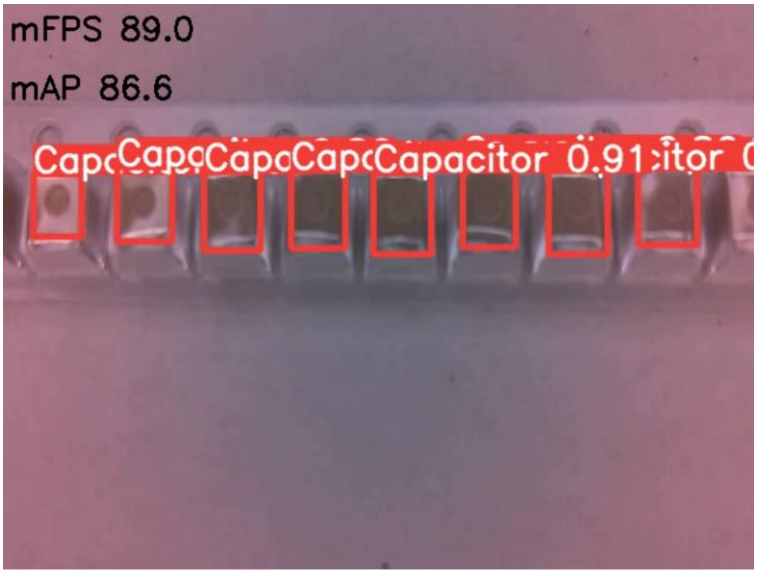
Eil. nr.	**Modelis**	**Tesla P100-16GB**			**Jetson Nano 4GB**		
		AP, %	**FPS**	**IT**, ms	**AP**, %	**FPS**	**IT**, ms
1.	SSD-Mobilenet-V1	67,3	**92**	**10,9**	67,2	**41**	**24,4**
2.	YOLOv3	**88,6**	16	62,5	**88,7**	1	1000
3.	YOLOv3-tiny	65,3	21	47,6	65,2	6	166,7
4.	YOLOv4	66,2	49	20,4	85,9	1	1000
5.	YOLOv4-tiny	76,2	**122**	**8,2**	76,5	**18**	**55,6**
6.	YOLOv4-Scaled	70,7	46	21,7	70	2	500
7.	YOLOv5s	82,8	79	12,7	81,2	8	125

Galiausiai tyrimas su pasiūlyta aparatine įranga, kurioje įdiegti giliojo mokymosi detektoriai, atliktas aptinkant visas keturias komponentų rūšis vienu metu. Rezultatai pateikti 4.4 lentelėje. Iš gautų rezultatų matyti, kad aukščiausi AP buvo gauti 97,2 % su YOLOv4 detektoriumi naudojant *Jetson Nano* mikrokompiuterį, o naudojant *Tesla P100-16GB* GPU gautas 1,8 % mažesnis aptikimo vidutinis tikslumas – 95,4 %. Toks pat aptikimo vidutinio tikslumo skirtumas buvo gautas su YOLOv5s detektoriumi – naudojant *Jetson Nano* (70,6 %) AP buvo didesnis nei naudojant *Tesla P100* (68,8 %). AP skirtumai tarp SSD-Mobilenet-V1, YOLOv3, YOLOv3-tiny, YOLOv4-tiny, YOLOv4-Scaled detektorių įdiegtuose įrenginiuose buvo mažesni nei 0,1 %. Greičiausi išvedimo laiko (IT) rezultatai buvo gauti naudojant *Tesla P100-16GB* – 11 ms su YOLOv3 detektoriumi ir vos 0,2 ms ilgesnis IT su SSD-Mobilenet-V1 modeliu. *Jetson Nano* mikrokompiuteris pasiekė 27,8 ms

išvedimo laiką su SSD-Mobilenet-V1 ir 58,8 ms su YOLOv4-tiny modeliais. Aptinkant visas keturias elektronikos komponentų rūšis sparčiausiai apdoroti ir pasiekti 91 FPS pavyko su YOLOv5s modeliu naudojant *Tesla P100-16GB* GPU ir 36 FPS su SSD-Mobilenet-V1 detektoriumi naudojant *Jetson Nano 4GB*. Iš visų aptikimo modelių tik su YOLOv4-tiny kiti objektai buvo identifikuojami kaip viena iš mokytų klasių, pvz., 4.1(a) paveiksle matomus vario apskritimus detektorius kartais atpažindavo kaip kondensatorius. Klaidingo aptikimo priežastį lėmė, jog kondensatorius turi šviesiai rudos arba keramikinės spalvos pagrindą, tačiau šonuose yra sidabrinės spalvos sienelės, kurių klaidingai aptiktì objektai iš tiesų neturėjo (aplink buvo žalias fonas).

4.4 lentelė. Visų komponentų aptikimo tyrimo rezultatai

Eil. nr.	**Modelis**	**Tesla P100-16GB**			**Jetson Nano 4GB**		
		AP, %	**FPS**	**IT**, ms	**AP**, %	**FPS**	**IT**, ms
1.	SSD-Mobilenet-V1	60,6	89	11,2	60,5	**36**	**27,8**
2.	YOLOv3	66,9	15	66,7	66,9	1	1000
3.	YOLOv3-tiny	63,5	19	52,6	63,5	6	166,7
4.	YOLOv4	**95,4**	46	21,7	**97,2**	1	1000
5.	YOLOv4-tiny	79,9	65	15,4	79,8	**17**	**58,8**
6.	YOLOv4-Scaled	69,6	41	24,4	69,5	2	500
7.	YOLOv5s	68,8	**91**	**11**	70,6	8	125



4.2 pav. Kadras iš kondensatorių atpažinimo tyrimo naudojant YOLOv5s detektorių

Remiantis objektų aptikimo tyrimo rezultatais lentelėse, galima teigti, jog gauti objektų atpažinimo vidutinio tikslumo (AP), apdorojamų kadrų per sekundę (FPS) ir išvedimo laiko (IT)

rezultatai tarp apmokytų tuo pačiu duomenų rinkiniu detektorių svyruoja. 4.2 paveiksle pateikta tinklo objektų aptikimo tyrimo eigos ištrauka, pavaizduotas kadras iš kondensatorių atpažinimo eigos su YOLOv5s modeliu naudojant *Tesla P100-16GB* GPU. Visiems tirtiems detektoriams pavyko aptikti kadruose rodomus elektronikos komponentus, bet pastebėta, kad su SSD-Mobilenet-V1 ir YOLOv3-tiny modeliais poroje kadrų neatpažindavo kraštuose esančių elektronikos komponentų dėl judėjimo greičio pasikeitimo (objektams judant stabiliai neaptiktų komponentų kadre nelikdavo). Pastebėtas aptikimo tikslumo sumažėjimas atliekant diodų ir visų komponentų tyrimą, tai lėmė mažesnis diodų ir tranzistorių duomenų, skirtų mokymams, kiekis. Tačiau pasirinkus optimalų modelių mokymų epochų skaičių – 100, gauti svoriai buvo pakankami bandymams, skirtiems SMD elektronikos komponentų klasėms aptikti. Taip pat pastebėta, kad objektų aptikimo spartai įterptinėje sistemoje *Jetson Nano 4GB* turėjo įtakos modelių dydis ir naudotos giliojo mokymosi bibliotekos, kurios buvo mažiau tinkamos pasirinktam mikrokompiuteriui.

4.2. Tinklų objektų skaičiavimo efektyvumo tyrimo rezultatai

Kitame etape, siekiant įgyvendinti elektronikos komponentų ant konvejerio skaičiavimą, buvo atliktas mokytų modelių tinklų objektų skaičiavimo efektyvumo tyrimas. Tyrimo metu atliktas objektų atpažinimas ir atpažintų elektronikos komponentų skaičiavimas, jiems pasiekus nustatytą koordinačių plokštumą. Skaičiavimo algoritmas įgyvendintas pagal 3.5 paveikslą. Tyrimo metu atlikta detektorių SSD-Mobilenet-V1, YOLOv3, YOLOv3-tiny, YOLOv4, YOLOv4-tiny, YOLOv4-Scaled, YOLOv5s lyginamoji analizė, buvo įvertintas suskaičiuotų komponentų kiekis, gaunamas aptikimo tikslumas (AP), vaizdo apdorojimo sparta (FPS) ir išvedimo laikas (IT). AP, FPS ir IT parametrai pasirinkti norint įvertinti, ar įdiegtas elektronikos komponentų skaičiavimo algoritmas daro įtaką modelių veikimo efektyvumui. Šis tyrimas atliktas norint išsiaiškinti dirbtinio intelekto pritaikymo galimybes pramonėje atliekant automatizuotą elektronikos komponentų atpažinimą ir skaičiavimą.

Norint eksperimentiškai ištirti ir įvertinti elektronikos komponentų skaičiavimo algoritmo efektyvumą, pasitelkti praeitame tyrime naudoti keturi vaizdo įrašai. Tyrimams atlikti pasirinkta naudoti *Tesla P100-16GB* GPU ir mikrokompiuterį *Nvidia Jetson Nano 4GB*, programos algoritmas įgyvendintas su trečiame skyriuje pateiktais giliojo mokymosi architektūrų modeliais, rezultatams saugoti ir vaizduoti panaudotos *CV2*, *Numpy* bibliotekos. Skaičiavimo algoritmui įgyventi atitinkamai naudotos *PyTorch*, *Keras*, *Darknet* bibliotekos, kad būtų nustatyti aptiktų objektų pozicijų parametrai.

Šio eksperimentinio tyrimo metu pasirinkti objektų aptikimo detektorių parametrai buvo: objekto pasitikėjimo slenkstinis koeficientas – 0,5, IoU NMS slenkstinis koeficientas – 0,75, įrenginys GPU, jau apmokyti tinklų svoriai panaudoti iš 3.2 lentelės. Tyrimo eigoje buvo nustatomi skirtingi aptikimo

juostos pločiai norint išvengti komponentų nesuskaičiavimo dėl jų dydžio arba komponentui palikus skaičiavimo ribas. Koordinačių vertės buvo priskirtos skirtingiems algoritmams ir naudotos vienodos abiem tyrimui naudojamiems įrenginiams.

Pirmiausia buvo atliekamas kondensatorių skaičiavimo tyrimas naudojant minėtą aparatinę įrangą, kurioje įdiegti apmokyti objektų aptikimo ir skaičiavimo algoritmai, gauti rezultatai pateikti 4.5 lentelėje. Matyti, kad daugiausiai komponentų pavyko suskaičiuoti su YOLOv5s detektoriumi – suskaičiuoti 137 vnt. naudojant *Tesla P100-16GB* GPU ir 133 vnt. naudojant *Jetson Nano 4GB* mikrokompiuterį. Aukščiausi vidutinio tikslumo (AP) įvertinimai buvo pasiekti – 97,3 % su YOLOv4-tiny detektoriumi naudojant *Tesla P100-16GB* ir tiek pat naudojant *Jetson Nano 4GB*. YOLOv4-tiny išvedimo laikas (IT) siekė 10,6 ms (94 FPS) naudojant *Tesla P100-16GB* ir 62,5 s (16 FPS) naudojant *Jetson Nano*. Bendrai suskaičiuotų komponentų kiekis tarp įrenginių skyrėsi nuo 1 iki 14 vnt., didžiausi skirtumai gauti su YOLOv3 ir YOLOv4 modeliais. Mažiausią įtaką skaičiavimui naudojant skirtingus įrenginius turėjo SSD-Mobilenet-V1, YOLOv4-tiny, YOLOv4-Scaled ir YOLOv5s modeliai. Blogiausiai komponentus skaičiuoti pavyko su YOLOv3 ir YOLOv3-tiny detektoriais, naudojant *Jetson Nano* mikrokompiuterį – suskaičiuoti 34 ir 35 vnt. kondensatorių. Remiantis gauta skaičiavimo paklaida, nuspręsta, jog YOLOv3 ir YOLOv3-tiny modeliams buvo sunkiau nustatyti ir fiksuoti judančių kondensatorių poziciją su *Keras* biblioteka. Rezultatuose matomas objektų aptikimo susikirtimas, t. y. vienu metu buvo du kartus fiksuojamas vieno objekto aptikimas ir tai neleido tinkamai įvertinti, kuomet objektas paliko skaičiavimo plokštumą.

4.5 lentelė. Kondensatorių skaičiavimo tyrimo rezultatai

Eil. nr.	**Modelis**	**Tesla P100-16GB**				**Jetson Nano 4GB**			
		Suskaičiuota, vnt.	**AP**, %	**FPS**	**IT**, ms	**Suskaičiuota**, vnt.	**AP**, %	**FPS**	**IT**, ms
1.	SSD-Mobilenet-V1	107	66,7	88	11,4	111	66,6	**39**	**25,6**
2.	YOLOv3	46	90,2	12	83,3	34	90,1	1	1000
3.	YOLOv3-tiny	34	83,1	17	58,8	35	82,7	4	250
4.	YOLOv4	106	94,3	47	21,3	95	94,6	1	1000
5.	YOLOv4-tiny	96	**97,3**	**94**	**10,6**	94	**97,3**	**16**	**62,5**
6.	YOLOv4-Scaled	112	87,8	42	23,8	111	86,7	1	1000
7.	YOLOv5s	**137**	85,2	87	11,5	**133**	85,3	6	166,7

Kitame etape buvo atliktas įpakuotų rezistorių ritėje skaičiavimas, rezultatai pateikti 4.6 lentelėje. Iš atlikto tyrimo matyti, kad naudojant *Jetson Nano 4GB* mikrokompiuterį su YOLOv3-tiny modeliu objektams aptikti ir skaičiuoti pavyko suskaičiuoti 88 vnt. komponentų išlaikant 76,1 % vidutinį tikslumą ir esant 250 ms išvedimo laikui (IT). Artimiausią kiekį rezistorių pavyko suskaičiuoti su YOLOv4-tiny detektoriumi – gauti 77 vnt. naudojant *Tesla P100-16GB* GPU ir 74 vnt. naudojant

Jetson Nano. Su abiem sistemomis pavyko pasiekti 98,5 % AP. Greičiausiai objektų aptikimas apdorotas su YOLOv4-tiny detektoriumi naudojant *Tesla P100-16GB* GPU – 90 FPS ir šiek tiek lėtesnis kadrų per sekundę apdorojimas – 87 FPS – gautas su SSD-Mobilenet-V1 modeliu. Kita vertus, naudojant įterptinę sistemą *Jetson Nano* gautas greičiausias 38 FPS apdorojimo greitis su SSD-Mobilenet-V1 detektoriumi ir 16 FPS su YOLOv4-tiny modeliu. Su YOLOv4, YOLOv4-Scaled ir YOLOv5s modeliais buvo gautas vienodas apskaičiuotų elektronikos komponentų kiekis – 64 vnt., tik naudojant *Jetson Nano* rezistorių suskaičiuota 68 vnt. Gauti rezultatai rodo, kad įpakuotus rezistorius pavyko lengviau skaičiuoti dėl jų išsidėstymo atstumo, tad esant greitesniam konvejerio greičiui algoritmas sugebėjo lengvai užfiksuoti, kuomet komponentas suskaičiuotas ir palieka savo poziciją. Taip pat skaičiuojant komponentus su YOLOv3 ir YOLOv4 modeliais ir esant 1 FPS spartai, gautas mažesnis skaičiavimo nuostolis, lyginant su kondensatoriais, kurių įpakavimo atstumas buvo kur kas mažesnis.

4.6 lentelė. Rezistorių skaičiavimo tyrimo rezultatai

Eil. nr.	**Modelis**	**Tesla P100-16GB**				**Jetson Nano 4GB**			
		Suskaičiuota, vnt.	**AP**, %	**FPS**	**IT**, ms	**Suskaičiuota**, vnt.	**AP**, %	**FPS**	**IT**, ms
1.	SSD-Mobilenet-V1	64	75	87	11,5	61	75,1	**38**	**26,3**
2.	YOLOv3	74	92,7	13	76,9	48	92,7	1	1000
3.	YOLOv3-tiny	**77**	76,3	16	62,5	**88**	76,1	4	250
4.	YOLOv4	64	97,7	46	21,7	68	**98,6**	1	1000
5.	YOLOv4-tiny	**77**	**98,5**	**90**	**11,1**	74	98,5	**16**	**62,5**
6.	YOLOv4-Scaled	64	89,1	42	23,8	64	89,1	1	1000
7.	YOLOv5s	64	91,1	66	15,2	64	91	6	166,7

Atlikus rezistorių aptikimo ir skaičiavimo bandymus, buvo pereita prie įpakuotų diodų ritėje aptikimo ir skaičiavimo tyrimo, gauti rezultatai pateikti 4.7 lentelėje. Iš gautų rezultatų matyti, jog daugiausiai diodų suskaičiuota ir tikrasis praslinkusių komponentų skaičius buvo viršytas su YOLOv4-tiny algoritmu – 113 vnt. naudojant *Tesla P100-16GB* GPU ir 126 vnt. naudojant *Jetson Nano 4GB*. Šios paklaidos priežastis – vienu metu vietoje vieno komponento identifikuojami ir suskaičiuojami keli komponentai. Tikėtina, kad iškraipymų atsiradimą lėmė pasirinktas mažesnis įpakuotų diodų nuotraukų kiekis mokymams. Taip pat su minėtu detektoriumi buvo pasiektas greičiausias vaizdo apdorojimo greitis – 120 FPS ir 8,3 ms IT diodų skaičiavimo bandymo metu naudojant *P100-16GB* vaizdo plokštę, o geriausias vidutinio tikslumo įvertinimas gautas 88,6 % su YOLOv3 modeliu. Su YOLOv3-tiny detektoriumi naudojant abu įrenginius suskaičiuota 84 vnt.

įpakuotų diodų ritėje. Naudojant *Jetson Nano* mikrokompiuterį aukščiausias AP įvertis gautas su YOLOv3 algoritmu, o greičiausiai apdoroti kadrai per sekundę su SSD-Mobilenet-V1 modeliu – pasiekti 38 FPS ir 67,2 % AP. Antroje vietoje pagal vaizdo medžiagos apdorojamo greitį YOLOv4-tiny – 16 FPS naudojant *Jetson Nano*.

4.7 lentelė. Diodų skaičiavimo tyrimo rezultatai

Eil. nr.	**Modelis**	**Tesla P100-16GB**				**Jetson Nano 4GB**			
		Suskaičiuota, vnt.	**AP**, %	**FPS**	**IT**, ms	**Suskaičiuota**, vnt.	**AP**, %	**FPS**	**IT**, ms
1.	SSD-Mobilenet-V1	68	67,3	90	11,1	52	67,2	**38**	**26,3**
2.	YOLOv3	63	**88,6**	13	76,9	58	**88,7**	1	1000
3.	YOLOv3-tiny	**84**	65,3	19	52,6	**84**	65,2	4	250
4.	YOLOv4	56	66,2	47	21,3	83	85,9	1	1000
5.	YOLOv4-tiny	**113**	76,2	**120**	**8,3**	**126**	76,5	**16**	**62,5**
6.	YOLOv4-Scaled	76	70,7	44	22,7	79	70	1	1000
7.	YOLOv5s	65	81,3	77	13	66	81,2	6	166,7

Paskutiniu etapu buvo atliekamas objektų skaičiavimo tyrimas su visų komponentų rūšių vaizdo medžiaga, kurioje vienu metu slenka prie spausdintos montavimo plokštės prilituoti kondensatoriai, rezistoriai, diodai ir tranzistoriai. Šiuo etapu buvo norima išbandyti skaičiavimo algoritmo gebėjimą aptikti prilituotus komponentus, kai šalia jų atsiranda minimalus kiekis lydmetalio. Remiantis tyrime gautais rezultatais, pastebėta, jog mažiausią įtaką minėti trikdžiai padarė YOLOv3, YOLOv4, YOLOv4-Scaled detektoriams. Daugiausiai komponentų pavyko suskaičiuoti naudojant *Jetson Nano* mikrokompiuterį – 86 vnt. su YOLOv3 detektoriumi, pasiektas 66,9 % AP, bet gautas per žemas apdorojimo greitis – 1 s. Aukščiausias vidutinis aptikimo tikslumas gautas su YOLOv4 modeliu naudojant *Jetson Nano* – 97,2 % AP ir naudojant *Tesla P100* – 95,4 % AP. Didžiausias apdorojimo greitis – 87 FPS – pasiektas naudojant *Tesla P100* su SSD-Mobilenet-V1 modeliu. Su YOLOv5s modeliu naudojant *Tesla P100-16GB* GPU pasiekti 89 FPS ir 70,6 % AP. Šiame etape nebuvo susidurta su trikdžiais dėl komponentų išsidėstymo atstumo, taip pat šiuo bandymu patikrinta, ar sistema nesuskaičiuoja vienu metu kelių komponentų.

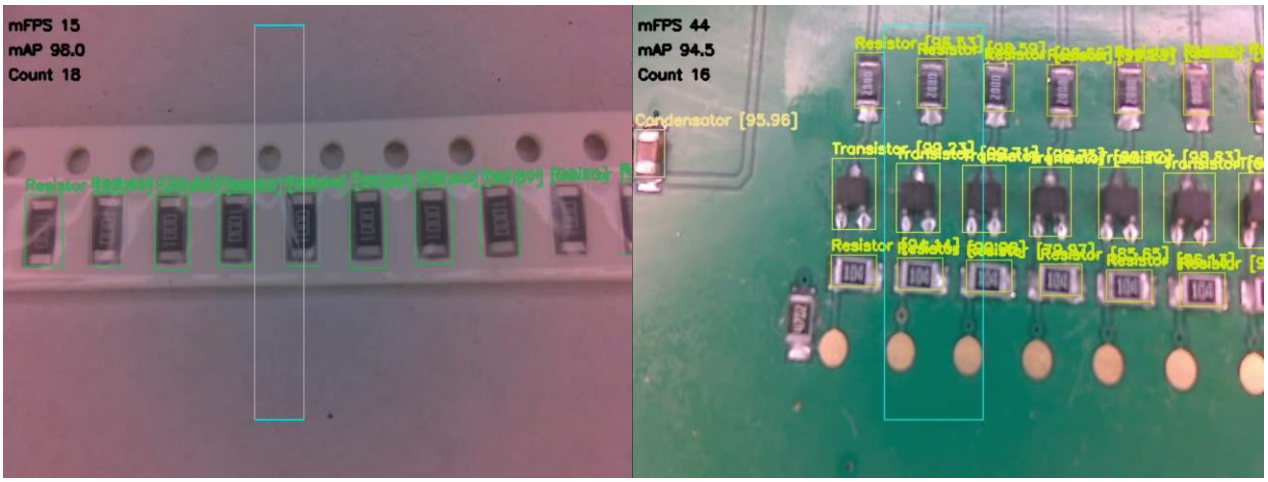
Pasiteisino įdiegtas skaičiavimo kontrolės metodas, kuris suteikia leidimą prie bendros sumos pridėti vienetą, tik tada, kai skaičiavimo perimetre suskaičiuoti komponentai šį plotą palieka. Norint, kad algoritmas skaičiuotų visas komponentų eiles, reikėtų įdiegti visų komponentų, kurie yra pažymėtame plote, skaičiavimą ir atlikti daugiau tikrinimų, ar bent vienas komponentas paliko skaičiavimo zoną. Ankstesniame tyrime pastebėta klaida, kad su YOLOv4-tiny modeliu vario

apskritimai vaizde kartais būdavo aptinkami kaip kondensatoriai, nepadarė įtakos skaičiavimo algoritmui ir nepridėjo papildomai suskaičiuotų vienetų.

4.8 lentelė. Visų komponentų skaičiavimo tyrimo rezultatai

Eil. nr.	Modelis	Tesla P100-16GB				Jetson Nano 4GB			
		Suskaičiuota, vnt.	**AP**, %	**FPS**	**IT**, ms	**Suskaičiuota**, vnt.	**AP**, %	**FPS**	**IT**, ms
1.	SSD-Mobilenet-V1	40	60,6	87	11,5	41	60,5	**33**	**30,3**
2.	YOLOv3	31	66,9	10	100	**86**	66,9	1	1000
3.	YOLOv3-tiny	23	63,5	17	58,8	24	63,5	4	250
4.	YOLOv4	49	**95,4**	43	23,3	54	**97,2**	1	1000
5.	YOLOv4-tiny	**51**	79,9	63	15,9	48	79,8	**15**	**66,7**
6.	YOLOv4-Scaled	47	69,6	39	25,6	45	69,5	1	1000
7.	YOLOv5s	42	70,6	**89**	**11,2**	42	70,6	6	166,7

Atlikus visus tinklo objektų skaičiavimo tyrimo etapus ir gavus rezultatus matyti, kad bandymų su kondensatoriais ir visais komponentais metu mažiausiai suskaičiuota su YOLOv3 ir YOLOv3-tiny detektoriais. Mažiausią paklaidą skaičiuojant išlaikė SSD-Mobilenet-V1, YOLOv4, YOLOv4-Scaled ir YOLOv5s modeliai. Elektronikos komponentų skaičiavimas, kai buvo atliekamas rezistorių skaičiavimas su YOLOv4-tiny detektoriumi naudojant *Jetson Nano* mikrokompiuterį, pavaizduotas 4.3 paveiksle.



4.3 pav. Kadras iš rezistorių aptikimo ir skaičiavimo tyrimo su YOLOv4-tiny ir YOLOv4 detektoriais

Paveiksle pavaizduoti tarpiniai gauti tarpiniai mFPS, mAP ir skaičiavimo (*Count*) rezultatai. Taip pat siekiant lengvesnio skaičiavimo kalibravimo yra nupiešti skaičiavimo ploto kraštai. Tiriant diodų skaičiavimą su YOLOv4-tiny detektoriumi buvo gauta paklaida, skaičiavimo perviršis, kai dėl

mažesnio įpakuotų diodų nuotraukų skaičiaus mokymams detektorius kartais aptikdavo vieną komponentą kaip kelis. Šiuos trikdžius turėtų pašalinti didesnis mokymo duomenų kiekis esant tiriamoms vaizdo sąlygoms ir didesnis mokymų epochų skaičius. Išlaikant tyrimo metu naudotas komponentams priskirtas skaičiavimo koordinates pavyko gauti vienodus arba mažą paklaidą turinčius skaičiavimo rezultatus.

4.3. Objektų aptikimo ir skaičiavimo algoritmų lyginamosios analizės rezultatai

Šiame poskyryje atliekama objektų aptikimo ir skaičiavimo algoritmų efektyvumo rezultatų lyginamoji analizė. Rezultatams įvertinti naudojamas vidutinio aptikimo tikslumo vidurkio įvertis mAP (angl. *mean Average Precision*), vidutinis apdorojamų kadrų per sekundę greitis mFPS (angl. *mean Frames Per Second*), vidutinis išvedimo laikas mIT (angl. *mean Inference Time*) ir suskaičiuotų komponentų vienetų skaičius. Tikimasi naudojant atliktų tyrimų bendrą rezultatų vidurkį išnagrinėti skaičiavimo įdiegimo įtaką modelių efektyvumui. Pirmiausia apžvelgiami 4.9 lentelėje gauti bendri komponentų aptikimo tyrimo rezultatai, kuriems buvo panaudoti 4.1 poskyrio tinklo objektų aptikimo efektyvumo tyrimo rezultatai.

4.9 lentelė. Bendri komponentų aptikimo tyrimo rezultatai

Eil. nr.	**Modelis**	**Tesla P100-16GB**			**Jetson Nano 4GB**		
		mAP, %	**mFPS**	**mIT**, ms	**mAP**, %	**mFPS**	**mIT**, ms
1.	SSD-Mobilenet-V1	67,4	89,75	**11,1**	67,35	**40**	**25,1**
2.	YOLOv3	84,6	15	66,8	84,6	1	1000
3.	YOLOv3-tiny	72,05	19,25	52,1	71,88	5,5	187,5
4.	YOLOv4	**88,43**	48	20,8	**94,08**	1	1000
5.	YOLOv4-tiny	87,98	**93,75**	11,2	88,03	**17,75**	**56,4**
6.	YOLOv4-Scaled	79,3	43,75	22,9	78,83	2,5	437,5
7.	YOLOv5s	83,28	82,25	12,3	82,03	8	125

Aukščiausią vidutinio aptikimo tikslumo vidurkį pavyko gauti 94,08 % mAP su YOLOv4 modeliu naudojant *Jetson Nano* mikrokompiuterį, o naudojant *Tesla P100-16GB* GPU įvertis siekė 88,43 %. Sparčiausiai modelį pavyko apdoroti su YOLOv4-tiny detektoriumi – 93,75 mFPS ir 11,2 ms mIT naudojant *Tesla P100-16GB* aparatinę įrangą. Naudojant *Jetson Nano* mikrokompiuterį greičiausias vidutinis kadrų per sekundę greitis buvo gautas su SSD-Mobilenet-V1 modeliu – 40 mFPS ir 25,1 ms mIT. Greičiausio detektoriaus YOLOv4-tiny gautas aptikimo tikslumas naudojant *Tesla P100* GPU skyrėsi 0,45 % nuo aukščiausią tikslumą turinčio YOLOv4 modelio. Taip pat

lyginant *Jetson Nano* mikrokompiuterio greičiausio ir tiksliausio detektorių rezultatus gautas tikslumo skirtumas tarp YOLOv4 (94,08 % mAP) ir SSD-Mobilenet-V1 (67,35 % mAP) modelių siekė 26,73 %. Kitas greičiausias modelis naudojant *Jetson Nano* mikrokompiuterį buvo YOLOv4-tiny – 17,75 mFPS, 56,4 ms mIT bei turėjo aukštesnį 88,03 % mAP.

Toliau šiame poskyryje apžvelgiami 4.2 poskyryje atlikto tinklo objektų skaičiavimo tyrimo su kondensatoriais, rezistoriais, diodais ir visomis keturiomis klasėmis rezultatai. Apskaičiuoti bendri komponentų skaičiavimo tyrimo rezultatai pateikti 4.10 lentelėje. Daugiausiai komponentų pavyko suskaičiuoti su YOLOv4-tiny modeliu – 342 vnt. naudojant *Jetson Nano* aparatinę sąranką ir 337 vnt. naudojant *Tesla P100-16GB* vaizdo plokštę. Tačiau, kaip buvo pastebėta atliekant 4.2 tyrimą, atliekant skaičiavimą su YOLOv4-tiny susidurta su skaičiavimo perviršio paklaida. Kitas detektorius, su kuriuo tinkamai suskaičiuota daugiausiai elektronikos komponentų, yra YOLOv5s – naudojant *Tesla P100-16GB* GPU iš viso suskaičiuota 308 vnt. elektronikos komponentų ir naudojant *Jetson Nano* mikrokompiuterį – 305 vnt. Iš atlikto objektų skaičiavimo tyrimo aukščiausias mAP buvo gautas su YOLOv4-tiny modeliu – 94,1 % mAP naudojant *Jetson Nano* ir 88,4 % naudojant *Tesla P100-16GB*. Sparčiausias objektų aptikimo detektorius iš bendrų rezultatų yra YOLOv4-tiny. Naudojant *Tesla P100-16GB* aparatinę įrangą, detektorius pasiekė 91,75 mFPS ir 11,5 ms mIT spartą. Naudojant *Jetson Nano* mikrokompiuterį greičiausiai buvo apdorojamas objektų aptikimas, skaičiavimas ir rezultatų išvedimas, su SSD-Mobilenet-V1 modeliu gautas vidutinis 37 kadrų per sekundę apdorojimo greitis ir 27,1 ms vidutinis išvedimo laikas.

4.10 lentelė. Bendri komponentų skaičiavimo tyrimo rezultatai

Eil. nr.	**Modelis**	**Tesla P100-16GB**				**Jetson Nano 4GB**			
		Suskaičiuota, vnt.	**mAP**, %	**mFPS**	**mIT**, ms	**Suskaičiuota,** vnt.	**mAP**, %	**mFPS**	**mIT**, ms
1.	SSD-Mobilenet-V1	279	67,4	88	**11,4**	265	67,4	**37**	**27,1**
2.	YOLOv3	214	84,6	12	84,3	226	84,6	1	1000
3.	YOLOv3-tiny	218	72,05	17,25	58,2	231	71,9	4	250
4.	YOLOv4	275	**88,4**	45,75	21,9	300	**94,1**	1	1000
5.	YOLOv4-tiny	**337**	87,98	**91,75**	11,5	**342**	88	**15,8**	**63,6**
6.	YOLOv4-Scaled	299	79,3	41,75	24	299	78,8	1	1000
7.	YOLOv5s	**308**	82,05	79,75	12,7	**305**	82	6	166,7

Apskaičiavus ir apžvelgus bendrus objektų aptikimo ir skaičiavimo tyrimų rezultatus, galima įvertinti įdiegto skaičiavimo algoritmo poveikį modelių veikimo efektyvumui naudotose aparatinėse sąrankose. Vidutinio aptikimo tikslumo vidurkio (mAP) įvertis naudojant *Jetson Nano* mikrokompiuterį sumažėjo labai nežymiai – nuo 0,02 % iki 0,05 %. Naudojant *Tesla P100-16GB* GPU pastebėtas mAP sumažėjimas – 0,03 % su YOLOv4 ir 1,23 % su YOLOv5s modeliais. Svarbiausias parametras, kuriam įdiegtas algoritmas turėjo padaryti įtaką, yra modelių objektų aptikimo apdorojimo sparta. Atlikus detektorių spartos įvertinimą, pastebėtas mFPS sulėtėjimas – nuo 1,75 iki 3 mFPS tyrimuose naudojant *Tesla P100* aparatinę įrangą ir nuo 1,5 iki 3 mFPS sumažėjimas tyrimuose naudojant *Jetson Nano* įrenginį. Turint gautus tinklų apdorojimo trukmės įverčius, nuspręsta, kad tinkama sparta turėtų būti daugiau nei 15 FPS ir ne didesnis nei 70 ms IT. Kitu atveju apdorojimas būtų vykdomas per ilgai ir neatitiktų realiojo laiko sistemos reikalavimų. Tyrimuose naudojant *Jetson Nano* modeliai, kurių pirminiai FPS buvo 1, liko nepakitę. Didžiausias objektų aptikimo modelio sulėtėjimas įdiegus skaičiavimo algoritmą pastebėtas YOLOv3 detektoriui – sparta naudojant *Tesla P100* GPU sumažėjo 3 mFPS ir 17,5 ms mIT. Taip pat atliekant komponentų skaičiavimus su *Jetson Nano* užfiksuotas 3 mFPS sumažėjimas su SSD-Mobilenet-V1 ir 563 ms mIT padidėjimas su YOLOv4-Scaled detektoriais.

4.4. Skyriaus apibendrinimas

Šiame skyriuje atlikti elektronikos komponentų aptikimo ir skaičiavimo metodų tyrimai. Atliekant tinklo objektų aptikimo efektyvumo įvertinimą, apskaičiuoti mokytų detektorių atpažinimo vidutinio tikslumo (AP), apdorojamų kadrų per sekundę (FPS) ir išvedimo laiko (IT) rezultatai. Pasitelkiant tirtus detektorius pavyko aptikti kadruose esančius elektronikos komponentus. Didžiausi vidutinio aptikimo tikslumo (AP) rezultatai buvo gauti aptinkant kondensatorius ir rezistorius, nes naudotame duomenų rinkinyje tinklams mokyti šių įpakuotų komponentų nuotraukų buvo daugiausiai. Sparčiausiai apdoroti objektų aptikimą pavyko su YOLOv4-tiny modeliu naudojant *Tesla P100-16GB* GPU ir su SSD-Mobilenet-V1 detektoriumi naudojant *Jetson Nano* mikrokompiuterį. Kitame etape atlikti tinklo objektų skaičiavimo tyrimai, kai objektų skaičiavimas yra įdiegtas į objektų aptikimo detektorių. Tyrimo metu naudotos modeliams priskirtos skaičiavimo koordinatės, kad esant skirtingiems detektoriams ir skirtingų pločių komponentams pavyktų gauti vienodus arba mažą paklaidą turinčius skaičiavimo rezultatus. Mažiausia paklaida skaičiuojant gauta su SSD-Mobilenet-V1, YOLOv4, YOLOv4-Scaled ir YOLOv5s detektoriais. Norint įvertinti įdiegto skaičiavimo algoritmo poveikį modelių efektyvumui, buvo apskaičiuoti ir palyginti bendri objektų aptikimo ir skaičiavimo tyrimų rezultatai. Iš gautų mAP rezultatų matyti, jog įdiegtas skaičiavimo algoritmas visiems modeliams, iš kurių mažiausias spartos nuostolis patirtas su SSD-Mobilenet-V1 modeliu ir didžiausias su YOLOv3 detektoriumi.

APIBENDRINIMAS. IŠVADOS

Šiame darbe buvo analizuoti ir tirti smulkiems objektams atpažinti ir aptikti taikomi intelektualieji metodai. Pasirinkti ir pritaikyti objektų aptikimo metodai smulkių elektronikos komponentų ant konvejerio skaičiavimo sistemai įgyvendinti.

Atsižvelgiant į iškeltus darbo uždavinius, buvo atlikta mokslinės literatūros analitinė apžvalga, paruoštas duomenų rinkinys ir mokyti objektų aptikimo detektoriai. Toliau atlikti eksperimentiniai tyrimai, įvertintas skirtingų tinklų objektų aptikimo ir skaičiavimo efektyvumas. Iš gautų analitinių ir eksperimentinių rezultatų padarytos šios išvados:

1. Mokslinės literatūros apie neuroninių tinklų mokymus apžvalga parodė, kad norint pasiekti tinklo modelio aukštą aptikimo tikslumą svarbu atsižvelgti į mokymo duomenų paruošimą, apdorojimą, kokybę ir kiekį. Turint tinkamai sužymėtą duomenų rinkinį yra sumažinama rizika, dėl kurios gali atsirasti mokymo IoU, objektyvumo ir klasifikavimo nuostolių.
2. Tinkamiausi detektoriai realiojo laiko objektų aptikimo sistemai yra veikiantys regresijos principu. Plačiausiai realiojo laiko sistemose smulkiems objektams aptikti yra taikomi YOLO, SSD, *SqueezeDet* detektoriai.
3. Elektronikos komponentų ant konvejerio skaičiavimo sistemai pasirinkta naudoti *Nvidia Jetson Nano 4GB* mikrokompiuterį dėl jo kokybės ir kainos santykio bei našumo šiuolaikiniams dirbtinio intelekto darbo krūviams. Norint palyginti sistemos veikimą su galingesne aparatine įranga, papildomai naudota *Tesla P100-16GB* GPU *Google Colab* debesijos paslaugos sistemoje.
4. Duomenų rinkiniui pildyti panaudoti duomenų didinimo metodai: nuotraukų horizontalūs apvertimai, vertikalūs apvertinimai, pasukimai iki 90° aukštyn-žemyn ir atsitiktiniai pasukimai nuo -15° iki +15° kampu. Tyrimo metu įsitikinta, kad naudojant šiuos metodus nepakenkiama duomenų rinkinio kokybei (pvz., nepašalinamos svarbios detalės, reikalingos objektui identifikuoti).
5. Tyrimams atlikti naudoti objektų aptikimo detektoriai buvo mokyti su sudarytu elektronikos komponentų duomenų rinkiniu. Šis rinkinys sudarytas iš keturių klasių (rezistoriai, diodai, kondensatoriai, tranzistoriai), 7661 vnt. SMD komponentų nuotraukų, kuriose yra 28536 vnt. objektų žymės – iš jų 10467 vnt. rezistorių, 8543 vnt. diodų, 8198 vnt. kondensatorių ir 1328 vnt. tranzistorių.
6. Naudota *Roboflow* terpė leido paspartinti bandymų su skirtingais neuroninių tinklų detektoriais procesą. Duomenų rinkinio žymų formatai iš pradinio *VOC* formato buvo automatiškai konvertuojami į architektūroms tinkamus formatus.

7. Aukščiausias vidutinio tikslumo (AP) įvertinimas atliekant kondensatorių aptikimą buvo pasiektas su YOLOv4-tiny detektoriumi – 97,3 % naudojant *Tesla P100-16GB* ir *Nvidia Jetson Nano 4GB* įrenginius. Šio modelio išvedimo laikas (IT) siekė 10,6 ms (94 FPS) naudojant *Tesla P100-16GB* ir 62,5 s (16 FPS) naudojant *Jetson Nano*.
8. Aukščiausi vidutinio tikslumo (AP) įvertinimai buvo gauti su YOLOv4-tiny detektoriumi – atliekant rezistorių aptikimą siekė 98,5 % AP. Tiksliausiai diodai aptikti su YOLOv3 modeliu – gautas 88,6 % AP. Atliekant visų komponentų aptikimą didžiausias AP buvo gautas su YOLOv4 detektoriumi – siekė 97,2 %.
9. Daugiausiai komponentų pavyko suskaičiuoti su YOLOv5s detektoriumi – buvo suskaičiuoti 308 vnt. elektronikos komponentų naudojant *Tesla P100-16GB* GPU ir 305 vnt. naudojant *Jetson Nano* mikrokompiuterį. Mažiausiai kondensatorių ir visų komponentų kartu atliktų bandymų metu buvo suskaičiuota su YOLOv3 ir YOLOv3-tiny detektoriais. Mažiausia paklaida skaičiuojant išlaikyta su SSD-Mobilenet-V1, YOLOv4, YOLOv4-Scaled ir YOLOv5s detektoriais.
10. Iš gautų mAP rezultatų matyti, kad įdiegtas skaičiavimo algoritmas visų modelių tikslumui padarė labai mažą įtaką – nuo 0,02 % iki 0,05 % naudojant *Jetson Nano* ir nuo 0,03 % iki 1,23 % naudojant *Tesla P100-16GB*.
11. Atlikus detektorių aptikimo ir skaičiavimo spartos lyginamąją analizę, pastebėtas mFPS sulėtėjimas nuo 1,75 iki 3 mFPS naudojant *Tesla P100-16GB* ir nuo 1,5 iki 3 mFPS sumažėjimas naudojant *Jetson Nano*.
12. Lyginant objektų aptikimo spartą atsižvelgta į mFPS, mIT rezultatus. Norint išlaikyti realiojo laiko sistemą optimalią, turėtų būti išlaikyta daugiau nei 15 FPS ir ne didesnis nei 70 ms IT. SSD-Mobilenet-V1 ir YOLOv4-tiny detektoriai atitiko FPS kriterijų naudojant *Jetson Nano* mikrokompiuterį. Spartesnis iš jų buvo SSD-Mobilenet-V1 modelis, su kuriuo pasiekta bendras 27,1 ms IT ir 37 FPS.

Remiantis suformuluotomis išvadomis, galima teigti, kad pavyko pasiekti darbo tikslą ir įgyvendinti išsikeltus uždavinius. Sudaryta ir ištirta elektronikos komponentų ant konvejerio skaičiavimo sistema yra tinkama SMD tipo kondensatorių, rezistorių, diodų ir tranzistorių identifikavimui ir skaičiavimui atlikti. Pastebėta galimybė pritaikyti sistemą spausdintinių plokščių su komponentais nuskaitymui ir kokybės kontrolei (pvz., ar netrūksta komponentų plokštėje, ar komponentai yra prilituoti). Darbo medžiaga pristatyta XXV-oje Lietuvos jaunųjų mokslininkų konferencijoje *Mokslas – Lietuvos ateitis* ir rezultatai paskelbti mokslo leidinyje *Applied Sciences-Basel*, straipsnio pavadinimas – *A System for a Real-Time Electronic Component Detecttion and Classification on a Conveyor Belt*. Tolimesniuose tyrimuose būtų galima plėsti duomenų rinkinį, įterpiant nepanaudotas elektronikos komponentų rūšis.

LITERATŪRA

1. Ball, J.; Anderson, D.; Chan, C. 2017. Comprehensive survey of deep learning in remote sensing: theories, tools, and challenges for the community, *J. Appl. Rem. Sens.* 11(4) 042609. <https://doi.org/10.1117/1.JRS.11.042609>
2. Liu, L.; Ouyang, W.; Wang, X. 2020. Deep Learning for Generic Object Detection: A Survey. *Int J Comput Vis* 128, 261–318. <https://doi.org/10.1007/s11263-019-01247-4>
3. Yang, J.; Li S., Wang, Z.; Yang, G. 2019. Real-Time Tiny Part Defect Detection System in Manufacturing Using Deep Learning, *IEEE Access, vol. 7, pp.* 89278-89291. <https://doi.org/10.1109/ACCESS.2019.2925561>
4. Ayush, S.; Pradeep, S.; Rakesh, P. 2017. Use of Deep Learning in Modern Recommendation System: A Summary of Recent Works, *International Journal of Computer Applications* (0975 – 8887) leidinys 180 – Nr.7. < https://doi.org/10.48550/arXiv.1712.07525>
5. Anguera, J.; Sataphy, S.; Bhateja, V.; Sunitha, K. 2018. Impact of Deep Learning in Image Processing and Computer Vision [interaktyvus]. Singapūras: Springer [žiūrėta 2020 m. gruodžio 5d.]. Prieiga per internetą: <https://doi.org/10.1007/978-981-10-7329-8_48>
6. Robertson, S.; Azizpour, H. 2018. Digital image analysis in breast pathology – from image processing techniques to artificial intelligence, *Transl Res.* 194: 19-35. <https://doi.org/10.1016/j.trsl.2017.10.010>
7. Skoryk, A.; Chyrka, Y.; Gorovyi, I.; Grechnyev, O.; Vyplavin P. 2020. Comparative Analysis of Classic Computer Vision Methods and Deep Convolutional Neural Networks for Floor Segmentation, *IEEE Third International Conference on Data Stream Mining & Processing (DSMP)* 217-221. <https://doi.org/10.1109/DSMP47368.2020.9204339>
8. Differentiable Programming for Image Processing and Deep Learning in Halide [interaktyvus]. 2018. Tzu-Mao L, Gharbi M. [žiūrėta 2020 m. gruodžio 5d.]. Prieiga per internetą: <https://doi.org/10.1145/3197517.3201383>
9. Demirovic, D.; Skejic, E.; Šerifović, A. 2018. Performance of some image processing algorithms in TensorFlow, *International Conference on Systems, Signals and Image Processing (IWSSIP)* 1-4. <https://doi.org/10.1109/IWSSIP.2018.8439714>
10. Imambi, S.; Prakash, K.B.; Kanagachidambaresan, G.R. 2021. PyTorch [interaktyvus]. Springer: Cham [žiūrėta 2022 sausio 25 d.]. Prieiga per internetą: <https://doi.org/10.1007/978-3-030-57077-4_10>
11. Cresson, R. 2019. A Framework for Remote Sensing Images Processing Using Deep Learning Techniques, *IEEE Geoscience and Remote Sensing Letters* 16(1): 25-29 <https://doi.org/10.1109/LGRS.2018.2867949>

12. Zhuang, X.; Zhang, T. 2019. Detection of sick broilers by digital image processing and deep learning, *Biosystems Engineering* 179: 106-116. <https://doi.org/10.1016/j.biosystemseng.2019.01.003>
13. Dongmei, H.; Qigang, L.; Weigou, F. 2018. A new image classification method using CNN transfer learning and web data augmentation, *Expert Systems with Applications* 95: 43-56. <https://doi.org/10.1016/j.eswa.2017.11.028>
14. Chandrasekar, P.; Bhattacharya, P. 2017. Improving the Prediction Accuracy of Decision Tree Mining with Data Preprocessing, *IEEE 41st Annual Computer Software and Applications Conference* 481-484. <https://doi.org/10.1109/COMPSAC.2017.146>
15. Ramírez, S.; Krawczyk, B.; Garcíaa, S.; Wozniak, M.; Herreraa, F. 2017. A survey on data preprocessing for data stream mining: Current status and future directions, *Neurocomputing* 239: 39-57. <https://doi.org/10.1016/j.neucom.2017.01.078>
16. Project Adam: Building an Efficient and Scalable Deep Learning Training System [interaktyvus]. 2014. Trishul, C.; Yutaka, S. [žiūrėta 2020 m. gruodžio 5d.]. Prieiga per internetą: <https://www.usenix.org/conference/osdi14/technical-sessions/presentation/chilimbi>
17. Garcia, J.; Barbedo, A. 2018. Impact of dataset size and variety on the effectiveness of deep learning and transfer learning for plant disease classification, *Computer and Electronics in Agriculture* 153: 46-53. <https://doi.org/10.1016/j.compag.2018.08.013>
18. Qimin, C.; Qian, Z.; Peng, F.; Conghuan, T.; Sen, L. 2018. A survey and analysis on automatic image annotation, *Pattern Recognition* 79: 242-259 <https://doi.org/10.1016/j.patcog.2018.02.017>
19. Chandrasekaran, J.; Feng, H.; Lei, Y.; Kacker, R.; Kuhn, R. 2020. Effectiveness of dataset reduction in testing machine learning algorithms, *IEEE International Conference On Artificial Intelligence Testing 133-140.* <https://doi.org/10.1109/AITEST49225.2020.00027>
20. Haoran, S.; Xiangyi, C.; Qingjiang, S.; Mingyi, H.; Xiao, F. 2017. Learning to optimize: training deep neural networks for wireless resource management, 2017 *IEEE 18th International Workshop on Signal Processing Advances in Wireless Communications*, 1-6. <https://doi.org/10.1109/SPAWC.2017.8227766 >
21. Training Deep Neural Networks for Visual Servoing [interaktyvus]. 2018. BATEUX Q, MARCHANDE, LEITNER J. [žiūrėta 2020 m. gruodžio 5d.]. Prieiga per internetą: <https://doi.org/10.1109/ICRA.2018.8461068>
22. Pelletier, M.; Holt, G. 2020. A Plastic Contamination Image Dataset for Deep Learning Model Development and Training, *IEEE International Conference on Robotics and Automation (ICRA)* 3307-3314 <https://doi.org/10.3390/agriengineering2020021>

23. Li, L.; Wanli, O.; Xiaogang, W. 2019. Deep Learning for Generic Object Detection: A Survey, *Int J Comput Vis* 128: 261-318. <https://doi.org/10.1007/s11263-019-01247-4>
24. Ball, J.; Anderson, D.; Chan, C. 2017. Comprehensive survey of deep learning in remote sensing: theories, tools, and challenges for the community, *J. of Applied Remote Sensing* 11(4) <https://doi.org/10.1117/1.JRS.11.042609>
25. Wozniak, M,; Połap, D. 2018. Object detection and recognition via clustered features, *Neurocomputing* 320: 76-84. <https://doi.org/10.1016/j.neucom.2018.09.003>
26. Chinthakindi, M.; Mohammad, H.; Neeraj, B.; Zong, G. 2020. Investigations of Object Detection in Images/Videos Using Various Deep Learning Techniques and Embedded Platforms – A Comprehensive Review, *Appl. Sci.* 10(9): 3280 <https://doi.org/10.3390/app10093280>
27. Yun, R.; Changren, Z.; Shunping, X. 2018. Small Object Detection in Optical Remote Sensing Images via Modified Faster R-CNN, *Appl. Sci.* 8(5): 813 <https://doi.org/10.3390/app8050813>
28. Alsing O. 2018. Mobile Object Detection using TensorFlow Lite and Transfer Learning <https://www.diva-portal.org/smash/get/diva2:1242627/>
29. Xiaolong, W.; Abhinav, S.; Abhinav, G. 2017. A-Fast-RCNN: Hard Positive Generation via Adversary for Object Detection, *Computer Vision and Pattern Recognition* <https://doi.org/10.48550/arXiv.1704.03414>
30. Liu, H. Y. 2018. Feature Extraction and Image Recognition with Convolutional Neural Networks, *J. Phys.* 1087(6): 062032 < http://dx.doi.org/10.1088/1742-6596/1087/6/062032 >
31. Hang, X.; Lewei, Y.; Wei, Z.; Xiaodan, L.; Zhenguo, L. 2019. Auto-FPN: Automatic Network Architecture Adaptation for Object Detection Beyond Classification, *IEEE/CVF International Conference on Computer Vision* 6648-6657. <https://doi.org/10.1109/ICCV.2019.00675
32. Chandan, G.; Ayush, J.; Harsh, J. 2018. Real Time Object Detection and Tracking Using Deep Learning and OpenCV, *International Conference on Inventive Research in Computing Applications* 1305-1308. <https://doi.org/10.1109/ICIRCA.2018.8597266>
33. Bichen, W.; Forrest, I.; Jin, P.; Keutzer, K. SqueezeDet: Unified, Small, Low Power Fully Convolutional Neural Networks for Real-Time Object Detection for Autonomous Driving, *IEEE Conference on Computer Vision and Pattern Recognition* 129-137. <http://dx.doi.org/10.1109/CVPRW.2017.60 >
34. Chen, P. -Y.; Hsieh, J. -W.; Wang, C. -Y.; Mark Liao, H. -Y. 2020. Recursive Hybrid Fusion Pyramid Network for Real-Time Small Object Detection on Embedded Devices, *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops* 1612-1621. <https://doi.org/10.1109/CVPRW50498.2020.00209>

35. Liu, W.; Arguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.; Berg, A. 2016. SSD: Single Shot MultiBox Detector [interaktyvus]. Springer [žiūrėta 2021 m. gegužės 19d.]. Prieiga per internetą: <https://doi.org/10.1007/978-3-319-46448-0_2>
36. Biswas, D.; Su, H.; Wang, C.; Stevanovic, A.; Wang, W. 2019. An automatic traffic density estimation using Single Shot Detection (SSD) with MobileNet-SSD, *Physics and Chemistry of the Earth, Parts A/B/C* 110: 176-184. <https://doi.org/10.1016/j.pce.2018.12.001>
37. Nvidia developer. DetectNet: Deep Neural Network for Object Detection in DIGITS [interaktyvus]. Nvidia [žiūrėta 2021 m. gegužės 19d.]. Prieiga per internetą: <https://developer.nvidia.com/blog/detectnet-deep-neural-network-object-detection-digits/>
38. Nvidia developer. Jetson Nano Developer Kit [interaktyvus]. Nvidia [žiūrėta 2021 m. gegužės 19d.]. Prieiga per internetą: <https://developer.nvidia.com/embedded/jetson-nano-developer-kit>
39. Raspberry. Camera Module [interaktyvus]. Raspberry Pi [žiūrėta 2021 m. gegužės 19d.]. Prieiga per internetą: <https://www.raspberrypi.org/documentation/hardware/camera/>
40. T6712DV Raspberry Pi Power Supply [interaktyvus]. Farnell [žiūrėta 2021 m. gegužės 19d.]. Prieiga per internetą: <https://www.farnell.com/datasheets/2693636.pdf >
41. Hendry.; Chen, R-C. 2019. Automatic License Plate Recognition via sliding-window darknet-YOLO deep learning, *Image and Vision Computing* 87: 47-56. <https://doi.org/10.1016/j.imavis.2019.04.007>
42. Qinjie, L.; Guo, Y.; Jiayi, W.; Han, L. 2022. RoboFlow: a Data-centric Workflow Management System for Developing AI-enhanced Robots, *Proceedings of the 5th Conference on Robot Learning, PMLR* 164: 1789-1794. <https://proceedings.mlr.press/v164/lin22c.html>
43. Qinjie, L,; Guo, Y.; Jiayi, Wang.; Han, L. 2021. Development of video-based emotion recognition using deep learning with Google Colab, *Journal of Electrical Engineering* 18(5): 2463-2471. <http://dx.doi.org/10.12928/TELKOMNIKA.v18i5.16717>
44. Parico, A.I.B.; Ahamed, T. 2021. Real Time Pear Fruit Detection and Counting Using YOLOv4 Models and Deep SORT, *Sensors* 21: 4803. <https://doi.org/10.3390/s21144803>
45. Fang, W.; Wang, L.; Ren, P. 2020. Tinier-YOLO: A Real-Time Object Detection Method for Constrained Environments, *IEEE Access* 8: 1935-1944. <10.1109/ACCESS.2019.2961959 >
46. Junsuo, Q.; Chang, S.; Zhiwei, Z.; Abolfazl, R. 2020. Dilated Convolution and Feature Fusion SSD Network for Small Object Detection in Remote Sensing Images, *IEEE Access* 8: 82832-82843. <https://doi.org/10.1109/ACCESS.2020.2991439>
47. Yu, J.; Zhang, W. 2021. Face Mask Wearing Detection Algorithm Based on Improved YOLO-v4, *Sensors* 21: 3263. <https://doi.org/10.3390/s21093263>

48. Valiati, G. R.; Menotti, D. 2019. Detecting Pedestrians with YOLOv3 and Semantic Segmentation Infusion, *2019 International Conference on Systems, Signals and Image Processing (IWSSIP)* 95-100 <https://doi.org/10.1109/IWSSIP.2019.8787210>
49. Koirala, A.; Walsh, K.B.; Wang, Z. 2019. Deep learning for real-time fruit detection and orchard fruit load estimation: benchmarking of 'MangoYOLO', *Precision Agric* 20: 1107–1135. <https://doi.org/10.1007/s11119-019-09642-0>
50. Kasper-Eulaers, M.; Hahn, N.; Berger, S.; Sebulonsen, T.; Myrland, Ø.; Kummervold, P.E. 2021. Short Communication: Detecting Heavy Goods Vehicles in Rest Areas in Winter Conditions Using YOLOv5, *Algorithms* 14: 114. <https://doi.org/10.3390/a14040114>
51. Vuong, C.; Phan, V.; Mou, R.; Vu, T.; Vu, D.; Nguyen, N. 2021. Vehicle Detection and Counting under Mixed Traffic Conditions in Vietnam using YOLOv4, *International Journal of Advanced Research in Engineering and Technology (IJARET)* 12(2): 722-730. <https://doi.org/10.34218/IJARET.12.2.2021.072>
52. Binghuang, C.; Xiren, M. 2019. Distribution Line Pole Detection and Counting Based on YOLO Using UAV Inspection Line Video, *Journal of Electrical Engineering and Technology* 15(3) <http://dx.doi.org/10.1007/s42835-019-00230-w>
53. Yang, Y.; Gao, W. 2019. A Method of Pedestrians Counting Based on Deep Learning, *2019 3rd International Conference on Electronic Information Technology and Computer Engineering (EITCE)* 2010-2013. <https://doi.org/10.1109/EITCE47263.2019.9094838>
54. Vishal, M.; Yaw, A-G. 2020. Object Detection and Tracking Algorithms for Vehicle Counting: A Comparative Analysis, *Computer Vision and Pattern Recognition 2007: 16198*. <https://doi.org/10.48550/arXiv.2007.16198>
55. Lin, C-J.; Jeng, S-Y. 2021. A Real-Time Vehicle Counting, Speed Estimation, and Classification System Based on Virtual Detection Zone and YOLO, *Mathematical Problems in Engineering* 1-10. <http://dx.doi.org/10.1155/2021/1577614>
56. Song, H.; Liang, H.; Li, H. 2019. Vision-based vehicle detection and counting system using deep learning in highway scenes, *Eur. Transp. Res.* 11: 51. <https://doi.org/10.1186/s12544-019-0390-4>
57. Xue, J.; Cheng, F.; Li, Y.; Song, Y.; Mao, T. 2022. Detection of Farmland Obstacles Based on an Improved YOLOv5s Algorithm by Using CIoU and Anchor Box Scale Clustering, *Sensors* 22: 1790. <https://doi.org/10.3390/s22051790>
58. Wang, C-Y.; Bochkovskiy, A.; Liao, H-Y. 2021. Scaled-YOLOv4: Scaling Cross Stage Partial Network, *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* 13024-13033. <https://doi.org/10.1109/CVPR46437.2021.01283>

59. Du, X.; Zoph, B.; Hung, W-C.; Lin, T. 2021. Simple Training Strategies and Model Scaling for Object Detection, *Computer Vision and Pattern Recognition* <https://doi.org/10.48550/arXiv.2107.00057>

PRIEDAI

A priedas. Elektronikos komponentų aptikimo ir skaičiavimo metodų tyrimo rezultatai

1 lentelė. Kondensatorių skaičiavimo tyrimo rezultatai

Eil. nr.	Modelis	Tesla P100-16GB				Jetson Nano 4GB			
		Suskaičiuota, vnt.	**AP**, %	**FPS**	**IT**, ms	**Suskaičiuota**, vnt.	**AP**, %	**FPS**	**IT**, ms
1.	SSD-Mobilenet-V1	107	66,7	88	11,4	111	66,6	**39**	**25,6**
2.	YOLOv3	46	90,2	12	83,3	34	90,1	1	1000
3.	YOLOv3-tiny	34	83,1	17	58,8	35	82,7	4	250
4.	YOLOv4	106	94,3	47	21,3	95	94,6	1	1000
5.	YOLOv4-tiny	96	**97,3**	**94**	**10,6**	94	**97,3**	16	62,5
6.	YOLOv4-Scaled	112	87,8	42	23,8	111	86,7	1	1000
7.	YOLOv5s	**137**	85,2	87	11,5	**133**	85,3	6	166,7

2 lentelė. Rezistorių skaičiavimo tyrimo rezultatai

Eil. nr.	Modelis	Tesla P100-16GB				Jetson Nano 4GB			
		Suskaičiuota, vnt.	**AP**, %	**FPS**	**IT**, ms	**Suskaičiuota**, vnt.	**AP**, %	**FPS**	**IT**, ms
1.	SSD-Mobilenet-V1	64	75	87	11,5	61	75,1	**38**	**26,3**
2.	YOLOv3	74	92,7	13	76,9	48	92,7	1	1000
3.	YOLOv3-tiny	**77**	76,3	16	62,5	**88**	76,1	4	250
4.	YOLOv4	64	97,7	46	21,7	68	**98,6**	1	1000
5.	YOLOv4-tiny	**77**	**98,5**	**90**	**11,1**	74	98,5	16	62,5
6.	YOLOv4-Scaled	64	89,1	42	23,8	64	89,1	1	1000
7.	YOLOv5s	64	91,1	66	15,2	64	91	6	166,7

3 lentelė. Diodų skaičiavimo tyrimo rezultatai

Eil. nr.	Modelis	Tesla P100-16GB				Jetson Nano 4GB			
		Suskaičiuota, vnt.	**AP**, %	**FPS**	**IT**, ms	**Suskaičiuota**, vnt.	**AP**, %	**FPS**	**IT**, ms
1.	SSD-Mobilenet-V1	68	67,3	90	11,1	52	67,2	**38**	**26,3**
2.	YOLOv3	63	**88,6**	13	76,9	58	**88,7**	1	1000
3.	YOLOv3-tiny	84	65,3	19	52,6	84	65,2	4	250
4.	YOLOv4	56	66,2	47	21,3	83	85,9	1	1000
5.	YOLOv4-tiny	**113**	76,2	**120**	**8,3**	**126**	76,5	16	62,5
6.	YOLOv4-Scaled	76	70,7	44	22,7	79	70	1	1000
7.	YOLOv5s	65	81,3	77	13	66	81,2	6	166,7

4 lentelė. Visų komponentų skaičiavimo tyrimo rezultatai

Eil. nr.	Modelis	Tesla P100-16GB				Jetson Nano 4GB			
		Suskaičiuota, vnt.	**AP**, %	**FPS**	**IT**, ms	**Suskaičiuota**, vnt.	**AP**, %	**FPS**	**IT**, ms
1.	SSD-Mobilenet-V1	40	60,6	87	11,5	41	60,5	**33**	**30,3**
2.	YOLOv3	31	66,9	10	100	**86**	66,9	1	1000
3.	YOLOv3-tiny	23	63,5	17	58,8	24	63,5	4	250
4.	YOLOv4	49	**95,4**	43	23,3	54	**97,2**	1	1000
5.	YOLOv4-tiny	**51**	79,9	63	15,9	48	79,8	15	66,7
6.	YOLOv4-Scaled	47	69,6	39	25,6	45	69,5	1	1000
7.	YOLOv5s	42	70,6	**89**	**11,2**	42	70,6	6	166,7

5 lentelė. Bendri komponentų aptikimo tyrimo rezultatai

Eil. nr.	Modelis	Tesla P100-16GB			Jetson Nano 4GB		
		mAP, %	**mFPS**	**mIT**, ms	**mAP**, %	**mFPS**	**mIT**, ms
1.	SSD-Mobilenet-V1	67,4	89,75	**11,1**	67,35	**40**	**25,1**
2.	YOLOv3	84,6	15	66,8	84,6	1	1000
3.	YOLOv3-tiny	72,05	19,25	52,1	71,88	5,5	187,5
4.	YOLOv4	**88,43**	48	20,8	**94,08**	1	1000
5.	YOLOv4-tiny	87,98	**93,75**	11,2	88,03	17,75	56,4
6.	YOLOv4-Scaled	79,3	43,75	22,9	78,83	2,5	437,5
7.	YOLOv5s	83,28	82,25	12,3	82,03	8	125

6 lentelė. Bendri komponentų skaičiavimo tyrimo rezultatai

Eil. nr.	Modelis	Tesla P100-16GB				Jetson Nano 4GB			
		Suskaičiuota, vnt.	**mAP**, %	**mFPS**	**mIT**, ms	**Suskaičiuota**, vnt.	**mAP**, %	**mFPS**	**mIT**, ms
1.	SSD-Mobilenet-V1	279	67,4	88	**11,4**	265	67,4	**37**	**27,1**
2.	YOLOv3	214	84,6	12	84,3	226	84,6	1	1000
3.	YOLOv3-tiny	218	72,05	17,25	58,2	231	71,9	4	250
4.	YOLOv4	275	88,4	45,75	21,9	300	**94,1**	1	1000
5.	YOLOv4-tiny	**337**	**87,98**	**91,75**	11,5	**342**	88	15,8	63,6
6.	YOLOv4-Scaled	299	79,3	41,75	24	299	78,8	1	1000
7.	YOLOv5s	308	82,05	79,75	12,7	305	82	6	166,7

B priedas. Pranešimo 25-oje Lietuvos jaunųjų mokslininkų konferencijoje medžiaga

VILNIUS TECH
Elektronikos fakultetas

XXV-toji Lietuvos jaunųjų mokslininkų konferencija
„MOKSLAS – LIETUVOS ATEITIS“

ELEKTRONIKA IR ELEKTROTECHNIKA

PAŽYMA

Dainius Varna

ELEKTRONIKOS KOMPONENTŲ SKAIČIAVIMO ANT KONVEJERIO METODŲ TYRIMAS

Mokslo komiteto pirmininkas prof. habil. dr. Romanas Martavičius

Sekcijos pirmininkas dr. Vytautas Abromavičius

Vilniaus Gedimino technikos universitetas Elektronikos fakultetas

2022 m. kovo 18 d.
Vilnius

VILNIUS TECH
Vilniaus Gedimino technikos universitetas

ELEKTRONIKOS KOMPONENTŲ SKAIČIAVIMO ANT KONVEJERIO METODŲ TYRIMAS

25-OJI LIETUVOS JAUNŲJŲ MOKSLININKŲ KONFERENCIJA
„MOKSLAS – LIETUVOS ATEITIS“

Dainius VARNA, dainius.varna@stud.vilniustech.lt
Vadovas: dr. Vytautas ABROMAVIČIUS
Elektroninių sistemų katedra

Vilnius, 2022

Elektronikos komponentų skaičiavimo metodų analizė

VILNIUS TECH

- **Tikslas –** iš gaunamos vaizdo medžiagos aptikti elektronikos komponentus:
  - Objektų aptikimas, identifikavimas;
  - Objektų atpažinimas.

2

Aktualumas, taikymo sritys

VILNIUS TECH

- Galimybė panaudoti naujus staklių automatizavimo metodus.
- Tokio tipo modeliai naudojami
  - Produkcijos skaičiavime;
  - Kokybės kontrolėje;
  - Gamyboje, rūšiavime.

3

Atlikti tyrimai

VILNIUS TECH

- Tyrimams atlikti naudojamas individualus sukurtas duomenų rinkinys;
- 5 modelių objektų aptikimo efektyvumo tyrimas;
- 5 modelių objektų skaičiavimo efektyvumo tyrimas.

  * Efektyvumui įvertinti naudota: modelio aptikimo sparta (angl. *Frames Per Second*), vidutinis objektų atpažinimo tikslumas (angl. *Mean Average Precision*), suskaičiuotų komponentų kiekis.

4

Tyrimų rezultatai

VILNIUS TECH

- Tyrimams atlikti naudotas individualus sukurtas duomenų rinkinys

1 lentelė. Duomenų rinkinių specifikacija

Nr.	Nuotraukų kiekis, vnt.				Klasifikavimo anotacijų kiekis, vnt.				
	Iš viso	Patvirtinimui	Bandymams	Mokymui	Iš viso	Rezistoriai	Diodai	Kondensatoriai	Tranzistoriai
I	3005	300	300	2405	11875	4478	3404	3297	696
II	4061	406	406	3249	15950	6007	4581	4470	892
III	7661	300	300	7061	28536	10467	8543	8198	1328

5

Tyrimų rezultatai

VILNIUS TECH

- Modelių objektų aptikimo efektyvumo tyrimo rezultatai (Tesla P100-16GB)

2 lentelė. Bendri komponentų aptikimo tyrimo rezultatai

Nr.	Modelis	Tesla P100-16GB					
		AP_1, %	AP_2, %	AP_3, %	AP_4, %	mAP, %	mFPS
1.	SSD-Mobilenet-V1	66,7	75	67,3	60,6	67,4	**89,8**
2.	YOLOv3-Keras	90,2	92,7	**88,6**	66,9	84,6	15
3.	YOLOv4	**94,4**	**97,7**	66,2	**95,4**	**88,4**	48
4.	YOLOv4-Scaled	87,8	89,1	70,7	69,6	79,3	43,8
5.	YOLOv5s	88,6	92,9	82,8	68,8	83,3	82,3

*AP – vidutinis tikslumas (angl. *Average Precision*)

*mAP – vidutinio tikslumo vidurkis (angl. *Mean Average Precision*)

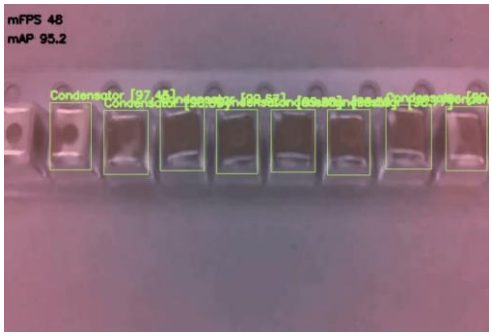
*mFPS – objekto aptikimo spartos vidurkis (angl. *Mean Frames Per Second*)

6

Tyrimų rezultatai

VILNIUS TECH

- Modelių objektų aptikimo efektyvumo tyrimo rezultatai (Tesla P100-16GB)



1 pav. Objektų aptikimo efektyvumo tyrimas

7

Tyrimų rezultatai

VILNIUS TECH

- Modelių objektų skaičiavimo efektyvumo tyrimo rezultatai (Tesla P100-16GB)

3 lentelė. Bendri komponentų skaičiavimo tyrimo rezultatai

Nr.	Modelis	Tesla P100-16GB						
		Suskaičiuota[1], vnt	Suskaičiuota[2], vnt	Suskaičiuota[3], vnt	Suskaičiuota[4], vnt	Bendra suma, vnt	mAP, %	mFPS
1.	SSD-Mobilenet-V1	107	64	68	40	279	67,4	**88**
2.	YOLOv3-Keras	102	**74**	63	31	270	84,6	12
3.	YOLOv4	106	64	56	**49**	275	**88,4**	46
4.	YOLOv4-Scaled	112	64	**76**	47	299	79,3	42
5.	YOLOv5s	**137**	64	65	40	**306**	83,3	80

*AP – vidutinis tikslumas (angl. *Average Precision*)

*mAP – vidutinio tikslumo vidurkis (angl. *Mean Average Precision*)

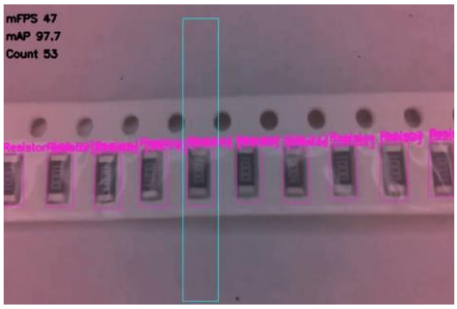
*mFPS – objekto aptikimo spartos vidurkis (angl. *Mean Frames Per Second*)

8

Tyrimų rezultatai

VILNIUS TECH

- Modelių objektų skaičiavimo efektyvumo tyrimo modelis



2 pav. Objektų aptikimo efektyvumo tyrimas

9

Išvados

VILNIUS TECH

- Iš ištirtų objektų aptikimo modelių:
  - Sparčiausias modelis SSD-Mobilenet-v1 (88 FPS);
  - Didžiausią tikslumą turintis modelis YOLOv4 (88,4 % mAP).
- Tinkamiausias objektų skaičiavimui modelis YOLOv5 (306 vnt., 83,3 mAP, 80 FPS);
- Tolimesniuose tyrimuose bus atliekami bandymai su Jetson Nano mikrokompiuteriu.

10

VILNIUS TECH
Vilniaus Gedimino technikos universitetas

Dainius Varna
dainius.varna@stud.vilniustech.lt

Ačiū už dėmesį