



**VILNIUS
TECH**

Elektroninių sistemų
katedra

Domantas SKUJA

**QR IR BRŪKŠNINIŲ KODŲ PAIEŠKA IR ATPAŽINIMAS VAIZDUOSE
TAIKANT GILIOJO MOKYMOSI METODUS**

**QR CODE AND BARCODE DETECTION AND RECOGNITION IN
IMAGES WITH DEEP LEARNING METHODS**

Magistro baigiamasis darbas

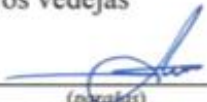
Informacinių elektroninių sistemų studijų programa, valstybinis kodas 6211BX015

Atvirojo kodo sistemų specializacija

Informatikos inžinerijos studijų kryptis

VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS
ELEKTRONIKOS FAKULTETAS
ELEKTRONINIŲ SISTEMŲ KATEDRA

TVIRTINU
Katedros vedėjas


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

Domantas SKUJA

QR IR BRŪKŠNINIŲ KODŲ PAIEŠKA IR ATPAŽINIMAS VAIZDUOSE
TAIKANT GILIOJO MOKYMOSI METODUS
QR CODE AND BARCODE DETECTION AND RECOGNITION IN
IMAGES WITH DEEP LEARNING METHODS

Magistro baigiamasis darbas

Informacinių elektroninių sistemų studijų programa, valstybinis kodas 6211BX015

Atvirojo kodo sistemų specializacija

Informatikos inžinerijos studijų kryptis

Vadovas

prof. dr. D. Matuzevičius

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

Lietuvių kalbos konsultantas

dr. A. Žemienė

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

Vilnius, 2022

Vilniaus Gedimino technikos universitetas Elektronikos fakultetas Elektroninių sistemų katedra	ISBN _____ ISSN _____ Egz. sk. Data-.....-.....
Antrosios pakopos studijų Informacinių elektroninių sistemų programos magistro baigiamasis darbas Pavadinimas QR ir brūkšninių kodų paieška ir atpažinimas vaizduose taikant giliojo mokymosi metodus Autorius Domantas Skuja Vadovas Dalius Matuzevičius	
Kalba: lietuvių	
Anotacija QR ir brūkšninių kodų paieška ir atpažinimas vaizduose taikant giliojo mokymosi metodus. Baigiamasis magistro darbas informatikos inžinerijos laipsniui. Vilniaus Gedimino technikos universitetas. Vilnius, 2022, 67 p., 37 iliustr., 7 lent., 32 bibl., 2 priedai. Šiame baigiamajame darbe buvo tiriama QR ir brūkšninių kodų paieška ir atpažinimas taikant giliojo mokymosi metodus. Tikslui pasiekti buvo atlikti giliojo mokymosi metodų, skirtų objektams vaizduose aptikti ir atpažinti, analitinė apžvalga. Taip pat išanalizuotos objektų atpažinimui reikalingos programinės įrangos ir jų licencijos. Atrinkti giliojo mokymosi metodai perspektyvus QR ir brūkšninių kodų atpažinimui vaizduose. Eksperimentiškai ištirti atrinktieji metodai ir įgyvendinta įterptinė sistema, veikianti realiuoju laiku ir naudojanti QR ir brūkšninių kodų paieškai ir atpažinimui tinkančius metodus. Gauta sistema parodė 20 kadrų per sekundę spartą ir pasiekia 94,3 % preciziškumą aptikdama QR ir brūkšninius kodus. Taip pat su šia sistema ištirtas neuronų tinklo ir kodų šifravimo greitis nuo įvesties kadro dydžio.	
Prasminiai žodžiai: kompiuterinė rega, QR kodas, brūkšninis kodas, gilusis mokymasis, sąsūkos dirbtiniai neuronų tinklai	

Vilnius Gediminas Technical University Faculty of Electronics Department of Electronic Systems	ISBN ISSN Copies No. Date-.....-.....
Master Degree Studies Information Electronics Systems study programme Master Graduation Thesis Title QR Code and Barcode Detection and Recognition in Images With Deep Learning Methods Author Domantas Skuja Academic supervisor Dalius Matuzevičius	
Thesis language: Lithuanian	
Annotation This work reviews capabilities of deep learning-based methods for QR code and barcode detection and recognition in images. To reach the goal, the following tasks were completed: analytical review of deep learning methods for object detection and recognition in images. Review of software and their licenses required for object recognition. Selection of perspective deep learning methods for QR and barcode recognition in images. The selected methods was investigated in the distributed embedded system (operating in real time) which are using QR and barcode search and recognition methods. The resulting system shows speed of 20 frames per second and achieves 94.3% accuracy in detecting QR and barcodes. The neural network and code decryption speeds compare to input frame size were also investigated with this system.	
Keywords: computer vision, QR code, barcode, deep learning, convolutional neural network	

VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS
ELEKTRONIKOS FAKULTETAS
ELEKTRONINIŲ SISTEMŲ KATEDRA

Technologijos mokslų sritis
Elektros ir elektronikos inžinerijos mokslo kryptis
Informatikos inžinerijos studijų kryptis
Informacinių elektroninių sistemų studijų programa, valst. kodas 6211BX015
Atvirojo kodo sistemų specializacija

TVIRTINU

Katedros vedėjas

prof. dr. A. Serackis

2022 m. 05 mėn. 23 d.

MAGISTRO BAIGIAMOJO DARBO
UŽDUOTIS

Studentui Domantas SKUJA

Baigiamojo darbo tema: QR ir brūkšninių kodų paieška ir atpažinimas vaizduose taikant giliojo mokymosi metodus

QR Code and Barcode Detection and Recognition in Images with Deep Learning Methods

Patvirtinta 2022 m. _____ mėn. ____ d. dekanų potvarkiu Nr. _____.

Baigiamojo darbo užbaigimo terminas 2022 m. _____ mėn. ____ d.

Darbo tikslas – ištirti giliojo mokymosi metodų taikymą QR ir brūkšninių kodų paieškai ir atpažinimui vaizduose.

Darbo uždaviniai

1. Atlikti giliojo mokymosi metodų, skirtų objektams aptikti ir atpažinti vaizduose, analitinę apžvalgą.
2. Atlikti objektų atpažinimui reikalingos programinės įrangos licencijų analizę.
3. Atrinkti giliojo mokymosi metodus, perspektyvius taikyti QR ir brūkšninių kodų paieškai ir atpažinimui vaizduose.
4. Paruošti duomenų rinkinį QR ir brūkšninių kodų paieškos ir atpažinimo metodų eksperimentiniams tyrimams.
5. Įgyvendinti ir eksperimentiškai ištirti atrinktus QR ir brūkšninių kodų paieškos ir atpažinimo metodus, ir pateikti tyrimo rezultatus.

Aiškinamojo rašto turinys

Įvadas

Darbo aktualumas ir tikslas

Darbo uždaviniai

Naudoti tyrimo ir analizės metodai

Darbo naujumas ir praktinė nauda

Darbo struktūra

1. Giliojo mokymosi metodų objektams atpažinti analitinė apžvalga
2. Giliojo mokymosi metodų taikymas QR ir brūkšninių kodų paieškai ir atpažinimui
3. Duomenų paruošimas QR ir brūkšninių kodų paieškos ir atpažinimo metodų eksperimentiniam tyrimui
4. QR ir brūkšninių kodų paieškos ir atpažinimo metodų tyrimo rezultatai

Apibendrinimas. Išvados

Literatūra

Priedai

A priedas. QR ir brūkšninių kodų paieškos ir atpažinimo metodų tyrimo rezultatai

Baigiamojo darbo rengimo konsultantas:

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

Vadovas


(parašas)

prof. dr. Dalius Matuzevičius

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

Užduotį gavau

202_ m. _____ mėn. ____ d.

(parašas)

VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS

Domantas Skuja 20200968

(Studento vardas ir pavardė, studento pažymėjimo Nr.)

Elektronikos fakultetas

(Fakultetas)

Informacinės elektroninės sistemos, AKSfm-20

(Studijų programa, akademinė grupė)

**BAIGIAMOJO DARBO (PROJEKTO)
SAŽININGUMO DEKLARACIJA**

2022 m. gegužės 24 d.

(Data)

Patvirtinu, kad mano baigiamasis darbas tema „QR ir brūkšninių kodų paieška ir atpažinimas vaizduose taikant giliojo mokymosi metodus“ patvirtintas 20__ m. _____ d. dekanų potvarkiu Nr. _____, yra savarankiškai parašytas. Šiame darbe pateikta medžiaga nėra plagijuota. Tiesiogiai ar netiesiogiai panaudotos kitų šaltinių citatos pažymėtos literatūros nuorodose

Prenkant ir įvertinant medžiagą bei rengiant baigiamąjį darbą, mane konsultavo mokslininkai ir specialistai: prof. dr. Dalius Matuzevičius. Mano darbo vadovas prof. dr. Dalius Matuzevičius.

Kitų asmenų indėlio į parengtą baigiamąjį darbą (projektą) nėra. Jokių įstatymų nenumatytų piniginių sumų už šį darbą niekam nesu mokėjęs (-usi).



(Parašas)

DOMANTAS SKUJA
(Vardas ir pavardė)

TURINYS

ŽYMENYS IR SANTRUMPOS	9
PAVEIKSLŲ SĄRAŠAS	10
LENTELIŲ SĄRAŠAS	11
IVADAS	12
Darbo aktualumas ir tikslas	12
Darbo uždaviniai	12
Naudoti tyrimo ir analizės metodai	13
Darbo naujumas ir praktinė nauda	13
Darbo struktūra	13
1. GILIOJO MOKYMOSI METODŲ OBJEKTAMS ATPAŽINTI ANALITINĖ APŽVALGA	14
1.1. Atvirojo kodo programinės įrangos licencijavimas	14
1.1.1. MIT	14
1.1.2. GPL-3.0	15
1.1.3. BSD	15
1.1.4. „Apache License 2.0“	15
1.2. Brūkšninių kodų tipų ir komercinių sistemų apžvalga	16
1.2.1. Vienmačiai brūkšniniai kodai	16
1.2.2. Dvimačiai brūkšniniai kodai	18
1.2.3. Brūkšninių kodų atpažinimo sistemos	19
1.3. Objektų aptikimas vaizduose	20
1.3.1. Dirbtiniai neuronų tinklai	20
1.3.2. Gilieji dirbtiniai neuronų tinklai	20
1.3.3. Sąsukos dirbtiniai neuronų tinklai	21
1.4. Sąsukos dirbtinių neuroninių tinklų architektūros	23
1.4.1. „EfficientNet“	23
1.4.2. „DenseNet“	23
1.4.3. „ResNet“	24
1.4.4. „Darknet53“	24
1.5. Giliojo mokymosi modeliai objektų atpažinimui	24
1.5.1. Regionais pagrįstas sąsukos dirbtinis neuronų tinklas	25
1.5.2. Greitas regionais pagrįstas sąsukos dirbtinis neuronų tinklas	25
1.5.3. Greitesnis regionais pagrįstas sąsukos dirbtinis neuronų tinklas	26
1.5.4. Regionais pagrįstas vien tik sąsukų dirbtinis neuronų tinklas	27
1.5.5. YOLO	27
1.5.6. „SSD“	28
1.5.7. Orientuotų gradientų histograma	28
1.6. Brūkšninių kodų aptikimo sistema paremta giliuoju mokymusi	29
1.7. Trumpas skyriaus apibendrinimas	30
2. GILIOJO MOKYMOSI METODŲ TAIKYMAS QR IR BRŪKŠNINIŲ KODŲ PAIEŠKAI IR ATPAŽINIMUI	31
2.1. Objekto aptikimo iššūkiai	31
2.1.1. Greitis aptikimui realiuoju laiku	31
2.1.2. Klasų disbalansas	31
2.1.3. Įrangos problemos	32
2.2. Metodų įvertinimo metrikos	32
2.2.1. Tikslumas, preciziškumas, jautrumas, F-rodiklis	33
2.2.2. IoU	33

2.2.3. mAP	33
2.2.4. Kadrai per sekundę.....	34
2.3. Tiriami giliojo mokymosi metodai.....	34
2.3.1. YOLOv4.....	35
2.3.2. YoloV5.....	36
2.3.3. SSD_MobileNet_V2	38
2.4. Trumpas skyriaus apibendrinimas	38
3. DUOMENŲ PARUOŠIMAS QR IR BRŪKŠNINIŲ KODŲ PAIEŠKOS IR ATPAŽINIMO METODŲ EKSPERIMENTINIAM TYRIMUI	39
3.1. Atsisiųsti duomenų rinkiniai	39
3.1.1. „ArTe-Lab 1D Medium Barcode“	39
3.1.2. „Muenster BarcodeDB“	40
3.2. Duomenų rinkinio sudarymas	40
3.2.1. Duomenų surinkimas	41
3.2.2. Duomenų žymėjimas.....	41
3.2.3. Duomenų papildymas	44
3.3. Duomenų rinkinio charakteristikos	44
4. QR IR BRŪKŠNINIŲ KODŲ PAIEŠKOS IR ATPAŽINIMO METODŲ TYRIMO REZULTATAI	46
4.1. Kompiuteriniai ištekliai ir programinė įranga	46
4.1.1. Programinės įrangos bibliotekos.....	46
4.1.2. Brūkšninio kodo dekodavimo bibliotekos	48
4.2. Paskirstytoji įterptinė sistema tyrimui	49
4.2.1. Architektūros struktūrinė diagrama	49
4.2.2. Paskirstytos įterptinės sistemos ryšio protokolai	50
4.2.3. Žiniatinklio serverio veikimo algoritmas	50
4.2.4. Jetson serverio veikimo algoritmas.....	52
4.3. Atrinktų metodų tyrimo rezultatai	53
Apibendrinimas	55
4.4. Paskirstytos įterptinės sistemos tyrimo rezultatai	56
Apibendrinimas	58
APIBENDRINIMAS, IŠVADOS	59
LITERATŪRA	60
PRIEDAI	63
A priedas. QR ir brūkšninių kodų paieškos ir atpažinimo metodų eksperimentinių tyrimų rezultatai	63
B priedas. ZBar dekoderio greičio tyrimo duomenų rinkinys pavyzdžiai.....	67

ŽYMENYS IR SANTRUMPOS

ASCII (<i>angl. American Standard Code for Information Interchange</i>)	amerikietiškas informacijos mainų koduotės standartas
BSD (<i>angl. Berkeley Software Distribution</i>)	Berklio programinė įranga
CPU (<i>angl. Central Processing Unit</i>)	centrinis procesoriaus įrenginys
EAN (<i>angl. European Article Number</i>)	Europos prekės numeris
FN (<i>angl. False negative</i>)	klaidingai neigiamas
FP (<i>angl. False positive</i>)	klaidingai teigiamas
GDNT (<i>angl. Deep Neural Network</i>)	gilusis dirbtinių neuronų tinklas
GPL (<i>angl. General Public License</i>)	bendroji viešojoji licencija
GPU (<i>angl. Graphics processing unit</i>)	grafikos procesorius
HOG (<i>angl. Histogram of Oriented Gradients</i>)	orientuotų gradientų histograma
YOLO (<i>angl. You Only Look Once</i>)	žiūrėti tik vieną kartą
IoU (<i>angl. Intersection over Union</i>)	sankirta padalinta iš sąjungos
MB (<i>angl. Mega Byte</i>)	megabaitas
MIT (<i>angl. Massachusetts Institute of Technology</i>)	Masačusetso technologijos institutas
MP (<i>angl. Mega-pixels</i>)	milijonas pikselių
QR (<i>angl. Quick Response</i>)	greito atsako
R-CNN (<i>angl. Region-based CNN</i>)	regioninis sąsūkos dirbtinis neuronų tinklas
R-FCN (<i>angl. Region-based Fully CN</i>)	regioninis pilnai sąsūkos dirbtinis neuronų tinklas
ResNet (<i>angl. Residual Neural Network</i>)	liekanų neuroninis tinklas
SDNT (<i>angl. Convolution Neural Network, CNN</i>)	sąsūkos dirbtinių neuronų tinklai
SPP (<i>angl. Spatial Pyramid Pooling</i>)	erdvinės piramidės sutelkimo
SSD (<i>angl. Single Shot Detector</i>)	vieno šūvio detektorius
TN (<i>angl. True negatives</i>)	teisingai neigiamas
TP (<i>angl. True positive</i>)	teisingai teigiamas
UPC (<i>angl. Universal Product Code</i>)	universalus produkto kodas
VP (<i>angl. mean average precision</i>)	vidutinis preciziškumas

PAVEIKSLŲ SĄRAŠAS

1.1 pav. EAN ir UPC pavyzdžiai	17
1.2 pav. Kodas 39 pavyzdys	17
1.3 pav. Kodas 128 pavyzdys	18
1.4 pav. QR pavyzdys	18
1.5 pav. PDF147 pavyzdys	18
1.6 pav. Duomenų matricos pavyzdys	19
1.7 pav. Dirbtinio intelekto aibės	21
1.8 pav. Aktyvacijos funkcijos: a) ELU b) linijinė c) ReLU d) SELU e) hiperbolinio tangento f) sigmoidinė g) pralaidi ReLU h) parametrinė ReLU	22
1.9 pav. „EfficientNet“ palyginimas	23
1.10 pav. „DenseNet“ architektūra [32]	23
1.11 pav. „DarkNet53“ architektūra [32]	24
1.12 pav. „DarkNet53“ architektūra [31]	24
1.13 pav. R-CNN algoritmo seka	25
1.14 pav. Fast R-CNN architektūros struktūrinė diagrama	26
1.15 pav. Faster R-CNN architektūros struktūrinė diagrama	26
1.16 pav. R-FCN architektūros struktūrinė diagrama	27
1.17 pav. YOLO architektūros veikimo pavyzdys	28
1.18 pav. 1D brūkšninio kodo aptikimo su integruotu giliuoju mokymusi ir geometrinio modeliu struktūrinė schema	29
1.19 pav. Kampo histogramos pavyzdys	29
2.1 pav. Bendra objektų aptikimo struktūrinė diagrama	34
2.2 pav. Pagrindinės objektų aptikimo architektūros dalys	35
2.3 pav. YOLOv5 architektūra	37
2.4 pav. SSD architektūra	38
3.1 pav. „ArTe-Lab 1D Medium Barcode“ duomenų rinkinio pavyzdys	39
3.2 pav. „ArTe-Lab 1D Medium Barcode“ duomenų rinkinio charakteristikos	40
3.3 pav. „Muenster BarcodeDB“ duomenų rinkinio pavyzdys	40
3.4 pav. „Muenster BarcodeDB“ duomenų rinkinio charakteristikos	40
3.5 pav. Naujai padarytų nuotraukų pavyzdžiai	41
3.6 pav. „Labelme“ programinės įrangos langas	42
3.7 pav. Sužymėto rinkinio pavyzdys	42
3.8 pav. Aprėpiamojo lango pavyzdys	44
3.9 pav. Objektų skaičius pagal klases	45
3.10 pav. Objektų pozicijos pasiskirstymas vaizde a) QR kodo, b) brūkšninio kodo, c) duomenų matricos	45
3.11 pav. Objektų skaičius nuotraukose pagal klases	45
4.1 pav. Supaprastinta algoritmo struktūrinės schemos	49
4.2 pav. Žiniatinklio serverio veikimo algoritmo struktūrinė schema	51
4.3 pav. „Jetson“ serverio veikimo algoritmo struktūrinė schema	52
4.4 pav. Apmokymo rezultatai	54
4.5 pav. Objektų skaičius nuotraukose pagal klases	57
4.6 pav. Apmokymo rezultatai	57
4.7 pav. Vykdomo greičio priklausomybė nuo įvesties vaizdo	58

LENTELIŲ SĄRAŠAS

2.1 lentelė. Maišaties matrica	32
2.2 lentelė. Stuburo SDNT tinklų palyginimo eksperimentų rezultatai [21]	35
4.1 lentelė. Debesijos paslaugos teikiamų GPU palyginimas	46
4.2 lentelė. Atrinktų metodų tyrimo rezultatai	53
4.3 lentelė. Tinklo apmokymo su skirtingais duomenis tyrimo rezultatai	54
4.4 lentelė. YOLOv5 skirtingų tinklo dydžių tyrimo rezultatai	55
4.5 lentelė. YOLOv5 tyrimo paskirstytoje įterptinėje sistemoje rezultatai	56

ĮVADAS

Darbo aktualumas ir tikslas

Brūkšninių kodų technologija yra viena iš pagrindinių sėkmingos įmonės priemonių. Tad šiandien juos galime surasti visur. Pavyzdžiui, pašto tarnybos plačiai naudoja brūkšninius kodus, kad galėtų stebėti siuntas visame pasaulyje. Taip pat gamybinės įmonės sandėliavimo tvarkai ir atsekamumui palaikyti, parduotuvių prekių kainoms sužinoti bei daug kitų sričių. Šiai dienai vienmačių kodų skanavimui yra naudojami rašiklio tipo, krūviniai ir lazeriniai skaitytuvai, o dvimačiams kodams yra naudojamos kameros. Tačiau visi šie skaitytuvai turi bendrų minusų. Pirmiausia, daugelis sistemų yra skirtos tik tam tikro tipo brūkšniniam kodams. Antroji dažnai išskylanti problema yra tokia, kad reikia surasti tikslią brūkšninio kodo vietą norint ją nuskanuoti. Šioms problemoms spręsti galima pasitelkti intelektualias sistemas, kurios papildo kameromis paremtų skaitytuvų algoritmus ir leidžia iš vaizdo užfiksuoti keletą brūkšninių kodų vienu metu ir visą tai vykdyti realiuoju laiku.

Kompiuterinė rega yra tarpdisciplininė sritis, kuri pastaraisiais metais sulaukia daugybės dėmesio. Viena iš neatsiejamų kompiuterinės regos sričių yra objektų aptikimas. Ši sritis nagrinėja žmonių, daiktų ir kitų objektų aptikimą ir klasifikavimą [1]. Norint naudoti kompiuterinę regą QR ir brūkšninių kodų aptikimui, pirmiausiai, kompiuterį reikia išmokyti suprasti vaizdus. Tad šiam tikslui yra naudojamas gilusis mokymasis ir jo modeliai greitam bei tiksliam objektų aptikimui ir atpažinimui.

Šio darbo tikslas – ištirti giliojo mokymosi metodų taikymą QR ir brūkšninių kodų paieškai ir atpažinimui vaizduose.

Darbo uždaviniai

Darbo tikslui pasiekti iškelti šie uždaviniai:

1. Atlikti giliojo mokymosi metodų, skirtų objektams aptikti ir atpažinti vaizduose, analitinę apžvalgą.
2. Atlikti objektų atpažinimui reikalingos programinės įrangos licencijų analizę.
3. Atrinkti giliojo mokymosi metodus, perspektyvius taikyti QR ir brūkšninių kodų paieškai ir atpažinimui vaizduose.
4. Paruošti duomenų rinkinį QR ir brūkšninių kodų paieškos ir atpažinimo metodų eksperimentiniams tyrimams.
5. Įgyvendinti ir eksperimentiškai ištirti atrinktus QR ir brūkšninių kodų paieškos ir atpažinimo metodus, ir pateikti tyrimo rezultatus.

Naudoti tyrimo ir analizės metodai

Šiame darbe buvo naudojama nemokama debesijos paslauga „Google Colab“, kuri suteikia galingų kompiuterių išteklius apmokant neuronų tinklus ir atliekant eksperimentus. Taip pat tyrimams naudota paskirstytoji įterptinė sistema įgyvendinta šiame baigiamajame darbe. Abiejose sistemose buvo taikoma aukštojo lygio mašininio mokymo sistemos „PyTorch“ ir „TensorFlow“ bei „OpenCV“, „Numpy“, „Zbar“ ir kitos *Python* programavimo kalba parašytos bibliotekos.

Darbo naujumas ir praktinė nauda

Teksto kodavimas yra plačiai naudojama sritis jau daugelį metų. Mintis panaudoti brūkšninius kodus patikimam ir greitam duomenų įvedimui buvo išvystyta po kompiuterio išradimo, dar 1934 m. Tačiau tuo metu trūko tinkamų skaitytuvų brūkšniniais kodams. Didelis prasiveržimas brūkšninio kodo sistemose įvyko 1960-aisiais ir 70-aisiais, kuomet buvo siekiama suvienyti sistemas ir standartus. Dėl daugelio brūkšninių kodų trūkumų, kompanija „Densio Wave“ nusprendė jį patobulinti. Ji 1994 m. automobilių dalių identifikacijai gamybos procese pradėjo naudoti dviejų dimensijų QR kodą. Vis populiarėjant QR kodams, jų atpažinimui prireikė modernių metodų. Todėl QR kodų, kaip objektų, atpažinimui buvo pradėtas naudoti dirbtinis intelektas. Vienas iš modernių objektų metodų yra 2015 m. Josepho Redmondo pristatytas YOLO tinklas, paremtas sąsūkos dirbtiniais neuronų tinklais, kuris parodė 95 proc. tikslumą atpažįstant veidus bei didelio veikimo spartą, kuri leido šį tinklą panaudoti realaus laiko sistemai. 2020m. atliktas tyrimas atskleidė, kad YOLO V5 yra bene optimaliausias sprendimas renkantis objektų atpažinimo metodą. Literatūroje nėra rasta daug šio metodo panaudojimo būdų, taip pat YOLO V5 nėra plačiai aprašytas moksliniuose straipsniuose. Tad dėl informacijos stygiaus, šis metodas yra ypatingai naujas ir aktualus.

Darbo struktūra

Šis baigiamasis darbas yra sudarytas iš keturių dalių. Pirmoje darbo dalyje aprašomi programinės įrangos licencijavimas, brūkšninių ir QR kodų skirtumai bei ypatumai, dirbtinių neuronų tinklai ir pagrindiniai objektų aptikimo metodai. Aprašomos pagrindinės sąsūkos dirbtinių neuronų tinklų architektūros dalys. Antroje darbo dalyje trumpai apibūdinami pagrindiniai iššūkiai aptinkant objektus vaizduose. Palyginami modernūs giliojo mokymosi metodai perspektyvūs kodų paieškai ir atpažinimui. Aprašomos pagrindinės įvertinimo metrikos ir modelių architektūros dalys. Remiantis įvairiais tyrimais, išrenkamas metodas, kuris yra perspektyviausias taikyti QR ir brūkšninių kodų paieškai. Trečioje darbo dalyje aprašomas duomenų rinkinio sudarymo etapai bei pagrindinės metrikos jo įvertinimui. Ketvirtoje darbo dalyje pateikiami atliktų eksperimentų rezultatai.

1. GILIOJO MOKYMO SI METODŲ OBJEKTAMS ATPAŽINTI

ANALITINĖ APŽVALGA

Šiame skyriuje aptariamos pagrindinės programinės įrangos licencijavimo kategorijos bei plačiai išanalizuojamos ir palyginamos laisvosios licencijos, kurios yra naudojamos šio darbo programinėje įrangoje. Taip pat supažindinama su pagrindiniais vienmačiais ir dvimačiais brūkšniniais kodais ir jų komercinėmis sistemos. Toliau aptariamas objektų aptikimas vaizduose, dirbtiniai neuronų tinklai ir nagrinėjama sąsūkos dirbtinių neuronų tinklų pagrindinės dalys, parametrai ir iš jų sudaryti pagrindiniai modeliai. Galiausiai apžvelgiami giliojo mokymosi modeliai objektų atpažinimui ir viena įgyvendinta brūkšninių kodų aptikimo ir dekodavimo sistema.

1.1. Atvirojo kodo programinės įrangos licencijavimas

Programinės įrangos licencija yra sutartis tarp subjekto, kuris sukūrė ir tiekė programinę įrangą, ir jo galutinio vartotojo. Licencija yra tekstinis dokumentas, skirtas apsaugoti programinės įrangos kūrėjo intelektualinę nuosavybę ir apriboti bet kokias pretenzijas, kurios gali kilti dėl jos naudojimo. Dauguma programinių įrangų patenka į vieną iš dviejų kategorijų, kurios skiriasi tuo, kaip į ją žiūrima pagal autorių teisių įstatymus:

- Nuosavybinė licencija (*angl. Proprietary Licence*) – teisės į programą ar kitą autorinį kūrinį priklauso jį užsakiusiai įmonei ar asmeniui. Vartotojui leidžiama naudotis programa licencijoje nustatytais sąlygomis.
- Laisvoji licencija (*angl. Free Licence*) – licencija, leidžianti vartotojui kopijuoti bei tobulinti kūrinį, platinti jį bei juo remiantis sukurtus kūrinius, tačiau gali riboti arba drausti toliau platinamų kūrinių licencijos keitimą.

Kadangi darbe daugiau skiriama dėmesio laisvosioms licencijoms, todėl šiame poskyryje yra apžvelgiama kokio tipo šių licencijų galima sutikti nagrinėjant QR ir brūkšninių kodų paieškos ir atpažinimo vaizduose.

1.1.1. MIT

MIT licencija suteikia vartotojams aiškų leidimą pakartotinai naudoti kodą bet kokiam tikslui, kartais net jei kodas yra patentuotos programinės įrangos dalis. Tol, kol vartotojai į savo platinimą įtrauks originalią MIT licencijos kopiją, jie gali atlikti bet kokius kodo pakeitimus ar modifikacijas, kad atitiktų savo poreikius. MIT yra viena paprasčiausiu ir lengviausiai įsisavinamų laisvųjų licencijų, kurios apibrėžime yra tik 172 žodžiai. Pagrindiniai reikalavimai yra kad į kodo kopiją ir (arba) modifikaciją reikia įtraukti du dalykus:

- Originalų pranešimą apie autorių teises

- Pačios licencijos kopiją

Svarbus MIT licencijos naudojimo pranašumas yra tas, kad ji vienodai naudinga abiem pusėms. Kai kurie kūrėjai mano, kad GNU GPL licencijos yra pernelyg liberalios, o kiti mano, kad visa programinė įranga ilgainiui turėtų tapti nuosavybės teise. Atvira MIT licencija tinka tiek bendruomenės kūrėjams, tiek komandoms, norinčioms kurti patentuotą programinę įrangą, naudojant esamus MIT licencijuoto kodo elementus.

1.1.2. GPL-3.0

Kaip ir GPL v2, GPL 3 yra stipri „copyleft“ licencija, o tai reiškia, kad bet kokia originalaus kodo kopija ar modifikacija taip pat turi būti išleista pagal GPL v3. Kitaip tariant, panaudojus GPL 3 licencijuotą kodą ir norint jį pradėti platinti, versijai yra taikomi tie patys licencijos reikalavimai. Tai reiškia, kad ji turi atitikti GPL v3 t. y. bet kas gali matyti ir pasiekti pakeistą kodą ir įdiegti jį savo reikmėms. Tad pagrindiniai licencijos reikalavimai:

- Pridėti viso licencijos teksto kopiją
- Nurodyti visus reikšmingus originalios programinės įrangos pakeitimus
- Pateikti pirminį šaltinio kodą, kai platinami dvejetainiai failai, pagrįsti licencijuotu darbu
- Pridėti originalaus pranešimo apie autorių teises kopiją

1.1.3. BSD

BSD atvirojo kodo programinės įrangos licencijų šeima turi ilgą istoriją. Pirmoji versija buvo sukurta UC Berkeley 1980 m., siekiant licencijuoti naują universiteto UNIX operacinę sistemą „Berkeley Software Distribution“ arba BSD OS. Ši pradinė versija, žinoma kaip 4-Clause BSD licencija, kuri šiandien jau nėra naudojama, taip pat kaip ir 0-Clause variantas (kuris iš esmės yra lygiavertė viešosios nuosavybės licencija). Tad šiai dienai yra likę 2 naudojami variantai: 2-Clause ir 3-Clause. BSD licencijai iš programinės įrangos kūrėjo daug nereikalauja. Tiesą sakant, BSD 3-Clause licencijos reikalavimai yra labai panašūs į plačiai naudojamos MIT licencijos reikalavimus. Kaip ir MIT licencija, BSD 3-Clause licencijos straipsnis yra labai trumpas. BSD 3-Clause licencijoje yra du pagrindiniai reikalavimai tiems, kurie nori naudoti licencijuotą kodą. Jie iš tikrųjų yra tokie patys kaip MIT licencijos. Norint kopijuoti, modifikuoti ar platinti bet kurį kodą, licencijuotą pagal BSD, turite įtraukti:

- Originalų pranešimą apie autorių teises
- Pačios licencijos kopiją

1.1.4. „Apache License 2.0“

„Apache License 2.0“ yra laisvoji licencija, o tai reiškia, kad vartotojai, naudodami kodą, gali daryti (beveik) viską, ko nori, išskyrus keletą išimčių. Tai suteikia programinės įrangos vartotojui

laisvę ją naudoti bet kokiems tikslams: platinti, modifikuoti ir platinti modifikuotas tos programinės įrangos versijas. Tačiau „Apache“ licencija nereikalauja, kad modifikuotos programinės įrangos versijos būtų platinamos naudojant tą pačią licenciją. „Apache“ licencijoje reikalaujama, kad gavėjai būtų informuojami tik apie tai, jog platinant buvo naudojamas Apache licencijos kodas, priešingai nei „copyleft“ licencijose. Pagrindiniai licencijos reikalavimai:

- Originalus pranešimas apie autorių teises
- Pačios licencijos kopija
- Svarbių pradinio kodo pakeitimų paskelbimas

Svarbių pradinio kodo pakeitimų paskelbimas yra pagrindinis reikalavimas Apache licencijos 2.0, lyginant su kitomis laisvosiomis licencijų skirtumas. Jei atliekami esminiai licencijuoto kodo pakeitimai, privaloma juos paskelbti viešai sekančioje atnaujintoje versijoje, kuri bus platinama.

Kaip ir MIT licencijoje, „Apache“ licencijoje 2.0 reikalaujama naudojant kodą įtraukti originalų pranešimą apie autorių teises bei pilną licencijos kopiją. Taip pat teigiama, kad visi, kurie žymiai modifikuoja kodą, privalo aprašyti savo pakeitimus. Be to, atvirojo kodo bibliotekoje turi būti failas „NOTICE“ su priskyrimo pastabomis. Nusprendus naudoti kodą su šia licencija yra privaloma įtraukti failą „NOTICE“. Kartu su šiais apribojimais yra keletas papildomų privalumų. „Apache 2.0“ aiškiai suteikia autorių teisių savininkams teisę reikalauti savo darbų patentų (nors daugelis ekspertų teigia, kad MIT licencijos tekste pateikiama panaši patentų apsauga). Be to, kiekvienas, kuris naudoja kodą, gali suteikti licencijuotos programinės įrangos garantiją.

1.2. Brūkšninių kodų tipų ir komercinių sistemų apžvalga

Brūkšninių kodų technologija yra viena pagrindinių sėkmingos įmonės priemonių. Tad šiandien juos galime surasti visur. Pavyzdžiui, ligoninėse vaistų brūkšniniai kodai užtikrina, kad vaistai skiriami tinkamam pacientui. Pašto tarnybos plačiai naudoja brūkšninius kodus, kad galėtų stebėti siuntas visame pasaulyje. Taip pat ši technologija taikoma industrinės įmonės sandėliavimo tvarkai ir atsekamumui bei daugelyje kitų sričių.

Vartotojai dažnai susiduria su QR kodais norėdami atsisiųsti mobiliąją programėlę ar užsisakyti maistą restorane. Tad nėra nei dienos, kai žmogus vienaip ar kitaip nesusidurtų su brūkšniais kodais. Šiame poskyryje trumpai aptariama su kokias kodais susiduriama bei koks kodavimo principas yra naudojamas.

1.2.1. Vienmačiai brūkšniniai kodai

Populiariausias brūkšninių kodų formatas yra universalus gaminio kodas (*angl. Universal Product Code*) formatas, kuris yra visuose JAV prekybos centrų produktuose. Pradėtas naudoti nuo 1970-ųjų pradžios, šis formatas yra žinomas visame pasaulyje ir yra visuotinai pripažintas. Įprastą UPC kodą sudaro 12 skaitmenų. Pirmasis skaitmuo nurodo, kokio tipo gaminyje yra kodas

(mažmeninė prekyba, farmacija ir kt.). Kiti penki skaitmenys identifikuoja konkretų to gamintojo gaminį. Paskutinis skaitmuo yra kontrolinis skaitmuo, naudojamas nustatyti, ar brūkšinių kodų skaitytuvas teisingai perskaitė vienuoliką pirmųjų skaitmenų. Europoje populiariausias yra Europos prekės numeris (*angl. European Article Number*), kuris panašus į UPC, tačiau turi papildomą skaitmenį, identifikuojantį šalį, iš kurios kilęs produktas.



1.1 pav. EAN ir UPC pavyzdžiai

Kodas 39 (*angl. Code 39*) yra dviejų lygių kodas, skirtas koduoti tiek raides, tiek skaičius. Standartinėje versijoje užkoduotos didžiosios raidės A – Z, skaičiai ir keli skyrybos ženklai. Žvaigždutė (*) visada naudojama kaip pradžios ir pabaigos simbolis. Išplėstinė šio kodo versija koduoja visus 128 ASCII simbolius. 39 kodas yra tapęs vyriausybės, gamybos, brūkšinių kodų pramonės, švietimo ir verslo programų standartu. Jis vadinamas 39 kodu, nes kiekvienas simbolis susideda iš devynių elementų, penkių juostų ir keturių tarpų. Trys iš devynių elementų yra platūs, o likę šeši – siauri (žr. 1.2 pav.).

Code 39



1.2 pav. Kodas 39 pavyzdys

Kodas 128 (*angl. Code 128*) – keturių lygių kodas, reiškiantis, kad juostos ir tarpai gali būti keturių skirtingų pločių. Yra trys kodo 128 versijos. A versija užkoduoja visus didžiuosius raidinius ir skaitmeninius simbolius bei visus ASCII valdymo simbolius. B versijoje užkoduoti visi viršutiniai ir apatiniai raidiniai skaitmeniniai simboliai. C versija koduoja tik skaičius. Kodo simbolių rinkinius galima perjungti naudojant „Shift“ simbolius. 128 kodo pranašumas yra tai, kad jis gali užkoduoti visus ASCII simbolius kuo trumpesniu kodo ilgiu. Trūkumas yra tas, kad jis turi keturis skirtingus juostų ir erdvės pločius, o ne du, todėl spausdinimo ir dekodavimo technologijoms keliama daugiau reikalavimų.

Code 128



1.3 pav. Kodas 128 pavyzdys

1.2.2. Dvimačiai brūkšniniai kodai

Greito atsako (*angl. Quick Response (QR)*) yra vieni plačiausiai pripažįstamų dvimačių brūkšninių kodų, nes jie plačiai naudojami skaitmeniniame lygmenyje. QR kodus pirmą kartą 1994 m. sukūrė „Denso“ įmonė, priklausanti „Toyota“ įmonių grupei. QR kodas sugeba užkoduoti iki 2 509 skaitinių arba 1 520 raidinių ir skaitinių simbolių ir turi tris klaidų aptikimo lygius. QR kodai yra mažiausiai 21×21 langeliai, tačiau jų dydis gali padidėti po 4×4 langelius iki maksimalaus 105×105 langelių dydžio.



1.4 pav. QR pavyzdys

PDF417 naudojamas įvairiose srityse: transportui, identifikavimo kortelėms ir atsargų valdymui. PDF reiškia nešiojamas duomenų failas (*angl. Portable Data File*), kurį sukūrė „Symbol Technologies“. PDF417 turi integruotą vidinį klaidų taisymą, siekiant užtikrinti geresnį įskaitomumą. Šie kodai gali užkoduoti nuo vieno iki dviejų šimtų ženklų simbolių arba daugiau nei kilobaitų duomenų etiketėje.



1.5 pav. PDF147 pavyzdys

Duomenų matrica (*angl. Data Matrix*) – brūkšninis kodas, galintis užkoduoti iki 3 116 simbolių iš 256 baitų ASCII simbolių rinkinio. Tai didelio tankio dvimatis brūkšninis kodas, siūlantis didesnę duomenų tankį, palyginti su PDF417 brūkšniniais kodais. Duomenų matricos brūkšniniai kodai yra sukonfigūruoti kvadratinėje lentelėje su iešiklio modeliu aplink simbolio kraštus, kurie leidžia skaitytuvams atpažinti brūkšninį kodą ir jį perskaityti, neatsižvelgiant į kodo orientaciją. Kaip ir

kituose 2D brūkšniniuose koduose, duomenų matricos brūkšniniuose koduose yra įmontuotos klaidų taisymo priemonės, užtikrinančios duomenų vientisumą, net jei kodas yra sugadintas fiziškai.



1.6 pav. Duomenų matricos pavyzdys

1.2.3. Brūkšinių kodų atpažinimo sistemos

Rašiklio stiliaus brūkšinių kodų skaitytuvas (*angl. Pen-type reader*) yra rašiklio dydžio kodų skaitytuvas, kurį sudaro šviesos diodas ir foto-diodas antgalyje. Vartotojas perbraukia šiuo skaitytuvu per brūkšninį kodą, tuo metu diodas matuoja šviesos atspindį nuo brūkšninio kodo ir apskaičiuoja kiekvienos juostos plotį. Ši informacija leidžia skaitmeniniu būdu nuskaityti brūkšninį kodą ir informacija perduodama į kitą skaitmeninį įrenginį apdoroti. Didžiulis šio skaitytuvo minusas, kad brūkšninį kodą surasti ir per jį perbraukti turi pats vartotojas, kas užima labai daug laiko [1].

Krūvio sąsajos įtaiso (*angl. charge coupled device*) skaitytuvo technologija yra žinoma kaip linijinis brūkšinių kodų skaitytuvas. Taikant šią technologiją, skaitytuvas turi šimtus mažų šviesos jutiklių, kurių pagalba yra fiksuojama brūkšninio kodo skleidžiama aplinkos šviesa. Didžiausias šių įrenginių minusas yra tai, kad jie brangūs, tačiau labai tikslūs ir universalūs.

Šiek tiek pažangesnis nei rašiklio skaitytuvas – lazerinis brūkšinių kodų skaitytuvas, kuris gali tiksliau skaityti šviesą, kad būtų išvengta klaidingų teigiamų rezultatų ar skaitytuvo klaidų. Lazerio skaitytuve lazerio spindulys šaudomas į veidrodį, esantį įrenginyje. Šis veidrodis šviesą nukreipia taip, kad lazeris tiesia linija perbrauktų brūkšninį kodą. Tuomet gaunamas šviesos atspindys diode, kurio lygis yra matuojamas bei paverčiamas skaitmeniniu brūkšninio kodo signalu. Lazeriniai skaitytuvai gali būti sumontuoti skenavimo įrenginyje arba būti rankinio įrenginio tipo [1].

Iš prieš tai aptartų brūkšinių kodų tipų žinoma, kad yra dvimačiai kodai, kurių neįmanoma nuskaityti standartinėmis aukščiau aptartomis priemonėmis. Norint perskaityti šiuos brūkšninius kodus, būtinas dvimatis kameros vaizdų skaitytuvas. Pagrindinis technologijos principas yra nufotografuojamas brūkšninis kodas ir nusiunčiamas į programinę įrangą kaip skaitmeninis signalas. Tuomet programinė įranga iššifruoja informaciją. Šis metodas gali skaityti visų tipų brūkšninius ir QR kodus, tačiau reikalauja geros techninės įrangos, dėl kurios skaitytuvo kaina yra gana didelė.

Apibendrinant visas sistemas galima išskirti keletą bendrų minusų, su kuriais susiduria kodų vartotojai. Pirmiausia daugelis sistemų yra skirtos tik tam tikro tipo brūkšniniams kodams, o universalios sistemos yra brangios. Antroji dažnai iškylanti problema tai, kad reikia surasti tikslią brūkšninio kodo vietą norint ją nuskenuoti. Tad šioms problemoms spręsti galima pasitelkti

intelektualiąsias sistemas, kurios papildys kameromis paremtų skaitytuvų algoritmus, kad iš padarytos nuotraukos galima būtų užfiksuoti keletą brūkšninių kodų vienu metu ir visa tai vykdytų realiu laiku.

1.3. Objektų aptikimas vaizduose

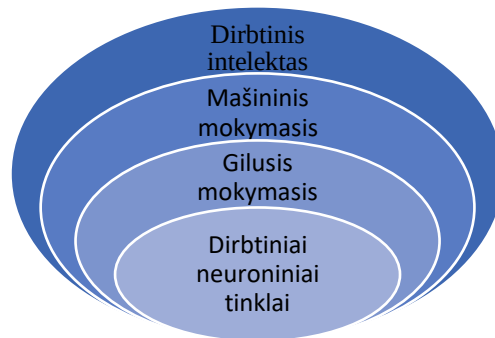
Kompiuterinė rega yra tarpdisciplininė sritis, kuri pastaraisiais metais sulaukia daugybės dėmesio. Viena iš neatsiejamų kompiuterinės regos sričių yra objektų aptikimas. Ši sritis nagrinėja žmonių, daiktų ir kitų objektų aptikimą ir klasifikavimą [1]. Pagrindinis skirtumas tarp aptikimo ir klasifikavimo algoritmų yra tai, kad pirmojo algoritmo rezultatas yra vaizde nubrėžtas keturkampis langas, vaizduojantis objekto vietą. Be to, objekto aptikimo atveju, nebūtinai reikia nupiešti tik vieną keturkampį, paveikslėlyje gali būti daugybė ribojančių langelių, vaizduojančių skirtingus objektus. Šiuo metu labiausiai paplitę metodai objekto aptikimo vaizde uždaviniui spręsti yra paremti giliuoju mokymusi ir dirbtiniais neuronų tinklais, kurie yra apžvelgiami šiame poskyryje.

1.3.1. Dirbtiniai neuronų tinklai

Pirmąjį neurono modelį 1943 m. paskelbė Varenas Makulokas ir Valteris Pitsas, o dar po penkiolikos metų Frankas Rosenblatas sukūrė pirmąjį kelių sluoksnių perceptroną (*angl. perceptron*). Po 1969 metų tyrimo, kurį atliko Minskis ir Popertas DNT srities tyrimai sulėtėjo, nes buvo nustatyta, kad vieno sluoksnio neuronų tinklai negali atlikti išskirtinių loginių operacijų, o to meto kompiuterių skaičiavimo pajėgumai negalėjo apskaičiuoti didelių matmenų DNT. Sekantis proveržis šioje srityje atsirado kai Verbosas 1974 m. savo disertacijoje pasiūlė dirbtinių neuronų tinklų mokymo metodą, pavadinimu, atgalinio klaidų sklidimo (*angl. Back Propagation*) algoritmas. Šis mokymo metodas pirmą kartą leido įgyvendinti daugiasluoksnius DNT, kuriuos pasiūlė Rosenblatas. Toliau sekė Lekuno ir bendraautorių 1990 m. paskelbtas darbas, kuris padėjo pamatus sąsūkos dirbtiniams neuronų tinklams (*angl. Convolution Neural Network*) [2].

1.3.2. Gilieji dirbtiniai neuronų tinklai

Pradėjus naudoti daugiasluoksnius perceptronus 1980 m. Fukushima pasiūlė neurokognitroną, kuris yra laikomas giliųjų dirbtinių neuronų tinklų (*angl. Deep Neural Network*) pirmtaku. Šie algoritmai pasiekė gerų rezultatų įvairiose srityse, įskaitant kompiuterinę regėjimą, kalbos atpažinimą, stalo ir vaizdo žaidimus. Vienas iš pagrindinių GDNT tyrinėjimo sričių yra architektūrų projektavimas (kartu su jų hiperparametrų optimizavimu) bei tinklo mokymo metodai. Pastaraisiais metais tinkamų architektūrų paieška susilaukia vis daugiau susidomėjimo. GDNT algoritmai yra mašininio mokymosi poaibis (žr. 1.8 pav.), naudojantis kelis sluoksnius, siekiant palaipsniui išgauti aukštesnio lygio funkcijas iš neapdorotų duomenų. Giliojo mokymosi metu kiekvienas sekantis lygis bando mokytis, kad įvesties duomenis paverstų vis labiau abstraktesniu ir bendresniu vaizdu [3].



1.7 pav. Dirbtinio intelekto aibės

Svarbu pabrėžti, kad GDNT evoliucija buvo pritaikyta kuriant daugybę kitų DNT, įskaitant sąsūkos dirbtinius neuronų tinklus (*angl. Convolution Neural Network*), automatinis dekodatorius (*angl. Autoencoders*), giluminio įsitikinimo tinklus (*angl. Deep Belief Networks*) ir pasikartojančius neuroninius tinklus (*angl. Recurrent Neural Networks*) [3].

1.3.3. Sąsūkos dirbtiniai neuronų tinklai

Pastaraisiais metais SDNT parodė puikių rezultatų įvairiose srityse, tokiose kaip vaizdų klasifikavimas (*angl. Image Classification*), modelių atpažinimas (*angl. Pattern Recognition*) ir daugialypės terpės glaudinimas (*angl. Multi-media Compression*) [2]. SDNT yra dirbtinis neuronų tinklas sudarytas iš sąsūkos sluoksnių kuriuose atskiri neuronai apdoroja įvesties vaizdą slenkančiu langu, apskaičiuojant lange esančių verčių sandaugų rezultatų sumą su sąsūkos filtrų vertėmis. SDNT sukurtas remiantis biologiniais procesais ir yra daugiasluoksniu perceptoru variacija, kuri pasiekia daug mažesnius skaičiavimo resursus bei leidžia iš vaizdo išgauti dvimačius arba trimačius požymius. SDNT naudojami kompiuterinės regos, automatinio kalbos atpažinimo ir natūralios kalbos apdorojimo, garso atpažinimo ir bioinformatikos srityse, kuriose pasiekia itin gerų rezultatų [4].

Klasikinė SDNT architektūra yra sudaryta iš sąsūkos sluoksnių, sutelkimo (*angl. Pooling*) arba atrinkimo (*angl. Subsampling*) sluoksnių, visiškai sujungtų DNT neuronų, susiorganizuojančių požymių žemėlapių (*angl. Self-organizing Maps*). Sąsūkos sluoksniuose per įėjimo duomenų matricą yra slenkamas pasirinkto dydžio $N \times M$ filtro langas su tam tikromis vertėmis žingsniu n per stulpelius ir m per eilutes. Po kiekvieno filtro jo išėjime yra suformuojamas požymių žemėlapis, kurio dydis $(N/n) \times (M/m)$ taškų. Naujai gauti požymių žemėlapiai apdorojami sutelkimo ar atrinkimo sluoksniuose kuriuose yra skaičiuojamas vienas iš pasirinktų dydžių: lango verčių vidurkis, mediana, minimumas, maksimumas ar kitokia funkcija. Po šio veiksmo gaunamas sutelktas požymių žemėlapis, kuris vėl yra apdorojamas sąsūkos sluoksniu. Šis kelių sluoksnių procesas kartojamas kol požymių žemėlapių masyvų dydis pakankamai sumažėja ir yra tinkamas visiškai sujungtų neuronų sluoksniui, kurie atlieka galutinę algoritmo užduotį – klasifikavimą.

SDNT naudoja atgalinio klaidos sklaidimo algoritmus neuroninio tinklo mokymosi procesui, o jo spartinimui dažniausia naudojamas išlyginta tiesinė aktyvavimo funkcija (*angl. Rectified Linear Unit, ReLU*). Aktyvacijos funkcija yra būtina, kad neuronų tinklui pridėtų netiesiškumo. Pagrindinės aktyvacijos funkcijos yra linijinė, kurios matematinė išraiška parodyta formulė 1.1, sigmoidinė (*angl. Sigmoid*) – formulė 1.2, hiperbolinio tangento (*angl. Tanh*) – formulė 1.3, ReLU – formulė 1.4, pralaidi išlyginta tiesinė aktyvavimo (*angl. Leaky ReLU*) – formulė 1.5, parametrinė išlyginta tiesinė aktyvavimo (*angl. Parametric ReLU*) – formulė 1.6, eksponentinė tiesinė aktyvavimo (*angl. Exponential Linear Unit, ELU*) – formulė 1.7, (žr. 1.8 pav.).

$$f(x) = x, \quad (1.1)$$

$$f(x) = \frac{1}{1+e^{-x}}, \quad (1.2)$$

$$f(x) = \frac{(e^x - e^{-x})}{(e^x + e^{-x})}, \quad (1.3)$$

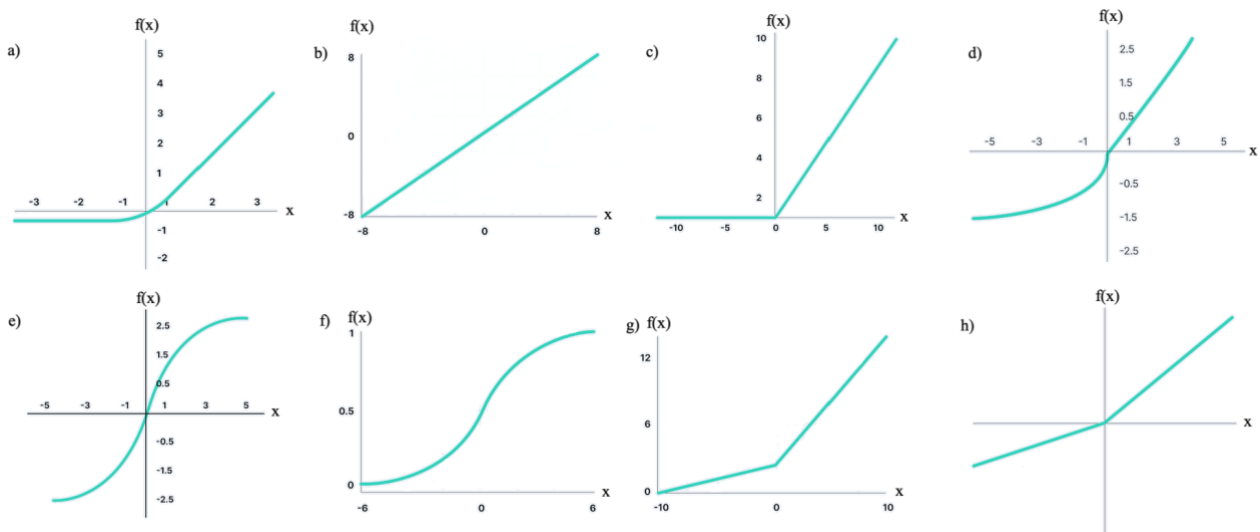
$$f(x) = \max(0, x), \quad (1.4)$$

$$f(x) = \max(0.1x, x), \quad (1.5)$$

$$f(x) = \max(ax, x), \quad (1.6)$$

$$f(x) = \begin{cases} x, & \text{kai } x \geq 0 \\ \alpha(e^x - 1), & \text{kai } x < 0 \end{cases} \quad (1.7)$$

$$f(\alpha, x) = \lambda \begin{cases} \alpha(e^x - 1), & \text{kai } x < 0 \\ x, & \text{kai } x \geq 0 \end{cases} \quad (1.8)$$

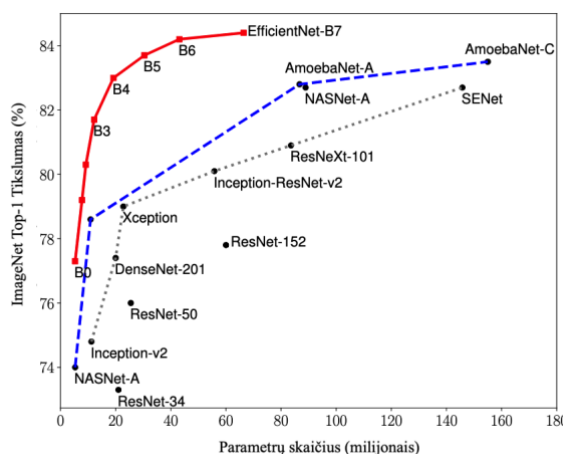


1.8 pav. Aktyvacijos funkcijos: a) ELU b) linijinė c) ReLU d) SELU e) hiperbolinio tangento f) sigmoidinė g) pralaidi ReLU h) parametrinė ReLU

1.4. Sąsukos dirbtinių neuroninių tinklų architektūros

1.4.1. „EfficientNet“

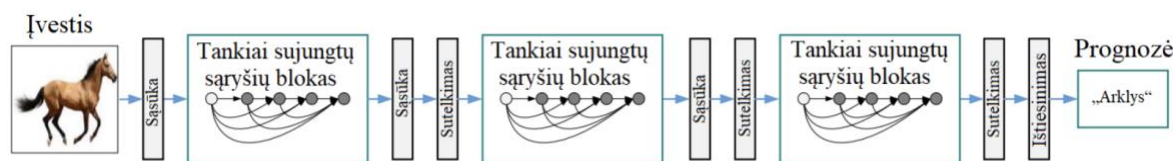
2019 metais M. Tanas ir V. Le pristatė naują tinklą, kuriame labai plačiai yra apžvelgiama SDNT tinklų didinimas į gylį arba plotį. Užuoat radę geriausią architektūrą, autoriai pasiūlė pradėti nuo palyginti mažo bazinio modelio ir palaipsniui didinti jo mastelį. Taip susiaurindami tinklo dizainą. Siekdami dar labiau apriboti dizaino erdvę, autoriai apriboja visus sluoksnius iki vienodo mastelio su pastoviu santykiu. Tokiu būdu pasiekė lengviau išsprendžiamą optimizavimo problemą [5]. 1.9 paveiksle yra palyginamas EfficientNet su kitomis populiariomis architektūromis. Nustatyta, kad tinklui pavyko pasiekti didesnę tikslumą su daug kartų mažiau parametrų.



1.9 pav. „EfficientNet“ palyginimas

1.4.2. „DenseNet“

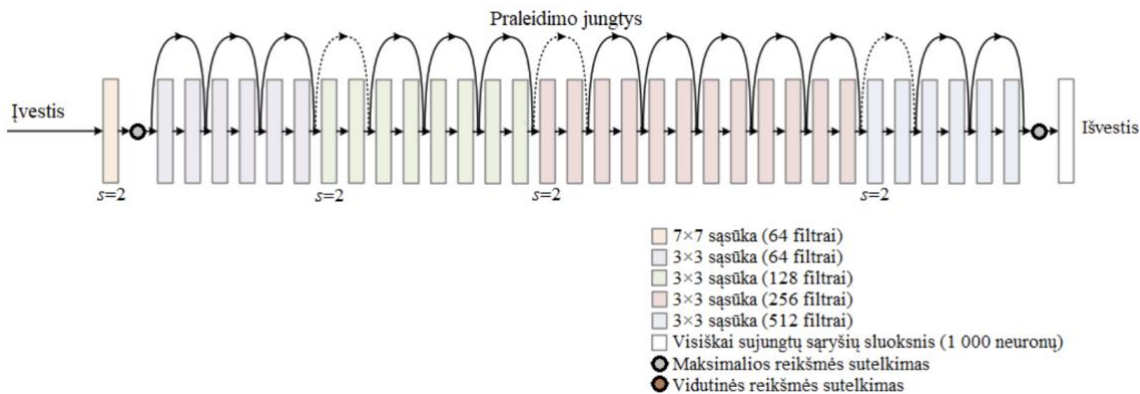
„DenseNet“ – tai tinklo architektūra, kuri užtikrina ankstesnių sąsukos operacijų duomenų prieinamumą kitiems sluoksniams. Architektūros pagrindinis elementas – tankiai sujungtų sąryšių blokas t. y. prieš tai buvusio sluoksnio išvesties požymių žemėlapis yra sujungtas su šiuo bloku. Kitaip nei „ResNet“ neuroniniame tinkle, čia požymių žemėlapiai yra sujungiami (*angl. Concatenated*), o ne sumuojami. Autoriai teigia, kad požymių žemėlapių, išmokyto skirtinguose sluoksniuose, sujungimas leidžia padidinti kitų sluoksnių įėjimo duomenų variacijas ir pagerinti efektyvumą. Sluoksnio išvesties požymių žemėlapių skaičius apibrėžiamas kaip augimo greitis (*angl. Growth Rate*), kuris kontroliuoja kiek naujos informacijos kiekvienas sluoksnis įneša į tinklą (žr. 1.10 pav.).



1.10 pav. „DenseNet“ architektūra [32]

1.4.3. „ResNet“

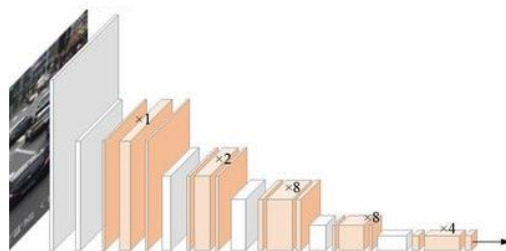
2015 metais „ILSVRC“ konkurse pirmą vietą užėmė K. He ir kt., kurie sukūrė ResNet neuronų tinklą (angl. *Residual Neural Network*). Pagrindinė tinklo architektūros idėja yra liekamieji blokai (angl. *Residual Block*), kurių pagalba kiekviename sluoksnyje yra naudojama ankstesnio sluoksnio išėjimas bei nepaliesti originalūs nauji duomenis, kurių pagalba išgaunami požymiai. ResNet naudoja trumpesnio kelio jungtis, kad būtų peršokama per 2–3 sluoksnius. Tinklo architektūra pavaizduota 1.11 paveiksle [6].



1.11 pav. „DarkNet53“ architektūra [32]

1.4.4. „Darknet53“

„Darknet53“ yra SDNT, kuris buvo išrastas kaip YOLOv3 objektų aptikimo metodo pagrindas. Jo pirmtako „Darknet-19“ patobulinimai apima likusių jungčių naudojimą bei papildytą sluoksnių skaičių. Apibendrinta architektūra pavaizduota 1.12 paveiksle.



1.12 pav. „DarkNet53“ architektūra [26]

1.5. Giliojo mokymosi modeliai objektų atpažinimui

Pagrindinė priežastis, kodėl aukščiau aptarti standartiniai sąsūkos dirbtiniai neuronų tinklai naudojami tik kaip sudedamoji dalis todėl, kad yra nežinoma kokia turėtų būti išvestis. Ar bus aptiktas tik vienas objektas ar jų grupė. Paprasčiausias būdas išspręsti šią problemą būtų pasirinkti įvairius dominančius regionus iš įvesties vaizdo ir jiems pritaikyti SDNT, kad klasifikuotų objektą pasirinktame regione. Šio būdo didžiulė problema yra ta, kad dominantys objektai gali turėti skirtingą vietą ir proporcijas vaizde. Todėl būtume priversti pasirinkti daug regionų ir atlikti begalę

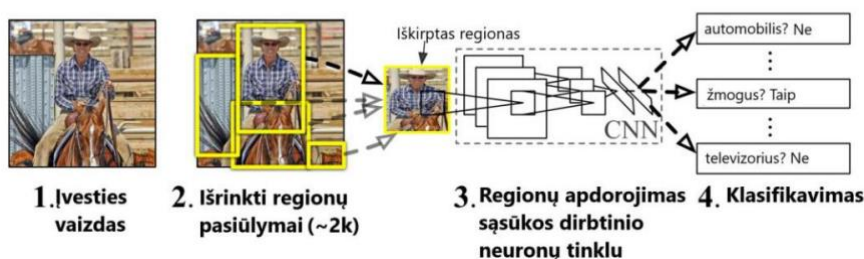
matematinų skaičiavimų, kuriems reikėtų daug kompiuterinių resursų. Spręsti šiai problemai buvo sukurti algoritmai, kurie šiame poskyryje yra aptariami.

1.5.1. Regionais pagrįstas sąsūkos dirbtinis neuronų tinklas

Vienas iš pirmųjų metodų sukurtas spręsti didelio langu skaičiaus problemai – regionais pagrįstas sąsūkos dirbtinis neuronų tinklas (*angl. region-based convolutional neural networks*). Buvo pasiūlyta naudoti atrankinę paiešką, kad iš įvesties vaizdo išrinktu apie 2000 langu, kurie tolimesnėje eigoje pavadinti regionų pasiūlymai (*angl. region proposals*) [7]. Ši atrankinė paieška yra vykdoma pasitelkiant selektyvios paieškos algoritmą, kuris sudarytas iš 3 etapų [8]:

- Regionų pogrupių sukūrimas iš smulkių langu
- Rekursyviai sujungiami panašūs regionai naudojant godųjį algoritmą (*angl. greedy algorithm*)
- Sujungtųjų regionų pagalba sugeneruojami galutiniai kandidatai.

Gauti du tūkstančiai regionų yra iškerpami ir apdorojami sąsūkos dirbtinio neuronų tinklo, kuris klasifikuoja kiekvieno iškirpto regiono objektą (žr. 1.13 pav.).



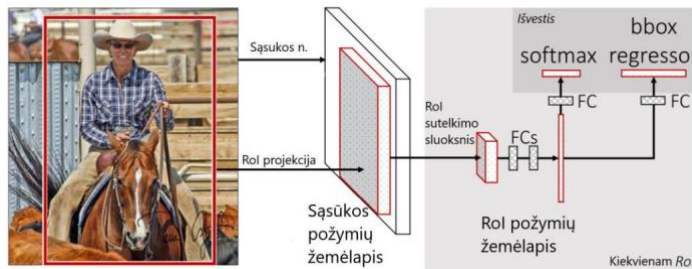
1.13 pav. R-CNN algoritmo seka

Pagrindinės šio metodo problemos yra tai, kad tinklo mokymas užima daug laiko, nes reikia klasifikuoti 2000 regionų vienam vaizdui, taip pat nėra galimybės įgyvendinti realiuoju laiku, nes kiekvienam bandomajam vaizdui reikia maždaug 47 sekundžių. Taip pat selektyvios paieškos algoritmas yra fiksuotas ir jo apmokymas negalimas, todėl yra nemažai siūloma prastų regionų.

1.5.2. Greitas regionais pagrįstas sąsūkos dirbtinis neuronų tinklas

Ankstesniojo metodo autorius (R-CNN) išsprendė kai kuriuos R-CNN trūkumus ir sukūrė greitesnį objektų aptikimo algoritmą, kurį pavadino – greitas regionais pagrįstas sąsūkos dirbtinis neuronų tinklas (*angl. Fast region-based convolutional neural networks*). Metodas yra panašus į R-CNN algoritmą, tačiau užuot pateikę visus regiono pasiūlymus sąsūkos dirbtiniam neuronų tinklui, metodas tinklui pateikia visą įvesties vaizdą, kad sukurtų sąsūkos požymių žemėlapi. Pagal šį žemėlapi yra pasiūlomi regionai ir sujungiami į kvadratus tuo pačiu selektyviosios paieškos metodu. Toliau naudojant „RoI“ sutelkimo sluoksnį juos pertvarko į fiksuoto dydžio ir panaudoja kaip įvestį

visiškai sujungtiems neuronams. Iš „RoI“ požymių vektoriaus „softmax“ klasifikatoriaus pagalba nustato siūlomo regiono klasę ir ribojančio langelio poslinkio vertes. Viso šio proceso architektūra pavaizduota 1.14 paveiksle [9].

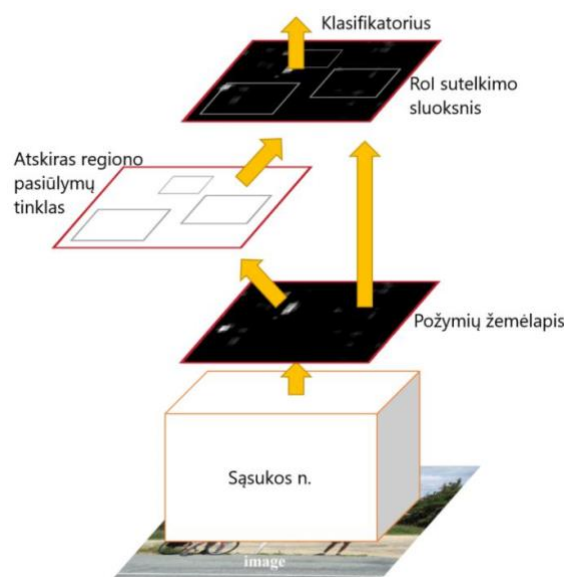


1.14 pav. Fast R-CNN architektūros struktūrinė diagrama

Viena iš priežasčių kodėl metodas yra greitesnis nei R-CNN ta, kad nereikia kiekvieną kartą tiekti 2000 regiono pasiūlymų į sąsukos neuronų tinklą. Vietoje to, sąsukos operacija atliekama tik vieną kartą kiekvienam paveikslėliui ir iš jo sukuriamas funkcijų žemėlapis.

1.5.3. Greitesnis regionais pagrįstas sąsukos dirbtinis neuronų tinklas

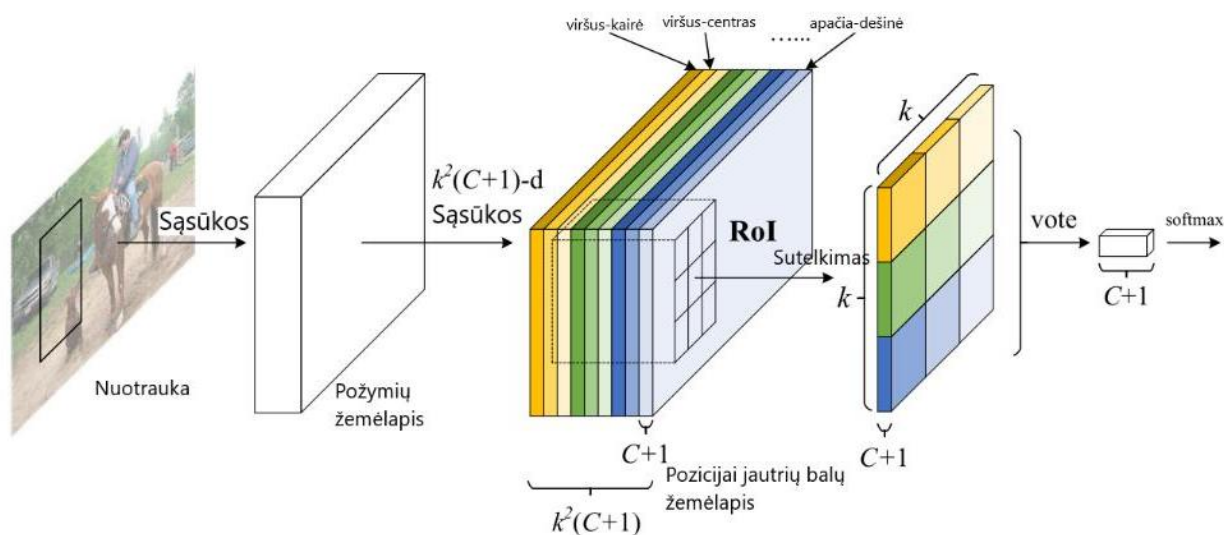
Abu aukščiau paminėti R-CNN algoritmai naudoja atrankinę paiešką, kad surastų regionų pasiūlymus. Tačiau šis procesas reikalaujantis labai daug laiko. Tad buvo sugalvota dar labiau patobulinti algoritmą pašalinant selektyvų paieškos algoritmą ir leidžiant tinklui pačiam išmokyti regionų pasiūlymus. Panašiai kaip Fast R-CNN vaizdas pateikiamas kaip įvestis į sąsukos dirbtinį neuronų tinklą, kuris sudaro požymių žemėlapi. Priešingai nei prieš tai aptartame metode šis nenaudoja selektyvaus paieškos algoritmo regiono pasiūlymus nustatyti, o naudoja atskirą tinklą. Pakeista architektūros struktūra pavaizduota 1.15 paveiksle [10].



1.15 pav. Faster R-CNN architektūros struktūrinė diagrama

1.5.4. Regionais pagrįstas vien tik sąsūkų dirbtinis neuronų tinklas

Regionais pagrįstas vien tik sąsūkų dirbtinis neuronų tinklas (*angl. region-based fully convolutional network*) yra dar viena R-CNN atmaina, skirta tiksliam ir efektyviam objektų aptikimui. Skirtingai nuo anksčiau aptartų regionais pagrįstų detektorių šis detektorius yra visiškai paremtas sąsūkomis ir beveik visi skaičiavimai dalijami visame įvesties vaizde. Šis principas pavaizduotas 1.16 paveiksle.



1.16 pav. R-FCN architektūros struktūrinė diagrama

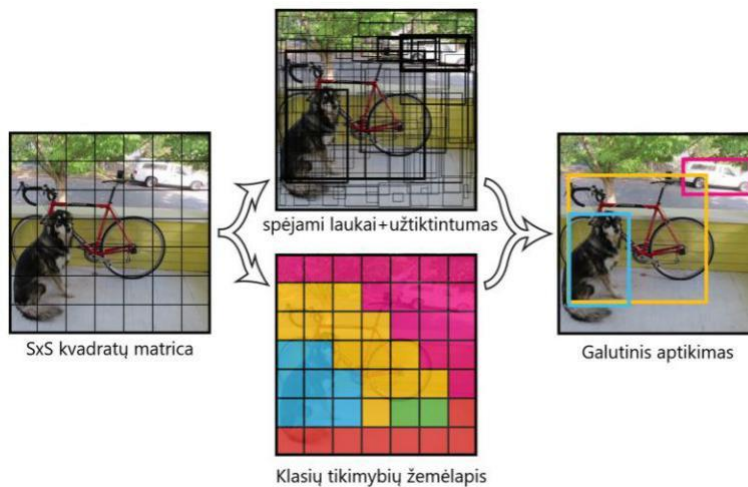
Pasiekti šiam tikslui yra naudojami pozicijai jautrūs balų žemėlapiai (*angl. position-sensitive score maps*), kad būtų išspręsta dilema tarp vertimo invariantiškumo (*angl. translation-invariance*) klasifikuojant vaizdą ir vertimo variacijos (*angl. translation-variance*) nustatant objektus [11].

1.5.5. YOLO

Tai visiškai kitoks vaizdo apdorojimo principas, nuo prieš tai aptartų. Šiame metode nėra naudojami pasiūlytieji regionai ir jų paieška. Išmetus šias architektūros dalis buvo sukurta pilnai tik sąsūkų dirbtiniais neuronų tinklais paremta sistema, galinti objektus vaizde atpažinti realiu laiku. Iš pradžių galėjęs atpažinti kelias dešimtis skirtingų klasių, vėliau, kitose versijoje patobulintas ir dabar atpažįstantis virš 9000 skirtingų klasių su aukštesniu tikslumu bei spartesniu atlikimu nei pirmojo versijoje.

YOLO algoritmas sukurtas taip, kad iš karto atliktų skaičiavimus visam paveikslėliui. Iš to ir kilo pavadinimas (*angl. You Only Look Once*). Tam įėjimo paveikslėlis padalinamas į $S \times S$ dydžio kvadratus. Jeigu objekto centras yra atitinkamoje ląstelėje, tokiu atveju yra klasifikuojamas objektas. Vienas langelis gali būti reikšmingas spėliojant B kiekį objektą apibrėžiančių stačiakampių bei kiekvienam tokiame spėjimui pateikiamas skaičius, įvertinantis, kaip užtikrintai tinklas siūlo tokį

klasifikavimo rezultata – objektą viduje ir stačiakampio koordinates. Jeigu nerandamas joks objektas, užtikrintumas dėl klasifikavimo yra lygus nuliui (žr. 1.17 pav.) [12].



1.17 pav. YOLO architektūros veikimo pavyzdys

1.5.6. „SSD“

SSD metodas pagrįstas SDNT, kuris sukuria fiksuoto dydžio rinkinį ribojančių dėžių ir įvertina ar objektas tuose laukeliuose atitinka turimus pavyzdžius, o po to atlieka ne maksimalaus slopinimo žingsnį (*angl. non-maximum suppression step*), kad būtų galima gauti galutinius aptikimus. Pirmieji tinklo sluoksniai remiasi standartine architektūra, naudojama aukštos kokybės vaizdų klasifikavimui. Tada prie tinklo pridedame pagalbinę struktūrą, pavadinta papildomų požymių sluoksniu, kad būtų galima aptikti šiuos pagrindinius požymius [13]:

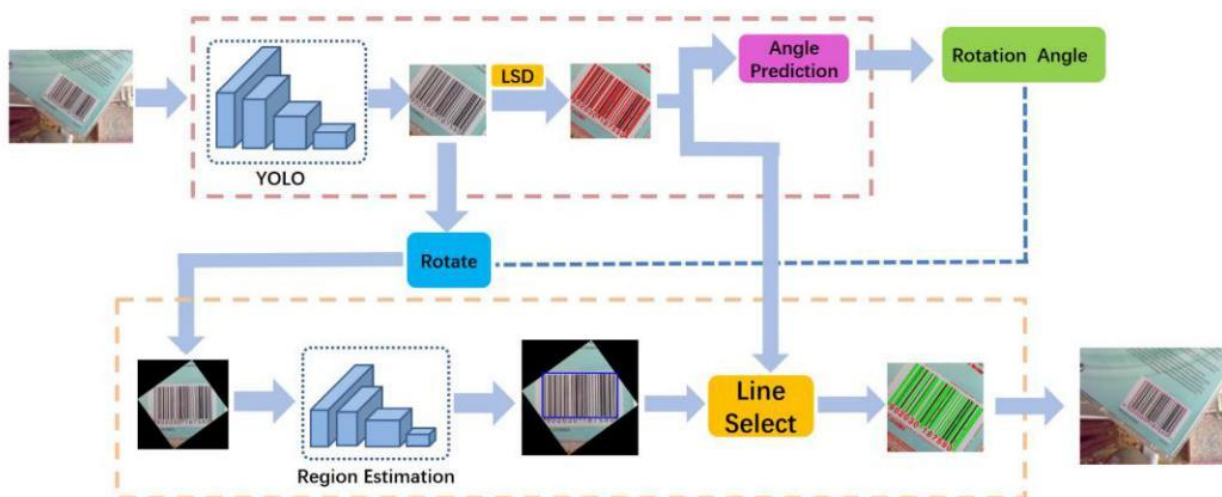
- Sąsūkos prognozės veiksniai skirti aptikimui (*angl. convolutional predictors*)
- Kelių mastelių požymių žemėlapis (*angl. multi-scale feature maps*)
- Numatytuosius langelius ir proporcijas (*angl. default boxes and aspect ratios*)

1.5.7. Orientuotų gradientų histograma

Orientuotų gradientų histograma (*angl. Histogram of Oriented Gradients (HOG)*) yra objektų aptikimui ir atpažinimui sukurtas metodas siekiant vaizdo medžiagoje atpažinti pėsčiuosius. Šis metodas yra paremtas vaizdo gradiento orientacijos tankiame tinklelyje (*angl. image gradient orientations in a dense grid*) normalizuotos histogramos vertinimu. Pagrindinė mintis yra, kad bandomo atpažinti objekto išvaizdą ir forma remiantis intensyvumo gradientų (*angl. intensity gradients*) arba krašto krypčių (*angl. edge directions*) pasiskirstymu [14]. Praktiškai tai yra įgyvendinama vaizdą dalijant į mažus nuotraukos langus. Kiekvienam langui yra sukaupiama vienmatė histograma, kuri yra sudaryta iš kiekvieno pikselio intensyvumo gradiento ir krašto krypties vertės [15].

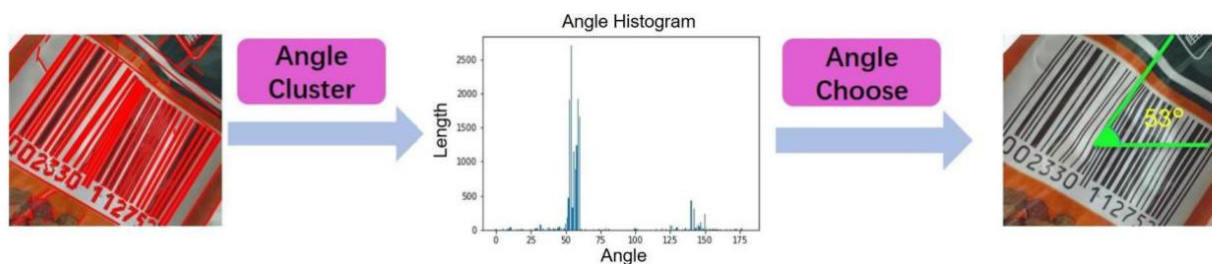
1.6. Brūkšinių kodų aptikimo sistema paremta giliuoju mokymusi

Šiame metode pasirinkta naudoti YOLO tinklą vienmačio brūkšninio kodo ribojimo laukelio aptikimui. Jis tiksliai suranda brūkšninius kodus pagal jų formą ir tekstūrą. Įvedus šaltinio vaizdą, kuriame yra 1D brūkšninis kodas, tinklas aptinka kodą ribojančią dėžutę ir iškerpa ją iš pirminio vaizdo palikdamas vaizde tik brūkšninį kodą. Nors sistema jau turi aptikta 1D kodą, tačiau jo nuskaityti negali dėl nežinomo pasukimo kampo.



1.18 pav. 1D brūkšninio kodo aptikimo su integruotu giliuoju mokymusi ir geometrinio modeliu struktūrinė schema

Tad toliau sekantis veiksmas yra kampo nustatymas. Remiantis tuo, kad brūkšninį kodą galima laikyti lygiagrečių linijų atkarpų rinkiniu ir jis jau užima pagrindinį plotą langelio regione, galima lengvai apibrėžti visų linijų segmentus pritaikant linijos segmento aptikimo (*angl. line segment detection*) algoritmą. Po šio veiksmo yra apskaičiuojama kiekvienos tiesės posvyrio kampas ir ilgis, naudojant tiesės kraštinius taškus. Tokio paties ilgio ir kampo tiesės yra susumuojamos ir sudaromas kampo histograma (žr. 1.19 pav.). Kampas su didžiausiu ilgiu ir nusako pasukimo kampą θ ($0 \leq \theta < 180$).



1.19 pav. Kampo histogramos pavyzdys

Turint iškirpta ir pasukta brūkšninio kodo nuotraukos dalį toliau seka jo dekodavimas. Jam pasiekti yra reikalinga nustatyti tiksli kodo buvimo vieta, kuri yra surandama regiono įvertinimo dalyje (*angl.*

region estimation) ir pavaizduota 1.18 paveiksle. Regionui surasti autoriai pasiūlė naudoti „Resnet-18“ SDNT su ReLU aktyvacijos funkcija, kuris suskaičiuoja keturias vertes padedančias apibrėžti stačiakampio sritį: dvi brūkšninio kodo ilgio ir aukščio santykio vertes, dvi poslinkio vertes nuo brūkšninio kodo stačiakampio centro iki vaizdo centro. Paskui šios vertės panaudojamos įvesties vaizde pažymėti brūkšninio kodo vietą ir reikšmę [16].

1.7. Trumpas skyriaus apibendrinimas

Šiame skyriuje buvo supažindinta su MIT, GPL, BSD, „Apache“ laisvosiomis licencijomis, kurios yra naudojamos šio darbo programinėje įrangoje. Taip pat supažindinta su populiariausiais brūkšniniais kodais bei apžvelgti jų aptikimo sistemų privalumai ir trūkumai. Toliau buvo apibrėžtas objektų aptikimas vaizduose uždavinys ir išanalizuota dirbtinių neuronų tinklų istorija. Plačiai išnagrinėti sąsūkos dirbtiniai neuronų tinklai, jų dalys, parametrai ir pagrindiniai modeliai. Galiausiai apžvelgta giliojo mokymosi modelių objektams atpažinti istorinė raida ir techninei pasiekimai bei aptarta viena įgyvendinta brūkšninių kodų aptikimo ir dekodavimo sistema.

2. GILIOJO MOKYMO SI METODŲ TAIKYMAS QR IR BRŪKŠNINIŲ KODŲ PAIEŠKAI IR ATPAŽINIMUI

Šiame skyriuje apžvelgiami pagrindiniai trys QR ir brūkšninių kodų aptikimo realiu laiku iššūkiai, su kuriais buvo susidurta šiame darbe. Aptariami pagrindiniai preciziškumo, atrinkimo, f-1 rodiklio, vidutinio preciziškumo metrikos giliojo mokymosi metodų įvertinimui. Taip pat paaiškinama maišaties matrica. Galiausiai aptariamas giliojo mokymosi metodų skirstymas į vienos ir dviejų pakopų algoritmus bei plačiai apžvelgiama tiriamųjų metodų architektūra, jų skirtumai ir panašumai.

2.1. Objekto aptikimo iššūkiai

Judančių objektų aptikimas yra esminis komponentas įvairioms kompiuterinio matymo ir vaizdo apdorojimo programoms: pėsčiųjų aptikimui, eismo stebėjimui, saugumo stebėjimui ir kt. Nors naujausi judančių objektų aptikimo metodai rodo gerus rezultatus, tikslus aptikimas vis dar yra sudėtingas dėl įvairių iššūkių, tokių kaip ypač spartus veikimas, klasių disbalansas ir įrangos problemos.

2.1.1. Greitis aptikimui realiuoju laiku

Objektų aptikimo algoritmai turi ne tik tiksliai klasifikuoti ir lokalizuoti svarbius objektus, bet ir būti neįtikėtinai greiti numatymo metu, kad atitiktų vaizdo apdorojimo realiuoju laiku poreikius. Vaizdo įrašai paprastai filmuojami bent 24 kadrų per sekundę greičiu, o tai reiškia, kad tikrai ne visi giliojo mokymosi metodai yra tinkami. Žinoma šiandieniniai objektų aptikimo algoritmai bando rasti greičio ir tikslumo pusiausvyrą, kurios rezultatams įtakos turi neuronų tinklo dizaino pasirinkimai, o ne tik pati aptikimo sistema. SDNT bazės pasirinkimas turi didelės įtakos greičio ir tikslumo balansui. Labai gilūs tinklai, tokie kaip 164 sluoksniai, naudojami „Inception-ResNet-V2“, užtikrina įspūdingą tikslumą, tačiau greičio atžvilgiu yra prasti, lyginant su sistemomis su „VGG-16“ kur yra mažas sluoksnių skaičius [17].

2.1.2. Klasių disbalansas

Klasių disbalansas yra daugelio klasifikavimo algoritmų problema, o objektų aptikimas nėra išimtis. Objektų disbalansas gali atsirasti dviem atvejais. Pirmasis kuomet yra vienas pagrindinis objektas ir daug visokių objektų už jo likusiame vaizde. Antrasis kuomet yra daugiau nei 2 pagrindiniai objektai vaizdo priekyje ir jų klasių pasiskirstymas yra ne vienodas.

Pirmasis paminėtas disbalansas yra neišvengiamas objekto aptikimo užduotyje, nes didžioji vaizdo dalis yra fonas. Šis disbalanso tipas sulaukė daug mokslinių tyrimų dėmesio, kurių metu buvo

įrodyta, kad jis veiksmingai sumažinamas treniruočių metu selektyviai imant foninius regionus [17], kad būtų išlaikytas fiksuotas ir subalansuotas fono ir priekinio plano objektų santykis.

Antrasis disbalansas priklauso nuo duomenų rinkinio, kai tam tikros objektų klasės yra per daug rinkinyje lyginant su kitomis. Jei toks netolygus rinkinys yra naudojamas neuronų tinklo apmokymui rezultate objektų klasės gali būti klaidingai klasifikuojamos bei pasiektas mažesnis tinklo preciziškumas.

2.1.3. Įrangos problemos

Ne tik objektų pasiskirstymas, bet ir parametrai susiję su vaizdo kamera daro įtaką objekto aptikimo preciziškumui. Vaizdo įrašas gali būti užfiksuotas drebant vaizdo kamerai arba su labai maža skyra. Taip pat įrašytas vaizdas gali turėti ribotą spalvų informaciją bei kitus iškraipymus. Visi šie faktoriai gali supainioti judančio objekto aptikimo algoritmą. Tad svarbu apmokant tinklą duomenų rinkinyje turėti panašaus trikdžio scenarijų bei skirtingų dydžių ir vietų nuotraukose objektų [18].

2.2. Metodų įvertinimo metrikos

Siekiant patikrinti giliojo mokymosi metodų efektyvumą yra naudojama maišaties matrica (*angl. confusion matrix*) (lentelė 2.1) ir pagrindiniai rodikliai, apskaičiuojami iš šios matricos verčių.

Dviejų ir daugiau klasių atveju atpažįstami objektai gali priklausyti klasei (tokius objektus vadiname teigiamais) ir nepriklausyti klasei (neigiami objektai). Taip pat kiekvienas objektas yra priskiriamas prie vienos iš klasių, todėl ir rezultatas gali būti arba teigiamas, arba neigiamas (objektas priskirtas prie nurodytos klasės arba ne). Jei objektas teigiamas ir klasifikatoriaus priskirtas prie nurodytos klasės, jis vadinamas teisingai teigiamu (*angl. true positive, TP*). O jeigu neigiamas objektas klasifikatoriaus nepriskirtas prie nurodytos klasės, jis vadinamas teisingai neigiamu (*angl. true negative, TN*). Teigiamas objektas, klasifikatoriaus klaidingai priskirtas prie klasės, vadinamas klaidingai neigiamu (*angl. false negative, FN*), o neigiamas objektas, klasifikatoriaus priskirtas prie tiriamos klasės, vadinamas klaidingai teigiamu (*angl. false positive, FP*) [19].

2.1 lentelė. Maišaties matrica

Prognozė Tikrasis rezultatas	Priimta	Atmesta
Teisingas	Teisingai teigiamu (TP)	Klaidingai neigiamu (FN)
Klaidingas	Klaidingai teigiamu (FP)	Teisingai neigiamu (TN)

2.2.1. Tikslumas, preciziškumas, jautrumas, F-rodiklis

Norint įvertinti pasiūlytus algoritmus, naudojami preciziškumo, atrinkimo, bendro klasifikavimo tikslumo ir F-rodiklio matai, apskaičiuojami pagal maišaties matricos reikšmes.

Bendras klasifikavimo tikslumas (*angl. Accuracy*) yra bendras teisingų prognozių skaičius ir apskaičiuojamas pagal formulę:

$$\text{Tikslumas} = \frac{TP+TN}{TP+TN+FP+FN}, \quad (2.1)$$

Preciziškumas (*angl. Precision*) apskaičiuojamas pagal formulę:

$$\text{Preciziškumas} = \frac{TP}{TP+FP}, \quad (2.2)$$

Atrinkimas (*angl. Recall*) apskaičiuojamas pagal formulę:

$$\text{Atrinkimas} = \frac{TP}{TP+FN}, \quad (2.3)$$

Vien preciziškumas ar atrinkimas negali apibūdinti klasifikatoriaus efektyvumo. Todėl įvedamas šių dviejų rodiklių derinys – F1-rodiklis. [20] Kuris apskaičiuojamas pagal formulę:

$$F1 - \text{rodiklis} = 2 * \frac{\text{Preciziškumas} * \text{Atrinkimas}}{\text{Preciziškumas} + \text{Atrinkimas}} = \frac{2TP}{2TP + FP + FN}, \quad (2.4)$$

2.2.2. IoU

Šis dydis dar vadinamas Džakardo (*angl. Jaccard*) indeksu ir yra viena dažniausiai naudojamų kompiuterinės regos įvertinimo metrių. Ji apibrėžiama kaip žmogaus sužymėtų segmentų ir modelio spėtų segmentų persidengimo sritis, padalyta iš žmogaus sužymėtų segmentų ir modelio spėtų segmentų sąjungos srities:

$$IoU = J(A, B) = \frac{|A \cap B|}{|A \cup B|}, \quad (2.5)$$

Daugelio klasių segmentavimo atveju yra naudojama vidutinis IoU (mIoU) įvertis.

2.2.3. mAP

Vidutinė preciziškumo (*angl. mean average precision*) reikšmė yra naudojama kaip objekto aptikimo našumo vertinimo indeksas. VP reikšmė yra plotas po Preciziškumo-Atrinkimo kreivės, kurioje atrinkimas yra X ašis, o preciziškumas - Y ašis. VP yra apskaičiuojamas pagal formulę **XX** ir atspindi vidutinį visų kategorijų tikslumą, kas leidžia išmatuoti modelio našumą visose objektų klasėse. VP50 reiškia VP reikšmę, kai IoU yra didesnis nei 0,5, o VP75 reiškia VP reikšmę, kai IoU slenkstis yra didesnis nei 0,75 [20].

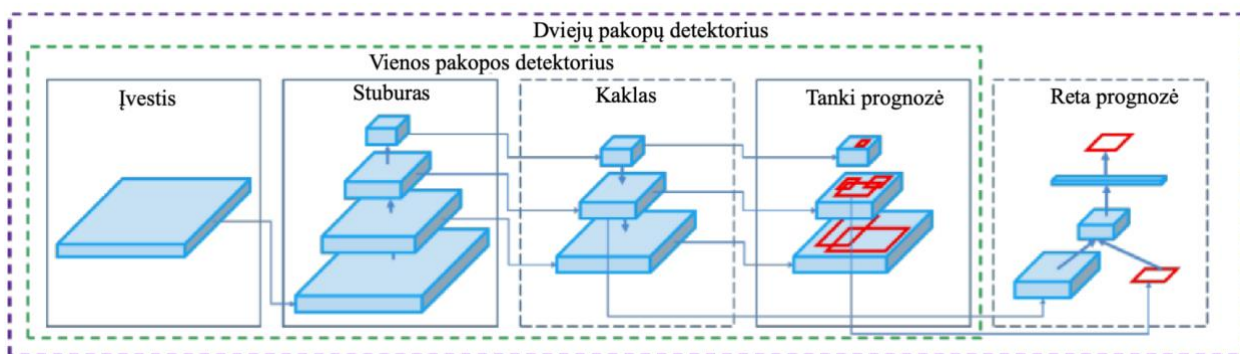
$$VP = \frac{1}{N} \sum_{i=1}^N \int_0^1 PdJ, \quad (2.5)$$

2.2.4. Kadrai per sekundę

Norint įvertinti modelio greitaveiką ir galimybę pritaikyti realaus laiko sistemos yra reikalingas kriterijus, kuris nurodo kiek modelis gali apdoroti kadrų per sekundę (*angl. frames per second, FPS*). Vidutiniškai šiuolaikinės kameros filmuoja bent 24 kadrus per sekundę. O sklandus vaizdas žmogaus akimis yra apie 30 FPS.

2.3. Tiriami giliojo mokymosi metodai

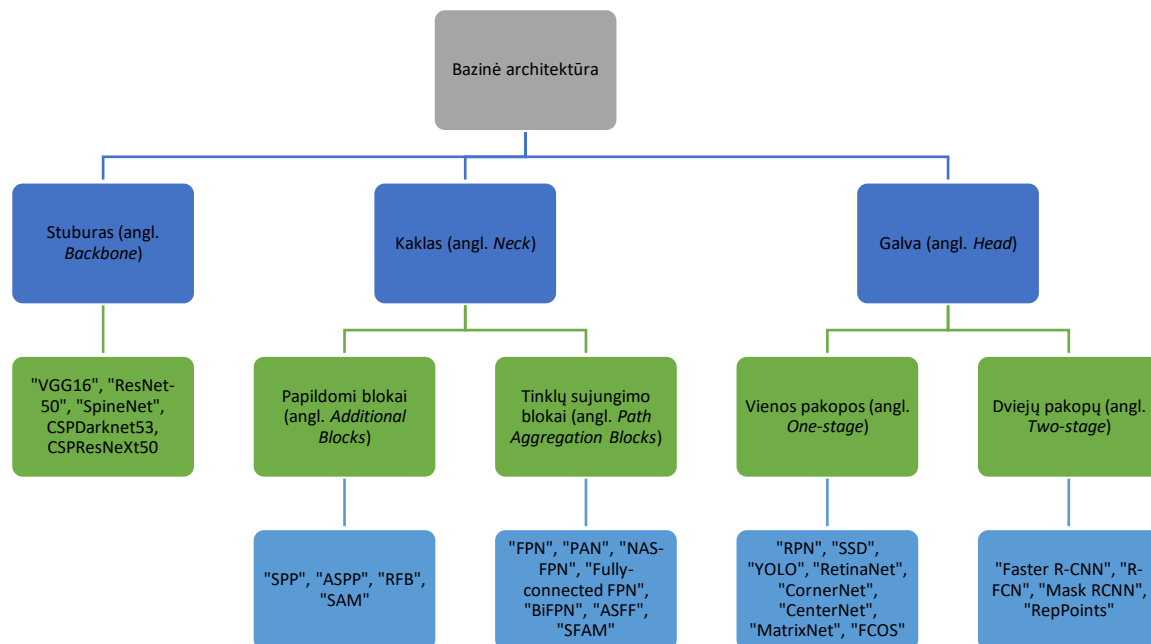
Objektų aptikimui yra naudojami du būdai. Tai yra vienos pakopos arba dviejų pakopų (žr. XX pav.). Dviejų pakopų detektoriai yra sudaryti iš dviejų etapų: regiono pasiūlymo (pirmas etapas) ir klasifikavimo bei lokalizavimo (antrasis etapas). Regiono pasiūlymo etapas paima aprėpiančius langus iš požymių generatoriaus (*angl. feature extractor*) ir priskiria šiems langams balą, kad patikrintų, ar juose yra objektas, o lokalizacija pakoreguoja šiuos aprėpiančius langus, kad objektas būtų užfiksuotas teisingai, visa tai užtikrina didesnę tikslumą. Vienos pakopos modeliai neapima regiono pasiūlymo ir ribojančius langelius perduoda tiesiai į antrąjį klasifikavimo ir lokalizavimo etapą, todėl sumažėja tikslumas, tačiau smarkiai išauga vykdymo greitis [21]. Kadangi apmokytą modeli yra norimą įgyvendinti realaus laiko įterptinėje sistemoje buvo tiriami YOLOR, YOLOv4, YOLOv5, “SSD_MobileNet_V2”, kurie yra vienos pakopos metodai. Dėl bendro abiejų būdų palyginimo buvo ištirtas ir vienas dviejų pakopų „Faster R-CNN“ metodas.



2.1 pav. Bendra objektų aptikimo struktūrinė diagrama

Bendra visų objektų aptikimo architektūrų esmė yra ta, kad įvesties vaizdo funkcijos bus suglaudintos per požymių generatorių (stuburas) ir persiunčiamos į objektų detektorius (įskaitant aptikimo kaklą ir aptikimo galvą). Aptikimo kaklas (arba kaklas) veikia kaip funkcijos, kurioms pavesta sumaišyti ir suderinti pagrindiniame tinkle suformuotas ypatybes, kad būtų pasirengta objekto aptikimo etapui galvoje. Galva yra atsakinga už aptikimą, įskaitant kiekvieno ribojamojo langelio lokalizaciją ir klasifikavimą. Dviejų etapų detektorius šias 2 užduotis įgyvendina atskirai ir vėliau sujungia jų rezultatus (reta prognozė), o vienos pakopos detektorius tai įgyvendina tuo pačiu metu

(tanki prognozė). Šiuolaikinių architektūrų stuburo, kaklo ir galvos pasirinkimai pavaizduoti 2.22 paveiksle.



2.2 pav. Pagrindinės objektų aptikimo architektūros dalys

2.3.1. YOLOv4

YOLOv4 stuburui naudoja skersinį dalinį tinką (*angl. Cross Stage Partial Network, CSPNet*) su „Darknet“ karkasu, kuris yra parašytas C programavimo kalba, sukurdamas naują požymių generavimo pagrindą, vadinamą „CSPDarknet53“. „CSPDarknet53“ sąsukos architektūra, pagrįsta modifikuotu „DenseNet“ [22]. Jis perkelia objektų žemėlapių kopiją iš pagrindinio sluoksnio į kitą sluoksnį per tankų bloką (*angl. dense block*). „DenseNet“ sumažina gradiento išnykimo problemas ir pagreitina atgalinio sklaidimo algoritmo naudojimą kas ženkliai patobulina mokymosi procesą. Taip pat „YOLOv4“ stuburui buvo bandyti 3 skirtingi tinklai: „CSPResNext53“, „CSPDarknet53“ ir „EfficientNet-B3“, kurie tuo metu buvo pažangiausi SDNT. Remiantis teoriniu pagrindimu ir daugybe eksperimentų [23] (žr. 2.2 lent.), „CSPDarknet53“ neuroninis tinklas buvo nustatytas kaip optimaliausias modelis.

2.2 lentelė. Stuburo SDNT tinklų palyginimo eksperimentų rezultatai [23]

Stuburo modelis	Tinklo rezoliucija	Lauko dydis	Parametrų skaičius	Vid. išeities sluoksnio dydis	BFLOPs	FPS (GPU RTX 2070)
„CSPResNext50“	512x512	425x425	20,6 M	1058 K	31	62
„CSPDarknet53“	512x512	725x725	27,6 M	950 K	52	66
„EfficientNet-B3“	512x512	1311x1311	12,0 M	668 K	11	26

Kaklas susideda iš erdvinės piramidės sutelkimo (*angl. Spatial Pyramid Pooling (SPP)*) sluoksnio ir sluoksnių sujungimo tinklo (*angl. Path Aggregation Network (PANNet)*). SPP sluoksnis ir PANet tinklas naudojami požymių agregavimui, siekiant pagerinti ir sutrumpinti svarbiausius iš stuburo išskirtus požymius [24].

Galva sudaryta iš „YoloV3“ sluoksnio. „YoloV4“ pagrindiniai 4 pagerinimai tinklui:

- Nemokamų dalykų krepšį (*angl. Bag of Freebies, (BoF)*) stuburui: „CutMix“ ir „Mosaic“ duomenų papildymą įvesties vaizdams, „DropBlock“ reguliavimą ir objekto klasės etiketės išlyginimą.
- Specialių dalykų krepšį (*angl. Bag of Specials, (BoS)*) stuburui: „Mish“ aktyvavimo funkcija, tarp pakopų dalines jungtis (*angl. Cross-stage partial (CSP)*), kelių įėjimų svertines liekamasias jungtis (*angl. Multi-input weighted residual, (MiWRC)*)
- Nemokamų dalykų krepšį detektoriui: „CIoU-loss“ ir „CmBN“ klaidos funkcijas, „DropBlock“ reguliavimą, „Mosaic“ duomenų papildymą, savarankišką priešpriešos apmokymą (*angl. Self-Adversarial Training*), tinklo jautrumo pašalinimą, optimalius hiperparametrus ir atsitiktines apmokymo formas.
- Specialių dalykų krepšį detektoriui: „Mish“ aktyvavimo funkcijas, „SPP“ ir „SAM“ blokus, „PANNet“ bloką, DIOU-NMS klaidos funkciją.

YOLOv4 kaip ir pirmtakai turi savo YOLO licenciją, kuri nusako, kad su programine įranga galima daryti viską.

2.3.2. YoloV5

Praėjus mėnesiui po YOLOv4 išleidimo, tyrėjas Glennas ir jo komanda paskelbė naują YOLO šeimos versiją, pavadintą YOLOv5. Glennas Jocheris yra „UltralyticsLLC“ tyrėjas ir generalinis direktorius. Ankstesni YOLO modeliai buvo sukurti pagal „Darknet“ sistemą, kurią C programavimo kalba parašė Aleksejus Bočkovskis. „Ultralytic“ yra įmonė, kuri konvertuoja ankstesnes YOLO versijas vienoje iš žinomiausių giliojo mokymosi sistemų „PyTorch“, kuri parašyta *Python* kalba.

Be to, Glennas Jocheris taip pat yra „Mosaic“ duomenų papildymo išradėjas, kurį Aleksejus Bočkovskis minėjo savo naujai išleistame straipsnyje apie YOLOv4 [23]. YOLOv5 modelis sukėlė daug ginčų kompiuterinės regos bendruomenėje dėl savo pavadinimo ir patobulinimų. Todėl, kad jis buvo išleistas praėjus mėnesiui po YOLOv4. Tačiau tai paaiškina, kad abiejų modelių tyrimų pradžia buvo labai panašiu laiku t. y. 2020 m. kovo – balandžio mėn. Kad išvengtų nesusipratimų, Glennas nusprendė pavadinti savo YOLO versiją YOLOv5. Taigi iš esmės abu tyrėjai taikė pažangiausias to meto kompiuterinės regos naujoves. Dėl to YOLOv4 ir YOLOv5 architektūros yra labai panašios kas nulėmė daugelio žmonių nepasitenkinimu, pavadinimu YOLOv5, kai jame nėra daug pažangesnių patobulinimų, lyginant su ankstesne YOLOv4 versija. Be to, Glennas nepaskelbė jokio dokumento,

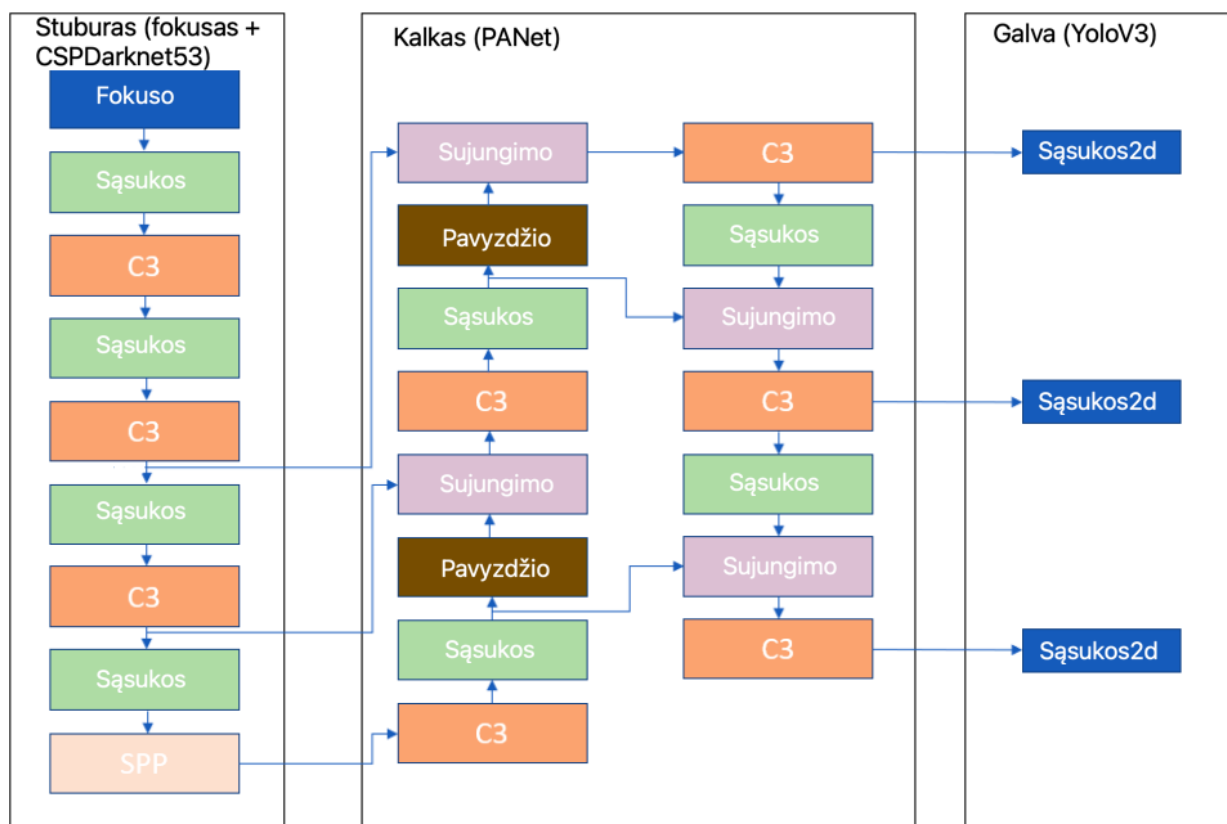
skirto YOLOv5, sukeldamas dar daugiau įtarimų dėl YOLOv5 [25]. Tačiau YOLOv5 susilaukė didelio susidomėjimo iš inžinierių, nes yra parašytas *Python* programavimo kalba, o ne *C*, kaip ankstesnėse versijose. Kas palengvina diegimą ir integravimą daiktų interneto įrenginiuose.

Kaip minėta aukščiau, YOLOv5 architektūroje integruoti pažangiausi modeliai, kurie panašūs į YOLOv4 architektūrą, todėl teorinių skirtumų nėra daug. Autorius nepaskelbė išsamaus dokumento, o tik sukūrė „Github“ saugyklą ir suteikė architektūrai *GPL-3.0* licenciją. YOLOv5 vietoj „Darknet“ savo karkasui naudoja „PyTorch“, kuris parašytas *Python* programavimo kalba [25], [26].

Stuburui YOLOv5 kaip ir YOLOv4 naudoja CSPDarknet53 kaip pagrindą. Šis pagrindas išsprendžia pasikartojančią gradiento informaciją dideliuose tinkluose ir integruoja jo keitimą į objektų žemėlapi, kuris sumažina tinklo vykdymo greitį, padidina tikslumą ir sumažina modelio dydį ir parametrų skaičių [27].

Kaklui naudojamas tas pats „PANet“ tinklas kaip ir YOLOv4, kad padidintų informacijos srautą tarp neuronų tinklų ir pagerina žemo lygio savybių sklaidą modelyje. „PANet“ taip pat pagerina klasifikavimą žemesniuose sluoksniuose, kas lemia geresnį objekto klasifikavimo tikslumą [27].

YOLOv5 galva yra tokia pati kaip YOLOv4 ir YOLOv3, kuri generuoja tris skirtingų požymių žemėlapius išėjime, kad būtų galima numatyti kelių mastelių objektų aptikimą. Tai taip pat padeda efektyviai nuspėti modelio mažus ir didelius objektus [27].



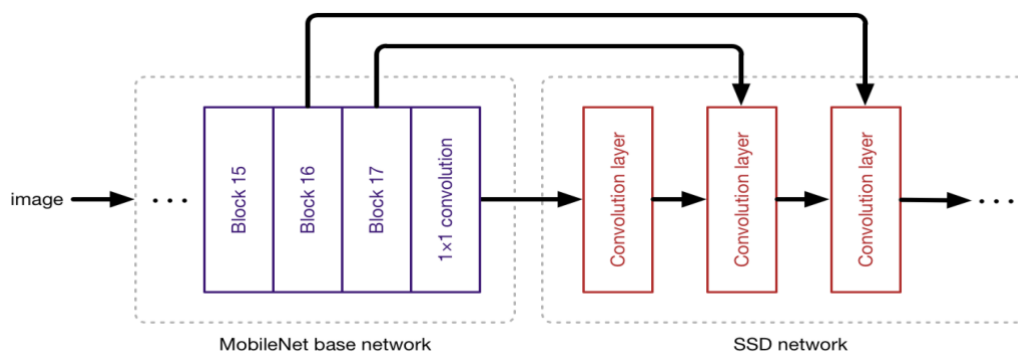
2.3 pav. YOLOv5 architektūra

2.3 paveiksle pateikta YOLOv5 algoritmo architektūra, kurioje matomas fokuso (*angl. Focus*) sluoksnis yra sukurtas iš YOLOv3 struktūros. Jis paima pirmuosius tris YOLOv3 sluoksnius ir sukuria vieną YOLOv5 sluoksnį. „C3“ sudarytas iš trijų sąsūkos sluoksnių ir modulio. „SSP“ yra sutelkimo sluoksnis, naudojamas pašalinti fiksuoto dydžio tinklo apribojimą. Pavyzdžio (*angl. upsample*) sluoksnis naudojamas ankstesnio sluoksnio sujungimui su artimiausio mazgo atrinkimo sluoksniu. Paskutiniai 3 „sąsūkos2d“ yra aptikimo moduliai, naudojami tinklo galvoje [26].

2.3.3. SSD_MobileNet_V2

„SSD_Mobilenet_V2“ yra vieno etapo objektų aptikimo modelis, kuris išpopuliarėjo dėl savo nedidelio dirbtinių neuronų tinklo ir naujų giliai atskiriamų konvoliucijų. Tai modelis, dažniausiai naudojamas mažus skaičiavimo resursus turinčiuose įrenginiuose kaip mobilusis telefonas.

„SSD_MobileNet_V2“ turi du komponentus: pagrindinį modelį „MobileNet“ (stuburas) ir SSD galvą. Stuburas yra iš anksto paruoštas vaizdų klasifikavimo tinklas, iš kurio buvo pašalintas galutinis visiškai prijungtas klasifikavimo sluoksnis ir palikti tik gilieji neuronų tinklai, galintys išgauti semantinę reikšmę iš įvesties vaizdo, išsaugant erdvinę vaizdo struktūrą, nors ir esant mažesnei raiškai. SSD galvutė yra keletas sąsūkos sluoksnių, pridėtų prie šio pagrindo.



2.4 pav. SSD architektūra

Pagrindinis šio metodo tikslas turėti greitą požymių ištraukiklį, kurį galėtų apskaičiuoti ir daug skaičiavimo resursų neturintis įrenginys. Metodas platinamas *Apache-2.0* licencija.

2.4. Trumpas skyriaus apibendrinimas

Šiame skyriuje buvo apžvelgti greičio, klasių disbalanso, įrangos iššūkiai objektų aptikimui realiu laiku naudojant vaizdo kamerą. Aptartos pagrindinės preciziškumo, atrinkimo, f-1 rodiklio, vidutinio preciziškumo metrikos giliojo mokymosi metodų įvertinimui. Aptartas giliojo mokymosi metodų skirstymas į vienos ir dviejų pakopų algoritmus bei plačiai apžvelgta tiriamųjų metodų architektūra. Nustatyta, kad pagrindiniai YOLOv4 ir YOLOv5 architektūros skirtumai yra tai, kad YOLOv4 architektūra naudoja „CSPdarknet53“ SDNT tinklą kaip stuburą, o YOLOv5 prie tokio pat stuburo yra pridėjusi „Focus“ sluoksnį, kuris pagreitina apmokymo procesą.

3. DUOMENŲ PARUOŠIMAS QR IR BRŪKŠNINIŲ KODŲ PAIEŠKOS IR ATPAŽINIMO METODŲ EKSPERIMENTINIAM TYRIMUI

Šiame skyriuje aptariamas duomenų rinkinio sudarymas išrinktų metodų apmokymui. Duomenų rinkinys sudarytas iš atsiųstų ir savo darytų nuotraukų. Tad pirmiausia yra apžvelgiami atsiųsti duomenų rinkiniai jų metrikos ir klasių pasiskirstymas. Antrojoje skyriaus dalyje aptarta kaip sudarytas savo duomenų rinkinys, kokios nuotraukos darytos bei pagrindinės duomenų rinkinio metrikos.

Kompiuterinės regos srityje duomenų rinkiniai įprastai yra suskirstomi į mokymo, validacijos ir testavimo rinkinius, siekiant išvengti persimokymo (*angl. Overfitting*) ir pasirinkimo šališkumo (*angl. Selection Bias*) problemų [28]. Šiame darbe galutinis rinkinys taip pat buvo suskirstytas į šiuos rinkinius: 70 % rinkinio buvo skirta apmokymui, 20 % – validacijai ir 10 % – testavimui.

3.1. Atsiųsti duomenų rinkiniai

Stengiantis sudaryti kuo didesnę duomenų rinkinį pirmiausiai buvo surasti visi duomenų rinkiniai sudaryti iš QR ir brūkšninių kodų ir turintis laisvąją licenciją. Šiuos kriterijus atitiko tik 2 duomenų rinkiniai „ArTe-Lab 1D Medium Barcode“ ir „Muenster BarcodeDB“. Rinkiniai buvo atsiųsti, patikrinti ar tinką tinklo apmokymui bei paruošti naudojimui. Plačiau apie šiuos rinkinius pateikta skyreliuose žemiau.

3.1.1. „ArTe-Lab 1D Medium Barcode“

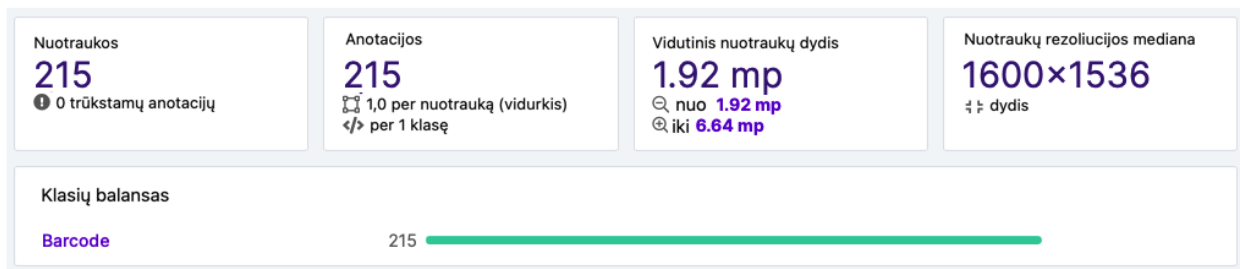
Šį duomenų rinkinį sudarė taikomųjų atpažinimo technologijų laboratorija („Arte-Lab“), kuri yra Insubrijos universiteto teorinių ir taikomųjų mokslų katedros tyrimų laboratorija. Ji atlieka mokslinius tyrimus ir plėtrą kompiuterinio matymo, modelių atpažinimo ir teksto gavybos srityse. „Arte-Lab“ didžiuojasi savo ryšiu su „7Pixel Srl“, nes jie suteikia puikią galimybę išbandyti naujus tyrimų sprendimus realioje įmonės aplinkoje.



3.1 pav. „ArTe-Lab 1D Medium Barcode“ duomenų rinkinio pavyzdys

Rinkinį sudaro 215 spalvotų nuotraukų (žr. 3.1 pav.) kurių rezoliucijos mediana yra 1600×1536 vaizdo taškų. Pagal anotacijas matome, kad nuotraukose yra tik po vieną objektą, kadangi yra 215

nuotraukų ir 215 anotacijų. Nuotraukų dydžio vidurkis yra 1,92 MP (žr. 3.2 pav.) bei duomenų rinkinį sudaro nuotraukos tik su brūkšninio kodu.



3.2 pav. „ArTe-Lab 1D Medium Barcode“ duomenų rinkinio charakteristikos

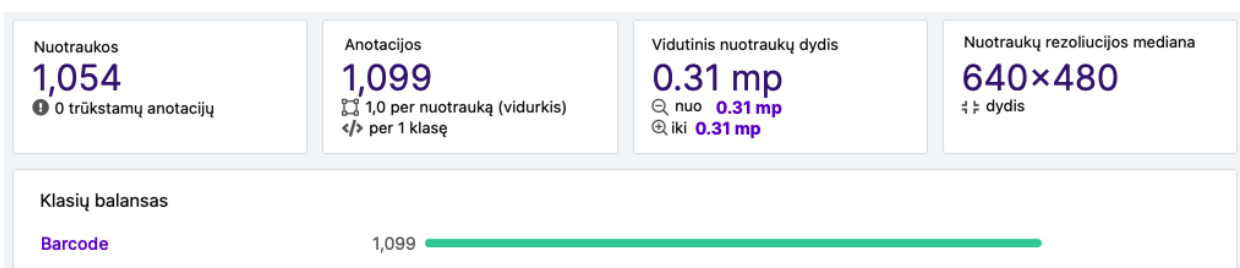
3.1.2. „Muenster BarcodeDB“

Šį duomenų rinkinį sudarė Miunsterio universiteto rašto atpažinimo ir vaizdų analizės mokslininkų grupė. Jų tyrimų sritys yra susijusi su duomenų klasifikavimu, vaizdų apdorojimu ir jų analize. Jie sukuria svarbius bei naujus dirbtinio intelekto komponentus ir juos pritaiko daugelyje sričių.



3.3 pav. „Muenster BarcodeDB“ duomenų rinkinio pavyzdys

Rinkinį sudaro 1054 spalvotų nuotraukų (žr. 3.3 pav.), kurių rezoliucijos mediana yra 640×480 vaizdo taškų. Anotacijų vidurkis per nuotrauką yra 1. Visų nuotraukų dydis vienodas ir yra 0,31 MP (žr. 3.4 pav.) bei duomenų rinkinį sudaro nuotraukos tik su brūkšninio kodu.



3.4 pav. „Muenster BarcodeDB“ duomenų rinkinio charakteristikos

3.2. Duomenų rinkinio sudarymas

Sudėjus atsisiųstus duomenų rinkinius į vieną buvo gauta 1269 nuotraukų, kurių apmokymui nepakako dėl to, kad abu rinkiniai yra sudaryti tik iš brūkšninių kodų bei nuotraukų skaičius per mažas. Tad buvo nuspręsta sudaryti savo duomenų rinkinį, kad būtų galima išlyginti klasių disbalansą

ir žinoma padidinti nuotraukų kiekį rinkinyje dėl geresnio giliojo mokymosi rezultato. Šiam tikslui pasiekti buvo įgyvendinti trys etapai: duomenų surinkimas, duomenų apdorojimas t. y. objektų žymėjimas nuotraukose, duomenų papildymas.

3.2.1. Duomenų surinkimas

Siekiant apmokyti tinklą pritaikytą naudoti visose aplinkose buvo stengtasi fotografuoti nuotraukas kaip įmanoma skirtingesnėse vietose. Iš viso buvo padaryta apie 2000 nuotraukų. Viena nuotraukų dalis buvo daryta su kompiuterine USB kamera ir gautos nuotraukos 500×300 vaizdo taškų, kitos su mobiliuoju telefonu – 3024×4032 vaizdo taškų (žr. 3.5 pav.). Nuotraukos buvo daromos namų aplinkoje, darbo aplinkoje, parduotuvėje. Taip pat prie natūralios dienos šviesos arba prie vidaus apšvietimo.



a) kompiuterinė kamera



b) mobilusis telefonas

3.5 pav. Naujai padarytų nuotraukų pavyzdžiai

3.2.2. Duomenų žymėjimas

Duomenų žymėjimui buvo naudojama „labelme“ programinė įranga, kuri yra parašyta *Python* programavimo kalba. „labelme“ yra grafinis vaizdų žymėjimo įrankis skirtas mašininio mokymosi entuziastams (žr. 3.6 pav.).



3.6 pav. „Labelme“ programinės įrangos langas

Pagrindiniai šio įrankio privalumai, kad galima žymėti daugiakampio, stačiakampio, apskritimo, linijos ir taško vaizdo anotacijas, taip pat galima žymėti objektus vaizdo įrašuose bei anotacijas eksportuoti į JSON ir XML kodo formatus. Programinė įranga yra platinama GPL-3.0 licencija, kuri leidžia ją naudoti nevaržomai. Programos naudojimas yra paprastas ir patogus, pelės pagalba yra pažymimas norimas objektas ir jam priskiriamas klasė. Taip yra nubrėžiamas aprėpiantis langas (*angl. Bounding Box*), kuris yra stačiakampis, supantis objektą ir nurodantis jo vietą bei klasę (pvz., brūkšninis kodas, QR). Šie langai toliau buvo naudojami objekto aptikimo užduočiai, kai siekiama nustatyti kelių objektų padėtį ir tipą vaizde.



3.7 pav. Sužymėto rinkinio pavyzdys

Pavyzdžiui, 3.7 paveikslo dešinėje pusėje matome nuotraukas su nubrėžtais raudonais stačiakampiais kurie pažymi QR kodo aprėpiančius langus, geltonai – duomenų matricos, žaliai – brūkšninio kodo.

3.1 kodas

```
{
  "version": "4.6.0",
  "flags": {},
  "shapes": [
    {
      "label": "QR",
      "points": [
        [
          1787.2222222222222,
          1143.8888888888889
        ],
        [
          1987.2222222222222,
          1366.1111111111111
        ]
      ],
      "group_id": null,
      "shape_type": "rectangle",
      "flags": {}
    }
  ],
  "imagePath": "IMG_1861.jpeg",
}
```

Iš kiekvienos sužymėtos nuotraukos yra gaunamas JSON formato kodas (žr. 3.1 kodą), kuriame yra kiekvieno aprėpiančiojo lango minimali ir maksimali x ir y ašies reikšmė (žr. pav. XX pavaizduota kaip x_{min} , y_{min} , x_{max} , y_{max}) ir klasės pavadinimas (*angl. label*). Šie aprėpiančių langų duomenis dėl tolimesnio apmokymo buvo paruošti YOLO formatui, kuris yra: [klasė, $x_{centras}$, $y_{centras}$, plotis, aukštis]. Šio formato dydžiai buvo apskaičiuoti pagal formules 2.1 – 2.4 [25] ir surašyti į tekstinį failą.

$$plotis = x_{max} - x_{min}, \quad (2.1)$$

$$aukštis = y_{max} - y_{min}, \quad (2.2)$$

$$x_{centras} = x_{min} + \frac{plotis}{2}, \quad (2.3)$$

$$y_{centras} = y_{min} + \frac{aukštis}{2}, \quad (2.4)$$



3.8 pav. Aprėpiančiojo lango pavyzdys

3.2.3. Duomenų papildymas

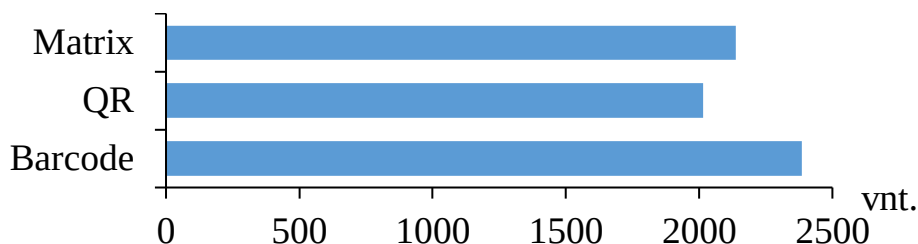
Kadangi surinkti labai didelį kiekį duomenų yra sunku, todėl dažnai yra naudojamas duomenų papildymas (*angl. Data Augmentation*) [29]. Kompiuterinės regos srityje duomenų papildymas – dirbtinis duomenų kiekio didinimas, taikant įvairias transformacijas. Kuris padeda kovoti su tokiomis problemomis kaip persimokymas ir klasių disbalansas. Pagrindiniai duomenų papildymo metodai yra:

- Mastelio keitimas – vaizdas yra didinamas, mažinamas, praplečiamas arba suspaudžiamas.
- Apkarpymas – išrenkama vaizdo dalis.
- Apvertimas – vaizdas apverčiamas vertikaliai arba horizontaliai.
- Užpildymas – vaizdas praplečiamas, sudarant dirbtinį rėmelį.
- Pasukimas – vaizdas pasukamas norimu kampu.
- Postūmis – objekto pozicijos keitimas vaizde.
- Ryškumas – vaizdo šviesinimas arba tamsinimas.
- Kontrastas – keičiamas kontrastas tarp ryškiausių ir tamsiausių vaizdo sričių
- Sodrumas – keičiamas spalvos intensyvumas
- Atspalvio suteikimas – keičiamas vaizdo spalvos
- Aštrinimas – pabrėžiamos vaizdo detalės
- Glotninimas – vaizdas yra suliejamas

3.3. Duomenų rinkinio charakteristikos

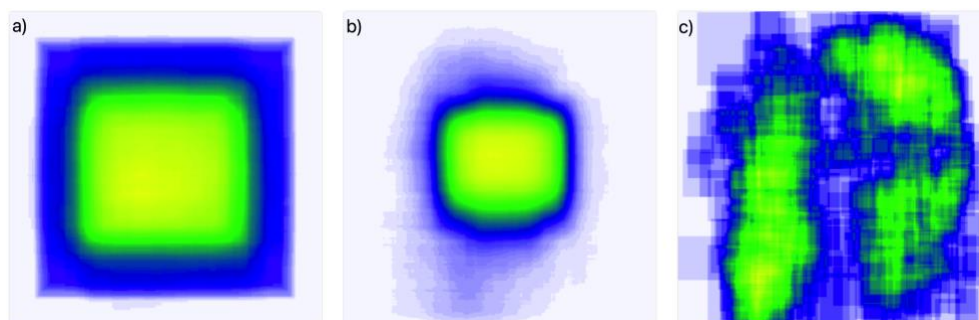
Galiausiai buvo sudarytas 3712 nuotraukų rinkinys, kuriame yra 6536 anotacijos. Buvo stengtasi išlaikyti vienodą klasių balansą, tad rinkinyje pažymėta 2385 brūkšniniai kodai, 2014 QR kodų ir

2137 duomenų matricių (žr. 3.9 pav.). Vidutiniškai nuotraukoje yra 1,7 objekto, o jos dydis prasideda nuo 0,15 milijono pikselių (MP) iki 12,19 MP, visų nuotraukų vidurkis 0,31 MP. Taip pat nuotraukų vaizdo taškų mediana yra 640×480 . Kad sulyginti šiuos parametrus visoms nuotraukoms buvo atliktas apdorojimas (angl. *Preprocessing*) į 416×416 vaizdo taškų.



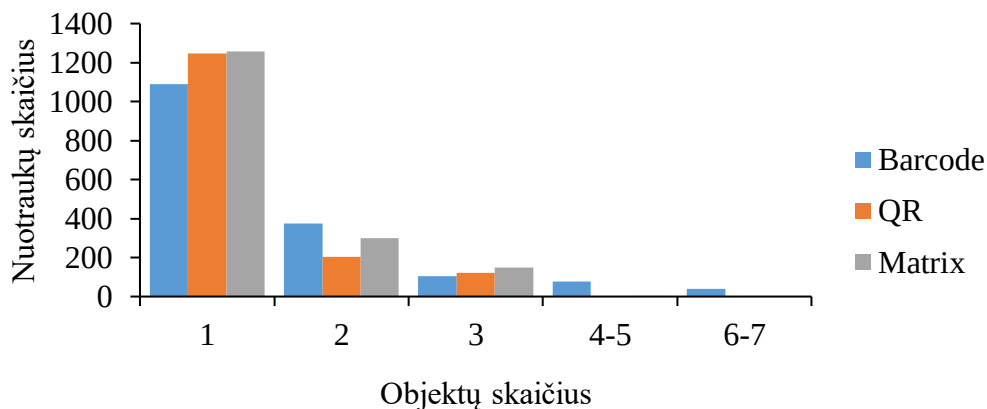
3.9 pav. Objektų skaičius pagal klases

Sekanti labai svarbi duomenų savybė yra objektų pozicijos pasiskirstymas visame vaizde (angl. *heat map*). Sudarytame rinkinyje šią savybę pavyko pilnai įgyvendinti tik su duomenų matricos klase. Prasčiausiai pasiskirstę liko vienmačiai brūkšniniai kodai ir QR kodai. (žr. 3.10 pav.)



3.10 pav. Objektų pozicijos pasiskirstymas vaizde a) QR kodo, b) brūkšninio kodo, c) duomenų matricos

Taip pat verta paminėti, kad duomenų rinkiniuose dominuoja iš klasės vienas objektas. Apie 20 procentų nuotraukų turi daugiau nei kelis tos pačios klasės objektus vaizde, o daugiausia – 6 brūkšniniai kodai buvo pažymėti viename vaizde. Šio pasiskirstymo histograma pavaizduota 3.11 paveiksle.



3.11 pav. Objektų skaičius nuotraukose pagal klases

4. QR IR BRŪKŠNINIŲ KODŲ PAIEŠKOS IR ATPAŽINIMO METODŲ TYRIMO REZULTATAI

4.1. Kompiuteriniai ištekliai ir programinė įranga

Šiame darbe metodų apmokymui buvo naudojama nemokama debesijos paslauga (angl. *Cloud Computing Services*) „Google Colab“, kuri yra „Google“ sukurta integruota kūrimo aplinka (angl. *integrated development environment*, IDE), skirta dirbtinio intelekto moksliniams tyrimams ir mokymuisi. „Google Colab“ aplinką yra panaši į „Jupyter Notebook“ ir leidžia vartotojams naršyklėje rašyti ir vykdyti *Python* kalba parašytus programos kodus bei turi iš anksto įdiegtas pačias populiariausias giliojo mokymosi bibliotekas „PyTorch“, „TensorFlow“, ir „OpenCV“, kurios yra naudojamos šiame darbe. Ši sistema leidžia nemokamai naudoti grafinį procesorių (GPU) (dažniausiai priskiriami „Nvidia Tesla K80“ ir „Nvidia Tesla T4“) ir tensorinį procesų bloką (TPU). Kadangi grafinio procesoriaus pasirinkti negalima, o pagal 4.1 lentelę matoma, kad „Tesla T4“ yra geresnis visuose parametruose tiek taktinis dažnis, tiek atminties dydis ir sparta, todėl rezultatuose bus pažymėta, kuris grafinis procesorius buvo naudojamas tyrime. Tiesa, kompiuteriniai ištekliai nėra garantuoti ir neriboti, o naudojimo laikas kartais svyruoja. Tai būtina, kad „Colab“ galėtų nemokamai teikti išteklius visiems vartotojams ir išlaikytų lankstumą.

4.1 lentelė. Debesijos paslaugos teikiamų GPU palyginimas

Parametras	„Tesla K80“	„Tesla T4“
„CUDA“ branduoliai	2496	2560
Taktinis dažnis, MHz	562	585
Atminties dydis, GB	12	16
Gamybos technologija, nm	28	12
Atminties sparta, GB/s	240,6	320
Slankiojo kablelio operacijų per sekundę	$\sim 1,87 \cdot 10^{12}$	$\sim 0,25 \cdot 10^{12}$

4.1.1. Programinės įrangos bibliotekos

„OpenCV“ yra atvirojo kodo kompiuterinės regos ir mašininio mokymosi programinė biblioteka. Jos pagrindinis tikslas yra sukurti infrastruktūrą, skirtą kompiuteriniai regai bei paspartinti jos naudojimą komerciniuose produktuose. Biblioteka turi daugiau nei 2500 optimizuotų algoritmų, kuriuos sudaro klasikiniai bei pažangiausi mašininio mokymosi algoritmai. Šie algoritmai gali būti naudojami veidams, objektams aptikti ir atpažinti. Taip pat klasifikuoti žmogaus veiksmus vaizdo įrašuose, stebėti fotoaparato judesius, sekti judančius objektus, išgauti trimačius objektų modelius, sukurti trimačius taškinis debesis iš keleto kamerų ir sujungti vaizdus, kad būtų sukurta didelė skiriamoji geba. „OpenCV“ biblioteka turi daugiau nei 54 tūkstančių vartotojų bendruomenę ir 18 milijonų atsisiuntimų. Pagrindinės sąsajos, kurias turi biblioteka yra *C++*, *Python*, *Java* ir *MATLAB*, kurias palaiko „Windows“, „Linux“, „Android“ ir „Mac OS“. „OpenCV“ daugiausia naudojama

realaus laiko programose ir jei įmanoma naudoja MMX ir SSE instrukcijas. Šiuo metu biblioteka aktyviai pritaikoma „CUDA“ ir „OpenCL“ sąsajoms. Taip pat lengvam sistemos panaudojimui prisideda ir jos licencijavimas. „OpenCV“ 4.5.0 versija ir aukštesnė yra licencijuota *Apache 2* licencija, o „OpenCV“ 4.4.0 ir žemesnė turi *3-clause BSD* licencija.

„TensorFlow“ yra mašininio mokymosi sistema, veikianti didelio masto, nevienalytėse aplinkose. Ji turi išsamią, lanksčią ekosistemą, kuri leidžia inžinieriams sukurti ir lengvai įdiegti pažangias mašininio mokymosi sistemas. Šią biblioteką sukūrė tyrėjai ir inžinieriai, dirbantys „Google Brain“ komandoje „Google“ dirbtinio intelekto tyrimų organizacijoje, kad palengvintų mašininio mokymosi ir giliųjų neuroninių tinklų tyrimų atlikimą. Sistema yra pakankamai bendra, tad ją galima pritaikyti ir įvairiose kitose srityse. „TensorFlow“ suteikia stabilų *Python* ir *C++* aplikacijų programavimo sąsają (*angl. application programming interface*), taip pat yra pritaikymai ir kitoms kalboms, tačiau jų stabilumas nėra garantuojamas. Biblioteka yra licencijuojama *Apache 2.0* licencija.

„CUDA“ tai „Nvidia“ sukurta lygiagreti skaičiavimo platforma bei programavimo modelis (*angl. Compute Unified Device Architectur*), skirtas bendram skaičiavimui grafiniuose procesoriuose (*angl. graphics processing unit*). Visa tai padeda išnaudoti GPU branduolius lygiagrečiam skaičiavimui, kurių yra tūkstančiai. Naudodami „CUDA“, kūrėjai turi galimybę programuoti tokiomis kalbomis kaip *C*, *C++*, *Fortran*, *Python* ir *MATLAB* kur lygiagrečius skaičiavimus gali išreikšti per plėtinius kelių pagrindinių raktinių žodžių pavidalu. „CUDA“ palaiko visas pagrindines „Nvidia“ vaizdo plokštes bei „Windows“ ir „Linux“ operacines sistemas ir yra licencijuota *nuosavybine* licencija.

„cuDNN“ tai „Nvidia“ giliųjų neuronų tinklų biblioteka, skirta grafiniais procesoriais paremtose sistemose su neuronų tinklais. „cuDNN“ siūlo pritaikytus standartinius metodus, tokius kaip SDNT, sutelkimo sluoksnius, normalizavimo ir aktyvinimo sluoksnius (*angl. activation layers/functions*). Ši biblioteka leidžia tyrėjams sutelkti dėmesį į neuroninių tinklų mokymą ir programinės įrangos kūrimą, o ne švaistyti laiką GPU našumo derinimui. „cuDNN“ paspartina plačiai naudojamas giliojo mokymosi sistemas, įskaitant „Caffe2“, „Chainer“, „Keras“, *MATLAB*, „MxNet“, „PaddlePaddle“, „PyTorch“ ir „TensorFlow“. Kaip ir „CUDA“ „cuDNN“ palaiko visas pagrindines „Nvidia“ vaizdo plokštes bei „Windows“ ir „Linux“ operacines sistemas ir yra licencijuota *nuosavybine* licencija.

„NumPy“ yra pagrindinis *Python* skaičiavimo paketas skirta mokslininkams. Tai yra biblioteka, leidžianti naudoti objektus sudarytus iš daugiamačių masyvų taip pat įvairius išvestinius objektus (pvz., užmaskuotas masyvus ir matricas) bei pagrindinius matematinius veiksmus greitoms masyvų operacijoms, įskaitant matematinę, loginę, formos manipuliavimą, rūšiavimą, pasirinkimą, įvestį / išvestį, diskrečias Furjė transformacijas, tiesinę algebra, statistiką, atsitiktinį modeliavimą ir daug daugiau. „NumPy“ licencijuota *BSD 3-Clause* licencija.

„PyTorch“ yra optimizuota tensorių biblioteka giliam mokymuisi naudojanti procesorius ir GPU. Ši biblioteka yra *Python* paketas, teikiantis dvi aukšto lygio funkcijas:

- Tensoriaus skaičiavimas (toks kaip „NumPy“) su dideliu GPU našumu.
- Gilūs neuroniniai tinklai, sukurti ant „tape-based autograd“ sistemos.

Pagrindinis PyTorch panaudojimas yra pakeisti NumPy biblioteką, kad būtų išnaudojama GPU galia bei suteiktų maksimalų lankstumą ir našumą. PyTorch licencijuota BSD licencija.

„Flask“ yra žiniatinklio sistema (*angl. framework*). Ji suteikia įrankius, bibliotekas ir technologijas, leidžiančias kurti žiniatinklio programą. Ši žiniatinklio sistema skirta kurti tinklalapius, elektroninius dienoraščius, wiki puslapius ir kt.. „Flask“ yra licencijuota *clause BSD* licencija.

„ZeroMQ“ (taip pat žinomas kaip „ØMQ“, „0MQ“ arba „zmq“) yra lygiagrečio ryšio sistema. Ji suteikia lizdus (*angl. socket*), kurių pagalba yra perduodami pranešimai įvairiais transportavimo būdais, pvz., procese, tarp procesų, TCP ir „multicast“. Sistema yra pakankamai greita, kad būtų galima naudoti daug klientų. Jo asinchroninis įvesties ir išvesties modelis suteikia keičiamo dydžio kelių branduolių programas, veikiančias asinchroniniame režime. „ZeroMQ“ turi daugybę kalbų API ir veikia daugelyje operacinių sistemų. „ZeroMQ“ yra *GPLv3* licencijos atvirojo kodo programinė įranga.

4.1.2. Brūkšninio kodo dekodavimo bibliotekos

„ZBar“ yra atviro kodo programinė įranga, skirta brūkšniniam kodams dekoduoti iš įvairių šaltinių, tokių kaip vaizdo srautai, vaizdo failai ir neapdoroto intensyvumo jutikliai. Ji palaiko daugelį populiarių brūkšninių kodų tipų, įskaitant EAN, UPC, 128 kodą, 39 kodą ir QR kodą. Biblioteka palaiko *Python*, *Perl*, *C++* programavimo sąsajas bei turi labai supaprastintą *C* bibliotekos versiją, kuri yra tinkama naudoti įterptinėse sistemose. Pagrindiniai sistemos privalumai:

- Pritaikytą daugeliui platformų – „Linux“, „Unix“, „Windows“, „iOS“ ir t.t.
- Didelis greitis – galima naudoti realaus laiko sistemose.
- Mažas kodo dydis – pagrindinis skaitytuvas ir EAN dekodevis užima tik 1 tūkstantį eilučių.
- Jokių slankiojo kablelio operacijų

„ZBar“ turi *GNU LGPL 2.1* licenciją, kad būtų galima plėtoti tiek atvirojo kodo, tiek komercinius projektus.

„ZXing“ ("zebra crossing") yra atvirojo kodo, vienmačių ir dvimačių brūkšninių kodų dekodavimo biblioteka, įgyvendinta su *Java*, bet prieinama ir kitoms programavimo kalboms. Biblioteka palaiko UPC, EAN, vienmačius industrinius ir net septynis dvimačius kodus. „ZXing“ licencijuota *Apache 2.0* licencija.

4.2. Paskirstytoji įterptinė sistema tyrimui

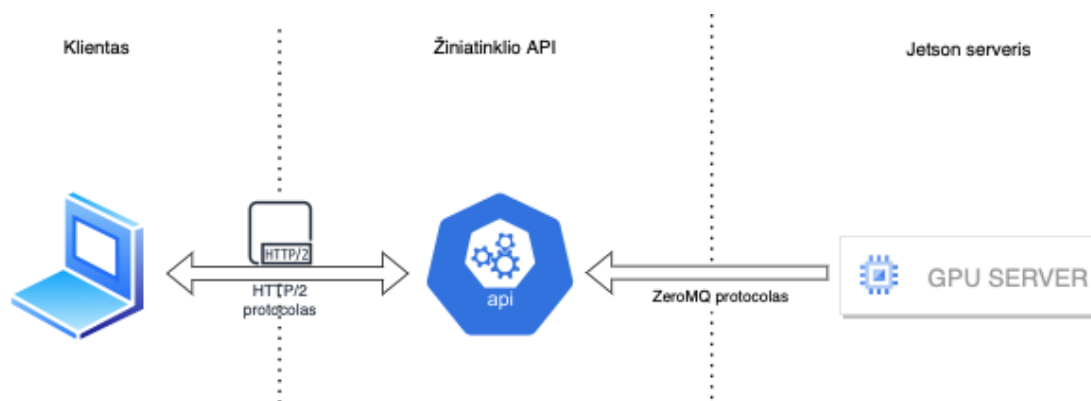
Tolimesniam metodų tyrimui ir pasirinkto metodo įgyvendinimui buvo suprojektuota ir įgyvendinta paskirstytoji įterptinė sistema. Ši sistema sudaryta iš 3 pagrindinių dalių: kliento, žiniatinklio API ir Jetson serverio. Visos šios dalys kartu leidžia atsidarius tinklalapį matyti tiesioginę vaizdo transliaciją iš „Jetson“ mikrokompiuteryje kameros, kuris kiekvieną vaizdo kadrą prieš išsiųsdamas apdoroja dirbtiniu neuronų tinklu. Šios programinės architektūros lankstumas yra tai, kad tiesioginę transliaciją galime pasiekti bet kuriame įrenginyje kuris turi naršyklę. Toliau skyreliuose pateikta sistemos struktūrinė diagrama ir aprašytos supaprastinti programų veikimo algoritmai.

4.2.1. Architektūros struktūrinė diagrama

„Jetson serveris“ programa buvo patalpinama į „Jetson TX2“, kuris yra vienas iš greičiausių, energiją taupančių integruotų AI skaičiavimo įrenginių. Šis modulyje esantis 7,5 vatų superkompiuteris užtikrina greitus matematinius skaičiavimus. Jis sukurtas aplink „NVIDIA Pascal™“ šeimos GPU, jame yra 8 GB operatyvios atminties ir 59,7 GB / s atminties pralaidumo. Jame yra įvairių standartinių aparatinės įrangos sąsajų, kurios leidžia lengvai integruoti jį į įvairius gaminius ir formos veiksnus.

Žiniatinklio API programa buvo paleista asmeniniame kompiuteryje, kurios OS sistemos pagrindas yra „UNIX“. O vartotojai norėdami pasiekti tinklalapį gali naudotis standartinėmis naršyklėmis. Kadangi serverio programa veikė tik kompiuteryje ir nebuvo paleista veikti į serverį, tad kreipimuisi į tinklalapį reikalingas kompiuterio prievado 5000 prirašymas.

Kaip pavaizduota 4.1 paveiksle galime matyti, kad yra keletą pagrindinių komunikacijos tarpusavyje metodų. Pirmasis tarp Jetson serverio ir žiniatinklio API ir antrasis tarp žiniatinklio API ir kliento.



4.1 pav. Supaprastinta algoritmo struktūrinės schemos

4.2.2. Paskirstytos įterptinės sistemos ryšio protokolai

„ZeroMQ“ pranešimas yra atskiras duomenų vienetas, perduodamas tarp programų ar tos pačios programos komponentų. Pačios „ZeroMQ“ požiūriu pranešimai laikomi neskaidriais dvejetainiais duomenimis, kurie yra bet kokio dydžio, pradedant nuo nulio iki dydžio maksimaliai telpančio atmintyje. Paprasčiausias „ZeroMQ“ pranešimas susideda iš vieno kadro (dar vadinamas pranešimo dalimi). „ZeroMQ“ garantuoja, kad pranešimui pristatys visas dalis. Tai leidžia siųsti arba gauti kadrų sąrašą kaip vieną tiesioginį pranešimą, kuris (vienos arba kelių dalių) turi tilpti į atmintį. O jei norima siųsti savavališko dydžio failus, reikia juos suskaidyti į dalis ir išsiųsti kiekvieną vienetą kaip atskirus vienos dalies pranešimus. Naudojant kelių dalių duomenis, atminties sąnaudos nesumažėja.

Iš pirmo žvilgsnio galima būtų pagalvoti, kad tai yra klasikinis TCP modelis nes matome, kad norėdami siųsti ir gauti pranešimus, naudojame „zmq_msg_send()“ ir „zmq_msg_recv()“ t. y. siuntimo ir gavimo metodus, tačiau „ZeroMQ“ įvesties / išvesties modelis pakankamai skiriasi nuo klasikinio TCP modelio pavaizduoto žemiau.

Pagrindinis skirtumas tarp TCP ir „ZeroMQ“ lizdų, kai reikia dirbti su duomenimis:

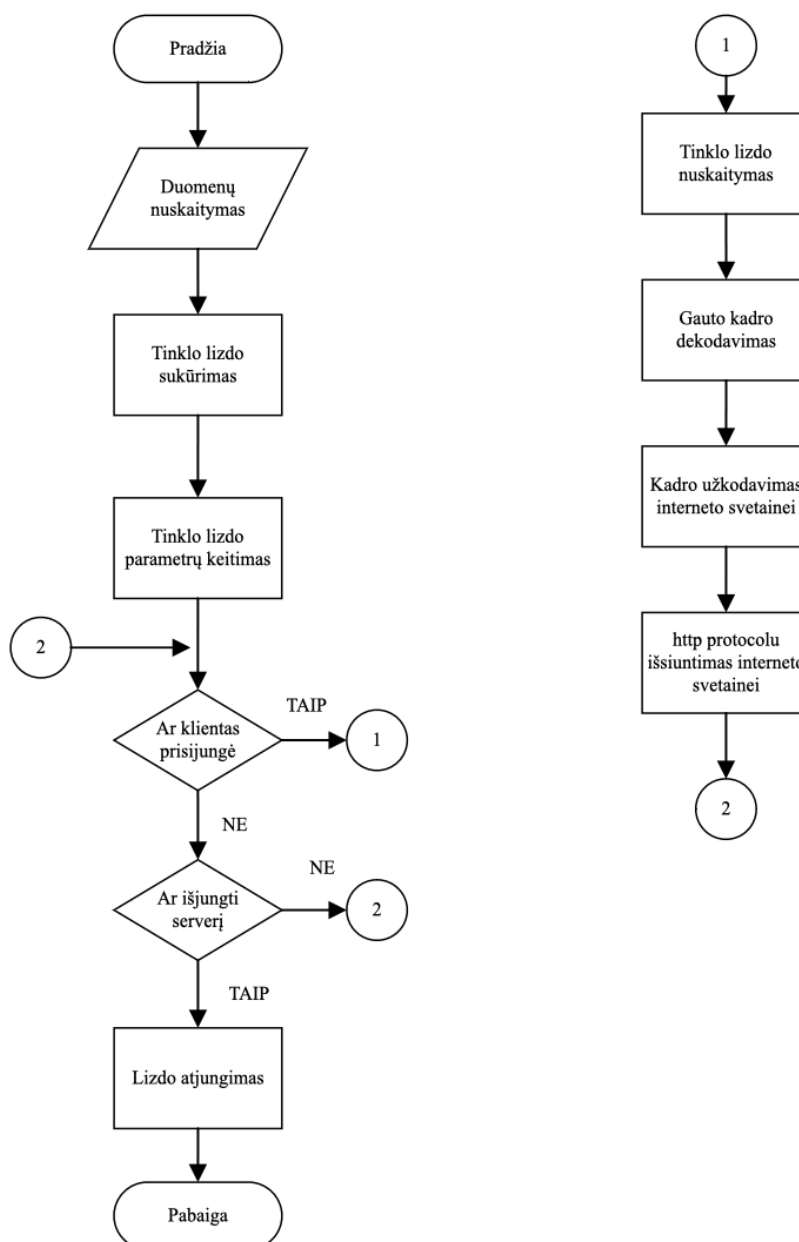
- „ZeroMQ“ lizdas perduoda pranešimus taip pat kaip UDP
- „ZeroMQ“ savo komunikaciją atlieka foninėje (angl. *background*) gijoje. Tai reiškia, kad pranešimai patenka į eilę ir siunčiami vienas paskui kitą iš eilės, nesvarbu, kuo užsiima programa.
- „ZeroMQ“ lizdai turi integruotą maršruto parinkimą, atsižvelgiant į lizdo tipą. T.y. galima pasiekti daug serverių vienu metu.

HTTP (angl. HyperText Transfer Protocol) – pagrindinis metodas informacijai pasauliniame tinkle pasiekti. Pradinė protokolo paskirtis – pateikti standartinį būdą *HTML* puslapiams skelbti ir skaityti. *HTTP* yra užklausų ir atsakymų protokolas, jungiantis klientą ir serverį. *HTTP* klientas, pavyzdžiui, naršyklė, inicijuoja užklausą, kai TCP / IP ryšiu jungiasi prie tam tikro nutolusio kompiuterio prievado (angl. *port*) (pagal nutylėjimą naudojamas 80). *HTTP* serveris klausosi šio prievado, laukdamas kliento užklausos, pavyzdžiui, „GET / HTTP/1.1“ ir susijusio informacinio pranešimo, nusakančio užklausos prigimtį, taip pat kartais po informacinio pranešimo gali eiti ir papildoma informacija. Gavęs užklausą, serveris atgal siunčia atsakymo eilutę („200 OK“ sėkmės atveju) ir susijusį pranešimą, kuris dažniausiai yra *HTML* puslapis. Šio darbo atveju *HTTP* pagalba yra atsiunčiamas *HTML* puslapis ir informaciją jame taip pat gaunamos vaizdo medžiagos nuotraukos realiu laiku.

4.2.3. Žiniatinklio serverio veikimo algoritmas

Pagrindinio žiniatinklio serverio veikimas prasideda sukuriant standartinius ryšio kelius *http* komunikacijai (angl. *endpoints*), kad naršyklėje išsiuntus užklausą į šį serverį jis sugražintu atgal

HTML aplikaciją. Toliau seka tinklo lizdo (*angl. socket*) sukūrimas ir jo parametrų nustatymas, kuris yra skirtas komunikacijai su „Jetson“ serveriais. Svarbu paminėti, kad šis tinklo lizdas veikia kaip sekėjas. T. y. kai „Jetson“ serveris pasisveikins (*angl. handshake*) su žiniatinklio serveriu, jis pradės sekėti jo siunčiamus duomenis. Tad toliau algoritme ir yra tikrinimo dalis ar „Jetson“ serveris yra prisijungęs. Jei taip tai serveris nuskaityto gautus duomenis iš tinklo lizdo juos dekoduoja, paruošia transliavimui per *http* ir juos išsiunčia į kliento programą. Šie paskutiniai veiksmai vykdomi tol kol serveris yra prisijungęs. Jei „Jetson“ serveris atsijungia programa nesustoja veikti, o laukia kol jis vėl prisijungs nebent yra paleidžiama sustabdymo komanda (žr. 4.2 pav.).

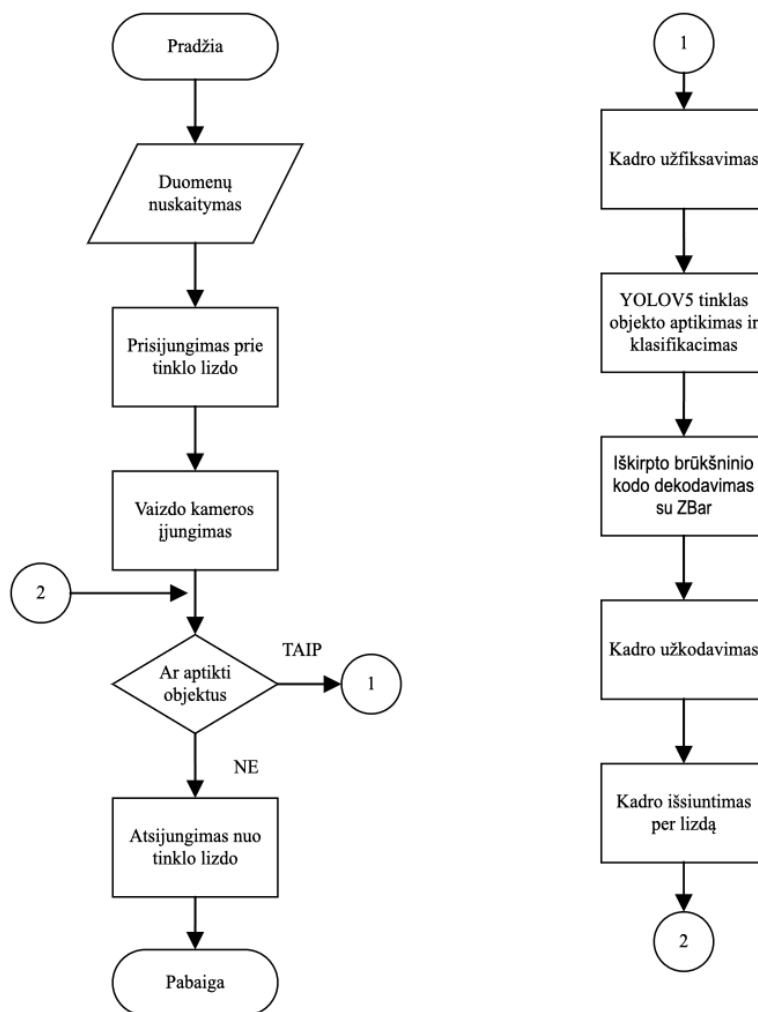


4.2 pav. Žiniatinklio serverio veikimo algoritmo struktūrinė schema

4.2.4. Jetson serverio veikimo algoritmas

„Jetson“ serverio programos veikimas prasideda susikuriant ir nuskaitant reikalingus duomenis bei įgalinant reikalingas techninės įrangos prievadas. Toliau seka tinklo lizdo sukūrimas ir jo parametrų nustatymas, kuris yra skirtas komunikacijai su žiniatinklio serveriu. Taip pat svarbu paminėti, kad šis tinklo lizdas veikia kaip skleidėjas. T. y. kai „Jetson“ serveris prisijungia prie žiniatinklio serverio ir pasisveikina jis pradeda siųsti savo duomenis nelaukdamas iš žiniatinklio atsakymo ar duomenis gauna. Sekantis veiksmas yra vaizdo kameros įjungimas ir jei kamera ir tinklo lizdas veikia tinkamai yra paleidžiamas QR ir brūkšninių kodų aptikimas ir dekodavimas, kuris vykdomas iki tol kol gauna komandą sustoti.

Aptikimo ir dekodavimo dalyje pirmiausia yra užfiksuojamas vaizdo kadras ir išsiunčiamas giliojo mokymosi neuronų tinklui apdoroti. Po neuronų tinklo iš vaizdo yra iškerpami langai pagal aptiktus apibrėžiančiuosius langus ir perduodami atvirojo kodo brūkšninių kodų dekodavimo programai „ZBar“, kuri gražina kodo tipą ir užkoduotą reikšmę. Toliau visi duomenis yra užkoduojami „Base64“ formatu ir perduodami išsiuntimui į tinklo lizdą (žr. 4.3 pav.).



4.3 pav. „Jetson“ serverio veikimo algoritmo struktūrinės schema

4.3. Atrinktų metodų tyrimo rezultatai

Pirmojo eksperimento metu naudojant „Google Colab“ sistemą ir šiam uždaviniui spręsti sudarytą duomenų rinkinį, buvo tiriami YOLOv5, YOLOv4, „SSD_MobileNet_V2“, YOLOR, „Faster R-CNN“ objektų aptikimo modeliai. Modelių įvestyje pateikiamos 416×416 vaizdo taškų nuotraukos. Tinklo struktūra ir tikslo funkcijos naudojamos kaip numatyta originaliuose modeliuose. Rezultatų įvertinimui naudotos preciziškumo, atrinkimo, FPS, vidutinio preciziškumo, kai IoU yra didesnis nei 0,5 (mAP@.5), mAP, kai IoU yra daugiau nei 0,5, bet mažiau nei 0.95 (mAP@.5:.95) ir F1-rodiklio metrikos.

Pagal preciziškumą ir F1-rodiklį geriausią rezultatą parodė YoloV5s modelis, kuris pasiekė 0,932 preciziškumą ir 0,925 F-1 rodiklį taip pat šis tinklas buvo greičiausias ir pasiekė 33,3 kadrų per sekundę spartą. Pagal atrinkimą ir vidutinį preciziškumą, kai IoU yra daugiau už 0,5 geriausius rezultatus parodė YOLOR (4.2 lentelė). Dar verta paminėti, kad dviejų pakopų „Faster R-CNN“ preciziškumas tik 0,488 nes modelis buvo apmokamas tiek pat laiko kiek kiti modeliai.

4.2 lentelė. Atrinktų metodų tyrimo rezultatai

Modelis	Preciziškumas	Atrinkimas	mAP@.5	mAP@.5:.95	F1-rodiklis	FPS
YOLOR	0,815	0,996	0,987	0,629	0,896	28,4
SSD_MobileNet_V2	0,723	0,760	0,932	0,592	0,741	12,2
Faster R-CNN	0,488	0,435	0,589	0,330	0,460	5,0
YOLOV4-Tiny	0,820	0,900	0,773	0,480	0,860	13,1
YOLOV5s	0,932	0,919	0,952	0,675	0,925	33,3

Remiantis tyrimu [26] YOLOv4 ir YOLOv5 preciziškumas ir atrinkimas yra subalansuoti, todėl jų F1-rodiklis yra didelis lyginant su kitais modeliais. Taip pat šiame tyrime nustatyta, kad YOLOv5 modelis yra greitesnis už YOLOv4, o tyrime [30] lyginant YOLOv4 su „SSD_MobileNet_V2“. Įrodyta, kad YOLOv4 yra 45,38 % greitesnis. Siekiant palyginti vienos pakopos ir dviejų pakopų modelius [31] tyrime nustatyta, kad dviejų pakopų modeliai pasiekia tokį pat preciziškumo lygį, tačiau turi didžiulį aptikimo greičio trūkumą.

Antruoju eksperimentu siekta patikrinti kaip skiriasi modelio apmokymas bei jo preciziškumas kai yra naudojami skirtingo dydžio duomenų rinkiniai apmokymui. Visi trys duomenų rinkiniai buvo sudaryti iš 416×416 vaizdo taškų nuotraukų. Pirmasis testui naudotas rinkinys buvo šiam darbui sudarytas ir trečiame skyriuje aptartas duomenų rinkinys. Antrasis rinkinys buvo sudarytas ištrinant nuotraukas ir paliekant tik 300 nuotraukų iš pirmojo rinkinio. Trečiasis rinkinys – pirmajam pritaikant, apkarpymo nuo 0 % iki 20 % priartinant, apvertimo tarp -15° iki 15° , ryškumo tarp -25 % iki 25 %, duomenų pildymą. Visus tris kartus apmokytas YOLOv5s tinklas su originaliai numatyta

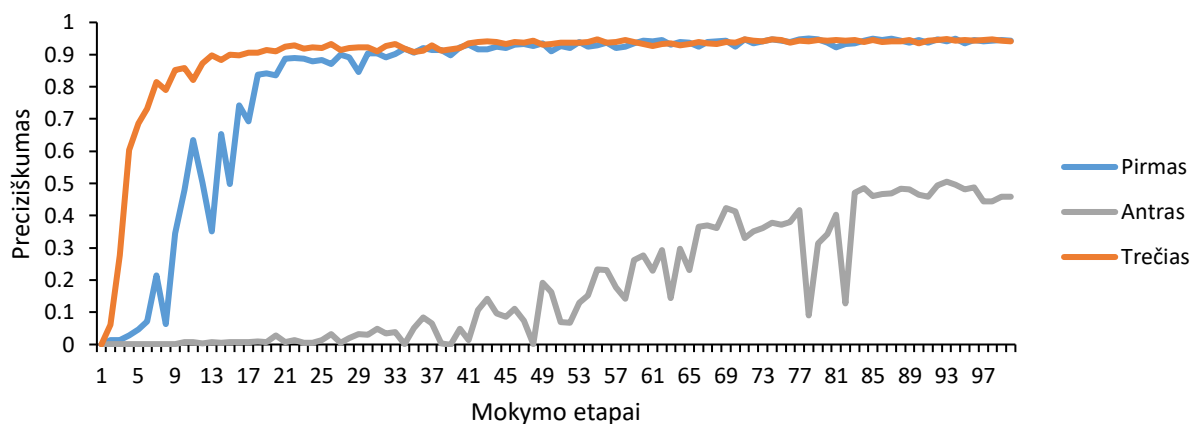
struktūra. Rezultatų įvertinimui naudotos preciziškumo, atrinkimo, mAP @,5, mAP@,5:95 ir F1-rodiklio metrikos.

Pagal visas metrikas modelis geriausius rezultatus pasiekė tinklą apmokant pirmuoju duomenų rinkiniu. Nors apmokant tinklą trečiuoju duomenų rinkiniu pasiektas toks pat 0,894 atrinkimas (4.3 lentelė)

4.3 lentelė. Tinklo apmokymo su skirtingais duomenis tyrimo rezultatai

Modelis	Duomenų rinkinys	Preciziškumas	Atrinkimas	mAP @,5	mAP@,5:95	F1-rodiklis
YOLOv5s	Pirmas	0,936	0,894	0,943	0,676	0,915
	Antras	0,691	0,435	0,459	0,237	0,534
	Trečias	0,931	0,894	0,941	0,647	0,912

Kadangi visų trijų duomenų rinkinių įvertinimui buvo naudojamas tas pats patvirtinimo rinkinys, kiekvieno apmokymo etapų rezultatus galima lyginti tarpusavyje. Nustatyta, kad daugiau duomenų turintis rinkinys (pirmas) greičiau pasiekia aukštą preciziškumo lygį (žr. 4.4 pav.). Bet kaip rodo rezultatai aukščiau (4.3 lentelė), toks duomenų rinkinys nebūtinai pasiekia geriausią preciziškumą ir atrinkimą. Taip pat nustatyta, kad smarkiai sumažinus apmokymo duomenų rinkinį (antras), modelis nepasiekia gerų rezultatų, t. y. aukšto F1-rodiklio.



4.4 pav. Apmokymo rezultatai

Trečiuoju – ištirti YOLOv5 skirtingu gyliu modelius ir įvirtintį jų pritaikymą realaus laiko sistemoms. Apmokymui buvo naudojamas šiam darbui sudarytas QR ir brūkšninių kodų duomenų rinkinys. Tirti 4 skirtingu gylių YOLOV5 modeliai: YOLOv5n, YOLOv5s, YOLOv5m, YOLOv5l. YOLOv5l yra modelis su nepakeistu tinklo gyliu ir pločiu, YOLOv5m – su 33 % mažesniu tinklo gyliu ir 25 % siauresnis, YOLOv5s – su 66 % mažesniu tinklo gyliu ir 50 % siauresnis, YOLOv5n – su 66 % mažesniu tinklo gyliu ir 75 % siauresnis. Apmokytų metodų rezultatų įvertinimui naudotos preciziškumo, atrinkimo, mAP @,5, mAP@,5:95, F1-rodiklio ir FPS metrikos. O tinklo įvertinimui naudotos parametrų skaičiaus, tinklo dydžio ir tinklo sluoksnių skaičiaus metrikos.

Pagal F1-rodiklį kuris įvertina ir preciziškumą, ir atrinkimą pastebėta, kad mažinant tinklo gyli ir plotį F1-rodiklis mažėja nuo 0,933 iki 0,910. Didžiausias nustatytas skirtumas tyrime yra parametrų skaičius kuris mažiausiame tinkle yra 1,77 milijono, o didžiausiame 46,64 milijono ir tinklo dydis kuris YOLOv5n siekia 3,6 MB, o YOLOv5l – 89,9 MB. Šis skirtumas atsispindi FPS skiltyje, jei tinklas turi daug sluoksnių ir parametrų jo įvykdymas užtrunka ilgai pvz.: YOLOv5l (didžiausias tinklas) pasiekia 9,8 kadrų per sekundę su „Tesla K80“, o YOLOv5n (mažiausias tinklas) 52,6 FPS.

Tad geriausi modeliai realaus laiko įterptinei sistemai yra YOLOv5s ir YOLOv5n, nes jų kadrų skaičius per sekundę viršiją 30 (žmogaus akims sklandaus vaizdo FPS) (4.4 lentelė).

4.4 lentelė. YOLOv5 skirtingų tinklo dydžių tyrimo rezultatai

Modelis	YOLOv5n	YOLOv5s	YOLOv5m	YOLOv5l
Preciziškumas	0,943	0,932	0,949	0,953
Atrinkimas	0,880	0,919	0,905	0,913
mAP @,5	0,942	0,952	0,945	0,957
mAP@,5:,95	0,667	0,675	0,691	0,695
FPS	52,6	33,3	16,7	9,8
Parametrų skaičius, milijonais	1,77	7,25	21,04	46,64
Tinklo dydis, MB	3,8	14,8	42,5	89,9
Tinklo sluoksnių skaičius	224	232	308	499
F1-rodiklis	0,910	0,925	0,926	0,933

Apibendrinimas

Apibendrinant atrinktų metodų tyrimų rezultatus galima teigti, kad pagal preciziškumą ir F1-rodiklį geriausią rezultatą parodė YOLOv5s modelis, kuris pasiekė 0,932 preciziškumą ir 0,925 F-1 rodiklį, o pagal atrinkimą ir vidutinį preciziškumą, kai IoU yra daugiau už 0,5 geriausius rezultatus parodė YOLOR“. Sekančiu eksperimentu nustatyta, kad apmokant YOLOv5s modelį su skirtingais duomenų rinkiniais geriausius rezultatus modelis pasiekė naudojant šiam baigiamajam darbui sudarytą duomenų rinkinį. Taip pat nustatyta, kad rinkiniui pritaikius duomenų papildymą aukštas preciziškumas yra pasiekiamas greičiau. Trečiuoju tyrimu įrodyta, kad geriausi modeliai realaus laiko įterptinei sistemai yra YOLOv5s ir YOLOv5n, nes tinklo apdorojimas viršiją 30 FPS.

4.4. Paskirstytos įterptinės sistemos tyrimo rezultatai

Antroji tyrimu dalis buvo vykdoma su įterptine sistema, kurios išsamus aprašymas pateiktas 4.2 poskyryje. Visuose tyrimuose buvo naudojama YOLOv5 architektūra, kuri iš atrinktų metodų buvo geriausia.

Pirmas tyrimas buvo nustatyti tinkamiausia YOLOv5 tinklo dydį „Jetson“ kompiuteriui bei ištirti objektų aptikimo tikslumą pagal klases. Buvo tiriami YOLOv5n, YOLOv5s, YOLOv5m modeliai. Jų apmokytus parametrus ir neuronų tinklo svorius naudojau iš aukščiau atlikto eksperimento, kuris buvo vykdomas „Google Colab“ platformoje. Testavimui naudotas atskirai sudarytas duomenų rinkinys iš 500 vienetų 416×416 vaizdo taškų nuotraukų. Stengtasi atrinkti įvairias nuotraukas (žr. priedą A) su skirtingais nuotraukos fonais bei QR ir brūkšninių kodų dydžiais. Tinklo dydžio rezultatų įvertinimui naudotos preciziškumo, atrinkimo, $mAP@,5$, $mAP@,5;95$, F1-rodiklio ir FPS metrikos.

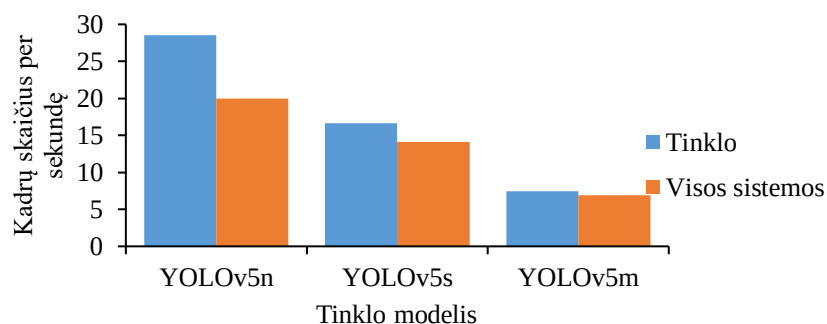
Nustatyta, kad pagal 0,948 F1-rodiklį duomenų matricas geriausiai aptinka YOLOv5n, o brūkšninius ir QR kodus YOLOv5m kurių F1-rodiklis 0,919 ir 0,941. Tačiau YOLOv5m yra 55,24 % lėtesnis už YOLOv5s ir 73,89 % lėtesnis už YOLOv5n (4.5 lentelė).

4.5 lentelė. YOLOv5 tyrimo paskirstytoje įterptinėje sistemoje rezultatai

Modelis	Klasė	Preciziškumas	Atrinkimas	$mAP@,5$	$mAP@,5;95$	F1-rodiklis	FPS
YOLOv5n	Brūkšninis kodas	0,867	0,858	0,911	0,607	0,862	28,57
	Duomenų matrica	0,947	0,949	0,956	0,707	0,948	
	QR	0,885	0,92	0,947	0,614	0,902	
YOLOv5s	Brūkšninis kodas	0,861	0,858	0,917	0,64	0,859	16,67
	Duomenų matrica	0,924	0,93	0,971	0,675	0,927	
	QR	0,94	0,935	0,97	0,702	0,937	
YOLOv5m	Brūkšninis kodas	0,947	0,893	0,943	0,665	0,919	7,46
	Duomenų matrica	0,954	0,872	0,951	0,701	0,911	
	QR	0,923	0,96	0,984	0,73	0,941	

Sekantis eksperimentas buvo ištirti kiek FPS galima pasiekti paskirstytos įterptinės sistemos vaizdo transliavime naudojant YOLOv5n, YOLOv5s, YOLOv5m. Sistemai panaudota 640×480 vaizdo taškų kompiuterinė „HAVIT HV-N5086“ vaizdo kamera, kurios didžiausias galimas FPS yra 30. Rezultatų įvertinimui naudojama FPS metrika.

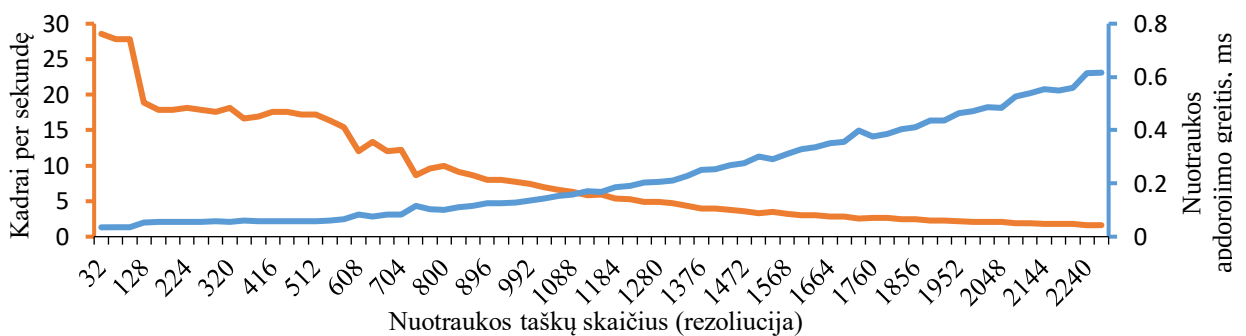
Nustatyta, kad YOLOv5n tinklas per sekundę apdoroja 28,57 kadrus, o tinklą įgyvendinus paskirstytai įterptinei sistemai pasiekiamas 20 kadrų per sekundę greitis. 8,57 kadrų skirtumas stebimas dėl ilgai užtrunkančio brūkšninio kodo dekodavimo programos „Zbar“, kuri vidutiniškai dekoduoja per 11 ms. YOLOv5s ir YOLOv5m FPS skirtumas tarp visos sistemos ir neuronų tinklo apdorojimo yra tik 2,59 ir 0,56 todėl, kad šie modeliai veikia daug lėčiau ir 11 ms nedaro tokios didelės įtakos (žr. 4.5 pav.).



4.5 pav. Objektų skaičius nuotraukose pagal klases

Trečiuoju eksperimentu tirta modelio greičio priklausomybė nuo įvesties vaizdo taškų skaičiaus. Tyrimui įgyvendinti naudotas YOLOv5s tinklas ir 4032×3024 vaizdo taškų nuotrauka su keliais brūkšniniais kodais. Iš tos pačios nuotraukos buvo sudarytas tyrimo duomenų rinkinys „OpenCV“, bibliotekos pagalba, keičiant nuotraukos rezoliuciją nuo 32×32 iki 2496×2496 didinat kas 32 vaizdo taškus. Rezultatams įvertinti naudotos FPS ir nuotraukos apdorojimo greičio (ms) metrikos.

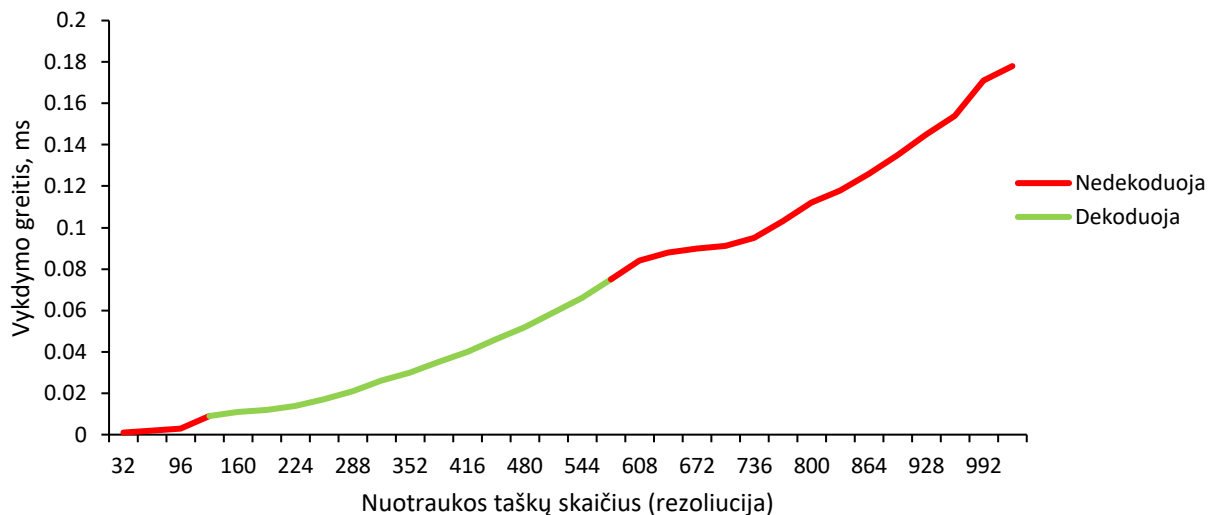
Nustatyta, kad tinklas panašiu greičiu t. y. 18-20 FPS apdoroja nuotraukas, kurių rezoliucija yra nuo 128 iki 608 vaizdo taškų. Vaizdas su labai maža rezoliucija iki 128 vaizdo taškų pasiekia 28-30 FPS. O nuotraukų didesnių nei 608 vaizdo taškų tinklo apdorojimo greitis kinta tolygiai (žr. 4.6 pav.). Taip pat verta paminėti kad „HD“ kokybės nuotrauka yra apdorojama per 0,205 s, „Full HD“ – 0,436 s, „4K“ – 1,8 s.



4.6 pav. Apmokymo rezultatai

Ketvirtasis eksperimentas įgyvendintas „Jetson“ mikrokompiuteryje buvo ištirti „Zbar“ dekoderio veikimo greitį bei surasti minimalų ir maksimalų galimą vaizdo dydį dekodavimui. Šiam tyrimui naudotas vienas vaizdas ir iš jo sudarytas tyrimo duomenų rinkinys „OpenCV“ bibliotekos pagalba keičiant nuotraukos rezoliuciją nuo 32×32 iki 1024×1024 didinat kas 32 vaizdo taškus (pavyzdys pateiktas B priede). Rezultatams įvertinti naudota vaizdo apdorojimo greičio (ms) metrika.

Ištirta, kad „Zbar“ dekoderis aptinka ir dekoduoja kodus, kurių rezoliuciją yra nuo 128 iki 608 vaizdo taškų. Atitinkamai užtrukdamas nuo 9 ms iki 75 ms. Visas kitas nuotraukas jis taip pat apdoroja tačiau nieko nedekoduoja (žr. 4.7 pav.).



4.7 pav. Vykdomo greičio priklausomybė nuo įvesties vaizdo

Apibendrinimas

Apibendrinant įterptinės sistemos tyrimo rezultatus galime teigti, kad pagal F1-rodiklį duomenų matricas geriausiai aptinka YOLOv5n tinklas, o brūkšninius ir QR kodus YOLOv5m. Tačiau YOLOv5m yra 55,24 % lėtesnis už YOLOv5s ir 73,89 % lėtesnis už YOLOv5n. YOLOv5n tinklas per sekundę apdoroja 28,57 kadrus, o tinklą įgyvendinus paskirstytai įterptinei sistemai pasiekiamas 20 kadrų per sekundę greitis. Taip pat paskirstyta įterptinė sistema su YOLOv5s pasiekia 14,08 FPS, o su YOLOv5m – 6,9 FPS. Ištyrus modelio greičio priklausomybę nuo įvesties vaizdo taškų skaičiaus gauta, kad greitis nesikeičia apdorojant nuotraukas nuo 128 iki 608 vaizdo taškų, o nuotraukų didesnių nei 608 vaizdo taškų tinklo apdorojimo greitis lėtėja tolygiai didinat vaizdą.

APIBENDRINIMAS, IŠVADOS

1. Šiame darbe buvo atlikta analitinė dirbtinių neuronų tinklų, jų rūšių ir pagrindinių parametrų apžvalga. Taip pat išanalizuoti pagrindiniai objektų aptikimo metodai, paremti giliojo mokymusi.

2. Atlikta atviros programinės įrangos licencijų apžvalga. Nustatyta, kokias licencijas naudoja programinė įranga, skirta QR ir brūkšniniais kodams atpažinti ir dekoduoti. Apibendrinta giliojo mokymosi metodų licencijavimo praktika.

3. Buvo atrinkti YOLOR, „Faster R-CNN“, „SSD_MobileNet_V2“, YOLOv4, YOLOv5 metodai, kurie yra perspektyviausi aptinkant QR ir brūkšninius kodus. Metodai atrinkti pagal jų preciziškumą ir vaizdų apdorojimo spartą. Taip pat išsamiai apžvelgta jų architektūra ir SDNT panaudojimas metode.

4. Sudarytas 3712 nuotraukų su 6536 anotacijomis duomenų rinkinys perspektyvių giliojo mokymosi metodų apmokymo tikslu. Rinkinys sudarytas iš dviejų viešai internete prieinamų duomenų rinkinių ir šiam darbui sudaryto duomenų rinkinio, su kuriuo stengtasi išlyginti klasių disbalansą, atsiradusį tarp atsiųstų rinkinių. Taip pat padidintas nuotraukų kiekį bendrame rinkinyje, siekiant geresnio giliojo mokymosi rezultato.

5. Naudojant nemokamas debesijos paslaugas buvo apmokyti ir eksperimentiškai ištirti atrinktieji giliojo mokymosi metodai. Gauta, kad geriausią rezultatą parodė YOLOv5s modelis, labai panašius rezultatus parodė ir YOLOR bei YOLOv4. Taip pat nustatyta, kad apmokant YOLOv5s modelį su skirtingais duomenų rinkiniais geriausius rezultatus modelis pasiekė naudojant šiam baigiamajam darbui sudarytą duomenų rinkinį bei tai, kad rinkiniui pritaikius duomenų papildymą aukštas preciziškumas yra pasiekiamas greičiau.

Tolimesniam metodų tyrimui ir pasirinkto metodo įgyvendinimui buvo suprojektuota ir įgyvendinta paskirstytoji įterptinė sistema. Jos paskirtis – atsidarius tinklalapį matyti dirbtinių neuronų tinklo apdorotą tiesioginę vaizdo transliaciją iš „Jetson“ mikrokompiuterio kameros. Gauta, kad duomenų matricas geriausiai aptinka YOLOv5n, o brūkšninius ir QR kodus YOLOv5m tinklas. Tačiau YOLOv5m yra 55,24 % lėtesnis už YOLOv5s ir 73,89 % lėtesnis už YOLOv5n. Be to, ištyrus modelio greičio priklausomybę nuo įvesties vaizdo taškų skaičiaus nustatyta, kad greitis nesikeičia apdorojant nuotraukas nuo 128 iki 608 vaizdo taškų, o nuotraukų, didesnių nei 608 vaizdo taškų, tinklo apdorojimo greitis lėtėja tolygiai didinant vaizdą.

LITERATŪRA

- [1] Hayat S., Kun S., Tengtao Z., Yu Y., Tu T., Du Y. 2018. A Deep Learning Framework Using Convolutional Neural Network for Multi-Class Object Recognition, *IEEE International Conference on Image, Vision and Computing*: 194–198.
- [2] Zhang, Q., Zhang, M., Chen, T., Sun, Z., Ma, Y., Yu, B. 2019. Recent advances in convolutional neural network acceleration. *Neurocomputing* 323: 37-51
- [3] Galván E., Mooney P. 2020. Neuroevolution in Deep Neural Networks: Current Trends and Future Challenges, *IEEE Transactions on Artificial Intelligence* 2020 2(6): 476–493
- [4] Šabanovič, E. 2019. Sąsukos dirbtinių neuronų tinklų įgyvendinimas spartinančiuosiuose įrenginiuose vaizdams analizuoti realiuoju laiku, *daktaro disertacija* Vilnius: VGTU leidykla „Technika“
- [5] Tan, M., Le, Q. 2019. Efficientnet: Rethinking model scaling for convolutional neural networks, *International conference on machine learning* 6105-6114.
- [6] Toliušis, R., Kurasova, O., Bernatavičienė, J. 2019. Semantic Segmentation of Eye Fundus Images Using Convolutional Neural Networks, *Information & Media* 85: 135-147.
- [7] Girshick, R., Donahue, J., Darrell, T., Malik, J. 2014. Rich feature hierarchies for accurate object detection and semantic segmentation, *Proceedings of the IEEE conference on computer vision and pattern recognition*: 580-587.
- [8] Uijlings, J. R., Van De Sande, K. E., Gevers, T., Smeulders, A. W. 2013. Selective search for object recognition, *International journal of computer vision* 104(2): 154-171.
- [9] Girshick, R. 2015. Fast r-cnn, *Proceedings of the IEEE international conference on computer vision*: 1440-1448.
- [10] Ren, S., He, K., Girshick, R., Sun, J. 2015. Faster r-cnn: Towards real-time object detection with region proposal networks, *Advances in neural information processing systems*, 28.
- [11] Dai, J., Li, Y., He, K., Sun, J. 2016. R-fcn: Object detection via region-based fully convolutional networks, *Advances in neural information processing systems*, 29.
- [12] Redmon, J., Divvala, S., Girshick, R., Farhadi, A. 2016. You only look once: Unified, real-time object detection, *Proceedings of the IEEE conference on computer vision and pattern recognition*: 779-788.
- [13] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., Berg, A. C. 2016. Ssd: Single shot multibox detector, *European conference on computer vision*: 21-37
- [14] Tomasi, C. 2012. Histograms of oriented gradients, *Computer Vision Sampler*: 1-6.

- [15] Surasak, T., Takahiro, I., Cheng, C. H., Wang, C. E., Sheng, P. Y. 2018. Histogram of oriented gradients for human detection in video, *2018 5th International conference on business and industrial research (ICBIR)*: 172-176.
- [16] Xiao, Y., Ming, Z. 2019. 1D barcode detection via integrated deep-learning and geometric approach, *Applied Sciences* 9(16): 3268.
- [17] Shrivastava, A., Gupta, A., Girshick, R. 2016. Training region-based object detectors with online hard example mining, *Proceedings of the IEEE conference on computer vision and pattern recognition*: 761-769.
- [18] *Deep Learning for Overcoming Challenges of Detecting Moving Objects in Video* [interaktyvus]. 2019. Katrenko A. [žiūrėta 2022 gegužės 29 d.]. Prieiga per internet: <<https://www.apriorit.com/dev-blog/607-deep-learning-moving-object-detection-video>>
- [19] Vaičiulytė, J. 2020. Paslėptųjų Markovo Modelių tyrimas ir taikymas daugiamačių sekų palaipsnei analizei, *daktaro disertacija* Vilnius: Vilniaus universitetas.
- [20] Fu, H., Song, G., Wang, Y. 2021. Improved YOLOv4 marine target detection combined with CBAM, *Symmetry* 13(4): 623.
- [21] Shek, S. C. K., Butterfield, J., Murphy, A., Spence, I. 2020. Current State of the Art in Object Detection for Autonomous Systems.
- [22] Iandola, F., Moskewicz, M., Karayev, S., Girshick, R., Darrell, T., Keutzer, K. 2014. Densenet: Implementing efficient convnet descriptor pyramids.
- [23] Bochkovskiy, A., Wang, C. Y., Liao, H. Y. M. 2020. YOLOv4: Optimal speed and accuracy of object detection, *arXiv*:2004.10934.
- [24] Liu, S., Qi, L., Qin, H., Shi, J., Jia, J. 2018. Path aggregation network for instance segmentation, *Proceedings of the IEEE conference on computer vision and pattern recognition*: 8759-8768.
- [25] Thuan, D. 2021. Evolution of yolo algorithm and yolov5: the state-of-the-art object detection algorithm
- [26] Nepal, U., Eslamiat, H. 2022. Comparing YOLOv3, YOLOv4 and YOLOv5 for Autonomous Landing Spot Detection in Faulty UAVs, *Sensors* 22(2): 464.
- [27] YOLOv5 [interaktyvus]. 2022. Glenn Jocher [žiūrėta 2022 gegužės 29 d.]. Prieiga per internet: <<https://github.com/ultralytics/yolov5>>
- [28] *Data splitting technique to fit any Machine Learning Model*. 2020. Kumar S. [žiūrėta 2022 gegužės 11 d.]. Prieiga per internet: <<https://towardsdatascience.com/data-splitting-technique-to-fit-any-machine-learning-model-c0d7f3f1c790>>

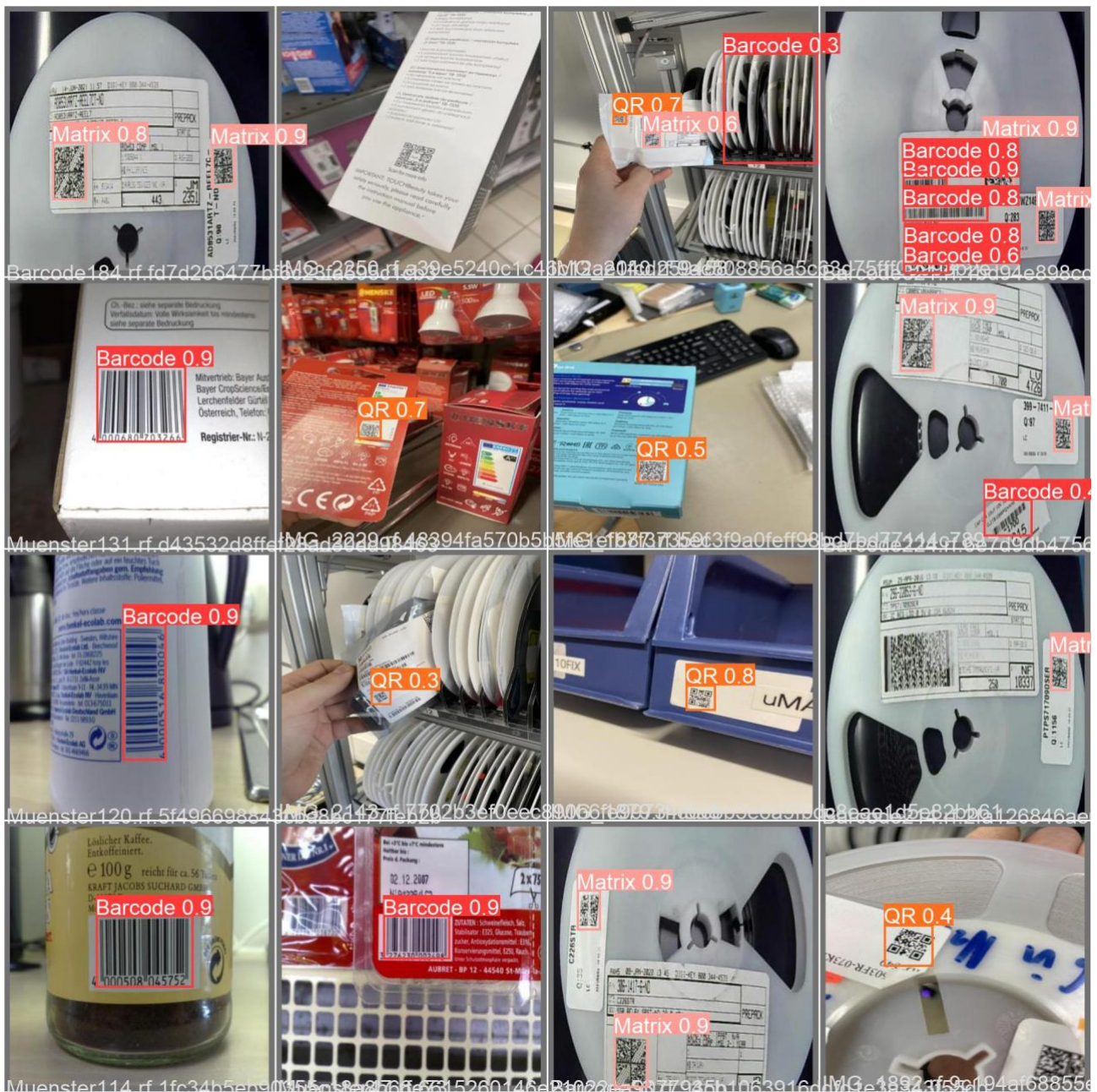
- [29] Zoph, B., Cubuk, E. D., Ghiasi, G., Lin, T. Y., Shlens, J., Le, Q. V. 2020. Learning data augmentation strategies for object detection, *European conference on computer vision*: 566-583.
- [30] Fairuzi, M. R., Zulkifli, F. Y. 2021. Performance Analysis of YOLOv4 and SSD Mobilenet V2 for Foreign Object Debris (FOD) Detection at Airport Runway Using Custom Dataset, *2021 17th International Conference on Quality in Research (QIR): International Symposium on Electrical and Computer Engineering*: 11-16.
- [32] Vyšniauskas, V. 2021. Automatinė aerofotografinių nuotraukų analizė taikant giliojo mokymosi metodus.

PRIEDAI

A priedas. QR ir brūkšninių kodų paieškos ir atpažinimo metodų eksperimentinių tyrimų rezultatai



Šiame paveiksle pateikta dalis testavimo rinkinio naudoto paskirstytajai įterptinei sistemai.



Šiame tinkle pateikta gautas rezultatas po YOLOv5n tinklo



Šiame paveiksle pateikta gautas rezultatas po YOLOv5s tinklo



Šiame paveiksle pateikta gautas rezultatas po YOLOv5m tinklo

B priedas. ZBar dekoderio greičio tyrimo duomenų rinkinys pavyzdžiai



Šiuose paveiksluose pateikta kaip atrodė dalis nuotraukų naudotų „Zbar“ dekodavimo greičiui tirti.