Kauno technologijos universitetas

Elektros ir elektronikos fakultetas

Giliųjų neuroninių tinklų taikymo automobilių atpažinimui nuotraukose tyrimas

Baigiamasis magistro projektas

Edvinas Boguša

Projekto autorius

Doc. Dr. Arūnas Lipnickas

Vadovas

Kaunas, 2019

Kauno technologijos universitetas

Elektros ir elektronikos fakultetas

Giliųjų neuroninių tinklų taikymo automobilių atpažinimui nuotraukose tyrimas

Baigiamasis magistro projektas

Valdymo technologijos (6211EX014)

Edvinas Boguša

Projekto autorius

Doc. Dr. Arūnas Lipnickas

Vadovas

Lekt. Gintautas Narvydas

Recenzentas

Kaunas, 2019

Kauno technologijos universitetas

Elektros ir elektronikos fakultetas

Edvinas Boguša

Giliųjų neuroninių tinklų taikymo automobilių atpažinimui nuotraukose tyrimas

Akademinio sąžiningumo deklaracija

Patvirtinu, kad mano, Edvino Bogušos, baigiamasis projektas tema „Giliųjų neuroninių tinklų taikymo automobilių atpažinimui nuotraukose tyrimas“ yra parašytas visiškai savarankiškai ir visi pateikti duomenys ar tyrimų rezultatai yra teisingi ir gauti sąžiningai. Šiame darbe nei viena dalis nėra plagijuota nuo jokių spausdintinių ar internetinių šaltinių, visos kitų šaltinių tiesioginės ir netiesioginės citatos nurodytos literatūros nuorodose. Įstatymų nenumatytų piniginių sumų už šį darbą niekam nesu mokėjęs.

Aš suprantu, kad išaiškėjus nesąžiningumo faktui, man bus taikomos nuobaudos, remiantis Kauno technologijos universitete galiojančia tvarka.

(vardą ir pavardę įrašyti ranka)	(parašas)

Edvinas Boguša. Giliųjų neuroninių tinklų taikymo automobilių atpažinimui nuotraukose tyrimas. Magistro baigiamasis projektas / vadovas Doc. Dr. Arūnas Lipnickas; Kauno technologijos universitetas, Elektros ir elektronikos fakultetas.

Studijų kryptis ir sritis (studijų krypčių grupė): elektronikos inžinerija, inžinerijos mokslai.

Reikšminiai žodžiai: Gilusis neuroninis tinklas, AlexNet, VGG, automobilių aikštelė, parkavimo vietų stebėjimas.

Kaunas, 2019. 50 p.

Santrauka

Pastarąjį dešimtmetį mašininio mokymosi srityje dominuoja vadinamieji gilieji neuroniniai tinklai, kurie panaudoja sparčiai besivystančias kompiuterių skaičiavimo ir duomenų apdorojimo technologijas. Darbo tikslas yra nustatyti, ištirti ir pritaikyti geriausią giliojo neuroninio tinklo architektūrą parkavimo aikštelės užimtumo nustatymui.

Pradžioje buvo atlikta literatūros analizė, kurioje apžvelgtos įvairios, regionais paremtos, giliųjų neuroninių tinklų architektūros ir trumpai supažindinama su spalvų erdvėmis, kurios dažnai yra pritaikomos nuotraukų apdorojimo uždavinių sprendimui.

Metodologinėje dalyje apžvelgiami tyrimo metu naudojami tinklai, jų struktūra bei panaudojimo galimybės pritaikant žinių perkėlimo metodą. Siekiant tyrimo metu palyginti skirtingų architektūrų tinklus, buvo atsižvelgta į literatūroje dažnai pritaikomą vidutinio tikslumo tinklų palyginimo metodą.

Projektinėje dalyje apžvelgiama naudota techninė įranga, plačiai pateikiama įvairių duomenų rinkinių paruošimo eiga ir tinklų apmokymo seka. Pristatomi tinkamiausio tinklo ir spalvų erdvės paieškos rezultatai. Apžvelgiamas tinklų pritaikymo ir greitaveikos tyrimas. Gauti rezultatai tarpusavyje palyginami. Galiausiai, remiantis tyrimų metu gautomis išvadomis, užsibrėžta sukurti automobilių parkavimo vietų žymėjimo ir skaičiavimo algoritmą.

Edvinas Boguša. Research of Convolutional Neural Networks Application for Cars Detection in Images. Master's Final Degree Project / supervisor Assoc. Prof. Dr. Arūnas Lipnickas; Electrical and Electronics Engineering, Kaunas University of Technology.

Study field and area (study field group): electronics engineering, engineering sciences.

Keywords: Convolutional Neural Networks, AlexNet, VGG, parking lot, parking space monitoring.

Kaunas, 2019. 50 pages.

Summary

In the last decade, machine learning has been dominated by so-called deep neural networks, which utilize rapidly developing computing and data processing technologies. The aim of this work is to identify, investigate and adapt the best architecture of the deep neural network for parking lot occupancy.

Initially, a literature review was carried out to review the various regional-based architectures of deep neural networks and to briefly introduce color spaces that are often adapted to solve photo processing tasks.

The methodological part provides an overview of the networks used in the research, their structure, and the possibilities of using them by applying the knowledge transfer method. In order to compare the networks of different architectures during the research, the method of comparison of average precision networks, often used in literature, was taken into account.

The project part focuses on the hardware used, the process of preparing various data sets and the training sequence of the networks. Introducing the most relevant network and color space search results. Network adaptation review and speed analysis are introduced. The results obtained are comparable. Finally, based on the findings of the research, an algorithm for marking and calculating parking spaces was designed.

Turinys

Lentelių sąrašas

Paveikslų sąrašas

Santrumpų ir terminų sąrašas

Santrumpos:

CNN (angl. *convoliutonal neural network*) – gilusis neuroninis tinklas;
RGB (angl. *red, green, blue*) – raudona, žalia, geltona;
HSV (angl. *hue, saturation, value*) – atspalvis, įsisotinimas, vertė;
ReLU (angl. *rectified linear unit*) – greitesnio ir efektyvesnio mokymo metodas, nustatantis neigiamas vertes iki nulio ir išlaikydamas teigiamas.

Terminai:

Sluoksnis (angl. *layer*) – neuronų grupė, turinti specifinę funkciją ir apdorojama kaip visuma. Dažniausias pavyzdys yra „feedforward“ tinklas, kuriame yra įvesties sluoksnis, išvesties sluoksnis ir vienas ar daugiau paslėptų sluoksnių.
Svoriai (angl. *weights*) – ryšio tarp dviejų neuronų stiprumas. Svoriai gali būti teigiami (sužadinantys) arba neigiami (slopinantys).
Žinių perkėlimas (angl. *transfer learning*) – yra mašininio mokymosi metodas, kai užduočiai sukurtas modelis pakartotinai naudojamas kaip pavyzdys antrajai užduočiai.
Epocha (angl. *epoch*) – sąvoka vartojama kalbant apie dirbtinius neuroninius tinklus. Epocha – toks laiko (ir skaičiavimų) tarpas, kurio metu visi įvesties aibės vektoriai yra pateikiami tinklui ir yra apdorojami. Vieną epochą sudaro n iteracijų, čia n – duomenų aibės dydis.
Iteracija (angl. *iteration*) – sąvoka vartojama kalbant apie dirbtinius neuroninius tinklus. Iteracija – toks laiko (ir skaičiavimų) tarpas, kurio metu vienas įvesties vektorius yra pateikiamas tinklui ir yra apdorojamas (kartu adaptuojami ir tinklo svoriai).
Forward Pass – tai skaičiavimo procesas tarp įėjimo duomenų ir išėjimo sluoksnių. Skaičiavimai atliekami nuo pirmo iki paskutinio sluoksnio. Paskutiniame sluoksnyje suskaičiuojant klaidos vektoriaus judėjimo kryptį.
Backward Pass – tai svorio pokyčių skaičiavimo algoritmas (de facto mokymasis). Skaičiavimas atliekamas iš paskutinio sluoksnio, atgal į pirmąjį. Svoriai atnaujinami remiantis gradientiniu klaidos vektoriumi ir mokymosi žingsniu.
Tikslumas (angl. *precision*) – nusako, koks tikslus buvo spėjimas. Tikslumas yra teisingų rezultatų skaičius padalintas iš visų gautų rezultatų skaičiaus.
Atmetimas (angl. *recall*) – nusako kaip gerai buvo rasti objektai. Recall yra teisingų rezultatų skaičius padalintas iš rezultatų, kurie turėjo būti gauti, skaičių.
Konvoliucija (angl. *convolution*) – vaizdų paleidimas per konvoliucinių filtrų rinkinį, kurių kiekvienas suaktyvina tam tikras funkcijas.
Pooling – netiesinės atrankos metodas, po kurio sumažinamas parametrų, kuriuos tinklas turi išmokti, skaičius.
Pilnai sujungtas sluoksnis (angl. *fully connected layer*) – CNN tinklo sluoksnis, kuriame aprašomos visos klasės, kurias tinklas gali spėti. Čia aprašomos kiekvieno objekto spėjimo tikimybės.
Softmax – CNN tinklo sluoksnis atiduodantis klasifikavimo rezultatą.

Įvadas

Laisvos automobilio parkavimo vietos radimas dažnai tampa varginančiu ir daug laiko užtrunkančiu procesu, dėl kurio kiekvieną dieną kenčia daugelis tankiai apgyvendintose vietose gyvenančių žmonių [1]. Todėl pageidautina sukurti konkurencingos kainos ir atpažinimo kokybės laisvų stovėjimo vietų aptikimo sistemą, kuri atpažintų laisvas parkavimosi vietas [1] bei informuotų vartotoją apie susidariusią situaciją [28]. Šią problemą yra bandoma spręsti įvairiais būdais [29]: parkavimo vietose montuojamos brangiai kainuojančios išmaniosios kameros (angl. *smart cameras*), diegiamos pastovaus palaikymo reikalaujančios jutiklių sistemos, užtvaru reguliuojamos stovėjimo aikštelės ir daugelis kitų. Tokių sistemų įdiegimas sutrumpina parkavimosi laiką miesto gatvėse, sumažina diegimo ir išlaikymo kaštus bei padidina patogumą, tačiau jų kaina ir / arba kokybė verčia ieškoti kitų variantų.

Išmaniųjų kamerų įdiegimas ir tobulėjančių algoritmų taikymas sumažina automobilių parkavimo vietų įrengimo laiką ir sudėtingumą [26], [33]. Tai veiksminga, efektyvi ir keičiamo mąsto stebėjimo sistema, kurią įdiegus galima pakeisti plačiai parkavimo vietose montuojamus jutiklius (pvz. magnetinis, atstumo jutiklis). Tačiau didelė diegimo ir išlaikymo kaina bei šių sistemų veikimo kokybę mažinantys įvairūs aplinkos veiksniai - šešėliai, atspindžiai, persidengimai, automobilių unikalumas - verčia ieškoti alternatyvų. Dėka tobulėjančių algoritmų bandoma spręsti keletą skirtingų problemų vienu metu, įskaitant automobilių radimo, sekimo ir atpažinimo uždavinius. Taip pat sudėtingesnio lygio problemas - transporto priemonių elgesio ir anomalijų nustatymą [27].

Dažniausiai sprendžiami nuotraukose ar vaizduose esančių objektų lokalizavimo ar klasifikavimo uždaviniai, kurie gali būti sprendžiami įvairiomis priemonėmis. [22] šaltinyje pateikiamas parkavimo aikštelėje esančių automobilių numerių atpažinimas pritaikant droną bei *Java* programavimo kalbą. Automobilio numeris atpažįstamas su licenzijuotu programiniu paketu *OpenALPR*, o dronas valdomas pagal iš anksto paruoštą GPS žemėlapį. Kitame šaltinyje [23] pateikiamas automobilyje integruotas laisvų parkavimo vietų sprendimas. Integruoti 4 „Fisheye“ jutikliai, kurie pagal *C++* programą, automobiliui važiuojant, realiu laiku, ieško laisvų parkavimo vietų. Ieškant kaip atpiginti sistemą, bandoma pritaikyti pigių mikrovaldiklių (Pvz.: *Arduino*, *SparkFun*) siūlomas technologijas [34], [35]. Tačiau šiuo metu smarkiai išpopuliarėjo autonominių mašinų parkavimui pritaikomi gilieji neuroniniai tinklai [24]. Šaltinyje pateikiamas autonominės mašinos parkavimo trajektoriją apskaičiuojančių giliųjų neuroninių tinklų palyginimas.

Gilieji neuroniniai tinklai (*CNN*) tapo viena iš įtakingiausių naujovių kompiuterinės regos srityje (angl. *computer vision*), kai Alexas Krizhevskis juos pritaikydamas, laimėjo 2012 m. vykusį „ImageNet“ konkursą. Renginio metu, klasifikavimo klaidos tikimybė buvo sumažinta nuo 26 % iki 15 %. Nuo tada daugybė įmonių savo paslaugų centrų apmokymui pradėjo diegti giliuosius neuroninius tinklus. Lietuvių įkurta įmonė *Pixevia* sukūrė realaus laiko parkavimo vietų stebėjimo ir informavimo sistemą. *Facebook* pritaikė neuroninius tinklus automatiniams nuotraukų žymėjimo algoritmams, *Amazon* – siūlomų produktų rekomendacijoms, *Pinterest* – pagrindinio puslapio personalizavimui, *Instagram* – paieškos infrastruktūros tobulinimui [2]. Šias ir kitas problemas padeda išspręsti giliųjų neuroninių tinklų taikymas.

Automobilių atpažinimo su giliaisiais neuroniniais tinklais taikymo sritys:

- Parkavimo vietų registravimo sistemoms – automobilių srauto registravimas įvažiavimo ir išvažiavimo metu, automatinis numerių atpažinimas, laisvų parkavimo vietų stebėjimas ir kita.
- Eismo priežiūrai – vairuotojų elgesio stebėjimui, greičio matavimui, prasižengimų registravimui, automobilių paieškai.
- Autonominėms mašinoms (angl. *self-driving cars*) – automobilio valdymui, parkavimui, eismo įvykių išvengimui ir kitų automobilių atpažinimui [24].
- Stebėjimo sistemoms – kaštų sumažinimui, senų vaizdo kamerų taikymas parkavimo vietų stebėjimo uždaviniams spręsti. Viena vaizdo kamera atstoja keliolika judesio jutiklių ir brangiai kainuojančių specifinių kamerų.
- Kelių stebėjimo kamerų atnaujinimui – gali būti įdiegiamos papildomos funkcijos (sekimas, judėjimo krypties nustatymas, adaptacija į aplinkos pokyčius), keičiamas stebimo objekto dydis, reguliuojamas jautrumas ir stebėjimo pozicija [31].

Darbo tikslas:

Ištirti giliųjų neuroninių tinklų taikymo galimybes atpažįstant automobilius ir suskaičiuojant laisvas parkavimo vietas su automobilių aikštelių nuotraukų rinkiniais.

Darbo uždaviniai:

1. sukurti programas giliųjų neuroninių tinklų apmokymui pritaikant žinių perkėlimo metodą;
2. palyginti giliųjų neuroninių tinklų mokymosi kokybę su stovėjimo aikštelių duomenų rinkiniais;
3. ištirti spalvų erdvių įtaką tinklų mokymosi rezultatams;
4. išanalizavus gautus rezultatus, rasti geriausią automobilių atpažinimo užduočiai tinkantį tinklą;
5. remiantis tyrimo metu gautais rezultatais sukurti automobilių parkavimo vietų žymėjimo ir skaičiavimo algoritmą.

Situacija:

Atlikus išsamią *CNN* architektūrų apžvalgą buvo nuspręsta automobilių atpažinimo uždavinių sprendimui panaudoti „faster R-CNN“ architektūros (*AlexNet, VGG-16* ir *VGG-19*) giliuosius neuroninius tinklus. Šie tinklai plačiau pristatomi 2.1 poskyriui – „Naudojamos tinklų architektūros“. Tinklų apmokymui panaudota keletas skirtingų duomenų rinkinių (*PUCPR+*, *CARPK*). Plačiau apie šiame darbe pasirinktus naudoti *PUCPR+* (stacionarios kameros) ir *CARPK* (drono) duomenų rinkinius pateikiama 3.2 poskyriui. Tinklų apmokymui su sąlyginai mažais duomenų rinkiniais panaudojamas žinių perkėlimo metodas.

Darbo metu išsikeltos problemos:

Duomenų rinkiniai išsiskiria skirtingais nuotraukų formatais, aplinkos sąlygomis, fotografavimo kampu, apšvietimu, blizgesiu ir daugeliu kitų faktorių, kurie gali daryti įtaką apmokyto tinklo rezultatams. Dėl šios priežasties reikalingas tyrimas, kurio metu būtų išsiaiškinta, kuriuo duomenų rinkiniu apmokytas tinklas yra tinkamesnis pagal 2.3 skyriuje pateikiamus tinklų vertinimo kriterijus.

Nėra aišku, kurios spalvų erdvės duomenų rinkiniu apmokytas tinklas veiks geriausiai. Todėl nuspręsta ištirti literatūroje dažnai minimas *RGB*, *HSV* ir *LAB* spalvų erdves, jų poveikį tinklų veikimui bei, atsižvelgiant į tinklų mokymosi rezultatus, rasti geriausiai tinkančią spalvų erdvę.

Literatūroje dažnai randama, kad tinkamam neuroninio tinklo apmokymui yra reikalingas didelis įvairių duomenų rinkinys. Todėl buvo nuspręsta apjungti *PUCPR+* ir *CARPK* duomenų rinkinius į vieną bendrą rinkinį ir patikrinti, kaip keičiasi apmokyto tinklo rezultatai. Tam sujungiami duomenų rinkiniai ir įvykdomas *MIX* rinkinio apmokymo tyrimas.

Su vienu duomenų rinkiniu apmokytas tinklas gali prastai veikti pateikus kito rinkinio tikrinimo informaciją. Norint patikrinti, ar prielaida yra teisinga, geriausiai apmokyti tinklai yra išbandomi su kitais duomenų rinkiniais.

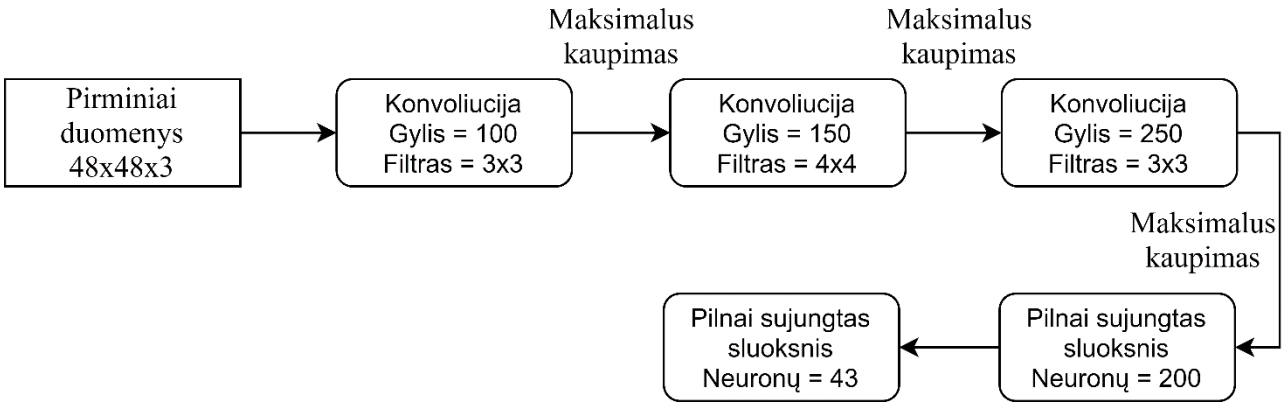
1. Teorinė dalis

1.1. CNN klasifikavimo architektūrų apžvalga

CNN architektūra paremtas tinklas buvo pristatytas 2011 metais, o nuo 2012 m. gilieji arba „konvoliuciniai“ neuroniniai tinklai (*CNN*, *ConvNet*) pradėjo dominuoti daugelyje kompiuterinės regos sričių. Šie tinklai tapo objektų klasifikavimo standartu, o 2015 m. „ImageNet“ konkurse aplenkė žmogų [10]. Gilieji neuroninai tinklai gali būti pritaikomi nuotraukų ir vaizdų atpažinimo sistemose, rekomendavimo sistemose [8], [31], vaizdų klasifikavimo, medicininių nuotraukų analizavimo [11], garso takelių klasifikavimo [14] ir natūralios kalbos apdorojimo sistemose [9].

Standartinis *CNN* buvo kuriamas objektų atpažinimui. Per paskutiniuosius keletą metų neatsiejama *CNN* dalimi tapo regionais paremti objektų atpažinimo metodai (angl. *regions-based object detection*), sutrumpintai vadinami *R-CNN* [3]. Šios architektūros tinklai apdoroja nuotrauką ir bando teisingai nustatyti vaizde esančių pagrindinių objektų buvimo vietą, juos pažymint aprėpties kontūrais (angl. *bounding box*).

Greitas šios srities vystymasis leido atsirasti efektyvesnėms šio tinklo kartoms. Per sąlyginai trumpą laikotarpį po *R-CNN* tinklo paskelbimo buvo sukurtas *Fast R-CNN* [4] tinklas, vėliau greitesnė jo versija *Faster R-CNN* [5] ir naujausia šio tinklo versija pavadinta *Mask R-CNN* [12]. Žemiau pateikiama pačio pirmojo *CNN* tinklo architektūra.



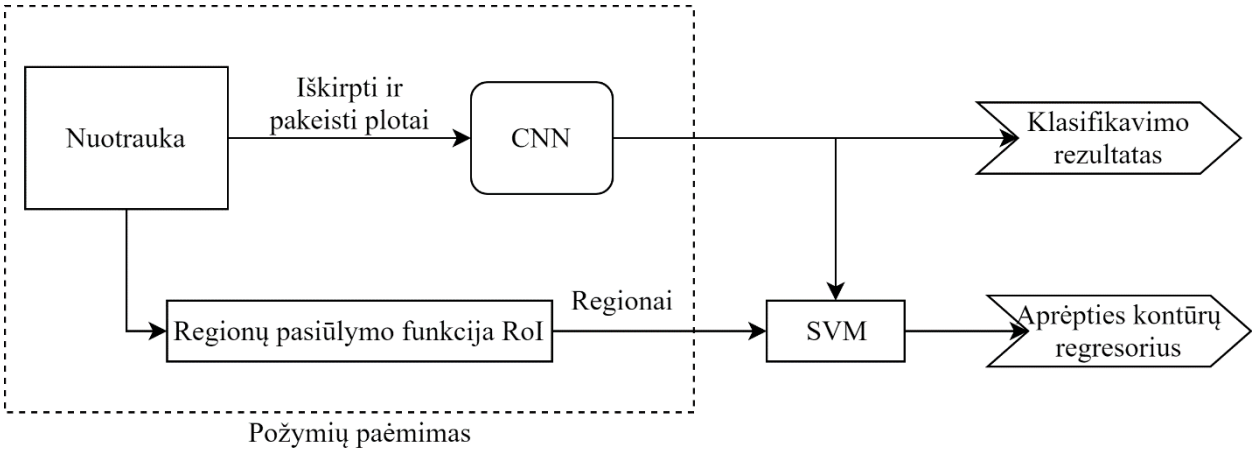
1 pav. *CNN* Architektūra

Pradinis *CNN* tinklo įėjimo duomenų dydis yra 48x48x3. Architektūrą sudaro 8 sluoksniai (žr. 1 pav.), kurių 3 įgyvendina konvoliucines operacijas, 3 maksimalaus kaupimo sluoksniai ir 2 pilnai sujungti sluoksniai. Architektūra pasižymi tuo, kad po kiekvieno konvoliucijos sluoksnio naudoja maksimalaus kaupimo sluoksnį (angl. *max-pool layer*). Ši architektūra tapo įvairių *R-CNN* architektūrų pagrindu.

1.1.1. R-CNN

R-CNN buvo pirmoji *CNN* architektūros modifikacija. Remiantis [3] šaltinio informacija, *R-CNN* pasiekė 53,7 % vidutinį tikslumą *MaP* (angl. *Mean average Precision*) su *PASCAL VOC 2010* duomenų rinkiniu. Pastaba: toliau pateikiamas *CNN* tinklas gali būti pakeičiamas bet kuriuo kitu tinklu, kuris yra palaikomas *R-CNN* architektūros.

R-CNN tinklo architektūra pavaizduota 2 pav., kuriame pateikiama, kaip iš nuotraukos paimti regionų pasiūlymų kandidatai sukeliami į *CNN*, kuris išėjime sukuria 4096 ilgio požymių vektorių (angl. *feature vector*). *CNN* išskiria požymius, kurių informaciją toliau siunčia į *SVM*, galutiniam objekto klasifikavimui. Taip pat, algoritmui spėjant objektą, yra spėjamas ir jo aprėpties kontūras, todėl tinklas sugeba vienu metu ne tik atlikti objektų lokalizavimo, bet ir klasifikavimo užduotį.



2 pav. *R-CNN* architektūra

Tinklas sukuria aprėpties kontūrus ir regionų pasiūlymus (angl. *region proposals*) pritaikant atrankinės paieškos (angl. *selective search*) metodą [13]. Šis regionų pasiūlymo metodas stebi nuotrauką per skirtingo mastelio langus ir bando sugrupuoti šalia esančius pikselius pagal tekstūrų ir spalvų panašumus, kontūro dydį ir užpildymą, apšvietimo sąlygas (šešėlį ir šviesos spalvą) ar intensyvumą. Visi šie kriterijai daro įtaką atrankinės paieškos metodo kokybei ir pasiūlymų kūrimo greičiui. Atrankos algoritmo pabaigoje *R-CNN* tinklas pakeičia rastų regionų mastelį į tinklui priimtiną dydį, pritaiko žinių perkėlimo metodą ir perduoda atrinktus požymius į tinklą (pvz.: *AlexNet*).

Paskutiniame *R-CNN* sluoksnyje esantis *SVM* (angl. *Support Vector Machine*) vektorinio palaikymo blokas yra atsakingas už objekto aprėpties kontūro patikslinimą ir supaprastinimą. Tai yra vienas iš apmokymo duomenų klasifikavimo metodų. Funkcijos tikslas yra apjungti mokymo rezultatus į atskiras erdvės dalis, paliekant tarp jų tik vektorius (angl. *support vectors*). Tokiu būdu apmokymas reikalauja mažiau darbinės atminties. Klasifikatoriui radus objektą, galima patikslinti turimą aprėpties kontūrą, kad atitiktų realius objekto išmatavimus. Tam pasitelkiamas paskutinis *R-CNN* žingsnis. Regiono pasiūlymui atliekama paprasta linijinė regresija, kad būtų sukuriamos griežtesnės kontūro koordinatės ir gautas galutinis rezultatas.

Apibendrinant *R-CNN* sistemą reikėtų paminėti šiuos veiksnius:

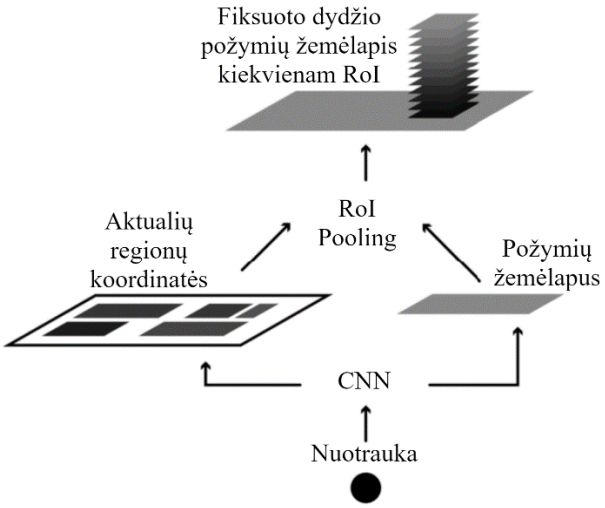
1. panaudojamas atrankinės paieškos metodas, kuris sukuria ~2000 regionų pasiūlymų (aprėpties kontūrų);
2. atrinkti regionai suformuojami į fiksuoto dydžio kvadratą ir perkeliami į tinklą (pvz.: *AlexNet*);
3. kiekvienos klasės išskirti požymiai yra įvertinami *SVM* bloko, priskiriant tikimybės įverčius;
4. objekto koordinatės patikslinamos pritaikant linijinį regresijos modelį.

R-CNN sukūrė pamatus šios srities tobulėjimui, tačiau eigoje susidūrė su keletu problemų. *CNN* privalo vykdyti atskirus perskaičiavimus (angl. *forward pass*) kiekvienam nuotraukos regionui (viso apie 2000 perskaičiavimų per nuotrauką). Taip pat tinklas yra sudarytas iš trijų atskirų modelių: *CNN*, kuris skirtas nuotraukos požymių išskyrimui, klasifikatoriaus, kuris nuspėja objekto klasę ir regresinio modelio, skirto aprėpties kontūro patikslinimui. Tokios struktūros tinklo apmokymui reikia daug skaičiavimo resursų. Pagal [3] šaltinio informaciją, vienos nuotraukos analizė trunka apie ~45 sekundes, todėl galima teigti, kad toks tinklas nėra tinkamas daugeliui realaus laiko uždavinių sprendimui. Taip pat tinkle naudojamas atrankinės paieškos algoritmas yra fiksuotas, t. y. šio etapo eigoje mokymasis nevykdomas. Šios priežastys prisidėjo prie prastesnių tinklo kokybės rezultatų. Daugelis *R-CNN* tinklo problemų buvo išspręstos su *Fast R-CNN* tinklu.

1.1.2. Fast R-CNN

2015 m., Ross Girshick, *R-CNN* autorius pristatė *Fast R-CNN* tinklą, kuris išsprendė pagrindines *R-CNN* tinklą stabdžiusias problemas [4]. Idėjos autorius suprato, kad tarp ~2000 randamų regionų pasiūlymų yra nemažai persidengiančių ar vaizduojančių tą patį objektą, todėl *CNN* buvo naudojamas atlikti pasikartojančius skaičiavimus kiekvienam regionui.

Perskaičiavimų problema buvo išspręsta paleidžiant *CNN* tik vieną kartą per nuotrauką ir pritaikant „RoI (angl. *Region of Interest*) Pooling“ sluoksnį. Veikimo principas pateikiamas paveikslėlyje žemiau.

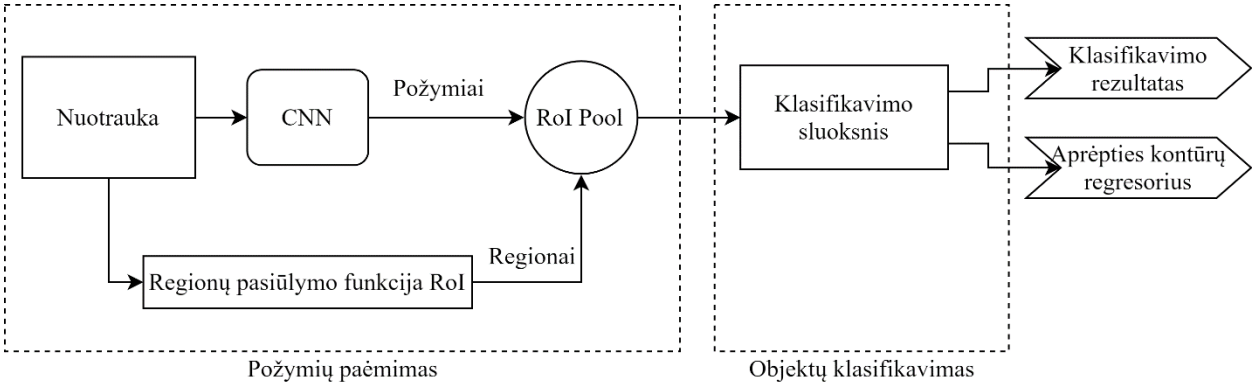


3 pav. *CNN* pritaikant „RoI Pooling“ sluoksnį

Pritaikius šį sluoksnį (pagal 3 pav.) *CNN* tinklas nuskaito nuotrauką ir atiduoda atrinktų regionų koordinates ir požymių žemėlapį. Toliau „RoI Pooling“ sluoksnyje nuotrauka suskirstoma į vienodo dydžio blokus, randamos didžiausios kiekvieno bloko vertės ir, priklausomai nuo pasirinkto santykio, gaunamas fiksuoto dydžio požymių žemėlapis kiekvienam *RoI*. Šis metodas pagreitina sistemos darbą, nes *CNN* sukurtą požymių žemėlapį galima panaudoti ne vienam regionui, o ~2000.

Kitas Fast *R-CNN* patobulinimas buvo kartu apmokyti *CNN*, klasifikatorių ir aprėpties kontūrų regresorių viename modelyje. Atsižvelgiant, kad *R-CNN* tinklas buvo sudarytas iš nuotraukos

požymių generavimo modelio (*CNN*), klasifikatoriaus (*SVM*) ir aprėpties kontūrų pataisymo algoritmo (linijinio regresinio modelio), buvo nuspręsta viską sujungti į vieną tinklą.

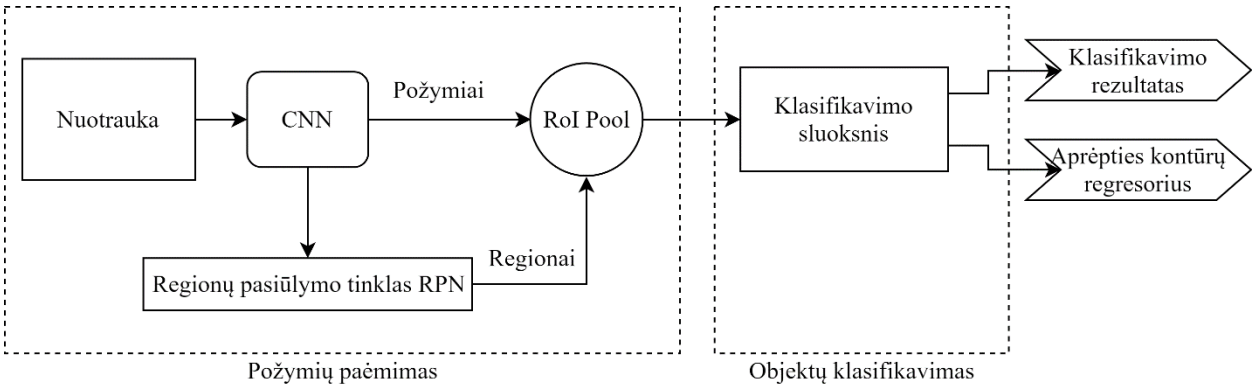


4 pav. *Fast R-CNN* architektūra

Fast *R-CNN* patobulinto modelio architektūra pateikta 4 pav., kuriame senas *SVM* klasifikatorius pakeistas nauju klasifikavimo sluoksniu, pavadintu „Softmax“. Lygiagrečiai pridėtas linijinis regresijos sluoksnis, skirtas aprėpties kontūrų išvedimui. Tokiu būdu visi reikalingi išėjimai gaunami iš vieno tinklo.

1.1.3. Faster R-CNN

Faster R-CNN tinklas atsirado su tikslu pakeisti visą procesą lėtinantį regionų pasiūlymo sluoksnį. *R-CNN* ir *Fast R-CNN* algoritmai naudojo atrankinės paieškos metodą, kuris yra lėtas ir daug laiko atimantis procesas, darantis didelę įtaką tinklo veikimui. Dėl šios priežasties Shaoqing Ren sugalvojo objektų aptikimo algoritmą, kuriam atrankinės paieškos metodas nereikalingas [5], [7]. Pirmojo „Forward Pass“ metu *CNN* gautus duomenis nuspręsta panaudoti tolimesniems regionų pasiūlymams. Pastebėta, kad regionais paremtuose detektoriuose, tokiuose kaip *R-CNN*, naudojamas požymių žemėlapis gali būti pritaikytas regionų pasiūlymų kūrimui. 5 pav. pateiktoje architektūroje matoma, kaip vienas *CNN* tinklas susitvarko su regionų pasiūlymais ir klasifikavimu. Tokiu būdu *CNN* turi būti apmokomas tik vieną kartą.



5 pav. *Faster R-CNN* architektūra

Panašiai kaip ir *Fast R-CNN*, nuotrauka įkeliama į *CNN*, kuriame sukuriamas požymių žemėlapis. Toliau, vietoje atrankinės paieškos algoritmo ir požymių žemėlapio panaudojimo regionų pasiūlymams gauti, panaudojamas atskiras *RPN* tinklas.

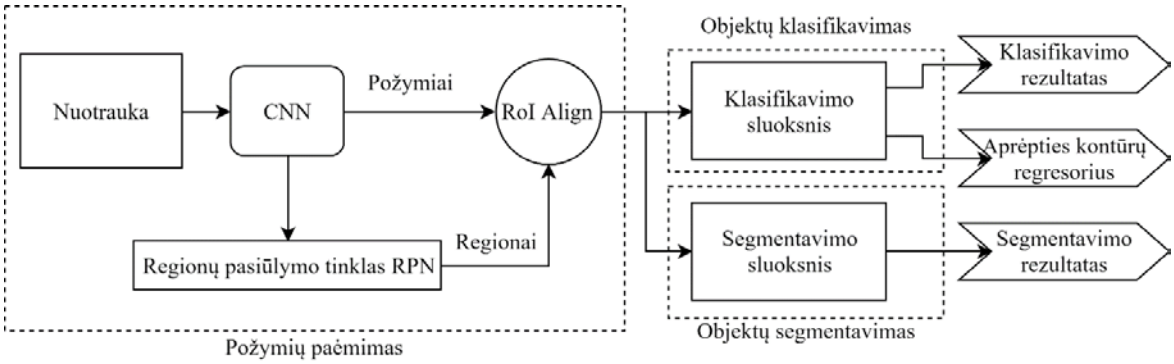
RPN algoritme panaudojamas „Sliding window“ principas kiekvienai požymių žemėlapio koordinatei. Yra priimta, kad tam tikri nuotraukos objektai turėtų tilpti į fiksuoto dydžio ir pločio santykius, kaip kad žmogus, kurio kvadratinis ir vertikalus aprėpties kontūras atspindi žmogaus siluetą. Todėl yra pritaikomi *k* standartiniai santykiai, vadinami inkaro kontūrais (angl. *anchor boxes*).

Kiekvienam šių inkarų išvedamas vienas aprėpties kontūras ir atitikimo rezultatas. Atsižvelgiant į šių inkarų principą panaudojant *CNN* požymių žemėlapį, išvedamas aprėpties kontūras kiekvienam inkarui ir atitikimo rezultatas, kuris nurodo tikimybę, kad tos nuotraukos aprėpties kontūre yra objektas. Numanomi regionų pasiūlymai pertvarkomi panaudojant „RoI pool“ sluoksnį, kuris paskui panaudojamas nuotraukos klasifikavimui pagal pasiūlytą regioną bei objekto kontūro sutvarkymui. Kiekvienas „RoI Pool“ narys pereina per objektų klasifikavimo sluoksnį, tokiu būdu objektui priskiriamas aprėpties kontūras bei etiketė (angl. *label*). Tokiu principu gaunamas šiuo metu vienas iš geriausių regionais paremtas objektų aptikimo ir klasifikavimo tinklas.

1.1.4. Mask R-CNN

Iki šio tinklo pasirodymo, *CNN* pateikiami požymiai buvo panaudojami daugeliu skirtingų būdų norint sėkmingai rasti nuotraukos objektus, pritaikant aprėpties kontūrus. Šiame tinkle ieškomi ne aprėpties kontūrai, o kiekvieno objekto pikseliai. Toks aptikimo metodas vadinamas segmentavimu. Vienas pirmųjų sėkmingai pritaikęs šį metodą buvo Kaiming He, kuris su mokslininkų komanda, panaudodamas *Facebook* duomenų rinkinį, sukūrė *Mask R-CNN* [12]. Šio tinklo pritaikymas, dirbant su aktualiu *PKLot* duomenų rinkiniu, pateikiamas [36] šaltinyje.

Mask R-CNN gaunamas panaudojant *Faster R-CNN* architektūrą ir pridedant išsišakojimą prieš *Faster R-CNN* aprėpties kontūrus ir klasifikavimo sluoksnius. Prie tinklo prijungiamas pilnai „konvoliucinis“ tinklas *FCN* (angl. *Fully Convolutional Network*), kuris išveda dvejetainę kaukę (angl. *binary mask*) įvertinančią, ar duotas pikselis yra objekto dalis. Ši nauja šaka įėjimui naudoja *CNN* požymių žemėlapį. Išėjimui sukuriama matrica, kurioje vienetais pažymimi objektams priklausantys pikseliai ir nuliais pažymima visa kita (todėl tai ir vadinama dvejetaine kauke). Tinklo architektūra pateikiama 6 pav.



6 pav. *Mask R-CNN* architektūra

Kadangi nuotraukų segmentavimas reikalauja pikselių lygio tikslumo, priešingai nei aprėpties kontūrai, eigoje buvo pastebėta, kad požymių žemėlapio regionai nežymiai nesutampa su

originalaus paveikslėlio regionais. Siekiant išspręsti šią problemą, buvo pakeistas *RoIPool* sluoksnis. *Faster R-CNN* tinklo autoriai sugebėjo išspęsti duomenų iškreipimo problemą pakeisdami *RoIPool* sluoksnį į atnaujintą RoIAlign. Šis metodas *RPN* sluoksnyje išvengia pikselių apvalinimo, kuris netiksliai pakeičia pasiūlytų regionų mastelį. *RoIAlign* sluoksnyje panaudojamas bilinearios interpoliacijos metodas norint sužinoti tikslią pikselio informaciją po nuotraukos transformacijos į požymių žemėlapį. Sukurtos kaukės *Mask R-CNN* sujungiamos su klasifikavimo rezultatais ir aprėpties kontūrais paimtais iš *Faster R-CNN*. Tokiu būdu *Mask R-CNN* lygiagrečiai atlieka objektų segmentavimą ir klasifikavimą.

1.2. Spalvų erdvių apžvalga

1931 m. tarptautinė apšvietimo komisija, žinoma akronimu *CIE* (angl. *Commission Internationale de l'Éclairage*), tyrinėjo žmogaus suvokiamų spalvų spektrą ir sukūrė standartą, vadinamą *CIE XYZ*. Tai standartas, apibrėžiantis trimatės erdvės pagrindinių spalvų koordinates (angl. *tristimulus*), kurios apibrėžia spalvą. Šis standartas šiandien vis dar plačiai naudojamas ir yra laikomas neapdorotu formatu, nuo kurio ir kilo kiti spalvų modeliai.

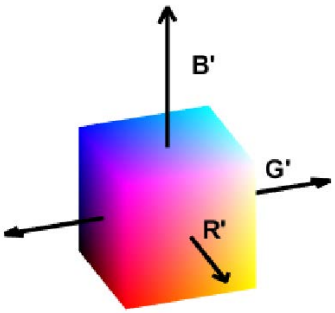
Spalvų modelis yra abstraktus matematinis modelis, kuris nusako, kaip spalvos gali būti aprašomos skaitmeninėje erdvėje. Standartiškai modelis užkoduojamas bitais, pavyzdžiui 8 bitų *RGB* modelio R dedamosios vertės yra nuo 0 iki 255. Daugelis šiuolaikinių spalvų modelių turi 3 erdvinius matmenis, todėl gali būti vaizduojami įvairiomis 3D figūromis, tiksliau spalvos gamut‘ais (angl. *gamut*). Tokie modeliai gali būti konvertuojami iš vieno į kitą, tam reikalingi matematiniai ir trigonometriniai perskaičiavimai. Yra 5 pagrindiniai spalvų modeliai: *CIE*, *RGB, YUV*, *HSL*/ *HSV* ir *CMYK*, kurie toliau skirstomi į smulkesnes kategorijas. Gali būti aparatiniai ir spalvos manipuliavimo modeliai. *RGB* yra aparatinis modelis, skirtas atvaizdavimui skaitmeniniuose ekranuose, o *CMYK* – pritaikomas spalvotuose spausdintuvuose. Modelį siejant su specifiniais reikalavimais, gaunamas spalvų rinkinys, kuris vadinamas „Spalvų erdve“. Spalvų erdvė gali būti apibūdinama kaip spalvų koordinačių sistema, kur kiekvienam koordinačių taškui yra priskirtas unikalus spalvos kodas.

Spalvų erdvės pakeitimas praktikoje yra dažnas problemų sprendimo metodas [25]. Medicinoje pritaikoma žmogaus odos požymių aptikimui, klasifikuojant skirtingų spalvų modelių skaitmenines nuotraukas [18]. Kitame šaltinyje atliktas tyrimas, kuriame skirtingais spalvų modeliais ir duomenų rinkiniais įgyvendinamas mikroskopinių audinių struktūrų klasifikavimo kokybės palyginimo tyrimas [19]. Dar viename šaltinyje pateikiami eksperimentų metu gauti rezultatai, veidų atpažinimui skirtą tinklą apmokant su skirtingomis spalvų erdvėmis, tarp kurių pateikiamos *RGB, HSV* ir *LAB* spalvų erdvės [17]. Remiantis šaltinyje pateiktais rezultatais, geriausiai pasirodė *LAB* spalvų erdvė. Šiuo spalvų formatu apmokytas tinklas, pagal tikslumo kriterijų, pasirodė 10 % geriau, lyginant su standartiniu *RGB* modeliu. Remiantis [19] šaltiniu, *RGB* spalvų erdvės konvertavimas pagerino segmentavimo rezultatus.

Praktikoje sėkmingai pritaikomi skirtingi spalvų modeliai ir spalvų erdvės, tačiau nėra aišku, kaip gilieji neuroniniai tinklai apmokymo metu reaguoja į skirtingus spalvų modelius. Norint atlikti tyrimą, pirmiausia reikia susipažinti su skirtingomis spalvų erdvėmis, jų skirtumais bei galimybėmis.

1.2.1. RGB

RGB spalvų modelis buvo sukurtas išskirtinai atvaizdavimui (ekranai, projektoriai ir kt.). Daugelis prekyboje esančių elektroninių prietaisų ekranų gali būti asocijuojami su *RGB*. Tai pridėtinių spalvų modelis, kuriuo pasinaudojus, galima sukurti bet kokią spalvą proporcingai suliejant raudoną, žalią ir mėlyną spalvas (žinomos kaip trys pagrindinės spalvos). Toks procesas kitaip dar vadinamas erdviniu spalvų maišymusi (angl. *spatial color mixing*). Remiantis šiuo modeliu buvo sukurta daug skirtingų spalvų erdvių: *sRGB, Adobe RGB, Adobe Wide Gamut RGB* bei kitos. Žemiau pateikiamas *RGB* spalvų erdvės grafinis atvaizdavimas.

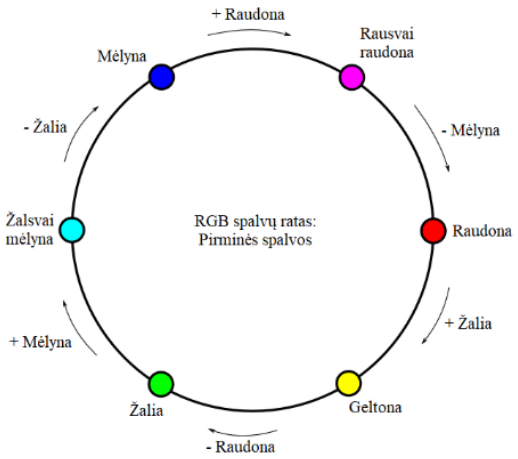


7 pav. *RGB* Spalvų erdvės iliustracija (pagal Pascal Getreuer skriptą)

7 pav. pateiktos modelio dedamosios šifruojasi taip:

- R – raudona (angl. *red*), kuri svyruoja nuo 0 iki 255;
- G – žalia (angl. *green*), kuri svyruoja nuo 0 iki 255;
- B – mėlyna (angl. *blue*), kuri svyruoja nuo 0 iki 255.

Šios dedamosios gali būti suskirstomos į taip vadinamą spalvų ratą, kurio iliustracija pateikiama 8 pav.



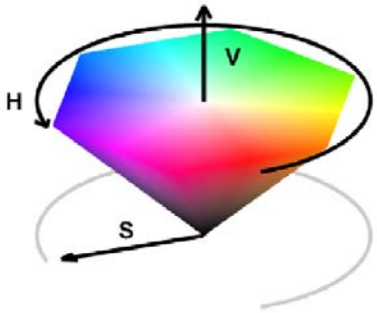
8 pav. *RGB* spalvų ratas

Remiantis spalvų rato metodu galima apibūdinti bet kokią gryną spalvą kaip raudonos, žalios ir mėlynos spalvos mišinį. Paveikslėlyje žiūrint nuo raudonos, pridedant žalios spalvos, gauta spalva pasislenka link geltonos. Iš geltonos spalvos atėmus dalį raudonos, gauta spalva pasislenka ties

žalia. Šiuo principu gaunamas uždaras RGB spalvų ratas, kuriame visos pirminės spalvos yra priklausomos viena nuo kitos.

1.2.2. HSV

HSV spalvų erdvė yra pamėgta menininkų ir kitų su spalvomis dirbančių žmonių, kuriems tenka atrinkinėti spalvas (dažus, rašalus, maketo spalvas). Tai yra todėl, kad *HSV* spalvų erdvė geriau atitinka žmogaus suvokiamų spalvų spektrą nei *RGB*.



9 pav. *HSV* Spalvų erdvės iliustracija (pagal Pascal Getreuer skriptą)

HSV spalvų erdvės grafinis vaizdas pateikiamas 9 pav. *HSV* spalvų modelio dedamosios yra šios:

- H – atspalvis (angl. *hue*), kinta nuo 0° iki 360° arba nuo 0 iki 1;
- S – įsisotinimas (angl. *saturation*), kinta nuo 0 iki 1;
- V – vertė arba šviesumas (angl. *value / whiteness*), kinta nuo 0 iki 1.

Atspalvio reikšmė gali kisti nuo 0 iki 1 arba 0° iki 360°, kai 0 atitinka raudoną spalvą, po kurios seka geltona, žalia, mėlyna ir kitos, ties vienetu arba 360° grįžtant į raudoną. Tokiu būdu raudona spalva šiame modelyje pasikartoja du kartus. Įsisotinimas kinta nuo 0 iki 1, atitinkamos spalvos (atspalviai) skiriasi nuo neutralių (pilkos spalvos atspalvių) iki visiškai įsisotinusių (su baltos spalvos komponentu). Vertė gali būti laikoma spalvos grynumu, kadangi vertė arba ryškumas svyruoja nuo 0 iki 1, artėjant prie 1 atitinkamos spalvos komponentas tampa vis ryškesnis.

RGB į *HSV* konvertavimas pateikiamas pagal Adrian Ford knyga „Colour Space Conversions“. Pritaikius (1) formulę gaunamos *R*, *G*, *B* vertės, intervale nuo 0 iki 1. Čia „$scale_x$“ kintamasis reiškiantis spalvos kanalo skalę, pavyzdžiui 255.

$$R = \frac{R'}{scale_r}, G = \frac{G'}{scale_g}, B = \frac{B'}{scale_b} \quad (1)$$

Įsistačius gautas *RGB* vertes į (2) ir (3) formules gaunamos minimalios ir maksimalios *RGB* spalvų vertės.

$$m_{max} = \max(R, G, B) \quad (2)$$

$$m_{min} = \min(R, G, B) \quad (3)$$

Pagal (4) formulę apskaičiuojamas spalvų verčių skirtumas Δ:

$$\Delta = m_{max} - m_{min} \quad (4)$$

Atspalvio *H* reikšmė randama pagal (5) formulę. Pirmiausia surandama didžiausia *RGB* spalvų erdvės kanalo vertė. Priklausomai nuo didžiausios m_{max}, randamas skirtumas tarp dviejų mažiausių *RGB* spalvų erdvės verčių ir gautas skirtumas padalijamas iš (4) formulėje gauto spalvų verčių skirtumo. Toliau atspalvis normalizuojamas prie gautos vertės pridedant 0, 2, 4. Rezultate gaunamas realaus tipo skaičius, kurio vertės 0 ir 6 ribose nurodo atspalvio *H* reikšmę, o mažesnės už 0 ir didesnės už 6 yra laikomos nereikalingomis.

$$H = \begin{cases} neapibrėžtumas, & jeigu\ \Delta = 0 \\ \dfrac{G-B}{\Delta}, & jeigu\ m_{max} = R, \Delta \neq 0 \\ \dfrac{B-R}{\Delta} + 2, & jeigu\ m_{max} = G, \Delta \neq 0 \\ \dfrac{R-G}{\Delta} + 4, & jeigu\ m_{max} = B, \Delta \neq 0 \end{cases} \quad (5)$$

Šviesumas *V* yra priklausomas nuo šviesiausios spalvos kanalo. Vertė gaunama pagal (6) ir (7) formules.

$$V = m_{max} \quad (6)$$

$$V' = V \times scale_v \quad (7)$$

Įsisotinimas, *S* yra skirtumas tarp didžiausios ir mažiausios spalvų kanalo vertės, padalintos iš šviesumo *V*, kaip pateikiama (8) ir (9) formulėse. Jeigu šviesumas *V* yra lygus 0, tai gautas įsisotinimas irgi lygus 0.

$$S = \begin{cases} 0, & jeigu\ V = 0 \\ \dfrac{\Delta}{V}, & kita \end{cases} \quad (8)$$

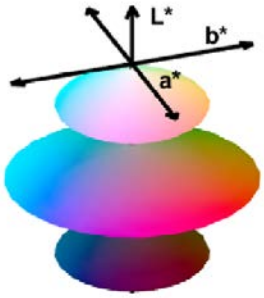
$$S' = S \times scale_S \quad (9)$$

Atlikus visus skyriuje pateikiamus skaičiavimus gaunama *HSV* spalvų erdvė, kurios kintamaisiais atvaizduojamos spalvos yra vienos tiksliausių ir lengviausiai suvokiamų spalvos kodavimo sistemų.

1.2.3. CIE L*a*b*

*CIE L*a*b** spalvų erdvė, toliau *LAB*, buvo sukurta 1976 metais, remiantis skyriaus pradžioje minėtu *CIE XYZ* neapdorotu spalvų modeliu. Šio modelio kūrimo metu buvo tikimasi suvienodinti suvokimą tarp skirtingų spalvos vektorių. Nors modelis buvo patvirtintas kaip de facto modelis, skirtas absoliučių spalvų koordinatėms ir spalvų skirtumams vaizduoti, jis niekada nebuvo oficialiai pripažintas tarptautiniu standartu.

Modelio spalvos informacija yra gaunama remiantis sistemos baltuoju tašku (angl. *white point*). O nelinijinė L*, a*, b* verčių priklausomybė yra tokia pati, kaip *CIE LUV*, ir yra skirta imituoti akies logaritminį atsaką. Šis modelis yra nepriklausomas nuo aparatūros, todėl gali būti vienodai atvaizduojamas skirtinguose įrenginiuose.



10 pav. *LAB* Spalvų erdvės iliustracija (pagal Pascal Getreuer skriptą)

LAB spalvų erdvės grafinis modelis pateikiamas 10 pav., kuriame *L*a*b** spalvų erdvės dedamosios šifruojasi taip:

- L* – šviesumas (angl. *lightness*);
- a* – raudona / žalia spalvų spektro spalvos;
- b* – mėlyna / geltona spalvų spektro spalvos.

Konvertavimas iš *RGB* į *LAB*, remiantis Gernot Hoffmann knyga „CIELab Color Space“. Kadangi *RGB* yra neiškreiptos vertės, kurios yra tiesiškai susijusios su *CIE XYZ*, konvertavimas atliekamas remiantis *XYZ* spalvų erdve, o prisitaikymo problema išsprendžiama linijine Bradfordo (angl. *Bradford*) transformacija. *XYZ* verčiama į *LAB*, kai X_n yra baltos spalvos koordinatė po balto taško korekcijos, pritaikant Bradfordo transformaciją. (11) formulėje įsistačius (10) formulės kintamuosius gaunamos XYZ_n vertės. Pagal (10) formulę realią spalvą padalinus iš transformuotos, gaunamas spalvų erdvės skirtumo koeficiento XYZ_1 vertės.

$$X_1 = \frac{X}{X_n}, \qquad Y_1 = \frac{Y}{Y_n}, \qquad Z_1 = \frac{Z}{Z_n} \tag{10}$$

Gautos XYZ_1 vertės įstatomos į (11) funkcijas:

$$X_1 = \begin{cases} X_1^{1/3}, & jeigu\ X_1 > 0.008856 \\ 7.787X_1 + 16/116, & kita \end{cases} \tag{11}$$

$$Y_1 = \begin{cases} Y_1^{1/3}, & jeigu\ Y_1 > 0.008856 \\ 7.787Y_1 + 16/116, & kita \end{cases}$$

$$Z_1 = \begin{cases} Z_1^{1/3}, & jeigu\ Z_1 > 0.008856 \\ 7.787Z_1 + 16/116, & kita \end{cases}$$

Gautos XYZ_1 vertės įstatomos į (12) formules, taip gaunant spalvų erdvės vertes L^*, a^*, b^*.

$$L^* = 116Y_1 - 16 \tag{12}$$

$$a^* = 500(X_1 - Y_1)$$

$$b^* = 200(Y_1 - Z_1)$$

Pritaikius šias formules gaunamas LAB spalvų erdvės spektras.

2. Metodologinė dalis

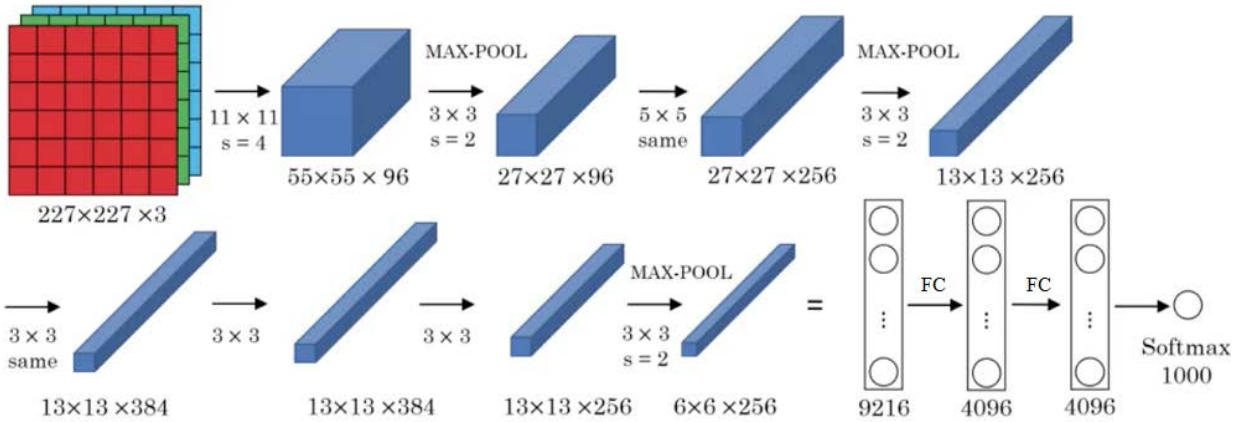
2.1. Naudojamos tinklų architektūros

Kasmetinis *ImageNet* konkursas *ILSVRC* (angl. *ImageNet Large Scale Vision Recognition Challenge*) nuo 2010 metų skatiną vaizdų atpažinimo technologijų vystymąsi. Konkurso dalyviams suteikiama prieiga apmokyti sukurtus *CNN* tinklus pasinaudojant 1.4 milijono nuotraukų turinčiu duomenų rinkiniu. *ILSVRC* konkurse yra laimėję šie tinklai: *LeNet-5*, *AlexNet*, *VGGNet*, *GoogLeNet*, *ResNet*. Šio tyrimo metu buvo lyginamos *AlexNet* ir *VGG-16* ir *VGG-19* tinklų architektūros.

2.1.1. AlexNet

AlexNet yra pavadintas jo autoriaus Alex Krizhevsky garbei, kai 2012 metais jo sukurtas tinklas laimėjo *ILSVRC* konkurso pirmąją vietą. Konkurso tikslas buvo sukurti efektyvų sprendimą 1000 objektų klasifikavimui (klaviatūra, puodelis, tušinukas, gyvūnai ir kita). Tinklas sukurtas apmokymui panaudojant virš milijono nuotraukų iš dalinio *ImageNet* duomenų rinkinio. Apmokymas truko 6 dienas, panaudojant dvi lygiagrečiai veikiančias Nvidia GeForce GTX 580 vaizdo plokštes. Šios architektūros sukūrimas paremtas *LeNet* architektūra įrodant, kad maksimalaus kaupimo sluoksnis (angl. *max-pooling*) po kiekvieno konvoliucijos sluoksnio yra neprivalomas.

Standartinį *AlexNet* tinklą sudaro 11 sluoksnių (5 konvoliuciniai, 3 maksimalaus kaupimo sluoksniai, 3 pilnai sujungti sluoksniai). Proceso pagreitinimui naudojama *ReLU* aktyvavimo funkcija vietoj „Sigmoid“ arba „Tanh“. Pritaikant šią funkciją greitis padidinamas 5 kartus, o tikslumas išlieka nepakitęs. Norint apsisaugoti nuo permokymo (angl. *overfitting*) pritaikoma „dropout“ funkcija, tačiau tinklo apmokymo laikas padvigubėja. Žemiau pateikiama platesnė šios architektūros informacija.



11 pav. *AlexNet* architektūra

AlexNet priima *RGB* spalvų standarto nuotraukas, kurių dydis yra 227x227. Tolimesnis konvoliucinis sluoksnis pritaiko 96 filtrus (angl. *kernel*), kurių dydis yra 11x11 su 4 pikselių dydžio filtro žingsniu. Tokiu būdu nuotraukos formatas sumažinamas iki 55x55 pikselių. Toliau naudojamas maksimalaus kaupimo sluoksnis, kurio parametrai 3x3 ir 2 pikselių dydžio filtro žingsnis. Pagal 11 pav. sluoksniai kartojami keičiant tik jų vidinius parametrus iki tol, kol yra

pasiekiamas pilnai sujungtas sluoksnis. Pirmas FC sluoksnis turi 9216 neuronų, o kiti du FC sluoksniai turi po 4096 neuronų. Pabaigoje panaudojama „softmax“ funkcija turinti 1000 išėjimo klasių. Bendrai *AlexNet* architektūra turi 60 milijonų parametrų.

2.1.2. VGG-16

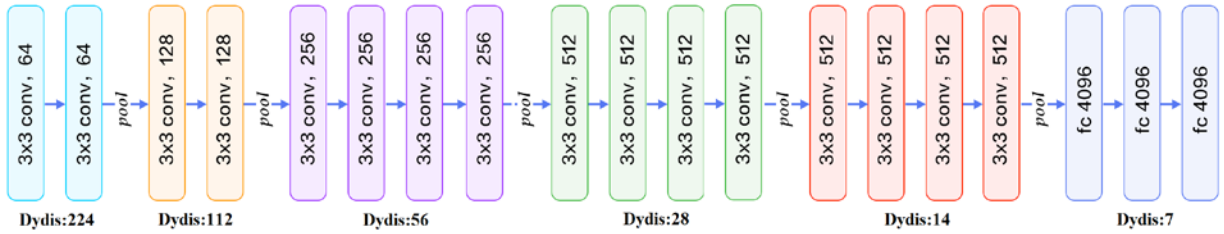
VGG-16 architektūra buvo sukurtas 2014 metų *ILSVRC* konkurse ir tais pačiais metais laimėjo. Tai iš anksto apmokytas konvoliucinis neuroninis tinklas, kurio apmokymui buvo panaudota daugiau nei 1 milijonas nuotraukų iš *ImageNet* duomenų rinkinio. Kaip ir *AlexNet* buvo kuriamas su tikslu klasifikuoti nuotraukas į 1000 objektų kategorijas. Kitaip dar vadinamas „OxfordNet“. Tinklas susideda iš paprastesnės architektūros modelio nei *AlexNet*, nes architektūra sudaryta iš paprastesnių ir pasikartojančių sluoksnių. Tačiau šis tinklas yra gilesnis nei *AlexNet* - susideda iš daugiau sluoksnių. Plačiau apie šio tinklo struktūrą pateikiama 12 pav.

12 pav. *VGG-16* architektūra

Šios architektūros tinklas buvo kuriamas remiantis idėja, kad kuo ilgesnis neuroninis tinklas, tuo tikslesnius rezultatus galima pasiekti, išmokant tinklą atpažinti daugiau požymių. Tinklas sudarytas iš 21 sluoksnio, iš kurių 5 yra maksimalaus kaupimo, 3 pilnai sujungti ir 13 konvoliucinėms operacijoms skirti sluoksniai. Tinklo įėjimas yra apibrėžtas 224x224x3 (*RGB* spalvų tipo) nuotraukos dydžiu. Konvoliuciniame sluoksnyje visada naudojamas 3x3 dydžio filtras su 1 pikselio dydžio filtro žingsniu, o maksimalaus kaupimo sluoksniuose buvo naudojamas 2x2 dydžio filtras, kurio žingsnio dydis yra 2 pikseliai. Bendrai *VGG-16* architektūra turi 138 milijonus parametrų.

2.1.3. VGG-19

VGG-19 tinklas nuo *VGG-16* skiriasi tik sluoksnių skaičiumi. Tinklo architektūra pateikiama 13 pav.



13 pav. *VGG-19* architektūra

Tinklas sudarytas iš 24 sluoksnių, kuriame 5 yra maksimalaus kaupimo, 3 pilnai sujungti ir 16 konvoliucinėms operacijoms skirti sluoksniai. Tinklo įėjimas yra apibrėžtas 224x224x3 (*RGB* spalvų tipo) nuotraukos dydžiu. Konvoliuciniame sluoksnyje visada naudojamas 3x3 dydžio filtras su 1 pikselio dydžio filtro žingsniu, o maksimalaus kaupimo sluoksniuose buvo naudojamas 2x2 dydžio filtras su 2 pikselių žingsnio dydžiu. Bendrai *VGG-19* architektūra turi 144 milijonus parametrų.

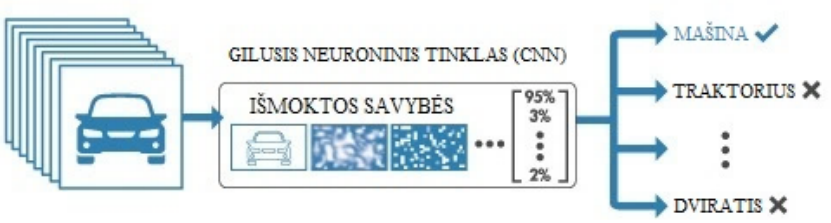
2.2. Žinių perkėlimo metodas

Žinių perkėlimo metodas (angl. *transfer learning*) yra dažnai naudojamas giliųjų neuroninių tinklų apmokymo algoritmuose. Tai *CNN* mokymosi metodas, kuriame vienam uždaviniui skirtas modelis yra naudojamas kaip pradinis taškas apmokant panašią užduotį atliekantį tinklą. Esamo tinklo žinių perkėlimo suderinimas (angl. *fine-tuning*) paprastai yra daug greitesnis ir lengvesnis nei tinklo mokymas nuo pradžių su atsitiktinai parinktais svoriais. Pritaikant šį metodą tinklo apmokymui galima ne tik pagreitinti naujo tinklo mokymo procesą, bet ir pagerinti naujo tinklo apmokymo kokybę [30].

Metodas dažniausiai yra naudojamas objektų aptikimo, vaizdų atpažinimo, kalbos atpažinimo programose ir daugelyje kitų sričių. Žinių perkėlimo metodas yra populiarus dėl šių metodo privalumų:

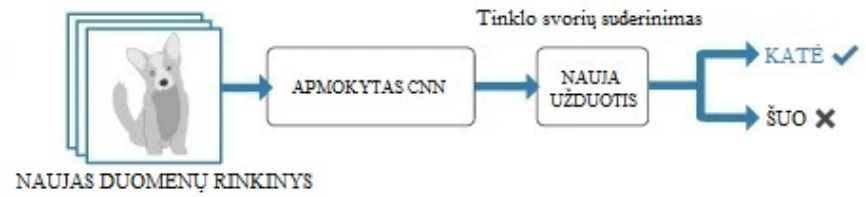
- drastiškai sumažinamas mokymosi laikas ir tausojami skaičiavimo ištekliai;
- reikalingas mažesnis mokymosi epochų skaičius;
- išsprendžiama problema, kai sužymėtų duomenų kiekis yra sąlyginai mažas, panaudojant populiarius modelius, kurie jau buvo apmokyti dideliuose duomenų rinkiniuose;
- lengvai iš vieno modelio į kitą perduodamos žinios, remiantis turima įranga ir darbo tikslais;
- sukuriamas tinklas yra stabilesnis ir gali būti plačiau pritaikomas;
- yra vienas iš žingsnių į ateities dirbtinį intelektą.

Standartiškai yra naudojami du giliojo mokymo metodai: mokymo nuo nulio ir žinių perkėlimo metodas. Abu metodai gali būti naudingi sprendžiant įvairias giliojo mokymo užduotis. Modelio kūrimas ir mokymas nuo nulio yra tinkamesnis labai specifinėms užduotims, kurioms negalima pritaikyti jau sukurtų modelių. Šio metodo neigiama savybė yra ta, kad paprastai tinklo apmokymui nuo nulio yra reikalingas žymiai didesnis apmokymo duomenų rinkinys. Modelio principinė schema pateikiama 14 pav.



14 pav. Apmokymo nuo nulio modelio principinė schema

Žinių perkėlimo metodas yra populiarus objektų atpažinimo srityje. Kaip pradžios taškas naujam tinklui gali būti panaudojama daugybė įvairių iš anksto apmokytų tinklų: *AlexNet, VGG, GoogleNet*, bei daugelis kitų. *Pavyzdžiui, turint projektą, kuriame reikia klasifikuoti skirtingų tipų automobilius ir yra ribotas duomenų kiekis, galima perkelti AlexNet tinklo sluoksnius ir svorius, pakeičiant tik paskutinį – klasifikavimo sluoksnį.* Žinių perkėlimo modelio principinė schema pateikiama žemiau.



15 pav. Žinių perkėlimo modelio principinė schema

Žinių perkėlimo modelio įgyvendinimui paprastai turi būti atliekami šie žingsniai:

- užkraunamas iš anksto apmokytas tinklas, pasirenkant tinklą, kuris yra panašus į norimą pasiekti tikslą;
- pakeičiamas klasifikavimo sluoksnis, o turint didelį duomenų rinkinį galima užsiimti svorių ir sluoksnių perderinimu (perderinant su mažu duomenų rinkiniu galima susidurti su permokymu);
- tinklo apmokymas su naujais duomenimis;
- naujo tinklo įvertinimas.

Rezultate gaunamas naujas tinklas, kurio apmokymui reikalingas žymiai mažesnis duomenų kiekis, apmokymui reikalingas laikas yra trumpesnis, o apmokymas reikalauja mažiau skaičiavimo resursų.

2.3. Tinklų vertinimo kriterijai

Mokymosi progresas gali būti įvertinamas įvairiais efektyvumo kriterijais. *PASCAL VOC* konkurso organizatoriai pasiūlė *MATLAB* algoritmą, kurį pritaikant galima įvertinti objektų atpažinimo kokybę. Pilna šio algoritmo struktūra bei informacija pateikiama [21] šaltinyje. Tinklai dažniausiai vertinami pagal šiuos tinklų vertinimo kriterijus: tikslumo ir atmetimo kreivę (angl. *precision and recall curve)* bei vidutinį tikslumą (angl. *average precision*). Toliau pateikiamas platesnis šių kriterijų paaiškinimas.

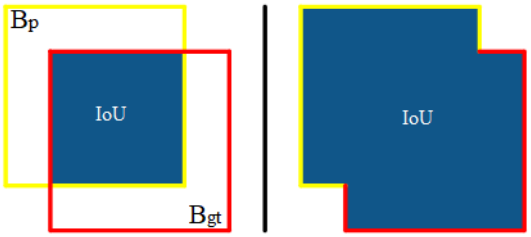
2.3.1. Susikirtimas per sąjungą (IoU)

Susikirtimas per sąjungą (angl. *Intersection over Union*) – tai matavimas, paremtas „Jaccard“ indeksu, kuris įvertina persidengimą tarp dviejų aprėpties kontūrų. Algoritmui turi būti pateikiami teisingi aprėpties kontūrai B_{gt} (angl. *ground truth*) ir spėjami aprėpties kontūrai B_p. Pritaikant *IoU* metodą galima sužinoti, ar objekto aptikimas yra teisingas (angl. *true positive*), ar klaidingas (angl. *false positive*).

IoU yra gaunamas pagal susikertantį plotą tarp spėjamų aprėpties kontūrų ir teisingų aprėpties kontūrų, padalintų iš tarp jų esančio sąjungos ploto:

$$IoU = \frac{plotas(B_p \cap B_{gt})}{plotas(B_p \cup B_{gt}} \tag{13}$$

Žemiau pateikiamas paveikslėlis, vaizduojantis *IoU* tarp teisingo aprėpties kontūro (geltona) ir spėjamo aprėpties kontūro (raudona). Mėlynas plotas vaizduoja dviejų aprėpties kontūrų persidengimą. Kontūrų persidengimui mažėjant (žr. 16 pav. dešinėje) objekto aptikimo tikslumas proporcingai didėja.



16 pav. *IoU* principine schema

2.3.2. Vertinimo kintamieji

Įvairių matematinių tikimybių palyginimui dažnai vartojamos šios sąvokos:

- teisingai teigiami (angl. *true positives*) – teisingas spėjimas, kai $IoU \geq$ slenkstinę ribą;
- neigiamai teisingi (angl. *false positives*) – klaidingas spėjimas, kai $IoU <$ slenkstinę ribą;
- klaidingai neigiamas (angl. *false negatives*) – nerastas teisingas aprėpties kontūras;
- teisingai neigiamas (angl. *true negatives*) – netaikomas.

Priklausomai nuo vertinimo kriterijų *IoU* slenkstinė riba dažniausiai nustatoma 50 %, 75 %, 95 %.

Tikslumas – vertinimo matas, nusakantis modelio gebėjimą rasti tik susijusius objektus. Tai procentais išreiškiama vertė, kuri nusako kiek tinklas rado teisingai teigiamų spėjimų. *Tikslumas* gali būti išreiškiamas (14) formule.

$$Tikslumas = \frac{TP}{TP + FP} = \frac{TP}{Visi\ spėjimai} \tag{14}$$

Atmetimas – vertinimo matas, nusakantis modelio gebėjimą rasti visus susijusius atvejus (visus teisingus arba sužymėtus aprėpties kontūrus). Tai procentinė dalis tarp visų teisingų teigiamų

spėjimų ir visų užduoties vaizdo teisingų aprėpties kontūrų. *Atmetimas* gali būti išreiškiamas (15) formule.

$$Atmetimas = \frac{TP}{TP + FN} = \frac{TP}{Visi\ aprėpties\ kontūrai} \tag{15}$$

2.3.3. Tikslumo ir atmetimo kreivė

Tikslumo ir atmetimo kreivė – tai įvertinimo metodas, kuris yra skirtas objektų detektoriaus veikimo įvertinimui. Detektorius yra vertinamas gerai, jeigu didinant atmetimo vertę, tikslumas išlieka didelis. Tai reiškia, kad keičiant užtikrintumo slenkstį, tikslumas ir atmetimas išlieka dideli.

Kitas būdas gero objektų detektoriaus sukūrimui yra tikrinti, ar jis gali atpažinti tik susijusius objektus (0 neigiamai teisingų = didelis tikslumas), arba rasti visus sužymėtus aprėpties kontūrus (0 klaidingai neigiamų = aukštas atšaukimas).

Šių vertinimo kriterijų priklausomybė aiškiai matoma tikslumo ir atmetimo kreivėje. Dažniausiai kreivė prasideda su didele tikslumo verte, tačiau, siekiant surasti visus sužymėtus aprėpties kontūrus, yra padidinamas randamų objektų skaičius (atmetimo vertei didėjant tikslumas mažėja). Apie šios kreivės taikymą pateikiama 3.4 poskyriuje.

2.3.4. Vidutinis tikslumas

Kitas būdas palyginti detektoriaus našumą yra apskaičiuoti 2.3.3 skyrelyje pateikiamo grafiko plotą po kreive. Tačiau ši užduotis tampa sudėtinga, kai atvaizduojami grafikai (skirtingi detektoriai) pradeda tarpusavyje kirstis, tokiu atveju detektoriaus našumas vertinamas pagal vidutinį tikslumą arba *AP*. Tai yra skaitinis vertinimo kriterijus su kuriuo galima palyginti skirtingus detektorius. *AP* yra vidurkis tarp visų atmetimo verčių intervale nuo 0 iki 1. Šio vertinimo kriterijaus apskaičiavimui naudojami skirtingi interpoliavimo metodai: 11 taškų interpoliacija ir visų taškų interpoliacija.

11 taškų interpoliacija bando apibendrinti kreivės formą skaičiuojant tikslumo vidurkį 11 vienodai išdėstytų atmetimo lygių [0, 0.1, 0.2, ... , 1]. Tikslumas 11 taškų interpoliacijos metodu paskaičiuojamas pagal (16), (17) formules. Remiantis [21] šaltinyje pateikiama informacija.

$$AP = \frac{1}{11} \sum_{r \in \{0, 0.1, \ldots, 1\}} \rho_{interp(r)} \tag{16}$$

Kiekvieno atmetimo lygio r tikslumas interpoliuojamas imant maksimalų išmatuotą tikslumą metodui, kuriam atitinkama atmetimo vertė viršija r vertę.

$$\rho_{interp(r)} = \max_{\tilde{r}:\tilde{r} \geq r} \rho(\tilde{r}) \tag{17}$$

Formulėje $\rho(\tilde{r})$ yra išmatuotas tikslumas prie atmetimo lygio $\tilde{r}$. Šiuo metodu *AP* gaunamas interpoliuojant tikslumą 11-oje atmetimo lygių r ir nuskaitant maksimalų tikslumą tik tų verčių, kurių atmetimo vertė yra didesnė už r.

Tačiau nuo 2010 m. algoritmas buvo pakeistas. Nuspręsta vietoje 11-os lygiais tarpais išdėstytų taškų interpoliavimo panaudoti visus taškus pagal (18),(19) formules.

$$\sum_{r=0}^{1}(r_{n+1}-r_n)\rho_{interp}(r_{n+1}) \tag{18}$$

čia n – interpoliavimo taškų r skaičius.

$$\rho_{interp}(r_{n+1}) = \max_{\tilde{r}:\tilde{r}\geq r_{n+1}} \rho(\tilde{r}) \tag{19}$$

čia $\rho(\tilde{r})$ – išmatuotas tikslumas prie atmetimo lygio $\tilde{r}$.

Vietoje naudojamų kelių taškų tikslumo nustatymui, *AP* skaičiavimo algoritmas pakeistas, kad interpoliuotų tikslumą kiekviename r lygyje paimant maksimalų tikslumą iš tų atmetimo verčių, kurios yra didesnės arba lygios $r+1$. Tokiu būdu tiksliai apskaičiuojamas plotas po tikslumo ir atmetimo kreive.

3. Projektinė dalis

3.1. Techninės įrangos apžvalga

AlexNet, *VGG-16* ir *VGG-19* giliųjų neuroninių tinklų apmokymui buvo panaudoti du kompiuteriai, kurių specifikacijos pateikiamos 1-oje ir 2-oje lentelėse. CNN apmokymo greičiui didžiausią įtaką daro procesorius – *CPU* ir grafikos apdorojimo įrenginys – *GPU*. Nuotraukų apdorojimo greitis drastiškai skiriasi priklausomai nuo pasirenkamo įrenginio. Dažniausiai naudojamas *GPU* dėl savo lygiagrečios skaičiavimų architektūros, tačiau išimtinais atvejais tą patį darbą, tik lėčiau, gali atlikti ir procesorius. Šio tyrimo metu tinklai apmokomi su skirtingais *GPU* įrenginiais, priklausomai nuo apmokomo tinklo grafinės atminties reikalavimų.

AlexNet skaičiavimams atlikti buvo panaudotas stacionarus, „Workstation“ tipo kompiuteris, kurio specifikacijos pateikiamos 1 lentelėje.

1 lentelė. Kompiuterio nr.1 specifikacijos

Eil. Nr.	Komponentas	Specifikacija
1.	Procesorius	Intel® Core™ i7-7700 CPU @ 3.60GHz
2.	Pagrindinė plokštė	ASUS PRIME Z270-P
3.	Grafikos apdorojimo įrenginys GPU	Nvidia Quadro K1200, 4GB GDDR5
	CUDA branduolių skaičius	512 branduolių
	Atminties pralaidumas	80 GB / s
4.	Operatyvioji atmintis RAM	16.0 GB
5.	Atmintis	SSD 250 GB
6.	Operacinė sistema	Windows 10 Pro
7.	OS architektūra	64 bitų

1 lentelėje pateiktų specifikacijų kompiuteris pasižymi *Intel® Core™ i7-7700* keturių branduolių 3.60 GHz procesoriumi, turinčiu „Intel® Hyper - Threading“ technologiją. Sumontuota *Nvidia Quadro K1200* vaizdo plokštė. Tyrimo metu buvo pastebėta, kad šioje plokštėje esančių 4GB grafinės atminties nepakanka *VGG-16* ir *VGG-19* tinklų apmokymui, todėl šių tinklų apmokymas buvo perkeltas į kitą kompiuterį, kurio parametrai pateikiami 2 lentelėje. Projekto tyrimai buvo atliekami dirbant su *Windows 10 Pro* operacine sistema.

2 lentelė. Kompiuterio nr.2 specifikacijos

Eil. Nr.	Komponentas	Specifikacija
1.	Procesorius	Intel® Core™ i5-2400 CPU @ 3.10GHz
2.	Pagrindinė plokštė	Gigabyte HA65M-D2H-B3
3.	Grafikos apdorojimo įrenginys GPU	Nvidia GeForce GTX 1060, 6GB GDDR5/X
	CUDA branduolių skaičius	1280 branduolių
	Atminties pralaidumas	192 GB / s
4.	Operatyvioji atmintis RAM	4.0 GB
5.	Atmintis	HDD 1000 GB
6.	Operacinė sistema	Windows 10 Pro
7.	OS architektūra	64 bitų

VGG-16 ir *VGG-19* tinklų apmokymui buvo panaudotas vienas iš naujausių šiuo metu rinkoje esančių grafikos apdorojimo įrenginių: *Nvidia GeForce GTX 1060*. Šis *GPU* pasižymi 6GB grafine atmintimi, 1280 *CUDA* branduolių bei dideliu atminties pralaidumu. Šių specifikacijų pakaktų visų tyrime apžvelgtų giliųjų neuroninių tinklų apmokymui, tačiau dėl ilgai trunkančio tinklo mokymo, *AlexNet* skaičiavimai buvo perkelti į 1-oje lentelėje pateikiamą kompiuterį.

3.2. Duomenų rinkinių paruošimas

Tyrimui įgyvendinti buvo pasirinkti [6] šaltinyje pateikti *PUCPR* ir *CARPK* duomenų rinkiniai, kurie yra atskira *PKLot* [25] rinkinio dalis. Tai didelės apimties duomenų rinkiniai, išsiskiriantys dinaminėmis savybėmis, dažnai naudojami įvairiems automobilių atpažinimo uždaviniams spręsti [16], [27]. Paveikslėlių rinkinio nuotraukose esantys automobiliai anotuojami aprėpties kontūro principu. Anotacija pažymėta užfiksuojant automobilio viršutinį kairį ir apatinį dešinį taškus, kaip pateikta 18 pav. a dalyje. Toks duomenų rinkinio formatas leidžia spręsti objektų skaičiavimo, lokalizavimo uždavinius, atlikinėti įvairius tyrimus, kuriems yra būtinas aprėpties kontūro formato duomenų rinkinio objektų žymėjimas. Žemiau pateikiama aktualių duomenų rinkinių palyginimo lentelė.

3 lentelė. Duomenų rinkinių palyginimas

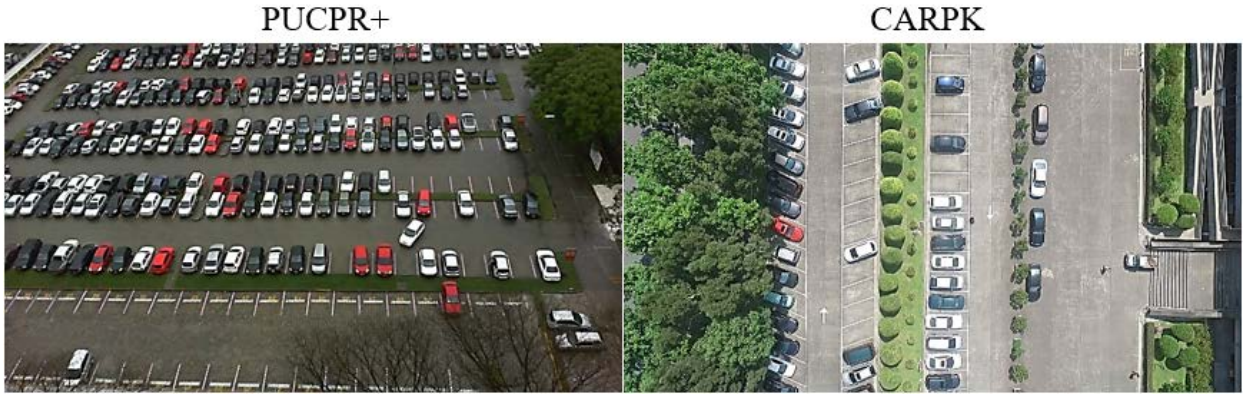
Statusas	Duomenų rinkinys	Įranga	Skirtingos scenos	Rezoliucija	Tipas	Anotacijų formatas	Mašinų kiekis
Nenaudojama	PUCPR	Kamera	×	1280x720	JPG	Aprėpties kontūras	192,216
Naudojama	PUCPR+	Kamera	×	1280x720	JPG	Aprėpties kontūras	16,456
Naudojama	CARPK	Dronas	✓	1280x720	PNG	Aprėpties kontūras	89,777
Naudojama	MIX	Kamera ir dronas	✓	1280x720	PNG	Aprėpties kontūras	106,233

PUCPR+ rinkinyje yra duomenys, surinkti iš parkavimo aikštelėje įrengtos stacionarios kameros, fiksuojančios parkavimo zoną įvairiomis aplinkos sąlygomis (saulėta, debesuota, lietinga). Rinkinyje yra 125 nuotraukos (100 mokymui / 25 tikrinimui). Rinkinio anotacijose pateikiamos 2 klasės (automobilis, fonas). Šio rinkinio pavyzdys pateikiamas 17 pav. kairėje. Tyrimui panaudotos tik ekspertų anotuotos šio rinkinio nuotraukos, kurių dalis vadinama *PUCPR+*, likęs rinkinys nenaudojamas.

CARPK duomenų rinkinį sudaro 4 skirtingų automobilių stovėjimo aikštelių nuotraukos, gautos 40 m. aukštyje skraidinant „PHANTOM 3 PROFESSIONAL“ droną. Gauti duomenys pritaikomi parkavimo vietų uždaviniams spręsti, atsižvelgiant į kintančios aplinkos faktorių. Rinkinyje yra 1448 nuotraukos (989 mokymui / 459 tikrinimui). Rinkinio anotacijose pateikiamos 2 klasės (automobilis, fonas). Šio rinkinio pavyzdinė nuotrauka pateikiama 17 pav. dešinėje. Surinkti duomenys sužymėti aprėpties kontūrų principu ir gautos 89.777 sužymėtų automobilių anotacijos.

MIX duomenų rinkinys gaunamas apjungiant *PUCPR+* ir *CARPK* duomenų rinkinių nuotraukas, anotacijas bei mokymo ir tikrinimo sąrašus. Rinkinyje yra 1873 nuotraukos (1389 mokymui / 484

tikrinimui). Rinkinio anotacijose pateikiamos 2 klasės (automobilis, fonas). Rezultate gautas apjungtas duomenų rinkinys, kuriame yra stacionarios kameros ir drono nuotraukos.



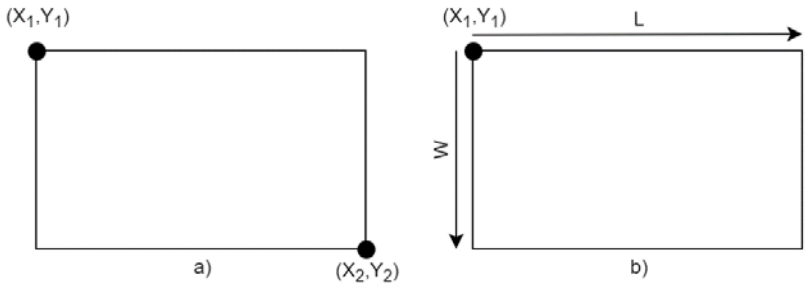
17 pav. Duomenų rinkinių pavyzdžiai

Projekto pradžioje buvo pastebėta, kad *PUCPR+* duomenų rinkinį sudaro sąlyginai mažas nuotraukų kiekis. Atsižvelgiant į duomenų kiekio skirtumą tarp rinkinių, buvo nuspręsta sukurti algoritmą, kuris padidintų tik *PUCPR+* apmokymui skirtų nuotraukų kiekį. Rezultate *PUCPR+* duomenų rinkinio apmokymui skirtų nuotraukų skaičius padidintas iki 400 nuotraukų, t. y. padidintas 4 kartus.

3.2.1. Koordinačių konvertavimas

Tyrime naudojami gilieji neuroniniai tinklai yra panašūs tuo, kad tinklo apmokymui reikalauja fiksuotos aprėpties langelių struktūros. Šio reikalavimo *PUCPR+* ir *CARPK* duomenų rinkiniai neatitiko, todėl, prieš naudojant nuotraukas tinklo apmokymui, reikėjo pakeisti anotacijų koordinačių struktūrą. Nuotraukų standartinis dydis pikseliais yra 1280x720x3, kur X = 1280 pikselių, Y = 720 pikselių, o 3 nurodo *RGB* spalvų kanalų skaičių.

Abiejuose duomenų rinkiniuose automobilio koordinatės apibrėžiamos taip: $\{X_1, Y_1, X_2, Y_2\}$, kur X_1, Y_1 – pradinės koordinatės; X_2, Y_2 – galinės koordinatės, kaip pateikta 18 pav. a dalyje. Remiantis gautais taškais apibrėžiamas keturkampis arba aprėpties kontūras, kuris nusako objekto buvimo vietą.



18 pav. Anotacijų konvertavimo principas

Norint įkelti duomenų rinkinį į vieną iš giliųjų neuroninių tinklų, anotacijų formatas turi tenkinti tokią struktūrą: $\{X_1, Y_1, L, W\}$, kuri pateikiama 18 pav. b dalyje. Ilgis ir plotis gaunamas pagal (1) ir (2) formules:

$$L = X_2 - X_1 \quad (1)$$

$$W = Y_2 - Y_1 \quad (2)$$

kur, X_1, Y_1 – pradinės koordinatės; L – ilgis (pikseliais); W – plotis (pikseliais).

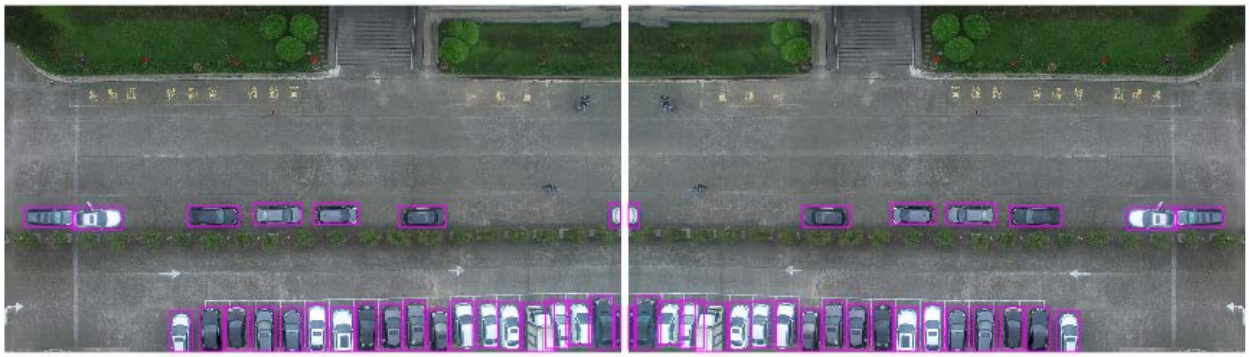
3.2.2. Duomenų rinkinio padidinimas

Giliųjų neuroninių tinklų architektūrose duomenų rinkinio vartymas padeda ne tik padidinti apmokymui skirtą duomenų rinkinį, bet ir sumažina sukurtos sistemos jautrumą pašaliniams aplinkos veiksniams. Mechaniškai keičiant tokias nuotraukų technines savybes kaip: šviesumas, kontrastas bei nuotraukų susiliejimas ir aštrumas galima papildomai didinti turimą duomenų rinkinį. Šio tyrimo metu buvo nuspręsta panaudoti veidrodinį duomenų kiekio didinimo metodą.

19 pav. Duomenų rinkinio didinimas, vartymo metodu

Duomenų rinkinio didinimui buvo parašytas algoritmas, sukuriantis esamų nuotraukų veidrodinius variantus per x, y bei abi kartu koordinačių ašis. (žr. 19 pav.). Algoritmas automatiškai pratęsia mokymui naudojamų nuotraukų sąrašą bei priskiria naujai sukurtoms nuotraukoms pavadinimus ir konvertuoja senas anotacijas pagal analogišką koordinačių ašį. Tokiu būdu tinklo apmokymui skirtas duomenų rinkinys praplečiamas 4 kartus, tuo pačiu perskaičiuojant anotacijose aprašomų aprėpties kontūrų koordinačių taškus.

20 pav. a dalyje matoma viena iš rinkinio nuotraukų kartu su anotacijose pateiktais automobilių aprėpties kontūrais, o dešinėje pateikiamas minėto algoritmo rezultatas konvertuojant per x ašį. Ši nuotrauka yra vienas iš pavyzdžių kaip buvo tikrinama, kad nuotraukų vartymo algoritmas, didindamas duomenų rinkinį, nesugadina sužymėtų objektų anotacijų ir pačių nuotraukų kokybės.



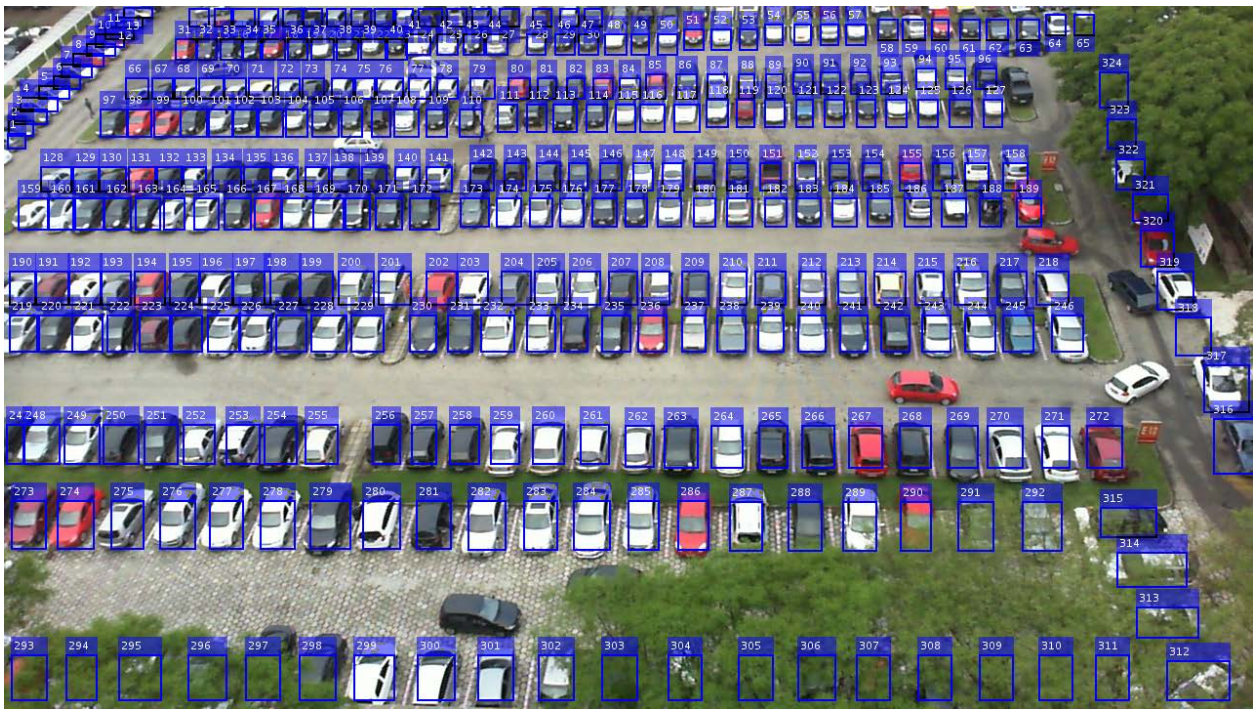
20 pav. Apversta ir sužymėta nuotrauka

Remiantis [6] šaltinio informacija *PUCPR+* duomenų rinkinyje iš viso pateiktos 125 nuotraukos, kuriuose aprėpties langelių principu sužymėtos 16.456 mašinos. Tyrimo metu *PUCPR+* duomenų rinkinio apmokymo nuotraukų kiekis buvo padidintas nuo 100 iki 400 nuotraukų, t. y. apie 52.600 mašinų. Atsižvelgiant, kad net nepanaudojus duomenų didinimo algoritmo *CARPK* apmokymo duomenų rinkinyje yra 61.318 mašinų, buvo nuspręsta toliau dirbti nekoreguojant *CARPK* duomenų rinkinio.

3.2.3. Anotacijų kaukės paruošimas

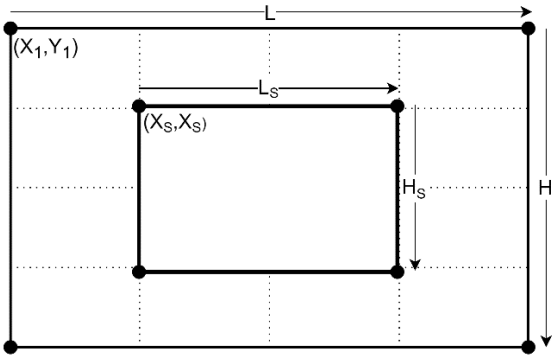
Automobilių žymėjimo algoritmo vienas iš pagrindinių komponentų yra anotacijų kaukė / šablonas. Ji yra naudojama norint palyginti realią situaciją su iš anksto paruoštu šablonu. Šiuo atveju buvo užsibrėžta palyginti parkavimo zonomis suskirstytos aikštelės užimtumą su kameros pateikiama realios situacijos informacija. Anotacijų kaukė - tai stacionarios kameros vaizdo šablonas, kuriame sužymėtos visos galimos parkavimo vietos. Kiekvienai parkavimo vietai priskiriamos paveikslėlio pikseliais išreiškiamos koordinatės (žr. 21 pav.). Šios koordinatės gaunamos pritaikant aprėpties kontūrų metodą.

Anotacijų kaukė buvo sudaroma remiantis toliau pateikiamais punktais. Pradiniam kaukės vaizdui panaudojama parkavimo aikštelės nuotrauka, kurioje yra daugiausiai automobilių. Gautos anotacijos rankiniu būdu sutvarkomos nuo nepageidaujamų persidengimų. Anotacijos sulygiuojamos ir sutvarkoma kaukės anotacijų sąrašo eilės tvarka (iš kairės į dešinę ir iš viršaus į apačią). Tokiu būdu, kiekvienai parkavimo vietai gali būti priskiriamas unikalus parkavimo vietos numeris. Kaukei priklausančios anotacijos nuspalvinamos laisvą parkavimo vietą žyminčia spalva. Sutvarkytos anotacijų kaukės pavyzdys pateikiamas 21 pav.



21 pav. Paruošta anotacijų kaukė

Norint sukurti stabilesnį automobilių skaičiavimo algoritmą, buvo būtina sumažinti anotacijų persidengimo plotą, todėl pabaigoje buvo atliekamas anotacijų mastelio sumažinimas. Mastelis sumažinamas per x ir y ašis kaip pateikiama 22 pav.



22 pav. Anotacijos mastelio pakeitimas

Automobilių anotacijų mastelis pakeičiamas remiantis 20 formulėmis. Atsižvelgiant į nuotraukoje esančių anotacijų skirtumus, įvedamas naujas kintamasis *scale*, kurio pagalba eksperimentiškai gaunama optimali anotacijų mastelio vertė.

$$X_1 = X + ((L/2)/2)/scale \tag{20}$$

$$Y_1 = Y + ((H/2)/2)/scale$$

$$L_s = (L/2) * scale$$

$$H_s = (H/2) * scale$$

čia X_1, Y_1 – anotacijos pradinio taško koordinatės; L, H – anotacijos ilgis ir plotis (pikseliais); X_S, Y_S – pakeistos anotacijos pradinio taško koordinatės; L_S, H_S – pakeistos anotacijos ilgis ir plotis (pikseliais).

Rezultate gaunama anotacijų kaukė, kuriuos vaizdas toliau taikomas 3.6 poskyriuje, sukuriant parkavimo aikštelės informavimo algoritmą.

3.3. Nustatymų parinkimas

Prieš atliekant tinklo apmokymą yra būtina parinkti apmokymo nustatymus. *Faster R-CNN* tinklas išsiskiria iš kitų tuo, kad apmokymo nustatymai parenkami keturiems tinklo mokymo etapams. Tinklų apmokymai buvo atliekami pagal 4 lentelėje pateiktus nustatymus, atsižvelgiant į panašaus tyrimo duomenis [28]. Pastaba: keičiantis mokymo etapams skiriasi tik paketo dydis.

4 lentelė. Tinklų apmokymo nustatymai

Nustatymai	**Nustatymų etapas**			
	1	**2**	**3**	**4**
Algoritmo tipas	sgdm	sgdm	sgdm	sgdm
Pradinis mokymosi greitis	0.001	0.001	0.001	0.001
Mokymosi greičio mažėjimo žingsnis	0.9	0.9	0.9	0.9
Mokymosi greičio mažinimo dažnis	1	1	1	1
Svorių mažinimo faktorius (L2Regularization)	0.0005	0.0005	0.0005	0.0005
Pagreitis (angl. *momentum*)	0.9	0.9	0.9	0.9
Maksimalus epochų skaičius	kintantis	kintantis	kintantis	kintantis
Paketų dydis	**512**	**256**	**512**	**256**
Duomenų rinkinio sumaišymas	Kiekvieną epochą	Kiekvieną epochą	Kiekvieną epochą	Kiekvieną epochą
Vykdymo aplinka	GPU	GPU	GPU	GPU

Neuroniniam tinklui optimizuoti panaudotas „sgdm“ – stohastinis gradiento mažėjimo su pagreičiu metodas (angl. *stochastic gradient descent with momentum*). Darbo metu pastebėta, kad kiti populiarūs optimizatoriai („adam“ ir „rmsprop“) nepalaiko *Faster R-CNN* tinklo. Todėl nutarta pasilikti prie „sgdm“ optimizatoriaus. Šio metodo pagrindinis skirtumas – šuolis nuo nusistovėjusių rezultatų. Kiekvieno svorių atnaujinimo metu, „sgdm“ neleidžia apsistoti ties klaidingais rezultatais.

Pradinis mokymosi greitis (angl. *initial learning rate*) yra gaunamas tinklų veikimą eksperimentiškai išbandant 0,01-0,0001 ribose. Stabiliausias tinklų veikimas yra pasiekiamas mokymosi greitį parenkant 0,001. Mokymosi greitis kinta pagal užduotą mokymosi greičio kitimo žingsnį. Po kiekvieno mokymosi žingsnio pradinis mokymosi greitis yra padauginamas iš 0,9.

Tinklų nustatymuose pritaikomas svorių mažinimo faktorius „L2Regularization“. Tai (angl. *ridge regression*) regresinis modelis, kuris svorius padaugina iš 0,0005 koeficiento kvadratinio dydžio. Šio metodo pagalba tinklo svoriai yra apsaugomi nuo staigaus persimokymo.

Paketo dydis (angl. *mini-batch size*) nusako, kiek nuotraukų paduoti tinklo mokymosi ciklo metu. Atsižvelgiant į teoriją, didinant paketo dydį, didėja tikslumas, bet tuo pačiu ir skaičiavimo laikas, o mažinant paketo dydį abu proporcingai mažėja. Todėl buvo nuspręsta paketo dydį keitinėti kiekvieno mokymosi etapo metu tarp 256 ir 512. Kiekvienos epochos pradžioje visas duomenų rinkinys yra iš naujo sumaišomas.

Maksimalus epochų skaičius keičiamas priklausomai nuo atliekamo tyrimo. Geriausiai veikiančio tinklo tyrimui nustatomas 30 epochų mokymo limitas. Šis tyrimas atliekamas su didesniu kiekiu epochų norint įsitikinti, kad mokymo eigoje tinklas nepersimokys ir sužinoti, kaip keisis tinklo rezultatai priklausomai nuo mokymo epochų skaičiaus. Visi kiti tyrimai atliekami su 15 epochų.

Pagal literatūroje pateikiamus šios srities tyrimus parenkamas 0,9 pagreitis, kuris, priklausomai nuo koeficiento, pakeičia mokymosi dažnį. Apmokymo skaičiavimams pagreitinti yra pasirenkamas grafinių skaičiavimų įrenginys (GPU).

3.4. Tyrimo eiga

Tyrimas buvo atliekamas pasinaudojant standartiniu *MATLAB R2018a* programiniu paketu bei įdiegtais šio paketo papildiniais, leidžiančiais dirbti su tyrimui aktualiais neuroniniais tinklais. Įdiegti *R-CNN* architektūrų paketai (*AlexNet, VGG-16* ir *VGG-19*) bei giliųjų neuroninių tinklų įrankių rinkinys.

Tyrimo tikslas buvo palyginti populiariausių *R-CNN* tinklų (žr. 2.1 poskyrių) kokybę (pagal 2.3 poskyriuje pateikiamus vertinimo kriterijus), sprendžiant automobilių atpažinimo uždavinius. Tyrimo skaičiavimai buvo padalinti dvejiems kompiuteriams, apie kuriuos plačiau pateikiama poskyriuje 3.1 Techninė apžvalga. Tinklų mokymui buvo panaudojami automobilių parkavimo aikštelių duomenų rinkiniai (plačiau 3.2 poskyriuje).

Prieš paleidžiant tinklo apmokymo algoritmą, detektoriui pateikiama tokia informacija:

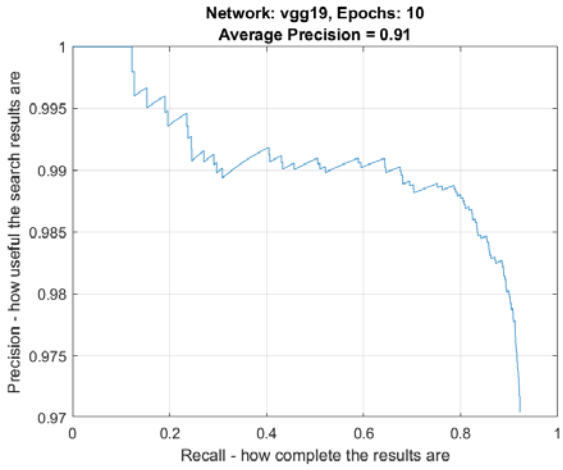
- apmokymui skirta duomenų rinkinio dalis (angl. *training data*);
- žinių perkėlimo metodą palaikantis tinklas (*AlexNet, VGG-16* ir *VGG-19*);
- tinklo mokymosi nustatymai.

Duomenų rinkinys, prieš perkeliant į detektorių, turi būti padalintas į apmokymo ir tikrinimo rinkinius. Abu rinkiniai turi iš anksto atrinktas mokymui ir testavimui skirtas duomenų dalis. Skirtinguose duomenų rinkiniuose nuotraukų formatai išsiskiria, todėl, kad būtų paprasčiau ir tyrimo eigoje neatsirastų papildomų problemų ir neatitikimų, visi duomenys buvo paversti į *PNG* nuotraukų formatą. *R-CNN* tinklai reikalauja, kad anotacijos prieš paduodant į tinklą būtų pagal reikalavimus apdorojamos. Todėl abiejų rinkinių anotacijų koordinatės buvo pakeičiamos kaip pateikiama skyrelyje 3.2.1 Koordinačių konvertavimas.

Tiriamas tinklas yra įkeliamas į *Faster R-CNN* klasifikatorių pritaikant žinių perkėlimo metodą (žr. 2.2 poskyrių). Tinklų apmokymui panaudoti *PUCPR+* ir *CARPK* duomenų rinkiniai, o paruošimo eiga pateikiama 3.2 poskyriuje.

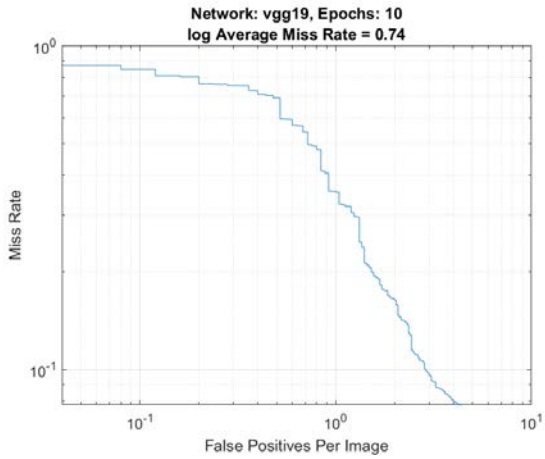
Prieš paleidžiant detektorių parenkami mokymo nustatymai apie kuriuos plačiau pateikiama 3.3 poskyriuje - Nustatymų parinkimas. Priklausomai nuo atliekamo tyrimo pakeičiamas epochų skaičius, duomenų rinkinio numeris, tyrimo numeris.

Apmokyto detektoriaus patikrinimas. Pasibaigus apmokymo etapui, prasideda tikrinimo etapas, kurio rezultatus įvertina „evaluateDetectionPrecision“ ir „evaluateDetectionMissRate“ *MATLAB* programinio paketo funkcijos. Pasinaudojant „evaluateDetectionPrecision“ funkcija galima gauti tikslumo ir atmetimo vertes. 23 pav. pateikiamas vienas iš tyrimo metu gautų *AP* grafikų pavyzdžių, kuriame *VGG-19* tinklas mokomas 10 epochų su *PUCPR+* duomenų rinkiniu. Gautas 91 % vidutinis tikslumas.



23 pav. Vidutinio tikslumo grafiko pavyzdys

Panaudojus „evaluateDetectionMissRate“ funkciją gaunamas neteisingai teigiamų nuotraukų skaičius *FPPI* (angl. *false positives per image*) bei nepataikymo dažnio vertė (angl. *miss Rate*). *LAMR* (angl. Log Average Miss Rate) kreivė dažniausiai naudojama norint grafiškai palyginti skirtingų tinklų našumą. Žemesnė kreivė rodo geresnį našumą. 24 pav. pateikiamas vienas iš tyrimo metu gautų *LAMR* grafikų pavyzdžių, kuriame *VGG-19* tinklas to pačio mokymo metu, su vienodais nustatymais, gauna 0,74 *LAMR* vertę.



24 pav. *LAMR* grafiko pavyzdys

Pagal šiame skyriuje aprašomą tyrimo eigą gaunami 3.5 poskyriuje pateikiamų tyrimų rezultatai.

3.5. Tyrimo rezultatai

Tyrimo sėkmingam atlikimui buvo būtina remtis 3.4 poskyriuje pateikiama tyrimo eiga. Pirmiausia buvo atliekamas tinkamiausio tinklo paieškos tyrimas, kurio metu buvo ieškoma geriausiai apsimokančio tinklo pagal skirtingus duomenų rinkinius. Šio tyrimo rezultatai pateikiami 3.5.1 skyrelyje.

Toliau buvo atliekamas tinkamos spalvų erdvės paieškos tyrimas, kuriame *PUCPR+* duomenų rinkinys buvo konvertuojamas iš *RGB* į *HSV* / *LAB* spalvų erdves. Tinklai apmokomi su skirtingų spalvų erdvių duomenų rinkiniais ir lyginami rezultatai. Atsižvelgiant į gautus rezultatus buvo nuspręsta tyrimo nepratęsti su *CARPK* duomenų rinkiniu. Tyrimo platesnė eiga ir rezultatai pateikiami 3.5.2 skyrelyje.

Norint sužinoti apmokytų tinklų patikimumą, buvo atliekamas tinklų pritaikymo tyrimas. 3.5.1 skyrelyje aprašyto tyrimo metu gauti detektoriai buvo patikrinami su priešingais duomenų rinkiniais. Plačiau apie šį tyrimą 3.5.3 skyrelyje.

Tinklų greitaveikai įvertinti buvo atliekamas tikrinimo trukmės įvertinimas. Duomenys gauti pagal 3.5.1 skyrelyje apmokytų tinklų vidutinį tikrinimo laiką. Tyrimo rezultatai ir eiga pateikiama 3.5.4 skyrelyje.

Norint išbandyti ir pritaikyti 3.5.1 skyrelyje geriausius rezultatus pasiekusį tinklą, buvo sukurtas parkavimo vietų užimtumo stebėjimo algoritmas. Apie šį algoritmą plačiau pateikiama 3.6 poskyriuje.

3.5.1. Tinkamiausio tinklo paieška

Vienas iš svarbiausių vertinimo kriterijų yra tikslumas. Tai kriterijus, parodantis kaip kokybiškai lokalizavimo algoritmas rado ieškomą objektą lyginant su objektui priskirta anotacija. Šio tyrimo metu vertinama pagal tinklo geriausią pasiektą vidutinį tikslumą. Standartiškai į skaičiavimą įtraukiami tik tie objektai, kurių persidengimo koeficientas didesnis nei 50 %.

Tyrimo tikslas buvo palyginti *AlexNet, VGG-16* ir *VGG-19* giliuosius neuroninius tinklus. Šiems tinklams apmokyti panaudojami įvairūs automobiliams atpažinti skirti duomenų rinkiniai.

Mokymo metu kiekvienos epochos detektorius yra išsaugojamas kaip atskiras kontrolinis taškas (angl. *checkpoint*). Šie kontroliniai taškai suteikia galimybę palyginti epochų rezultatus, pratęsti mokymą ir paleisti tinklą su specifiniu epochų skaičiumi. Tinklai vertinami pagal mažiausią epochų skaičių, kuris yra reikalingas pasiekti geriausią įmanomą tikslumo procentinę dalį. Plačiau apie tinklų vertinimo kriterijus pateikiama 2.3 poskyriuje. Tinklai apmokomi su nekintančiais *Faster R-CNN* mokymosi nustatymais bei 30 epochų. Tinkamiausio tinklo paieškos rezultatai pateikiami 5 lentelėje.

5 lentelė. Tinkamiausio tinklo paieškos rezultatai

Rinkinys	PUCPR+		CARPK		MIX	
Tinklas	**Min. epochų skaičius**	**Vidutinis tikslumas, %**	**Min. epochų skaičius**	**Vidutinis tikslumas, %**	**Min. epochų skaičius**	**Vidutinis tikslumas, %**
AlexNet	25	79,5	4	58,2	2	59,8
VGG-16	12	91,2	20	**88,9**	13	**89,5**
VGG-19	12	**92,8**	10	87,5	9	88,3

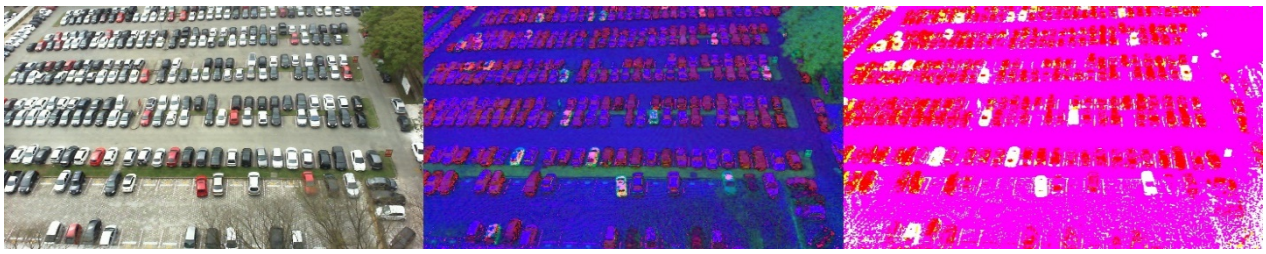
Pagal 5 lentelėje pateiktą informaciją geriausią spėjimo vidutinį tikslumą (92,8 %) pasiekė *VGG-19* tinklas, kuriam apmokyti panaudotas *PUCPR+* duomenų rinkinys. *CARPK* ir *MIX* duomenų rinkinių kategorijose geriausius rezultatus gavo *VGG-16* tinklas. *MIX* duomenų rinkiniu apmokytas tinklas pagal vidutinio tikslumo kriterijų pasirodė geriau už *CARPK*, tačiau atsiliko nuo *PUCPR+*. Tinklams apmokyti su *MIX* duomenų rinkiniu reikia pastebimai mažiau epochų, norint pasiekti panašius ar tik truputį blogesnių rezultatus.

Šio tyrimo gauti duomenys gali būti palyginami su [15] šaltinio rezultatais, kuriame autoriaus sukurtas tinklas pasiekė net 97 % vidutinį tikslumą. Šio tyrimo metu nepavyko pasiekti tokio tikslumo rezultatų.

3.5.2. Tinkamos spalvų erdvės paieška

Remiantis 1.2 poskyriuje pateikiama spalvų erdvių apžvalga bei literatūroje dažnai randamu teiginiu, kad spalvų erdvės daro įtaką apmokomo tinklo kokybei, buvo nuspręsta atlikti neuroninių tinklų priklausomybės nuo spalvų erdvių tyrimą. Prieš atliekant spalvų erdvės tyrimą su abejais duomenų rinkiniais, buvo nuspręsta pirmiausia ištirti *PUCPR+* rinkinį ir tik tada, priklausomai nuo gautų rezultatų, tęsti arba atšaukti spalvų erdvės tyrimą su *CARPK* rinkinio duomenimis.

Tyrimo pradžioje sukurtas algoritmas, kuris, panaudodamas 3.2 poskyriuje paruoštą *PUCPR+* duomenų rinkinį, sukūrė papildomas duomenų kopijas, kurias konvertavo iš *RGB* spalvų erdvės į *HSV* ir analogiškai iš *RGB* į *LAB*. Mokymui ir testavimui skirtų nuotraukų sąrašai bei sužymėtų objektų anotacijos buvo nekeičiamos. Gautų duomenų rinkinių pavyzdžiai pateikiami 25 pav. Čia iš kairės į dešinę vaizduojamos (*RGB*, *HSV*, *LAB*) spalvų erdvės. Taip pat, dėl nereikšmingų *LAB* ir *LUV* spalvų erdvių skirtumų, buvo nuspręsta netirti labai panašios *LUV* spalvų erdvės.



25 pav. Tiriamųjų spalvų erdvių pavyzdžiai

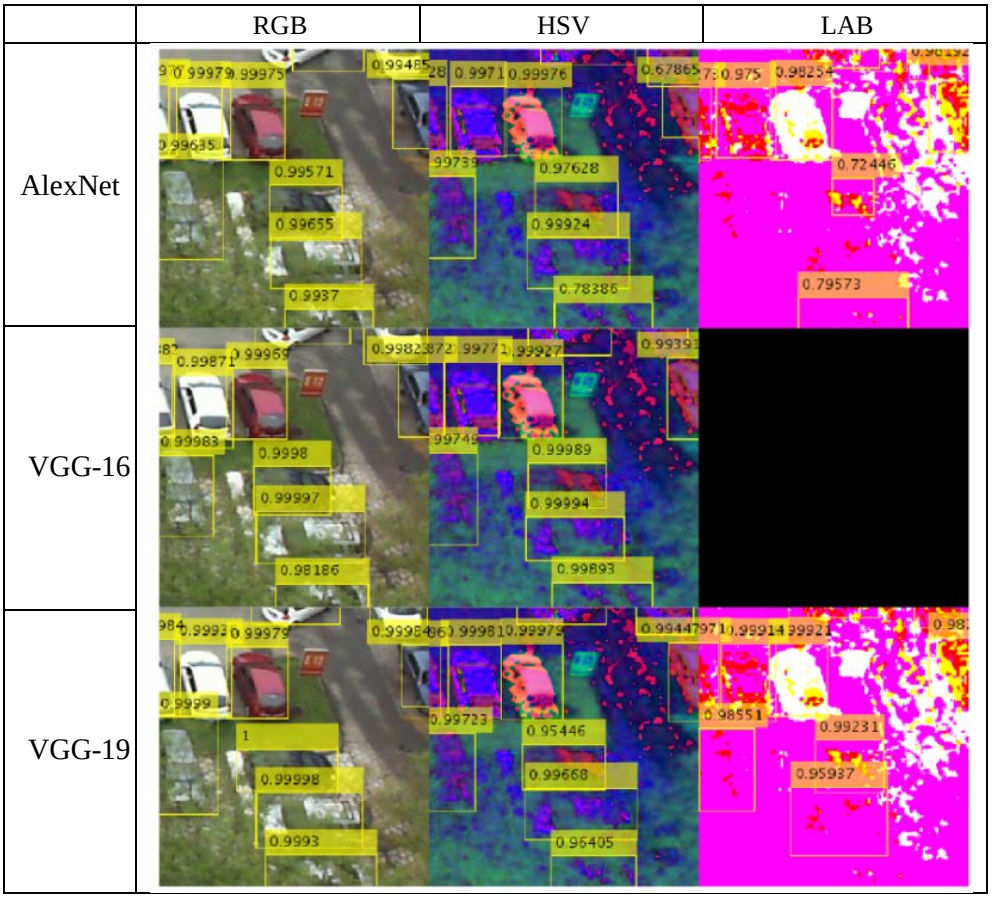
Šio tyrimo metu *AlexNet*, *VGG-16* ir *VGG-19* tinklai buvo mokomi 15 epochų. Neuroninio tinklo mokymo nustatymai viso tyrimo metu nekeičiami. Tinklų apmokymo kokybė vertinama pagal

vidutinio tikslumo kriterijų aprašomą, 2.3.4 skyrelyje. Tinklus apmokius su konvertuotos spalvų erdvės duomenų rinkiniais, gaunami 6 lentelėje pateikti rezultatai.

6 lentelė. Spalvų erdvės tyrimo rezultatai

Rinkinys: PUCPR+, 15 epochų			
Tinklas	**Spalvų erdvės vidutinis tikslumas, %**		
	RGB	**HSV**	**LAB**
AlexNet	**79,0**	68,7	57,5
VGG-16	**91,2**	90,9	-
VGG-19	**92,8**	88,4	82,8

Remiantis tyrimo rezultatais, geriausią vidutinį tikslumą pasiekė *VGG-19* tinklas, kai apmokymui panaudojamas *RGB* spalvų erdvės duomenų rinkinys. *HSV* bei *LAB* spalvų erdvių kategorijose matomas akivaizdus tikslumo sumažėjimas. Spalvų erdvės pakeitimas didžiausią neigiamą įtaką turėjo *AlexNet* tinklui. Pastarasis tinklas apmokytas su *LAB* spalvų erdvės duomenų rinkiniu pasirodė daugiau nei 20 % blogiau. *VGG-16* tinklui nepavyko apsimokyti su konvertuotu *LAB* spalvų erdvės rinkiniu.



26 pav. Spalvų tyrimo rezultatų palyginimas

Spalvų erdvės tyrimo metu gautų rezultatų vizualizacija pateikiama 26 pav. Čia pateikiamas kiekvieno tinklo tikrinimo etapo pavyzdys su detektoriaus spėjamų objektų aprėpties kontūrais bei

koeficientais. 26 pav. pateikta pasikartojanti tikrinimo sąrašo nuotrauka aiškesniam palyginimui. Galima pastebėti, kad *HSV* ir *LAB* spalvų erdvėse visų tinklų spėjimo koeficientai sumažėjo, o *LAB* dalis automobilių buvo visiškai neatpažinti. Reikia paminėti, kad, lyginant spalvų tyrimo rezultatus, juodas langas nurodo, kad *VGG-16* tinklui nepavyko užbaigti mokymosi proceso su *LAB* spalvų erdvės duomenų rinkiniu.

Remiantis 6 lentelėje bei 26 pav. pateikiamais rezultatų pavyzdžiais, galima daryti išvadą, kad automobilių atpažinimo uždaviniams spręsti skirti gilieji neuroniniai tinklai geriausiai apsimoko su *RGB* formato spalvų erdvės duomenų rinkiniais. Kadangi *PUCPR+* duomenų rinkinyje buvo gauti neigiami rezultatai, t. y. nepatvirtino hipotezės, todėl buvo nuspręsta spalvų erdvių tyrimo nepratęsti su *CARPK* duomenų rinkiniu.

3.5.3. Tinklų pritaikymo tyrimas

Tinklų pritaikymo tyrimas buvo atliekamas norint sužinoti, kaip vienu duomenų rinkiniu apmokytas tinklas prisitaiko prie kito rinkinio tikrinimo duomenų. Atsižvelgiant į 3.5.1 skyrelyje pateikiamus rezultatus, *MATLAB* pakete įkeliami geriausius rezultatus gavusių tinklų kontroliniai taškai. Kontrolinis taškas gali būti naudojamas kaip nepriklausomas tinklas, kuriam apmokyti panaudotas specifinis skaičius epochų. Toliau kiekvienam duomenų rinkiniui priklausantys tinklai išbandomi su likusių rinkinių tikrinimo dalimi. Gauti vidutinio tikslumo rezultatai pateikiami 7 lentelėje.

7 lentelė. Tinklų pritaikymo tyrimo rezultatai

Apmokytas rinkiniu		**PUCPR+**		**CARPK**		**MIX**	
Patikrintas rinkiniu		**MIX, %**	**CARPK, %**	**MIX, %**	**PUCPR+, %**	**PUCPR+, %**	**CARPK, %**
Tinklas	**AlexNet**	19	15	54	6	76	59
	VGG-16	77	76	84	16	**91**	**89**
	VGG-19	80	79	82	19	89	86

Remiantis 7 lentelėje pateikiamais rezultatais galima daryti išvadas apie tinklų pritaikymą. *MIX* duomenų rinkiniu apmokyti tinklai pasirodė geriausiai lyginant su *PUCPR+* ir *CARPK* apmokytų tinklų vidutinio tikslumo rezultatais. Patikimo detektoriaus apmokymui geriausia naudoti bendrą duomenų rinkinį, kuris vidutiniškai surinko 81,7 %, kai *PUCPR+* surinko 57,7 %, o *CARPK* 43,5 %. Geriausiai pasirodė 91–89 % vidutinį tikslumą pasiekęs *VGG-16* tinklas, kuriam apmokyti panaudotas *MIX* duomenų rinkinys. Blogiausius rezultatus gavo *PUCPR+* rinkiniu apmokytas *AlexNet* tinklas su 19–15 % vidutiniu tikslumu. Kadangi *MIX* duomenų rinkinys susideda iš įvairesnių duomenų, gaunamas geresnis tikslumas išbandant kituose duomenų rinkiniuose.

3.5.4. Tinklų tikrinimo trukmės įvertinimas

Objektų atpažinimo algoritmui labai svarbus kriterijus yra laikas. Todėl šio tyrimo metu, panaudojus *MATLAB* funkciją „Tic-Toc“, buvo skaičiuojamas tinklo tikrinimo duomenų sąrašo apdorojimo laikas. Sąlygos: tikrinimui skirti duomenys negali būti naudojami tinklo mokymosi etapo metu ir viso tyrimo metu tikrinimo duomenų rinkinys nekinta. Analogiškas procesas

atliekamas su visais duomenų rinkiniais. Tyrimo rezultatai pateikiami 8 lentelėje, kurioje fiksuojamas kiekvieno apmokyto tinklo tikrinimo laikas.

8 lentelė. Tinklų testavimo greičio palyginimas

Duomenų rinkinys	**PUCPR+**	**CARPK**	**MIX**
Tinklas	**Vidutinis laikas, s**		
AlexNet	**0,212**	**0,257**	**0,227**
VGG-16	0,446	0,607	0,444
VGG-19	0,478	0,580	0,523

Greičiausiai su tikrinimo duomenų rinkiniu susitvarkė *AlexNet* tinklas, apmokytas su *PUCPR+* duomenų rinkiniu. Nepriklausomai nuo duomenų rinkinio *AlexNet* tinklas vienai nuotraukai apdoroti užtruko ~0,2 sekundės. *VGG* architektūros tinklai vienos nuotraukos apdorojimui užtruko dvigubai ilgiau. Blogiausiai pasirodė *VGG-16* tinklas, apmokytas su *CARPK* duomenų rinkiniu.

3.6. Geriausio tinklo taikymas

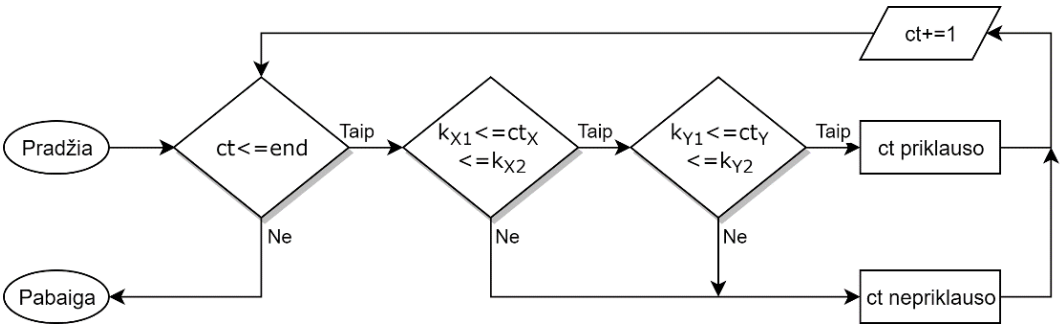
Tyrimo rezultatų taikymui buvo nuspręsta sukurti demonstracinę programą, kuri stebėtų ir informuotų vartotoją apie parkavimo aikštelėje esančią situaciją, pritaikant šio tyrimo metu geriausiu vidutiniu tikslumu apmokytą tinklą. Sukurtas algoritmas, kuris leistų informaciniuose šaltiniuose pateikti parkavimo aikštelėje esančių laisvų parkavimosi vietų žemėlapį bei užimtumą ir statusą. Algoritmas suteiktų galimybę realiu laiku informuoti vartotoją apie aikštelės parkavimo vietų būseną. Šaltinyje [32] pateikiamas kitoks šio uždavinio sprendimo metodas.

Pasirinkta naudoti stacionarios kameros pagalba surinktą *PUCPR+* duomenų rinkinį. Plačiau apie šį rinkinį pateikiama 3.2 poskyriuje. Apmokant *PUCPR+* rinkiniu geriausius rezultatus pasiekė *VGG-19* gilusis neuroninis tinklas, pasiekęs 92,8 % vidutinį tikslumą. Todėl šiai užduočiai įgyvendinti nuspręsta panaudoti šį detektorių.

Pradžioje įsikeliamas detektorius ir, jį paleidus ant labiausiai užimtos nuotraukos, gaunamas netvarkingas parkavimo vietų žemėlapis. Pagal 3.2.3 skyrelį „Anotacijų kaukės paruošimas“ gautas netvarkingas žemėlapis yra sutvarkomas ir paruošiamas tolimesniam darbui. *Pastaba: toliau šis žemėlapis bus vadinamas parkavimo aikštelės kauke.* Paruoštos kaukės anotacijos algoritmo eigoje yra lyginamos su apmokyto detektoriaus siūlomų objektų koordinatėmis. Norint rasti geriausiai tinkantį metodą, anotacijos yra lyginamos trimis skirtingais metodais. Toliau plačiau pateikiami šio darbo metu išbandyti palyginimo metodai.

Taško priklausymo anotacijos kontūrui metodas

Šiame metode yra lyginami apmokyto detektoriaus rastų objektų centriniai taškai su aikštelės kaukės anotacijų koordinatėmis. Žemiau pateikiama šiuo metodu pagrįsto algoritmo principinė schema.



27 pav. Taško ir kontūro algoritmo schema

Šioje schemoje *ct(x, y)* yra detektoriaus pateiktų anotacijų centrinių taškų masyvas, *kx* anotacijų kaukės x ašies pradžios ir pabaigos vektoriaus taškai. *ky* - anotacijų kaukės y ašies, pradžios ir pabaigos vektoriaus taškai.

Esminiai šio metodo požymiai:

+ atpažįsta per kelias parkavimosi vietas stovinčius automobilius ir juos pažymi;
– žinomos tik užimtos parkavimosi vietos, bet ne realus automobilių skaičius.

Didžiausio persidengimo (IoU) metodas

Šis metodas kiekvienam apmokyto tinklo objektui priskiria po aikštelės kaukės anotaciją. Kaukės anotacija, kurios persidengimas yra didžiausias, yra pažymima kaip užimta. Plačiau apie persidengimo metodą pateikiama 2.3.1 skyrelyje „Susikirtimas per sąjungą (IoU)“.

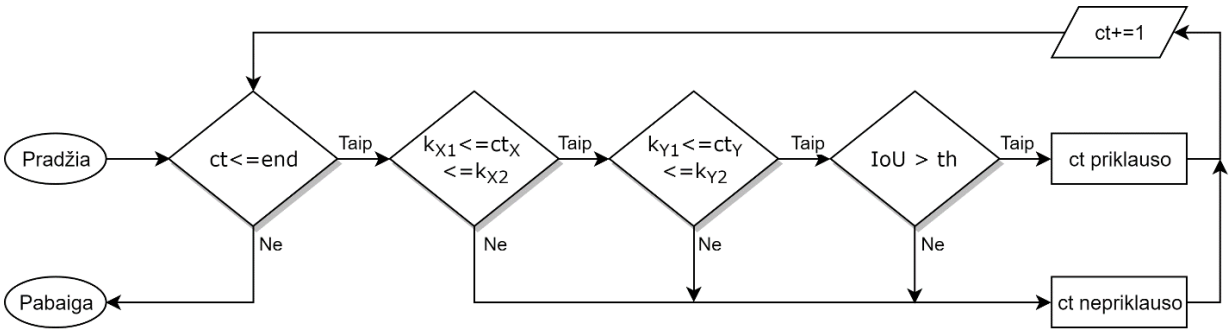
Esminiai šio metodo požymiai:

+ kiekvienas tinklo atpažintas objektas užima po vieną parkavimosi vietą;
+ besiparkuojantis automobilis greičiau persidengia su kauke, todėl ankščiau pakeičiamas parkavimosi vietos statusas;
– per kelias parkavimosi vietas prisiparkavę automobilių vairuotojai užima vieną vietą.

Taško palyginimo ir didžiausio IoU metodas

Metodas apjungia abu ankščiau minėtus palyginimo būdus. Pradžioje algoritmas palygina apmokyto detektoriaus rastų objektų centrinius taškus su aikštelės kaukės anotacijų koordinatėmis. Tokiu būdu paryškinami visi kaukės kontūrai, kurie kerta objekto centro tašką. Toliau pritaikomas persidengimo metodas, kuris padeda išspręsti netvarkingai pastatomų automobilių problemą.

Algoritmas priskiria užimtas parkavimosi vietas toms pozicijoms, kurių persidengimas su aikštelės kauke yra didžiausias. Taip kiekvienam detektoriaus objektui yra griežtai priskiriama tik po vieną užimtą vietą. Žemiau pateikiama šio metodo principinė schema.



28 pav. Atrinkimo algoritmas su *IoU*

Schemoje *IoU > th* sąlygos blokas nurodo, kad *ct* objekto kontūras yra palyginamas su visais kaukėje esančiais kontūrais ir *Taip* sąlyga yra įvykdoma tik esant didžiausiam *IoU*.

Esminiai šio metodo požymiai:

+ kiekvienas tinklo atpažintas objektas užima po vieną parkavimosi vietą;
+ algoritmas įvertina netvarkingai prisiparkavusių automobilių užimamą plotą;
+ algoritmas leidžia atvaizduoti abiejų metodų gautus rezultatus atskirai;
– lyginant su kitais metodais, šis yra lėtesnis.

Sukurtas parkavimo aikštelės informacinis langas pateikiamas 29 pav.

29 pav. Parkavimo aikštelės užimtumas, (18 nuotrauka)

Sukurta programa suteikia informaciją apie parkavimo aikštelės statusą, laisvų ir užimtų parkavimosi vietų skaičių ir priskiria parkavimo vietoms numeraciją. Papildomai suteikiama informacija apie tinklo rastų objektų ir žymėjimo algoritmo tikslumą ir nuotraukos skaičiavimo

laiką. Rezultate gaunamas parkavimo vietas žymintis algoritmas, kuris užimtas aikštelės kaukės anotacijas nuspalvina raudonu fonu, o parkavimo vietų, kurios yra užimtos, arba tikėtina, kad užimtos, kontūras pakeičiamas iš žalios į baltą.

Rezultatai ir išvados

Atliekant šį darbą buvo siekiama taikyti giliuosius neuroninius tinklus parkavimo aikštelėse esančių automobilių atpažinimo ir lokalizavimo uždavinių sprendimui. Šio tyrimo metu buvo išspręsti išsikelti uždaviniai ir gautos išvados:

1. Atliekant literatūros apžvalgą buvo susipažinta su *CNN* technologija, *R-CNN* tobulėjimo istorija ir vaizdų klasifikavimo konkursuose geriausius rezultatus pasiekusiais tinklais. Pasirinkta naudoti *Faster R-CNN* architektūrą, pritaikant žinių perkėlimo metodą, *AlexNet, VGG-16* ir *VGG-19* tinklams. Tinklų apmokymui nutarta naudoti *PUCPR+*, *CARPK* ir *MIX* duomenų rinkinius.

2. Ieškant tinkamiausio tinklo buvo lyginami skirtingais duomenų rinkiniais apmokomi *AlexNet* ir *VGG* architektūrų tinklai. Geriausius tyrimo rezultatus gavo *VGG-19* tinklas apmokytas su *PUCPR+* duomenų rinkiniu, pasiekęs 92,8 % atpažinimo tikslumą. O *VGG-16* tinklas geriausiai apmokytas su *CARPK* ir *MIX* duomenų rinkiniais, pasiekusiais 88,9 % ir 89,5 % tikslumą. Tyrimo pabaigoje buvo palyginti kiekvieno tinklo vienos nuotraukos vidutinio greičio rezultatai ir išsiaiškinta, kad *AlexNet* tinklas informacijos apdorojimui vidutiniškai užtrunka 0,232s, lyginant su *VGG* architektūros tinklais, kurių skaičiavimai analogiškai truko 0,499s ir 0,527s.

3. Atliekant geriausios spalvų erdvės paieškos tyrimą buvo nustatyta, kad standartinės *RGB* spalvų erdvės duomenų konvertavimas į *HSV* / *LAB* spalvų erdvę pablogina *PUCPR+* rinkiniu apmokytų tinklų atpažinimo rezultatus. *AlexNet* tinklo rezultatai kito nuo 79 % iki 68,7 / 57,5 %. *VGG-16* tinklo rezultatai beveik nepakito su *HSV* spalvų erdve, tačiau su *LAB* apmokymas nepavyko. *VGG-19* tinklo apmokymo rezultatai kito nuo 92,8 % iki 88,4 / 82,8 %. Dėl neigiamų tyrimo rezultatų tyrimas nebuvo tęsiamas su kitais duomenų rinkiniais.

4. Tinklų pritaikymo tyrimas padėjo nustatyti apmokytų tinklų stabilumą ir objektų aptikimo rezultatus išbandant su kitais duomenų rinkiniais. Blogiausiai pasirodė *CARPK* duomenų rinkiniu apmokyti tinklai, kurie vidutiniškai pasiekė 43,5 % tikslumą. *PUCPR+* rinkiniu apmokyti tinklai pasiekė 57,6 % tikslumą. O geriausius rezultatus parodė *MIX* duomenų rinkinys, kuriuo apmokyti tinklai vidutiniškai gavo 81,6 % tikslumą. Geriausius rezultatus pasiekė *MIX* rinkinio *VGG-16* tinklas su 91–89 % tikslumu, o blogiausiai pasirodė *PUCPR+* rinkinio *AlexNet* tinklas su 15–19 % tikslumu. Remiantis šio tyrimo rezultatais galima daryti išvadą, kad tinklo stabilumas tiesiogiai priklauso nuo apmokymo duomenų įvairovės.

5. Remiantis tyrimo metu geriausiai pasirodžiusiu *VGG-19* tinklu, buvo sukurtas parkavimo vietų užimtumo stebėjimo algoritmas, kuris leidžia paprastos kameros pagalba nuotoliniu būdu nustatyti automobilių stovėjimo aikštelės užimtumą ir / ar parodyti laisvų vietų skaičių bei poziciją.

Literatūros sąrašas

[1] **Marc Tschentscher, Christian Koch, Markus König, Jan Salmen, Marc Schlipsing,** „Scalable real-time parking lot classification: An evaluation of image features and supervised learning algorithms“, IEEE, 2015, [žiūrėta 2019 m. Kovo 25 d.] Prieiga per internetą: https://ieeexplore.ieee.org/document/7280319

[2] **Mei Wang, Weihong Deng,** „Deep Face Recognition: A Survey“, 2018, [žiūrėta 2019 m. Sausio 20 d.] Prieiga per internetą: https://arxiv.org/abs/1804.06655

[3] **Ross Girshick, Jeff Donahue, Trevor Darrell, Jitendra Malik**, „Rich feature hierarchies for accurate object detection and semantic segmentation“, UC Berkeley, 2014, [žiūrėta 2018 m. Gruodžio 2 d.] Prieiga per internetą: https://arxiv.org/pdf/1311.2524.pdf

[4] **Ross Girshick**, „Fast R-CNN, Microsoft Research“, 2015, [žiūrėta 2018 m. Gruodžio 2 d.] Prieiga per internetą: https://arxiv.org/pdf/1504.08083.pdf

[5] **Shaoqing Ren, Kaiming He, Ross Girshick, Jian Sun**, „Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks“, 2016, [žiūrėta 2018 m. Gruodžio 2 d.] Prieiga per internetą: https://arxiv.org/pdf/1506.01497.pdf

[6] **Meng-Ru Hsieh, Yen-Liang Lin, Winston Hsu**, „Drone-based Object Counting by Spatially Regularized Regional Proposal Networks“, ICCV, 2017, [žiūrėta 2018 m. Gruodžio 1 d.] Prieiga per internetą: https://dblp.uni-trier.de/pers/hd/h/Hsieh:Meng=Ru

[7] **Huaizu Jiang, Erik Learned-Miller**, „Face Detection with the Faster R-CNN“, IEEE 2017, [žiūrėta 2019 m. Sausio 5 d.] Prieiga per internetą: https://ieeexplore.ieee.org/abstract/document/7961803

[8] **Aaron van den Oord, Sander Dieleman, Benjamin Schrauwen**, „Deep content-based music recommendation“, 2013, [žiūrėta 2019 m. Sausio 10 d.] Prieiga per internetą: http://papers.nips.cc/paper/5004-deep-content-based-music-recommendation.pdf

[9] **Wenpeng Yin, Katharina Kann, Mo Yu, Hinrich Schütze**, „Comparative Study of CNN and RNN for Natural Language Processing“, 2017, [žiūrėta 2019 m. Sausio 10 d.] Prieiga per internetą: https://arxiv.org/abs/1702.01923

[10] **Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton**, „ImageNet Classification with Deep Convolutional Neural Networks“, 2012, [žiūrėta 2019 m. Sausio 11 d.] Prieiga per internetą: https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks

[11] **Dinggang Shen, Guorong Wu, Heung-Il Suk**, „Deep Learning in Medical Image Analysis“, 2017, [žiūrėta 2019 m. Sausio 11 d.] Prieiga per internetą: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5479722/

[12] **Kaiming He, Georgia Gkioxari, Piotr Dollár, Ross Girshick**, „Mask R-CNN“, 2018, [žiūrėta 2019 m. Sausio 11 d.] Prieiga per internetą: https://arxiv.org/abs/1703.06870

[13] **J.R.R. Uijlin, van de Sande, T. Gevers, Smeulders**, „Selective Search for Object Recognition“, 2012, IJCV, [žiūrėta 2019 m. Sausio 11 d.] Prieiga per internetą: https://www.koen.me/research/pub/uijlings-ijcv2013-draft.pdf

[14] **Shawn Hershey, Sourish Chaudhuri, Daniel P. W. Ellis, Jort F. Gemmeke**, „CNN architectures for large-scale audio classification“, 2017, IEEE, [žiūrėta 2019 m. Sausio 11 d.] Prieiga per internetą: https://ieeexplore.ieee.org/abstract/document/7952132

[15] **Sherzod Nurullayev, Sang-Woong Lee**, „Generalized Parking Occupancy Analysis Based on Dilated Convolutional Neural Network“, 2019, MDPI, [žiūrėta 2019 m. Sausio 20 d.] Prieiga per internetą: https://www.mdpi.com/1424-8220/19/2/277/htm

[16] **Debaditya Acharya, Weilin Yan, Kourosh Khoshelham,** „Real-time image-based parking occupancy detection using deep learning“, The University of Melbourne, 2018, [žiūrėta 2019 m. Sausio 20 d.] Prieiga per internetą: https://www.researchgate.net/publication/323796590_Real-time_image-based_parking_occupancy_detection_using_deep_learning

[17] **Hyung-Il Kim ir Yong Man Ro**, „Collaborative facial color feature learning of multiple color spaces for face recognition“, Image and Video Systems Lab, School of Electrical Engineering, KAIST, Republic of Korea, 2016, [žiūrėta 2019 m. Kovo 3 d.] Prieiga per internetą: https://ieeexplore.ieee.org/abstract/document/7532642

[18] **Rahman, M. A., Haque, M. T., Shahnaz, C., Fattah, S. A., Zhu, W. P., & Ahmed, M. O**, „Skin Lesions Classification Based on Color Plane-Histogram-Image Quality Analysis Features Extracted from Digital Images“, Bangladesh University of Engineering and Technology, IEEE, 2017, [žiūrėta 2019 m. Kovo 10 d.] Prieiga per internetą: https://ieeexplore.ieee.org/abstract/document/8053183

[19] **Cecilia Di Ruberto ir Lorenzo Putzu,** „A feature learning framework for histology images classification“, Department of Mathematics and Computer Science, University of Cagliari, Italija, 2016, [žiūrėta 2019 m. Kovo 10 d.] Prieiga per internetą: https://www.researchgate.net/publication/301354024_A_Feature_Learning_Framework_for_Histology_Images_Classification

[20] **Xi Wang, Ronny Hansch, Lizhuang Ma ir Olaf Hellwich**, „Comparison of Different Color Spaces for Image Segmentation using Graph-cut“, IEEE, 2014, [žiūrėta 2019 m. Kovo 20 d.] Prieiga per internetą: https://ieeexplore.ieee.org/document/7294824

[21] **Mark Everingham, Luc Van Gool, Christopher K. I. Williams, John Winn, Andrew Zisserman,** „The PASCAL Visual Object Classes (VOC) Challenge“, 2010, [žiūrėta 2019 m. Kovo 20 d.] Prieiga per internetą: https://link.springer.com/article/10.1007/s11263-009-0275-4

[22] **Julian Dasilva, Ricardo Jimenez, Roland Schiller ir Sanja Zivanovic Gonzalez**, „Unmanned Aerial Vehicle-based Automobile License Plate Recognition System for Institutional Parking Lots“, 2017, [žiūrėta 2019 m. Kovo 24 d.] Prieiga per internetą: http://www.iiisci.org/Journal/CV$/sci/pdfs/SA270TX17.pdf

[23] **Houben, S., Komar, M., Hohm, A., Luke, S., Neuhausen, M., & Schlipsing, M.,** „ On-Vehicle Video-Based Parking Lot Recognition with Fisheye Optics“, IEEE, 2013, [žiūrėta 2019 m. Kovo 24 d.] Prieiga per internetą: https://ieeexplore.ieee.org/document/6728595

[24] **Yi-Lun Lin, Li Li**, „Master General Parking Skill via Deep Learning“, IEEE, 2017, [žiūrėta 2019 m. Kovo 24 d.] Prieiga per internetą: https://ieeexplore.ieee.org/abstract/document/7995836

[25] **Paulo R.L. de Almeida, Luiz S. Oliveira, Alceu S. Britto Jr., Eunelson J. Silva Jr. Alessandro L. Koerich** „ PKLot – A robust dataset for parking lot classification“, 2015, [žiūrėta 2019 m. Kovo 25 d.] Prieiga per internetą: https://dl.acm.org/citation.cfm?id=2785180

[26] **Giuseppe Amato, Fabio Carrara, Fabrizio Falchi, Claudio Gennaro, Carlo Meghini, Claudio Vairo,** „ Deep learning for decentralized parking lot occupancy detection “, ScienceDirect, 2016,

[žiūrėta 2019 m. Kovo 25 d.] Prieiga per internetą: https://www.sciencedirect.com/science/article/pii/S095741741630598X

[27] **Martin Ahrnbom, Kalle, Astrom ir Mikael Nilsson,** „Fast Classification of Empty and Occupied Parking Spaces Using Integral Channel Features“, Lund University, Sweden, 2016, [žiūrėta 2019 m. Kovo 25 d.] Prieiga per internetą: https://ieeexplore.ieee.org/document/7789690

[28] **Sepehr Valipour, Mennatullah Siam, Eleni Stroulia, Martin Jagersand,** „Parking Stall Vacancy Indicator System Based on Deep Convolutional Neural Networks“, University of Alberta, IEEE, 2016, [žiūrėta 2019 m. Kovo 25 d.] Prieiga per internetą: https://ieeexplore.ieee.org/document/7845408

[29] **Guobin Chen, Wongun Choi, Xiang Yu, Tony Han, Manmohan Chandraker**, „Learning Efficient Object Detection Models with Knowledge Distillation“, NIPS, 2017, [žiūrėta 2019 m. Gegužės 13 d.] Prieiga per internetą: http://papers.nips.cc/paper/6676-learning-efficient-object-detection-models-with-knowledge-distillation

[30] **Mehdi Noroozi, Ananth Vinjimoor, Paolo Favaro, Hamed Pirsiavash,** „Boosting Self-Supervised Learning via Knowledge Transfer“, IEEE, 2018, [žiūrėta 2019 m. Gegužės 13 d.] Prieiga per internetą: http://openaccess.thecvf.com/content_cvpr_2018/html/Noroozi_Boosting_Self-Supervised_Learning_CVPR_2018_paper.html

[31] **Sepehr Valipour, Mennatullah Siam, Eleni Stroulia, Martin Jagersand,** „Parking-stall vacancy indicator system, based on deep convolutional neural networks“, IEEE, 2016, [žiūrėta 2019 m. Gegužės 13 d.] Prieiga per internetą: https://ieeexplore.ieee.org/abstract/document/7845408

[32] **Tomas Fabian**, „An Algorithm for Parking Lot Occupation Detection“, Department of Computer Science, 2008, [žiūrėta 2019 m. Gegužės 14 d.] Prieiga per internetą: https://ieeexplore.ieee.org/abstract/document/4557855

[33] **Saifa Khantasak, Nattha Jindapetch, Pakpoom Hoyingcharoen, Kanadit Chetpattananondh, Masami Ikura,** „Parking Violation Detection System based on Video processing“, IEEE, 2018, [žiūrėta 2019 m. Gegužės 14 d.] Prieiga per internetą: https://ieeexplore.ieee.org/abstract/document/8688790

[34] **Abdullah, Maruf, Sabbir Ahmed, Tangir Ahmed, Antor Roy, Zannatul Ferdous Nitu**, „A Proposed Model of Integrated Smart Parking Solution for a city“, American International University-Bangladesh, ICREST, 2019, [žiūrėta 2019 m. Gegužės 14 d.] Prieiga per internetą: https://ieeexplore.ieee.org/abstract/document/8644414

[35] **Yunqiang Li, Guobin Lin**, „Design of intelligent parking lot based on Arduino“, IOP, 2019, [žiūrėta 2019 m. Gegužės 14 d.] Prieiga per internetą: https://iopscience.iop.org/article/10.1088/1757-899X/490/4/042010/meta

[36] **Bill Yang Cai, Ricardo Alvarez, Michelle Sit, Fábio Duarte, and Carlo Ratti**, „Deep Learning Based Video System for Accurate and Real-Time Parking Measurement“, IEEE, 2019, [žiūrėta 2019 m. Gegužės 14 d.] Prieiga per internetą: https://ieeexplore.ieee.org/abstract/document/8660429