Kauno technologijos universitetas

Elektros ir Elektronikos fakultetas

Giliųjų neuroninių tinklų taikymo kelio trūkių aptikimui nuotraukose tyrimas

Baigiamasis magistro projektas

Artūras Gulbinas

Projekto autorius

Doc. Arūnas Lipnickas

Vadovas

Kaunas, 2019

Kauno technologijos universitetas

Elektros ir Elektronikos fakultetas

Giliųjų neuroninių tinklų taikymo kelio trūkių aptikimui nuotraukose tyrimas

Baigiamasis magistro projektas

Valdymo technologijos (6211EX014)

Artūras Gulbinas

Projekto autorius

Doc. Arūnas Lipnickas

Vadovas

Prof. Rimvydas Simutis

Recenzentas

Kaunas, 2019

Kauno technologijos universitetas

Elektros ir elektronikos fakultetas

Automatikos katedra

Artūras Gulbinas

Giliųjų neuroninių tinklų taikymo kelio trūkių aptikimui nuotraukose tyrimas

Akademinio sąžiningumo deklaracija

Patvirtinu, kad mano, Artūro Gulbino, baigiamasis projektas tema „Giliųjų neuroninių tinklų taikymo kelio trūkių aptikimui 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)

Gulbinas Artūras. Giliųjų neuroninių tinklų taikymo kelio trūkių aptikimui nuotraukose tyrimas. Magistro baigiamasis projektas / vadovas doc. Arūnas Lipnickas; Kauno technologijos universitetas, Elektros ir elektronikos fakultetas.

Studijų kryptis ir sritis (studijų krypčių grupė): Elektronikos inžinerija. Inžinerijos mokslai

Reikšminiai žodžiai: vaizdų analizė, gilieji neuroniniai tinklai, kelio trūkis

Kaunas, 2019. 61p

Santrauka

Moksliniame darbe nagrinėjami vaizdo analizės, tiksliau giliųjų neuroninių tinklų panaudojimo, kelio trūkiams atpažinti iš nuotraukų, būdai. Apmokomi gilieji neuroniniai tinklai taip, kad pateikus nuotrauką, parodytų kelio trūkio vietą. Atliekami eksperimentai, jeigu reikia – koreguojami parametrai. Gauti duomenys yra palyginami ir atvaizduojami lentelėmis, bei grafikais.

Tiriamojoje dalyje aprašyta giliųjų neuroninių tinklų veikimo principas, bei giliau išnagrinėjami tyrimui naudojami būdai: „Faster R-CNN“, semantinė segmentacija ir U-Net. Taip pat apžvelgiama duomenų bazė, tikslumo kokybės įverčiai, bei iš anksto apmokyti gilieji neuroniniai tinklai.

Eksperimentinėje dalyje suprogramuoti visi trys skirtingi giliųjų neuroninių tinklų panaudojimo būdai, pasirinkta duomenų bazė yra pritaikoma esamai užduočiai, ir išplečiama. Gilieji neuroniniai tinklai yra apmokomi ir ištestuojami. Testai atliekami bent po tris kartus, gauti duomenys yra apdorojami, atvaizduojami ir palyginami. Pateikiamos išvados.

Eksperimentai, programavimas, skaičiavimas ir visų duomenų apdorojimas atliekamas naudojant programinę įrangą *MATLAB*.

Gulbinas Artūras. Research of Convolutional Neural Networks Application for Road Cracks Detection in Images. Master's Final Degree Project / doc. Arūnas Lipnickas; Faculty of Electrical and Electronics Engineering, Kaunas University of Technology.

Study field and area (study field group): Electronics engineering. Engineering science

Keywords: image processing, convolutional neural network, road crack.

Kaunas, 2019. 61p.

Summary

The research explores ways of image analysis, the use of precisely deep neural networks. Methods of using deep neural networks, to recognize road cracks from photographs, are investigated. Experiments are performed, adjusting parameters if necessary. Obtained data is compared and displayed in tables and graphs.

In the exploratory part, the principle of deep neural networks is described, and the methods used for the research are analyzed deeper: Faster R-CNN, semantic segmentation and U-Net. It also reviews the database, accuracy quality estimates, and pre-trained deep neural networks.

In the experimental part, all three different ways of using deep neural networks are programmed, the selected database is adapted to the existing task and expanded. Deep neural networks are trained and tested. The tests are performed at least three times, obtained data are processed, displayed and comparable. Conclusions are provided.

Experiments, programming, computing, and data processing are all done using the *MATLAB* software.

Turinys

Lentelių sąrašas

Paveikslų sąrašas

Santrumpų ir terminų sąrašas

Santrumpos:

Bbox – ribinis langelis (angl. *boundary box*)

CNN – gilusis neuroninis tinklas (angl. *convolutional neural network*)

F1 – tikslumo matas

FC – pilnai prijungtas sluoksnis (angl. *fully connected layer*)

MCC – tikslumo matas

R-CNN – regioninis gilusis neuroninis tinklas (angl. *regional convolutional neural network*)

SVM – matematinis modelis analizuojantis duomenis (angl. support-vector machine)

Įvadas

Pramonė, gamyba, viskam ką šiais laikais kuria žmogus, norima pasiekti aukščiausią kokybę. Tačiau realybėje situacija yra kiek kitokia, kadangi ypač masinėje produkcijoje neišvengiama broko ir defektų. Tiekėjai turi užtikrinti, kad į rinką nepatektų gaminiai su defektais, todėl būtina kokybės patikra. Nuo pačių seniausių laikų, visi defektai ir gamybos brokai buvo tikrinami žmogaus akimi. Ir niekas nenuginčys, jog būtent žmogaus akis yra geriausias defektų aptikimo mechanizmas, kadangi žmogus gali vertinti defektus pagal reikiamus kriterijus, ir tai gali daryti sekundžių ar net milisekundžių laiko tarpsnyje. Deja, šiais laikais produkcija ir gamyba beveik visur yra masinė ir nepertraukiama, todėl žmogaus akimis tikrinti tokius didelius kiekius gaminamos produkcijos, yra ne tik neįmanoma, bet ir finansiškai brangu. Reiktų daug žmonių, kelių pamainų, ir žmogus pavargtų. Todėl defektų aptikimas ir kokybės kontrolė yra masiškai automatizuojama. Tam panaudojama skaitmeninė vaizdų analizė, kuri yra sparčiai vystoma, ir gali būti pritaikyta praktiškai bet kokiai užduočiai atlikti.

Skaitmeninis vaizdų apdorojimas ir vaizdo analizė šiuo metu yra ko gero bene dažniausiai naudojamos skaitmeninių signalų apdorojimo sričių. Platus naudojimas išpildytas pramonėje, buityje, transporte. Pramonėje daugelis automatizuotų linijų, ypač tų, kuriose yra robotai, naudoja įvairias kameras, vaizdui apdoroti ir analizuoti. Vaizdų apdorojimas – duomenų apdorojimo sistemos panaudojimas vaizdams sudaryti, skaityti, analizuoti, suvokti, gerinti, atvaizduoti. Vaizdų analizė apima sritis tokias, kaip segmentavimas, transformavimas, požymių išgavimas, objekto klasifikavimas. Pramonėje vaizdo analizė naudojama daugeliui užduočių vykdyti: gamybos klaidų aptikimas, produktų klasifikavimas, roboto valdymas, produkcijos surinkimas ir daugelis kitų.

Vaizdų analizė gali būti taikoma ne tik gamyboje, bet ir ten kur reikia stebėti objekto esamą būklę ir ją įvertinti. Šiais laikais automobilinių kelių infrastruktūra yra taip stipriai išplėtota, jog susiekimas būtų užtikrintas visoms gyvenamosios teritorijoms. Taip pat verta paminėti, jog kelių infrastruktūra su laiku vis tik plečiama, ir didėja. Tačiau, keliai nėra amžini ir jiems reikia priežiūros bet rekonstravimo darbų. Dėl netinkamos kelių būklės gali kilti pavojus žmonių gyvybėms, todėl itin svarbu užtikrinti, kad keliai atitiktų reikalavimus ir būtų tvarkomi laiku. Visos rimtos kelių būklės problemos prasideda nuo pirmojo požymio – kelio trūkio. Kai kelias ima trūkinėti, tai pirmasis ženklas, jog būtina imtis priemonių, norint išvengti didesnių problemų, arba planuoti rekonstrukciją.

Yra be galo daug priežasčių kodėl keliai ima trūkinėti: pernelyg didelis apkrovimas, silpnas arba plonas paviršius, pagrindas arba pogrindis, prasta drenažo sistema, senas arba sausas mišinys, prastas mišinys, padidėjęs mažas eismo intensyvumas, dideli temperatūrų svyravimai paros metu,

mažos oro angos ir daugelis kitų. Visos šios priežastys veda prie kelio defektų, kurie gali privesti prie skaudžių pasekmių.

Kelio trūkių aptikimas niekada nebuvo prioritetinis dalykas, todėl nėra smarkiai automatizuotas, o Lietuvoje išvis neatliekamas. Dažniausiai kelių rekonstrukcija ima vykdyti, tik kai kelių būklė būna tokia prasta, jog naudojimasis jais kelia riziką saugumui, arba kai sulaukiama daugybės gyventojų skundų dėl esamos kelių būklės. Todėl galima sakyti, jog kelių trūkių aptikimas yra atliekamas žmogaus akies. Net jeigu ir būtų skirti žmonės specialiai šiai užduočiai, t.y. tikrinti kelių būklę, ir ieškoti trūkių, tai būtų absurdiška, kadangi užimtų be galo daug laiko ir didelių žmogiškųjų resursų. Sprendimas: automatizuoti kelio trūkių aptikimą. Šiai užduočiai geriausiai atlikti tiktų skaitmeninė vaizdų analizė, kadangi jos galimybės yra labai didelės ir pati vaizdų analizė jau yra gan smarkiai išplėtota.

Tiriamojo darbo tikslas

Atlikti analizę apie skaitmeninį vaizdų apdorojimą, ir ištirti giliųjų neuroninių tinklų panaudojimo galimybes kelio trūkiams iš nuotraukų aptikti.

Tiriamojo darbo uždaviniai

Atlikti analizę apie giliųjų neuroninių tinklų panaudojimo galimybes. Ištirti šiuos panaudojimo būdus: R-CNN, semantinė segmentacija ir U-Net tinklas. Pasirinkti kelio trūkių duomenų bazę, ją pritaikyti tiriamajam darbui. Apmokyti giliuosius neuroninius tinklus atpažinti kelio trūkius iš nuotraukų. Gautus rezultatus palyginti ir išskirti geriausią būda kelio trūkiams aptikti.

1. Metodinė dalis

1.1. Gilieji neuroniniai tinklai

Panašiai kaip vaikai išmoksta atpažinti objektus, taip ir kompiuteriams reikia parodyti didelius kiekius paveikslėlių prieš jam sugebant ką nors atpažinti ar suklasifikuoti. Kompiuteriai mato paveikslėlius kitokiu būdu nei žmogaus akis. Kompiuterių pasaulis susidaro tik iš skaičių, taigi visi paveikslėliai gali būti atvaizduoti kaip 2 dimensijų skaičių masyvai, vadinami pikseliais. Tačiau nors ir kompiuteris supranta paveikslėlius skirtingai nei žmogus, nereiškia, kad nėra galimybės išmokyti kompiuterį atpažinti tam tikrų modelių ar panašumų, tarp skirtingų vaizdų.

Vaizdų atpažinimo uždavinys yra įvesti paveikslėlį ir išvesti atsakymą – ar yra objektas jame, kokia tikimybė kad jis yra, ar net koks tai objektas. Žmonėms ši atpažinimo užduotis yra pirmas dalykas kuris išmokstamas tik gimus, ir vėliau atliekamas suaugus be menkiausių pastangų. Kai žmogus mato paveikslėlį ar tiesiog aplinką aplink jį, dažniausiai jis sugeba iškart charakterizuoti sceną ir priskirti kiekvienam objektui vardą, net sąmoningai to nepastebėdamas. Būtent šituo žmonės ir skiriasi nuo kompiuterių.

2 pav. Vaizdas matomas žmogaus akimis ir kompiuterio

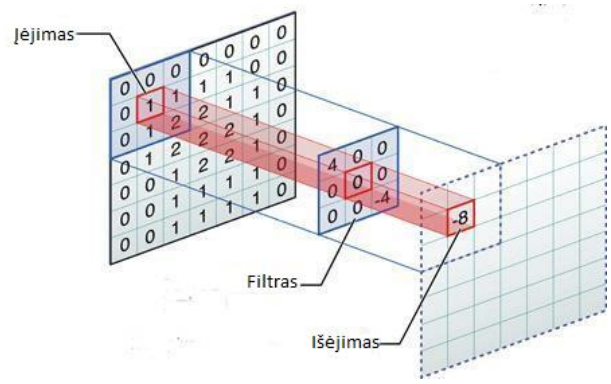
Kai kompiuteris mato vaizdą (įveda vaizdą kaip įvestį), jis mato pikselių reikšmių masyvą. Priklausomai nuo atvaizdo skiriamosios gebos ir dydžio, bus matomas 32 x 32 x 3 masyvų skaičius (3 reiškia RGB reikšmes). Tarkime, kad yra spalvotas vaizdas JPG formoje, o jo dydis yra 480 x 480. Pikselių reikšmių masyvas yra 480 x 480 x 3. Kiekvienas iš šių numerių turi reikšmę nuo 0 iki 255, kuri apibūdina taško pikselių intensyvumą. Šie skaičiai, nors ir beprasmiški žmogui, tačiau atliekant vaizdų atpažinimą, yra vieninteliai kompiuteriui prieinami duomenys. Idėja yra tokia, kad suteikus kompiuteriui šį skaičių masyvą yra gaunamas išvesties skaičius apibūdinančius tikimybę, kad vaizde bus ieškomas objektas.

Atpažinimo procesas žmogaus smegenyse vyksta pasąmoningai. Kai žmogus žiūri į šuns vaizdą, gali jį klasifikuoti, jei nuotraukoje yra identifikuojamų bruožų, pvz., letenų ar keturių kojų. Panašiu būdu kompiuteris gali atlikti vaizdų atpažinimą ir klasifikavimą ieškodamas žemo lygio funkcijų, pvz., kraštų ir kreivių, o vėliau sukurdamas daugiau abstrakčių sąvokų per keletą neuroninių tinklų sluoksnių.

Gilinantis labiau į specifiką apie tai, ką daro CNN, yra tai, kad paveikslėlis perduodamas per keletą pirmųjų neuroninių sluoksnių, tada netiesinių, sujungimų (sumažinimo) ir visiškai prijungtų sluoksnių ir gaunama išvestis (rezultatas). Išėjimas gali būti viena klasė arba tikimybė, kad klasės geriausiai apibūdina vaizdą. Tačiau reiktų suprasti, ką daro kiekvienas iš šių sluoksnių. [1]

1.1.1. Pirmas gilaus neuroninio tinklo sluoksnis

CNN naudoja filtrus (dar vadinamus kerneliais), kad aptiktų vaizdo savybes, pavyzdžiui briaunas. Filtras yra tik savybių matrica, vadinama svoriais, kuria mokoma aptikti konkrečias ypatybes. Filtras palaipsniui juda per kiekvieną vaizdo dalį, kad patikrintų, ar yra funkcija, kurią jis turi aptikti. Norint pateikti vertę, rodančią patikimumą, kad yra tam tikros funkcijos, filtras atlieka sąsukos operaciją, kuri yra elementinis produktas ir suma tarp dviejų matricų.[2]



2 pav. Kernelio (filtro) veikimas

Kai ypatybė yra tam tikroje vaizdo dalyje, sąsukos operacija tarp filtro ir tos vaizdo dalies yra tikras skaičius su didele verte. Jei savybės nėra, gaunama vertė yra maža.

Trečiame paveikslėlyje pavaizduotas filtras, kuris yra atsakingas už dešiniųjų kreivių patikrinimą, ir filtruoja tik dalį vaizdo. Kadangi toje vaizdo dalyje yra ta pati kreivė, kurios ieško filtras, sąsukos operacijos rezultatas yra didelis skaičius (6600).

0	0	0	0	0	0	30
0	0	0	0	50	50	50
0	0	0	20	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0

*

0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Imlus vaizdo laukas | Imlaus vaizdo lauko atvaizdavimas pikseliais | Filtro atvaizdavimas pikseliais

Daugyba ir suma = (50*30)+(50*30)+(50*30)+(20*30)+(50*30) = 6600

3 pav. Dešiniųjų kreivių buvimą tikrinantis filtras

Tačiau kai tas pats filtras filtruoja vaizdą su visai skirtingu briaunų rinkiniu, sąsukos išėjimas yra mažas, o tai reiškia, kad dešinioji kreivė, kurios ieško filtras nėra stipri.

0	0	0	0	0	0	0
0	40	0	0	0	0	0
40	0	40	0	0	0	0
40	20	0	0	0	0	0
0	50	0	0	0	0	0
0	0	50	0	0	0	0
25	25	0	50	0	0	0

*

0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Filtro atvaizdavimas paveikslėlyje | Imlaus vaizdo lauko atvaizdavimas pikseliais | Filtro atvaizdavimas pikseliais

Daugyba ir suma = 0

4 pav. Dešiniųjų kreivių buvimą tikrinantis filtras

Šio filtro perdavimo per visą vaizdą rezultatas yra išvesties matrica, kurioje šio filtro sąsukos saugomos įvairiose vaizdų dalyse. Filtro kanalas turi būti toks pat kaip ir įvesties vaizdas, kad elementas galėtų būti dauginamas. Pavyzdžiui, jei įvesties vaizdas yra trejų kanalų (pvz., RGB), filtras irgi turi būti trejų kanalų. Be to, filtras gali būti perslinktas per įvesties atvaizdą įvairiais intervalais, naudojant reikšmę „stride". Ši reikšmė lemia, per kiek filtras turėtų judėti kiekvienu žingsniu.

Taigi, kad gilusis neuroninis tinklas galėtų išmokti filtro vertes, aptinkančias įvesties duomenis, filtras turi būti perduotas nelinijiniu būdu. Sąsukos operacijos tarp filtro ir įvesties vaizdo

išvesties susumuojama su šališkumo terminu ir nelinijine aktyvavimo funkcija. Aktyvinimo funkcijos tikslas yra įvesti netiesiškumą į tinklą.

Paprastai tinklas per sluoksnį naudoja daugiau nei vieną filtrą. Tada kiekvieno filtro sąsukos išėjimai per įvesties vaizdą yra susieti išilgai paskutinės ašies, sudarydami galutinę 3D išvestį.

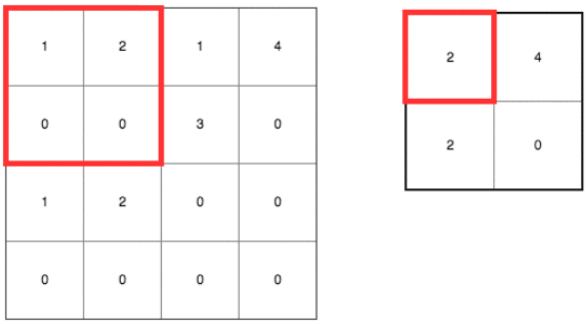
Po vieno ar dviejų neuroninių tinklų sluoksnių, kuriuose vykdomi sąsukos veiksmai, paprastai sumažėja sąsukos sluoksnio pateiktas vaizdas. Šis reprezentacijos dydžio sumažinimas yra vadinamas mažinimu.

1.1.2. Mažinimas

Mokymo procesui paspartinti ir sumažinti tinkle naudojamos atminties kiekį, stengiamasi sumažinti įvesties funkcijos atleidimą. Yra keletas būdų, kaip sumažinti vaizdą, tačiau labiausiai paplitęs būdas: maksimalus sujungimas (angl. *max pooling*).

Maksimalaus sujungimo metu langas perduoda vaizdą pagal nustatytą reikšmę „stride“ (kokiu žingsniu reikia judėti). Kiekviename žingsnyje maksimali reikšmė lango viduje yra sujungiama su išvesties matrica, taigi todėl ir vadinasi maksimalus sujungimas (maksimalių reikšmių suma).

Toliau, 5 pav. pateiktoje schemoje matyti, kad perduodamas vaizdas su 2 žingsniu (2x2), tai yra maksimalaus sujungimo lango matmenys (raudonas kvadratėlis) ir juda x ir y kryptimis. Kiekviename žingsnyje parenkama maksimali vertė lange:



5 pav. Maksimalaus sujungimo veikimo principas

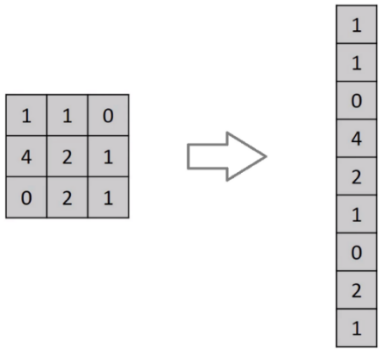
Maksimalus sujungimas žymiai sumažina vaizdo dydį, tuo pačiu sumažindamas ir reikalingos atminties bei operacijų kiekį, atliekamų vėlesniame tinkle.

Papildoma nauda iš maksimalaus sujungimo yra ta, kad ji verčia tinklą sutelkti į keletą neuronų, kurie turi reguliarų poveikį tinklui, o ne naudoti visus iškart. Dėl to mažiau tikimybė prastam duomenų apibendrinimui ir perpildymui.

Po kelių pirmųjų neuroninių sluoksnių ir mažinimo operacijų, 3D vaizdo pateikimas paverčiamas funkcijos vektoriumi, kuris perduodamas į daugiasluoksnį perceptroną, kuris yra neuroninis tinklas, turintis mažiausiai tris sluoksnius. Tai vadinama visiškai prijungtu sluoksniu.

1.1.3. Visiškai prijungtas sluoksnis

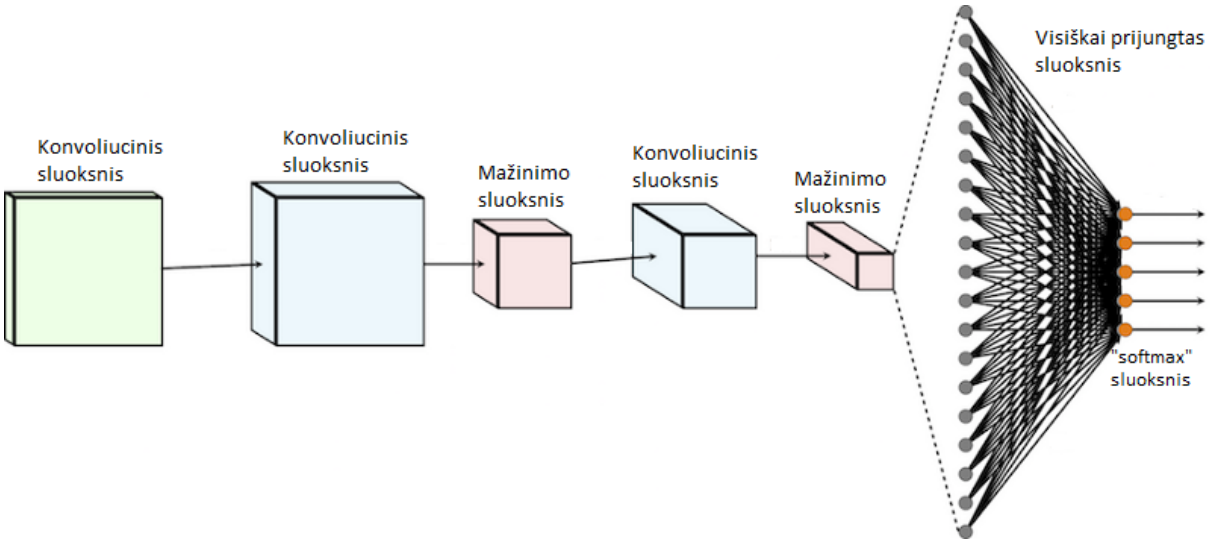
Visiškai sujungto neuroninio tinklo veikloje įvesties reprezentacija išlyginama į funkcijų vektorių ir pernešama per neuronų tinklą, kad būtų galima prognozuoti išėjimo tikimybes. Lyginimo operaciją grafiškai pavaizduota 6 paveikslėlyje.



6 pav. Lyginimo operacija

Šios eilutės sujungiamos, kad sudarytų ilgą funkcijos vektorių. Jei yra keli įvesties sluoksniai, jų eilutės taip pat sujungiamos, kad sudarytų dar ilgesnį funkcijos vektorių. Po to funkcijos vektorius perduodamas per kelis tankiuosius sluoksnius. Kiekviename tankiajame sluoksnyje funkcijos vektorius padauginamas iš sluoksnio svorio ir perteikiamas netiesiškai.

Visiškai prijungta operacija ir tankūs sluoksniai pavaizduoti 7 paveikslėlyje:



7 pav. Visiškai prijungta operacija ir tankieji sluoksniai

Visiškai prijungtas sluoksnis yra sąsukinė operacija su 1x1 išėjimo branduoliu (kerneliu arba filtru). Tai reiškia, kad jei perduodami 128 n-to-n filtrai per matmenį n-by-n, gaunamas 128 ilgio vektorius.

1.1.4. Išvesties sluoksnis

CNN išvesties sluoksnis yra atsakingas už kiekvienos klasės (kiekvieno skaitmens) tikimybę, atsižvelgiant į įvesties vaizdą. Šios tikimybės gaunamos inicializuojant galutinį tankųjį sluoksnį, kuriame neuronų yra tiek pat kiek ir klasių.[3]

1.1.5. Regioniniai gilieji neuroniniai tinklai

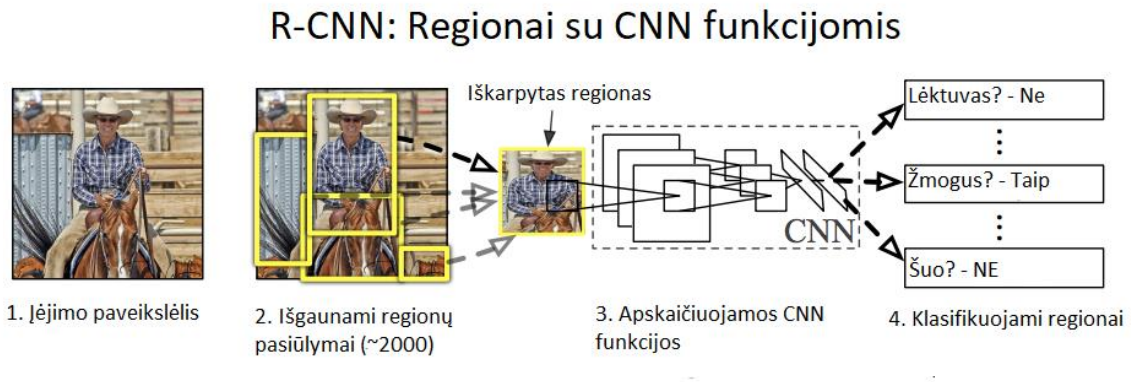
Skirtumas tarp objekto aptikimo algoritmų ir klasifikavimo algoritmų yra tas, kad aptikimo algoritmuose stengiamasi nukreipti ribojantį langelį aplink norimą objektą, kad jis būtų surastas vaizde. Be to, objekto aptikimo atveju nebūtinai gali būti nupieštas tik vienas ribojantis langelis, o gali jų būti daug.

Pagrindinė priežastis, kodėl negalima kurti standartinio konvoliucinio tinklo, po kurio yra visiškai prijungtas sluoksnis, yra ta, kad išvesties sluoksnio ilgis yra kintamas, t.y dominančių objektų skaičius nėra fiksuotas.

Kvailas sprendimas būtų paimti skirtingus dominančius regionus iš vaizdo ir naudoti CNN, kad šis klasifikuotų objekto buvimą šiame regione. Tada atsiranda problema, kad įdomūs objektai gali turėti skirtingas erdvines vietas vaizde ir skirtingus aspektų santykius. Taigi tektų pasirinkti didžiulį skaičių regionų ir tai būtų neįvykdoma užduotis. Todėl buvo sukurti tokie algoritmai kaip R-CNN, YOLO ir kt.[4]

1.1.6. R-CNN

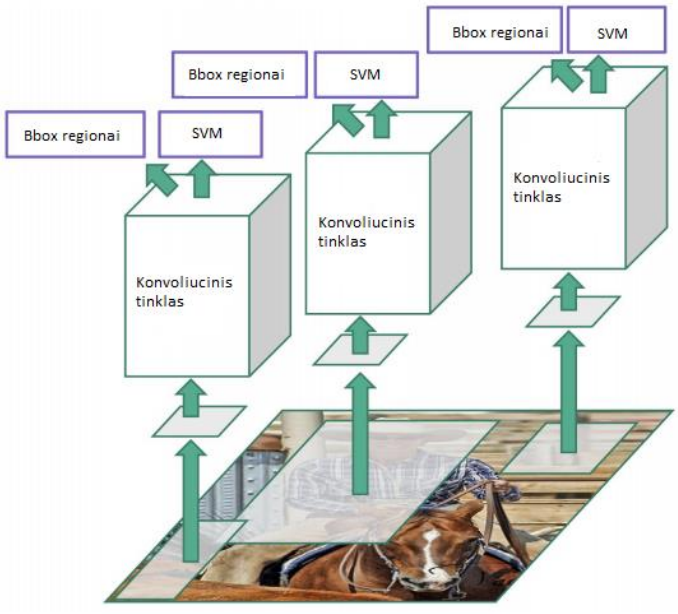
Norint apeiti didžiulių regionų pasirinkimo problemą, buvo pasiūlytas metodas, kai naudojama atrankinė paieška, kad iš atvaizdo būtų gaunami tik 2000 regionų. Iš ko ir kilo pavadinimas „Regional CNN“. Todėl dabar, užuot bandžius klasifikuoti didžiulį skaičių regionų, galima dirbti tik su 2000 regionų. Šie 2000 regiono pasiūlymai pateikiami naudojant atrankinį paieškos algoritmą. [5]



8 pav. R-CNN veikimo principinė schema

Šie 2000 regiono pasiūlymų yra nukreipti į kvadratą ir įeina į konvoliucinį neuroninį tinklą, kuris sukuria 4096 matmenų funkcijų vektorių kaip išvestį. CNN veikia kaip funkcijų ekstraktorius, o išvesties tankusis sluoksnis susideda iš atvaizdo išgautų savybių, o išgautos ypatybės įtraukiamos į SVM, kad būtų galima klasifikuoti objekto buvimą tame regione.

Be prognozuojamo objekto buvimo regioniniuose pasiūlymuose, algoritmas taip pat prognozuoja keturias vertes, kurios yra nuokrypio vertės, kad padidintų ribojimo dėžutės tikslumą. Pavyzdžiui, atsižvelgiant į regiono pasiūlymą, algoritmas būtų numatęs asmens buvimą, tačiau to asmens veidas šiame regione galėjo būti sumažintas perpus. Todėl kompensavimo reikšmės padeda koreguoti regiono pasiūlymo rėmelį.

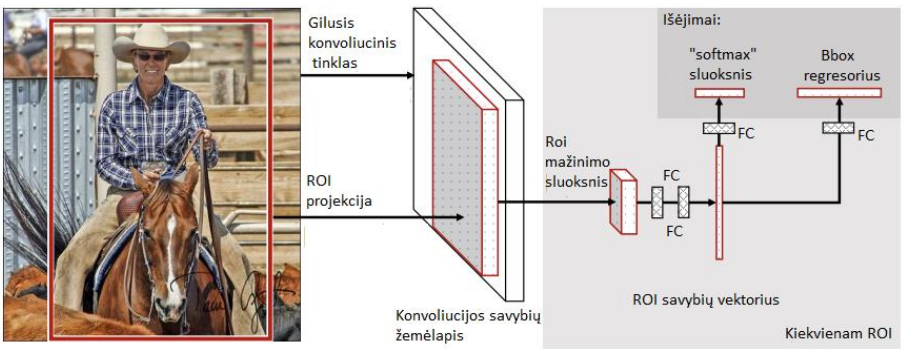


9 pav. CNN veikimas

Problemos su R-CNN:

- Vis dar trunka daug laiko treniruoti tinklą, nes reiktų klasifikuoti 2000 regiono pasiūlymų vienam vaizdui.
- Jis negali būti įgyvendintas realiu laiku, nes kiekvienam bandomam vaizdui apdoroti prireikia apie 47 sekundžių.
- Selektyvus paieškos algoritmas yra fiksuotas algoritmas. Todėl šiame etape mokymasis nevyksta. Tai gali lemti blogų regionų pasiūlymą.

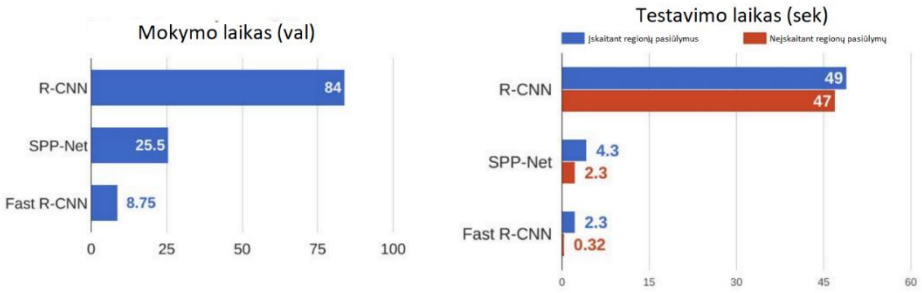
1.1.7. Fast R-CNN



10 pav. Fast R-CNN principinė veikimo schema

Išsprendus kai kuriuos R-CNN trūkumus buvo sukurtas greitesnis objekto aptikimo algoritmas ir pavadintas „Fast R-CNN“. Šis metodas panašus į R-CNN algoritmą. Tačiau vietoj to, kad CNN būtų pateikiami regioniniai pasiūlymai, yra įvedamas įvestas vaizdas į CNN, kad būtų sukurtas konvoliucinis funkcijų žemėlapis. Iš konvoliucinio bruožo žemėlapio nustatomas pasiūlymų regionas ir deformuojamas į kvadratus ir, naudojant RoI sutelkimo sluoksnį, pertvarkomas į fiksuotą dydį, kad šis būtų tiekiamas į visiškai sujungtą sluoksnį. Iš RoI funkcijos vektoriaus naudojamas „softmax“ sluoksnis, kad būtų galima prognozuoti siūlomo regiono klasę ir taip pat ribines dėžutės nuokrypio vertes. Priežastis kodėl „Fast R-CNN“ yra greitesnis už R-CNN yra ta, kad nereikia kaskart maitinti 2000 regioninių pasiūlymų į konvoliucinį neuroninį tinklą. Vietoj to, konvekcijos operacija atliekama tik vieną kartą per vaizdą ir iš jo sukuriamas funkcijų žemėlapis.

Iš 11 paveikslėlyje pateikto grafiko galima daryti išvadą, kad „Fast R-CNN“ yra žymiai greitesnis treniruočių ir bandymų metu nei R-CNN. Žiūrint į „Fast R-CNN“ našumą bandymo metu, įskaitant regiono pasiūlymus, algoritmas žymiai sulėtėja, lyginant su algoritmu kuris nenaudoja regiono pasiūlymų. Todėl regiono pasiūlymai tampa sparčiojo R-CNN algoritmo kliūtimis, turinčiomis įtakos jo veikimui.



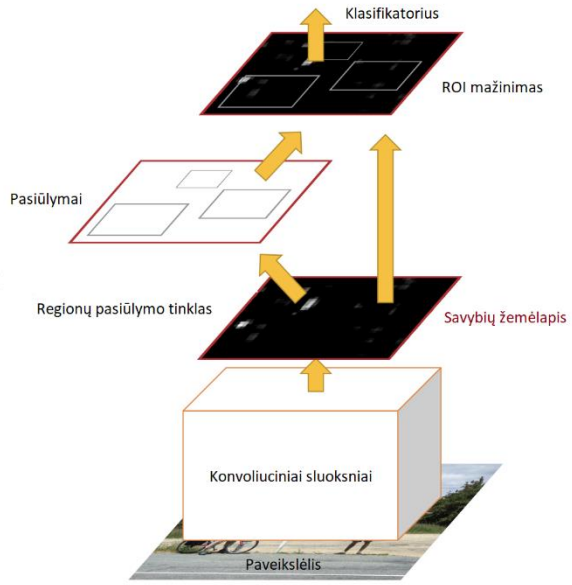
11 pav. Tinklų mokymo ir testavimo laikų palyginimo diagrama

1.1.8. Faster R-CNN

Abu anksčiau aptarti algoritmai (R-CNN & Fast R-CNN) naudoja atrankinę paiešką, kad sužinotų regiono pasiūlymus. Atrankinė paieška - tai lėtas ir daug laiko reikalaujantis procesas, turintis įtakos tinklo veikimui. Todėl atsirado objekto aptikimo algoritmas, kuris pašalina atrankinį paieškos algoritmą ir leidžia tinklui išmokti regiono pasiūlymus.

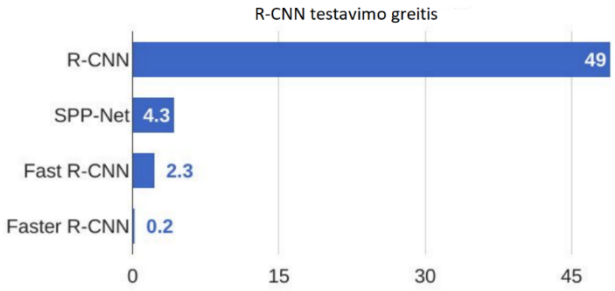
Panašiai kaip ir „Fast R-CNN“, vaizdas pateikiamas kaip įėjimas į konvoliucinį tinklą, kuris suteikia konvoliucinį funkcijų žemėlapį. Vietoj pasirinkimo paieškos algoritmo, esančio funkcijų žemėlapyje, norint nustatyti regiono pasiūlymus, regioniniam pasiūlymui prognozuoti naudojamas atskiras tinklas. Prognozuojami regiono pasiūlymai tada pertvarkomi naudojant „RoI“ sutelkimo

sluoksnį, kuris naudojamas klasifikuoti vaizdą siūlomame regione ir prognozuoti ribojančių langelių poslinkio vertes.



12 pav. Faster R-CNN principinė veikimo schema

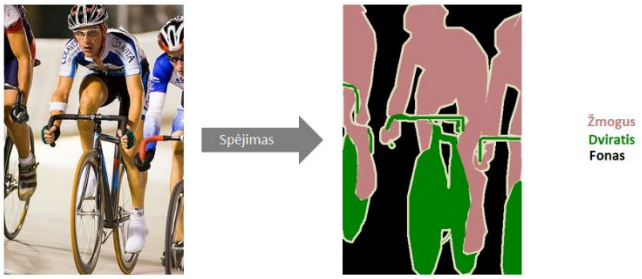
Iš 13 paveikslėlyje pateikto grafiko matyti, kad „Faster R-CNN“ yra daug greitesnis nei jo pirmtakai. Todėl jis net gali būti naudojamas realaus laiko objektų aptikimui. [6]



13 pav. R-CNN tinklų testavimo greičių palyginimo diagrama

1.2. Semantinė segmentacija

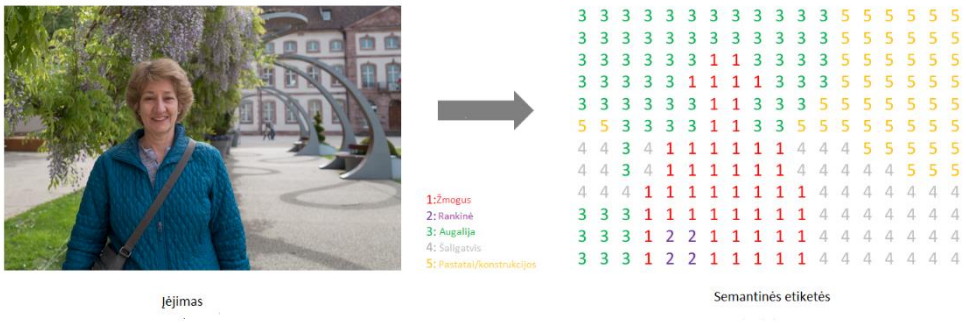
Semantinio vaizdo segmentavimo tikslas yra pažymėti kiekvieną vaizdo tašką atitinkamai, kas yra atstovaujama. Kadangi yra prognozuojamas kiekvienas vaizdo taškas, ši užduotis dar vadinama tankiu prognozavimu. [7]



14 pav. Semantinės segmentacijos vizualizacija

Svarbu paminėti, kad neišskiriami tos pačios klasės egzemplioriai. Rūpinamasi tik kiekvieno pikselio kategorija. Kitaip tariant, jei įvestame vaizde yra du tos pačios kategorijos objektai, segmentavimo žemėlapis neatsižvelgia į juos kaip į atskirus objektus. Yra įvairių modelių klasių, žinomų kaip instancijos segmentavimo modeliai, kurie atskiria atskirus tos pačios klasės objektus.

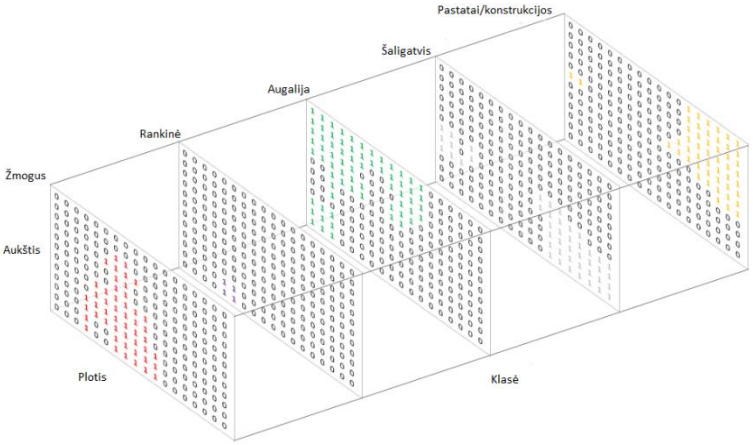
Paprasčiausiai, užduotis yra paimti RGB spalvų vaizdą (aukštis × plotis × 3 aukštis × plotis × 3) arba pilkos spalvos vaizdą (aukštis × plotis × 1 aukštis × plotis × 1) ir išvesti segmentavimo žemėlapį, kuriame kiekviename taške yra klasės žyma, pateikiamas kaip sveikasis skaičius (aukštis × plotis × 1 aukštis × plotis × 1).



15 pav. Segmentavimo žemėlapio vizualizacija

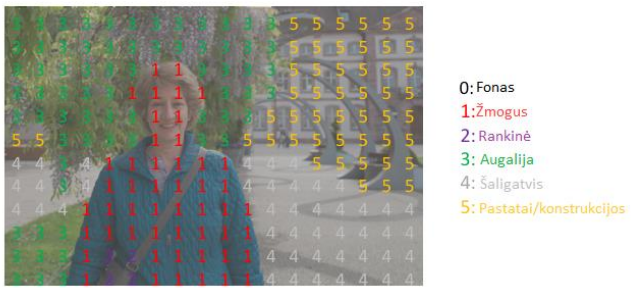
Pastaba: vizualiniam aiškumui pažymėtas mažos skiriamosios gebos prognozės žemėlapis. Iš tikrųjų, segmentavimo žymos skiriamoji geba turi atitikti pradinės įvesties rezoliuciją.

Panašiai, kaip su standartinėmis kategorinėmis vertėmis, sukuriamas kiekvienam iš galimų klasių išvesties kanalas.



16 pav. Segmentavimo žemėlapio, kiekvienai klasei, vizualizacija

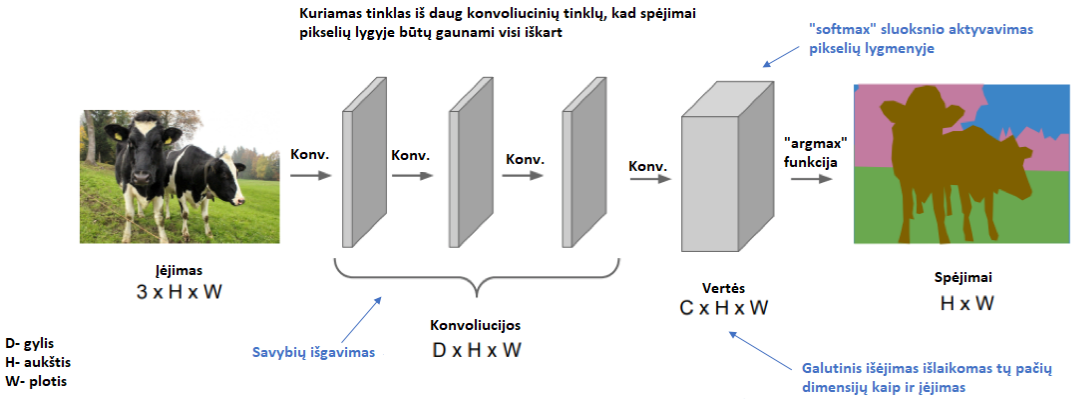
Prognozę galima suskirstyti į segmentavimo žemėlapį (kaip parodyta 18 paveikslėlyje), imant kiekvieno pikselio gylio maksimalų vektorių.



17 pav. Segmentavimo žemėlapio ir įėjimo vaizdo jungtis

Kai persidengiamas vienas tikslo arba prognozės kanalas, tai vadinama kauke, kuri apšviečia atvaizdo sritis, kuriose yra tam tikra klasė. [8]

Paprastas, bet ne itin gudrus požiūris į neuroninio tinklo architektūros sukūrimą šiai užduočiai yra paprasčiausiai sukrauti kelis konvoliucinius sluoksnius (su tais pačiais įdėklais, kad išsaugotų matmenis) ir išduotų galutinį segmentavimo žemėlapį. Tai tiesiog įvesties atvaizdas priskiriamas atitinkamam segmentavimui, persijungiant funkcijų atvaizdavimą. Tačiau tinkle yra brangu taupyti visą rezoliuciją.

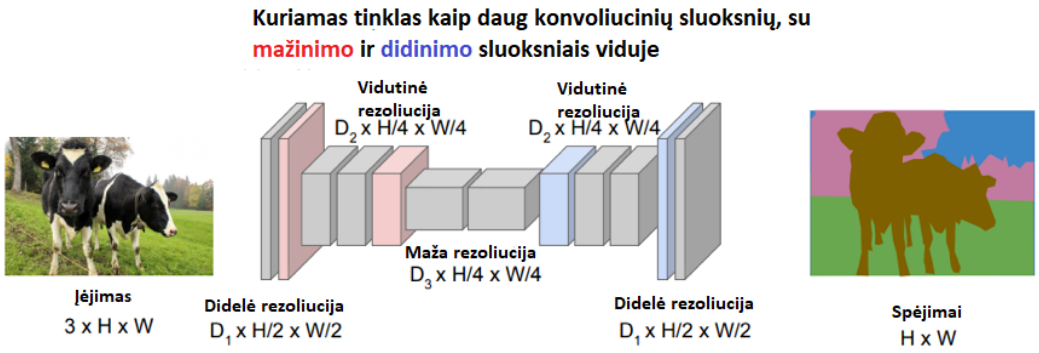


18 pav. Tinklo segmentacijai kūrimo schema

Giliems konvoliuciniams tinklams ankstesniuose sluoksniuose yra tendencija išmokti žemo lygio sąvokas, o vėlesniuose sluoksniuose yra aukšto lygio (ir specializuotų) bruožų. Norint išlaikyti išraiškingumą, turime padidinti funkcijų žemėlapių (kanalų) skaičių, einant giliau į tinklą.

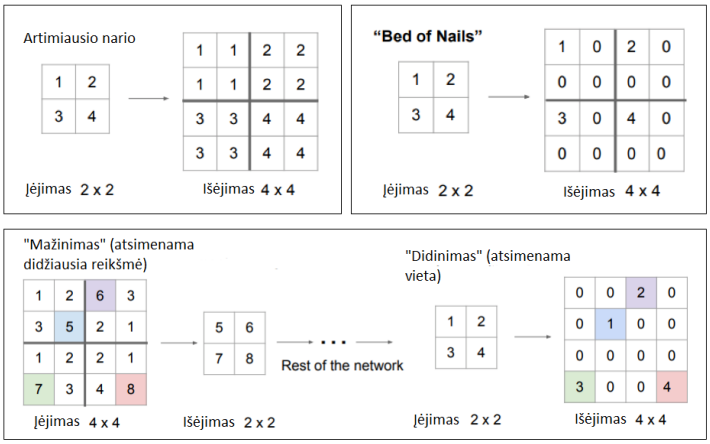
Tai nebūtinai sukelia problemą vaizdų klasifikavimo užduočiai, nes dėl šios užduoties rūpinamasi tik tuo, kas yra vaizde (o ne ten, kur jis yra). Taigi, galima sumažinti skaičiavimo naštą, periodiškai nukreipiant funkcijų žemėlapius per susivienijimą arba susikertančias sąryšius (ty sumažinant erdvinę skiriamąją gebą). Tačiau vaizdo segmentavimui reikia, kad modelis sukurtų pilnos rezoliucijos semantinę prognozę.

Vienas populiarus būdas yra sekti kodavimo / dekoderio struktūrą, kurioje sumažinama įvesties erdvinė skiriamoji geba, kuriant mažesnės skiriamosios gebos funkcijų atvaizdus, kurie yra labai efektyvūs diskriminuojant klases ir atkuriant funkcijų atvaizdų pavyzdžius į pilnos rezoliucijos segmentavimo žemėlapius. [9]



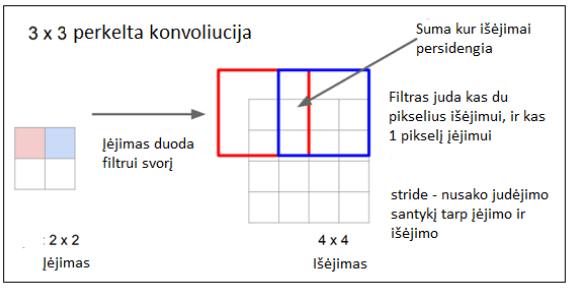
19 pav. Tinklo segmentacijai kūrimo schema, patobulinta

Yra keletas skirtingų metodų, kurie gali būti panaudoti, norint parodyti funkcijų žemėlapio skiriamąją gebą. Nors sutelkimo operacijos išreiškia rezoliuciją, apibendrindamos vietinę zoną, kurioje yra viena reikšmė (ty vidutinis arba maksimalus sutelkimas), operacijos „išjungimas“ išryškina rezoliuciją, paskirstydamos vieną vertę į didesnę skiriamąją gebą.



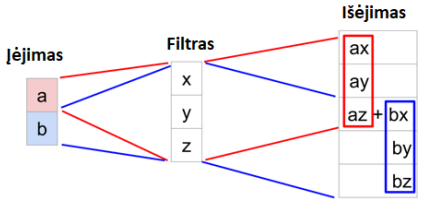
20 pav. Metodai, skirti parodyti funkcijų žemėlapio skiriamąją gebą

Tačiau perkeltos konvoliucijos būdas yra pats populiariausias.



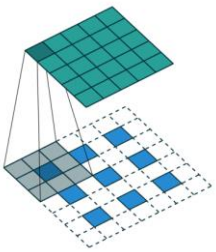
21 pav. Perkeltos konvoliucijos būdas

Nors tipinė konvoliucinė operacija išreiškia šiuo metu filtro rodinyje esančių reikšmių tašką, ir sukuria vieną vertę atitinkamai išėjimo padėčiai, perkeliama konvoliucija iš esmės daro priešingą. Perkeliant konvoliuciją, iš mažos skiriamosios gebos savybių žemėlapio yra paimama viena vertė ir dauginami visi filtro svoriai pagal šią vertę, projektuojant šias svertines vertes į išvesties funkcijų žemėlapį.



22 pav. Filtro veikimas

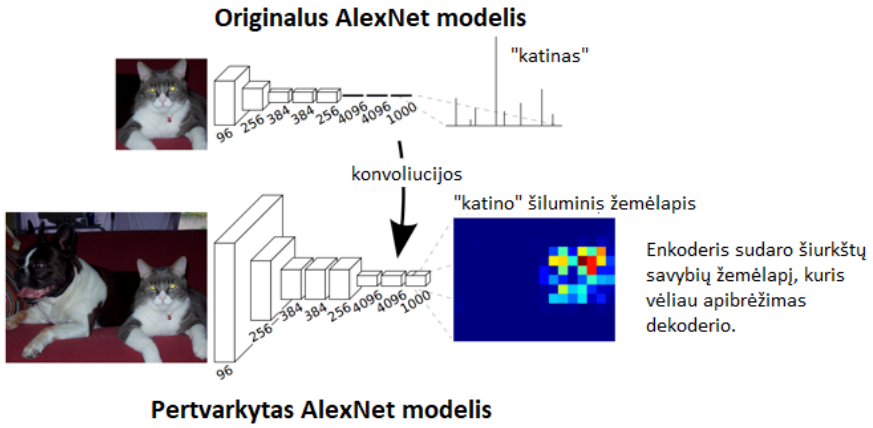
Filtrų dydžiams, kurie sukuria persidengimą išvesties funkcijų žemėlapyje (pvz., 3x3 filtras su 2 žingsniu - kaip parodyta 23 paveikslėlyje), persidengiančios vertės yra tiesiog sudedamos. Deja, tai yra linkusi gaminti šaškių lentos artefaktą ir yra nepageidaujama, todėl geriausia užtikrinti, kad filtro dydis nesudarytų persidengimo.



23 pav. Filtras 3x3 su 2 žingsniu

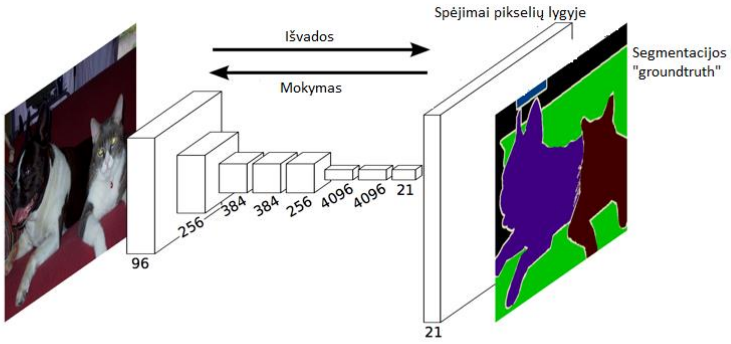
1.2.1. FCN

2014 m. Pabaigoje buvo pristatytas „visiškai konvuliucinis“ tinklas naudojamas vaizdo segmentavimo užduotims atlikti. Kūrėjai siūlo pritaikyti esamus, gerai ištirtus vaizdo klasifikavimo tinklus (pvz., „AlexNet“) tarnauti kaip tinklo kodavimo modulis, pridedant dekoderio modulį su transponuojamais konvoliuciniais sluoksniais, kad išryškintų šiurkščius požymių žemėlapius į pilnos skiriamosios gebos segmentavimo žemėlapį.



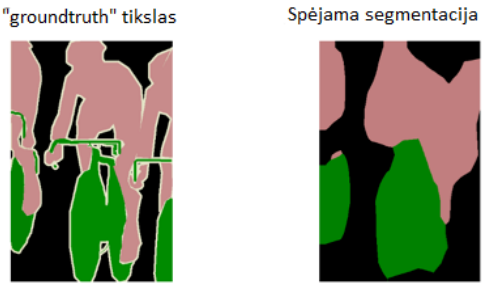
24 pav. AlexNet modelių pertvarkymas

Visas tinklas, kaip 25 paveikslėlyje, yra apmokytas pagal pikselių mąstymo kryžminį entropijos praradimą.



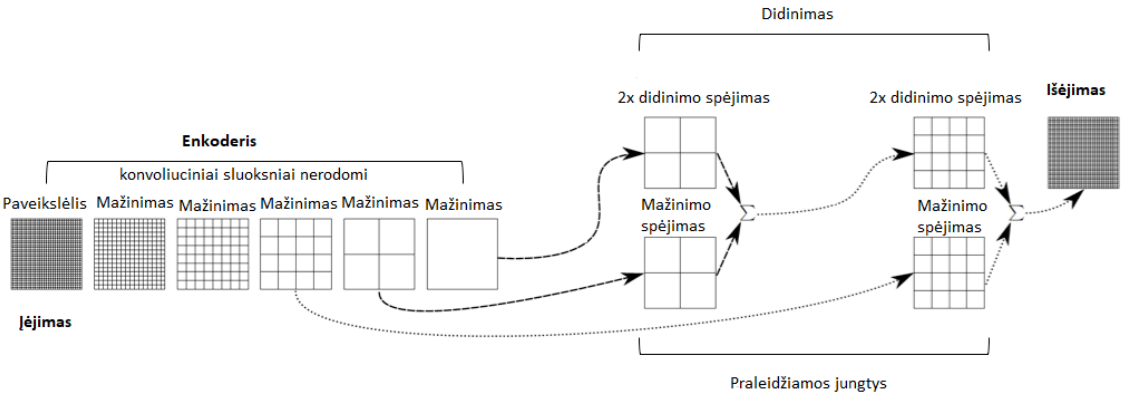
25 pav. AlexNet modelis pritaikytas segmentacijai

Tačiau, kadangi kodavimo modulis sumažina įvesties skiriamąją gebą 32 kartus, dekoderio modulis stengiasi gaminti smulkiagrūdžius segmentus (kaip parodyta 26 paveikslėlyje).



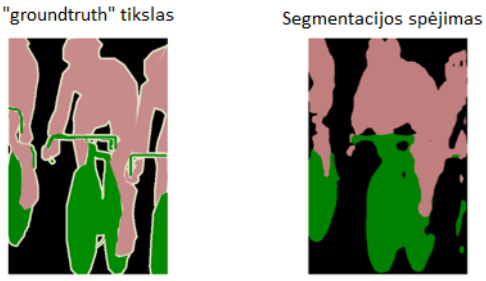
26 pav. Segmentacijos tikrieji rodmenys ir spėjamieji

Ši problema sprendžiama lėtai imant (etapais) užkoduotą vaizdą, pridedant „praleisti ryšius“ iš ankstesnių sluoksnių ir apibendrinant šiuos du funkcijų žemėlapius.



27 pav. „Praleisti ryšius“ taikymas, schema

Šie ankstesnių tinklo sluoksnių praleidimo ryšiai (prieš imties operaciją) turėtų suteikti reikiamą informaciją, kad būtų galima atkurti tikslias figūrų segmentavimo ribas. Iš tiesų, galima susigrąžinti ir daugiau smulkiagrūdžių detalių, pridėjus šiuos praleidimo ryšius.

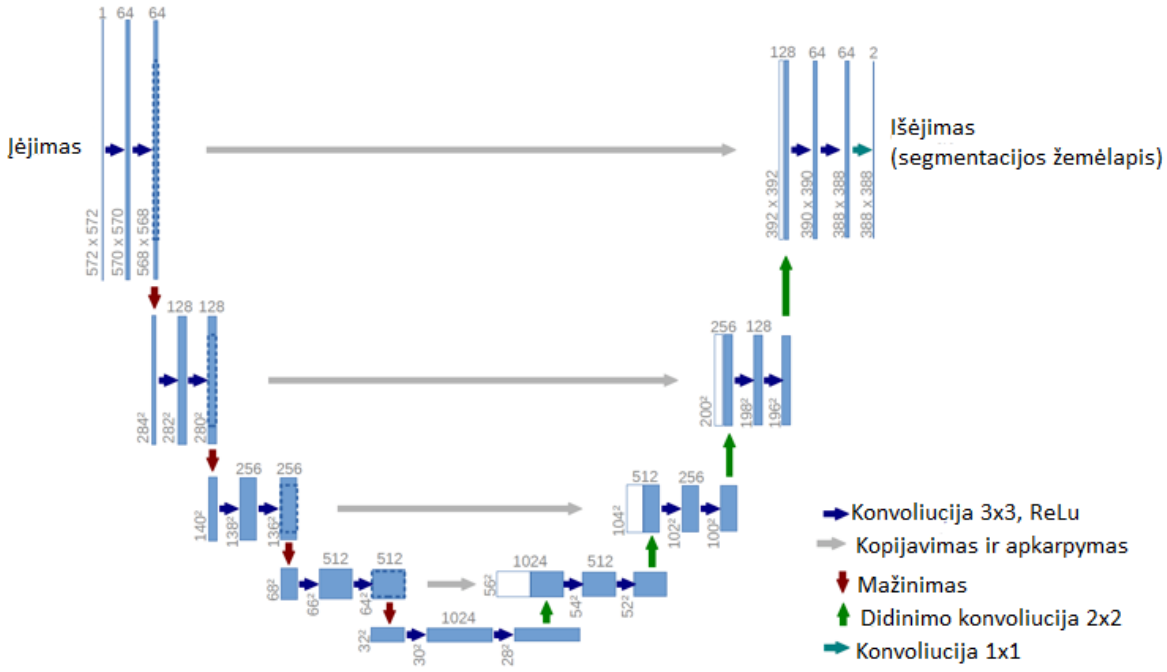


28 pav. Segmentacijos tikrieji rodmenys ir spėjamieji

1.2.2. U-NET

Remiantis visa kodavimo-dekoderio ir praleidimo ryšio koncepcija, visiškai konvoliucinio tinklo idėja išsiplėtė į U-net. „U-net“ pristato simetriją FCN, didindamas dekoderio dydį, kad jis atitiktų koduotąją, ir pakeičia sumų operaciją praleidimo jungtyse su susiejimu. [10]

Dėl simetrijos galima perduoti daug daugiau informacijos iš mėginių atrankos sluoksnių į atrankos mėginius (nes dabar yra daugiau funkcijų žemėlapių), tokiu būdu pagerinant galutinės produkcijos skiriamąją gebą. [11] [12]



29 pav. U-Net architektūros schema

Standartinis „U-Net“ modelis susideda iš kiekvienos architektūros „bloko“ konvekcijos operacijų serijos. Kaip buvo minėta dėl konvoliucinių tinklų architektūrų, egzistuoja keletas pažangesnių „blokų“, kuriais galima pakeisti sukrautus konvoliucinius sluoksnius. [13]

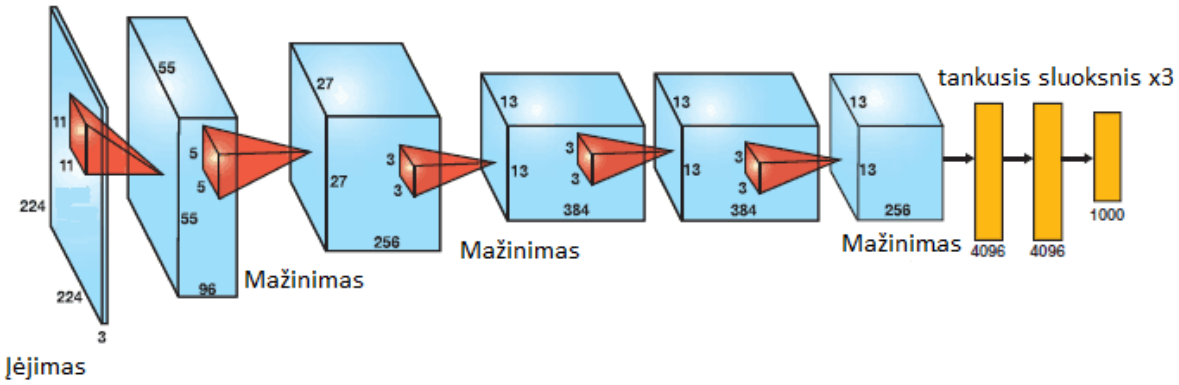
1.3. Iš anksto apmokyti tinklai

1.3.1. „Alexnet“

„AlexNet“ – tai gilusis neuroninis tinklas, kuris yra apmokytas daugiau nei milijonu vaizdų iš „ImageNet“ duomenų bazės. Tinklas turi 8 sluoksnius (5 gilieji (sąsukiniai) ir 3 pilnai sujungti) ir gali klasifikuoti vaizdus į 1000 skirtingų kategorijų, pavyzdžiui, klaviatūra, pelė, pieštukas ar net daugelis gyvūnų. Tinklo įvesties dydis yra 227 iki 227. Tinklas taip pat gali būti naudojamas ir

defektų aptikimui, kadangi filtrai ir sluoksniai kurie atpažįsta objektų savybes, veikia taip pat kaip ir klasifikavimo užduotyse. [14]

„AlexNet“ architektūra tai pirmasis gilusis neuroninis tinklas kuris „ImageNet“ duomenų bazės paveikslėlių klasifikacijos tikslumą padidino labai ženkliai, lyginant su tradiciniais metodais. Tinklas turi 5 giliuosius (convulutional) sluoksnius ir 3 pilnai sujungtus sluoksnius, kaip parodyta 30 paveikslėlyje. [15]



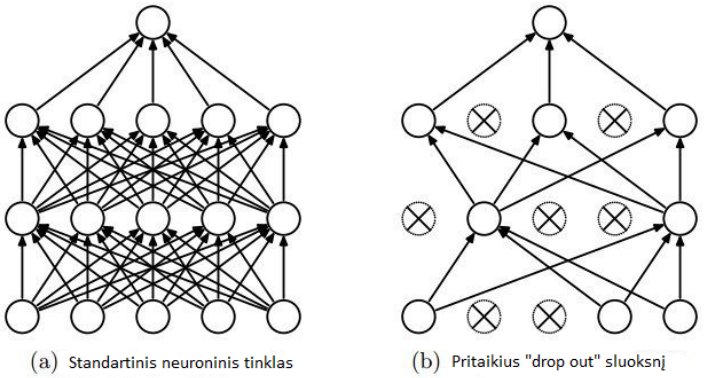
30 pav. „AlexNet“ giliojo neuroninio tinklo architektūra

„AlexNet“ netiesinei daliai naudoja „ReLu“ (ištaisytas linijinis vienetas), o ne „Tanh“ ar „Sigmoid“ funkciją, kurios buvo ankstesnis standartas tradiciniams neuroniniams tinklams.

ReLu privalumas lyginant su sigmoid yra tas, kad jis mokosi daug greičiau nei pastarasis, nes sigmoido išvestis prisotintame regione tampa labai maža, todėl svorių atnaujinimai beveik išnyksta. Tai vadinama išnykimo gradiento problema.

Tinkle ReLu sluoksnis dedamas po kiekvieno giliojo/pirmojo (convulutional) ir visiškai prijungto sluoksnio.

Kita problema, kurią ši architektūra išsprendė, buvo sumažintas pernelyg didelis perpildymas, panaudojant „Dropout“ sluoksnį po kiekvieno pilnai sujungto sluoksnio. Pašalinimojo sluoksnio (Dropout layer) tikimybė yra (p), susijusi su ja ir yra taikoma kiekviename atsako žemėlapio neurone atskirai. Jis atsitiktinai išjungia aktyvavimą su tikimybe p, kaip matyti 31 paveikslėlyje.

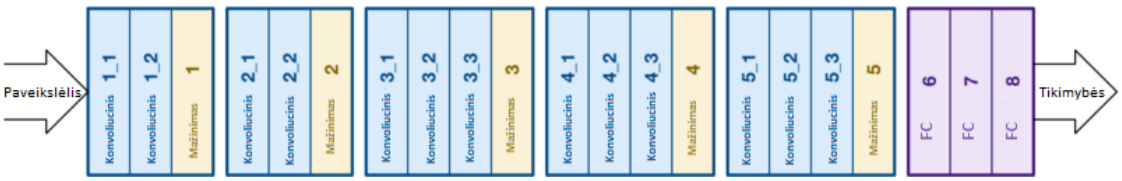


31 pav. „Drop out“ sluoksnio pritaikymas

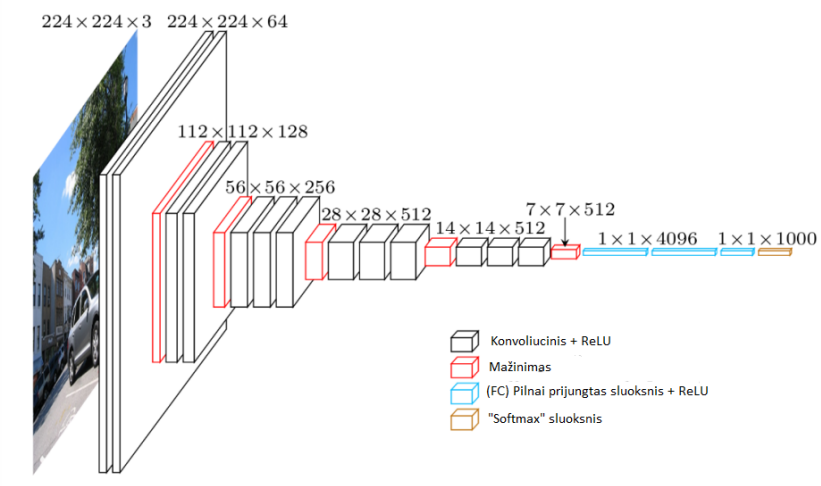
Pašalinimo idėja yra panaši į modelių ansamblius. Dėl pašalinamojo sluoksnio skirtingi išjungtų neuronų rinkiniai reprezentuoja skirtingą architektūrą ir visos šios skirtingos architektūros yra mokomos lygiagrečiai su kiekvienam pogrupiui suteiktu svoriu, o svorių apibendrinimas yra vienas. N neuronams, prijungtam prie pašalinamojo sluoksnio, sudarytų subsetų architektūrų skaičius yra 2^n. Taigi tai reiškia, kad šių modelių ansamblių vidurkis yra prognozavimas. Tai suteikia struktūrizuotą modelio reguliavimą, kuris padeda išvengti pernelyg didelio perpildymo. Dar vienas „DropOut“ sluoksnio privalumas yra tas, kad neuronai yra atsitiktinai parinkti, jie linkę vengti tarpusavyje prisitaikyti, tokiu būdu sudarant jiems galimybę kurti prasmingas savybes, nepriklausomas nuo kitų. [16]

1.3.2. VGG-16 ir VGG-19

VGG-16 ir VGG-19 yra konvoliuciniai neuroniniai tinklai, kurie yra apmokyti daugiau nei milijonu vaizdų iš „ImageNet“ duomenų bazės. Tinklai atitinkamai yra 16 ir 19 sluoksnių ir gali klasifikuoti vaizdus į 1000 objektų kategorijas. [17]



32 pav. VGG-16 architektūros schema



33 pav. VGG-16 architektūros schema, 3D

Šios gilaus neuroninio tinklo architektūros yra iš VGG grupės, Oksfordo. Jos geresnės už „AlexNet“, tuo , kad pakeičia didelius kernelius (atitinkamai 11 ir 5 pirmame ir antrame konvoliuciniame sluoksnyje) vienu metu po kelis 3X3 kernelius. Naudojant tam tikrą imtuvinį lauką (efektyvų įvesties įvaizdžio plotą, kuriam priklauso išėjimas), didesnis skaičius mažesnių kernelių yra geriau už didesnio dydžio kernelį, nes keli nelinijiniai sluoksniai padidina tinklo gylį, kuris leidžia išmokti daugiau sudėtingų funkcijų, ir tai taip pat mažesnėmis sąnaudomis [18]

VGG yra keli blokai, turintys tą patį filtro dydį, kad būtų išgauti sudėtingesni ir tipiškesni bruožai. Ši blokų / modulių samprata tapo plačiai naudojama po VGG atsiradimo.

Po VGG konvoliucių sluoksnių seka trys visiškai sujungti sluoksniai. Tinklo plotis prasideda nuo mažos 64 vertės, o po kiekvieno mėginių ėmimo / kaupimo sluoksnio padidėja 2 kartus. „VGG“ pasiekia 92,3% tikslumą..

1.4. Kokybės įverčiai

mAP (mean average precision) yra matas, skirtas kiekybiškai įvertinti objektų detektoriaus tikslumą, pvz., naudojant giliuosius neuroninius tinklus.

Visi kokybės matai yra apskaičiuojami iš 4 pagrindinių verčių: FN, FP, TN, TP. Todėl būtent šiuos matus svarbu išgauti tuo pačiu būdu, o toliau viskas apskaičiuojama pagal bendras formules.[19]

TP – (*angl*. true positive) – nurodo tikrąsias reikšmės, kuriose yra defektas.

FP – (*angl.* false positve) – nurodo vietas, kurias tinklas klasifikuoja kaip defektą, tačiau jo ten nėra.

TN – (*angl.* true negative) – nurodo vietas kuriose tinklas nurodė, kad defekto nėra, tačiau jis yra.

FN – (*angl.* false negative) – nusako vietas kuriose tinklas teisingai nurodė, jog defekto nėra.

„Precision“ nusako prognozės tikslumą. t. y. procentas teisingų teigiamų prognozių.

„Recall“ nusako kaip gerai randamos visos teisingos teigiamos prognozės. [20]

$$TP = \text{True positive}$$
$$TN = \text{True negative}$$
$$FP = \text{False positive}$$
$$FN = \text{False negative}$$

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

34 pav. Matematinės matavimo matų reikšmės

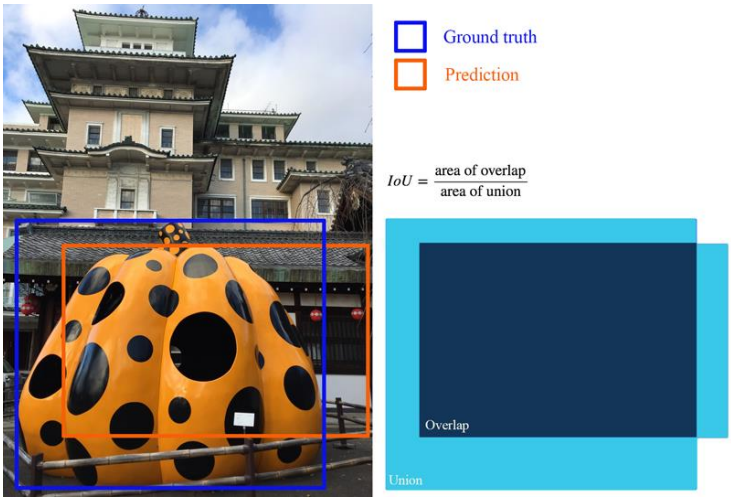
Reikia paminėti, jog tam tikrais atveju tikslumo matai skaičiuojami kiek kitaip, nes atsiranda neapibrėžti kraštiniai atvejai kai tikrieji teigiami (TP) yra 0, nes tai yra tiek Precision tiek Recall vardiklis. Tada reikia įvertinti jog: [21]

- Recall = 1, kai FN = 0, nes buvo atrasta 100% TP.
- Precision = 1, kai FP = 0, nes nebuvo jokių klaidingų rezultatų.

IoU (Intersection over union) nusako, kiek sutampa 2 regionai, t.y tikrosios objekto ribos su detektoriaus prognozuotomis ribomis.

$$IoU = \frac{\text{area of overlap}}{\text{area of union}}$$

35 pav. IoU matematinė išraiška



36 pav. IoU grafinis atvaizdavimas

F1 (taip pat F-balas arba F-matas) yra bandymo tikslumo matas. Jis įvertina prognozės tikslumą ir kaip gerai randamos teigiamos prognozės. Šis skaitinis matas plačiai naudojamas jei reikia ieškoti pusiausvyros tarp tikslumo ir atkūrimo. Taip pat F1 matas veiksmingai nurodo tikruosius teigiamus teigiamo ir tikrojo teigiamo teiginio aritmetinį vidurkį, ty apskaičiuotą normą, normalizuotą pagal idealizuotą vertę ir išreikštą tokia forma, jis statistikoje yra žinomas kaip susitarimo dalis, nes yra taikomas tik tam tikruose uždaviniuose.

$$F_1 = \left(\frac{\mathrm{recall}^{-1} + \mathrm{precision}^{-1}}{2} \right)^{-1} = 2 \cdot \frac{\mathrm{precision} \cdot \mathrm{recall}}{\mathrm{precision} + \mathrm{recall}}.$$

37 pav. F1 kokybės mato matematinė išraiška

MCC yra tikslo ir prognozės koreliacijos koeficientas. Jis paprastai svyruoja nuo -1 iki +1. -1, kai yra puikių nesutarimų tarp aktualių ir prognozavimo reikšmių, 1 kai yra puikus susitarimas tarp aktualių reikšmių ir prognozių. 0, kai prognozė taip pat gali būti atsitiktinė aktualių reikšmių atžvilgiu. Kadangi tai apima visų keturių painiavos matricos vertybių vertes, MCC laikomas subalansuota priemone kokybei įvertinti. [22]

Apsvarstykime atvejį, kai teigiamo arba neigiamo atvejų skaičius yra per mažas, o klasifikatorius grąžina arba TP, arba TN kaip 0, tada vidutinis TPR ir TNR grąžins rezultatą be jokios krypties. MCC apima visų keturių painiavos matricos vertybių vertę ir yra subalansuota priemonė, todėl grąžins vertę su kryptimi (+ ve ir -ve).

Trumpai apie MCC koreliacijos koeficiento privalumus:

- MCC galima apskaičiuoti naudojant painiavos matricą.

- MCC metrika apskaičiuojama naudojant keturis kiekius (TP, TN, FP ir FN), kurie suteikia geresnę klasifikavimo algoritmų veikimo santrauką.
- MCC nėra apibrėžta, jei bet kuris iš TP + FN, TP + FP, TN + FP arba TN + FN kiekių yra nulis.
- MCC atsižvelgia į intervalo [−1, 1] reikšmes, o 1 rodo pilną susitarimą, −1 pilną nesutikimą ir 0, rodantį, kad prognozė buvo nekoreliuota su žeme.

$$\mathrm{MCC} = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

38 pav. F1 kokybės mato matematinė išraiška

1.5. Teorinė tikimybė eksperimentų rezultatams

Atlikus literatūros analizę galima tikėtis aukštų rezultatų iš R-CNN, kurių tikslumas gali siekti net 80-90%, tačiau šis būdas nurodo netikslią defekto vietą, o tik regioną kuriame defektas yra. Semantinė segmentacija pateikia atsakymą pikselių lygyje ir yra gana tiksli, tačiau skirtą dideliems pikselių regionams, o šio darbo metu bus tiriame kelio trūkiai, todėl rezultatai gali būti ne itin aukšti. Sukūrus U-Net tinklą, kuris yra tikslus pikselių lygyje, galima tikėtis geriausio tikslumo rezultato iš visų trijų būdų, kadangi šis puikiai dirba ir su mažais pikselių kiekiais.

1.6. Naudojama įranga

Tyrimo ir eksperimentų metu naudojama įranga:

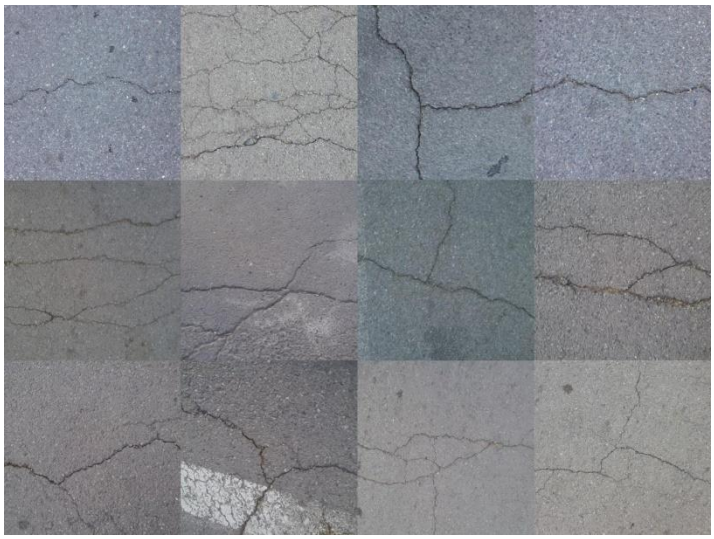
- Procesorius: Intel core i5 2500S, 2.7Ghz
- Vaizdo plokštė: NVIDIA GeForce GTX 1060 3GB
- Operatyvioji atmintis: 8GB DDR3
- Operacinė sistema: Windwos 7 Ultimate 64-bit SP1
- Programinis paketas: MATLAB

2. Duomenų bazė

Kadangi darbas yra apie automatizuotą automobilių kelių defektų aptikimą, todėl ir duomenų bazė turėtų būti atitinkama. Kurti savo duomenų bazę reikia daug laiko ir resursų, todėl nuspręsta pasirinkti jau esamą duomenų bazę iš interneto.

Šiuo metų internete prieinama nemažai duomenų bazių kurias galima panaudoti įvairioms užduotims įgyvendinti. Buvo pasirinkta „CrackForest Dataset“ duomenų bazė, kuri puikiai tinka reikiamai užduočiai spręsti. [23]

„CrackForest Dataset“ yra anotuota kelių kreko vaizdų duomenų bazė, kuri gali bendrai atspindėti miesto kelių būklę.

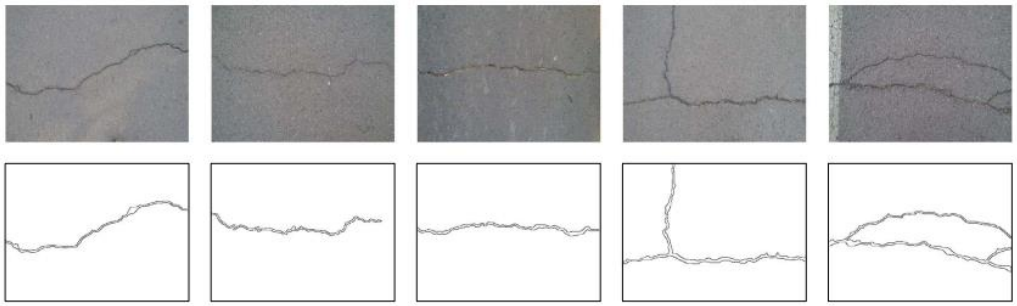


39 pav. „CrackForest Dataset“ duomenų bazėje esantys vaizdai

Šią duomenų bazę sudaro 155 paveikslėliai su įvairiais kelio defektais arbe be jų, kas be ko didžioji dalis su defektais. Tačiau ne kiekvienas paveikslėlis yra tinkamas, kadangi tik 118 paveikslėlių yra kokybiškai išfiltruoti, atlikta segmentacija ir gauti „groundtruth“.

Kitaip sakant tik 118 paveikslėlių turi žymas (labels), t.y. nurodymai kuriose vietose atvaizde yra įskilimas (defektas) ir kuriose nėra, tai pateikta matricose.

Darbui atlikti naudosime tik tuos 118 paveikslėlių, kiti nebus naudojami, dėl nepakankamo apdirbimo.



40 pav. „CrackForest Dataset“ duomenų bazės vaizdai ir žymos

2.1. Duomenų bazės trūkumai

Peržiūrėjus visas duomenų bazėje esančias nuotraukas ir jų žymas, buvo pastebėta, jog duomenų bazė nėra itin kokybiška, dėl kelių pagrindinių priežasčių:

- Kai kuriose nuotraukose ne visi kelio defektai yra sužymėti, t.y. nevisose nuotraukose žymos yra pilnos. Yra nuotraukų kuriose matomas aiškus kelio defektas, tačiau jis nėra pažymėtas.
- Nuotraukų žymėjimas yra labai nepastovus, žiūrint pikselių mastu. Vienose nuotraukose defektas pažymėtas gan tiksliai, neįtraukiant asfalto pikselių, o kitose defektas pažymėtas atmestinai, į žymas įtraukiant ir asfalto pikselius.
- Kadangi atlikti kokybiškam gilaus neuroninio tinklo mokymui 118 gali būti mažokai, tie patys paveikslėliai buvo panaudoti taip, jog neuroninis tinklas jį matytų kaip visiškai kitą atvaizdą. Tai buvo įgyvendinta paveikslėlius sukant 90, 180 ir 270 laipsnių variacijomis, bei naudojant veidrodžio funkciją, taip duomenų bazę padidinant iš 118 atvaizdų, į 944 atvaizdus.

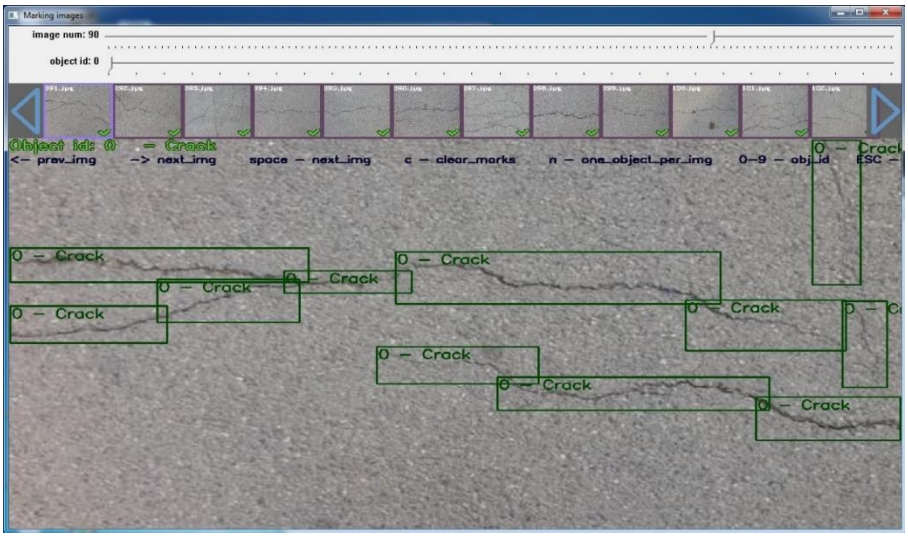
Dėl šių duomenų bazės trūkumų prastės kokybė ir tikslumas. Siekiant išlaikyti eksperimentų kokybę kuo aukštesnė, duomenų bazė buvo padidinta.

Trūkumai galėjo atsirasti, kai kuriant duomenų bazę dirbo ne vienas ekspertas, o keletas, kurie skirtingai juos ir sužymėjo.

2.2. Regionų žymėjimas

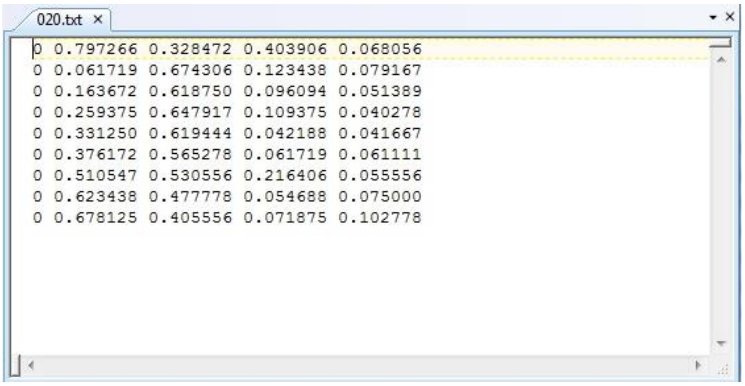
Defektų žymėjimo automatizuoti negalima, kadangi tuo atveju visi sužymėti ROI bus vienodo dydžio, ir tinklas išmoks rasti defektus tik tuo esamu dydžiu, kas nėra tikslinga. Todėl defektai atvaizduose sužymėti rankiniu būdu, taip siekiant išlaikyti atsitiktinį sužymėjimo dydį, kuris pageriną tinklo mokymosi kokybę ir pateikia geresnius rezultatus.

Šiai užduočiai pasirinkta *„YoloMark“* programinis paketas. [24]



41 pav. Defektų žymėjimas naudojant *„YoloMark‘* programinį paketą

Kadangi užduotis yra defektų aptikimo, o ne klasifikavimo, todėl klasė yra tik viena: defektas. Ant kiekvieno paveikslėlio ranka sužymima vieta kurioje yra defektas (trūkis), o likusi atvaizdo teritorija priima kaip fonas, kuris yra nereikšmingas.



42 pav. Santykinės koordinatės gautos sužymėjus defektus rankiniu būdu

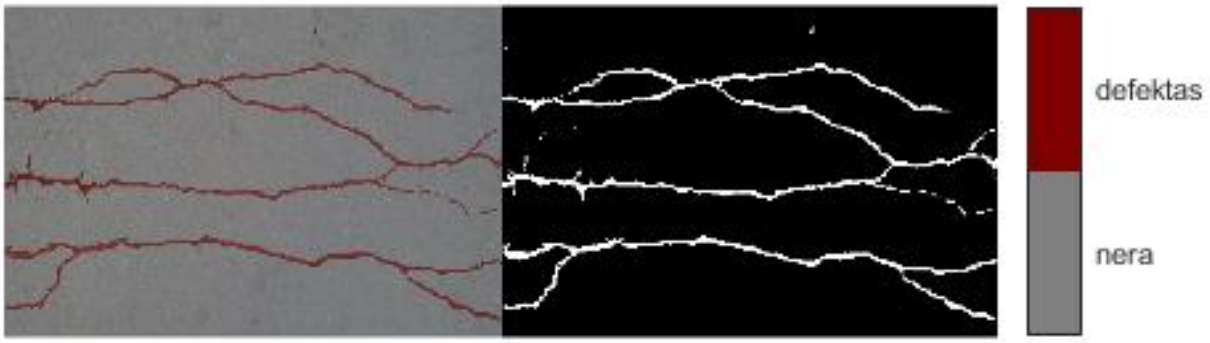
Kiekvienas sužymėjimas yra santykinių koordinačių nuėmimas, kad vėliau *Matlab* programiniame pakete, būtų galima nurodyti giliajam neuroniniam tinklui, kur yra defektas.

2.3. Defektų žymėjimas pikselių lygmenyje

Patikrinus duomenų bazėje esančius paveikslėlius ir žymas, pastebėta jog 7 iš 118 yra netinkami naudojimui, todėl buvo pašalinti. Tačiau atlikti kokybiškam gilaus neuroninio tinklo mokymui, kaip

ir naudojant R-CNN tiek paveikslėlių gali būti mažokai todėl duomenų bazė buvo padidinta tuo pačiu būdu. Paveikslėliai sukami 90, 180 ir 270 laipsnių variacijomis, tačiau šiuo atveju nenaudojant veidrodžio efektą, kadangi žymoms nėra galimybės naudoti veidrodžio efektą. Duomenų bazė padidinta nuo 111 iki 444 paveikslėlių su žymomis.

Ruošiant duomenis mokymo procesui, visos žymos yra pakraunamos ant originalių paveikslėlių ir kartu patikrinama ar žymos atitinka paveikslėlius, po duomenų bazės praplėtimo. Toliau pateiktas pavyzdys gauto rezultato.



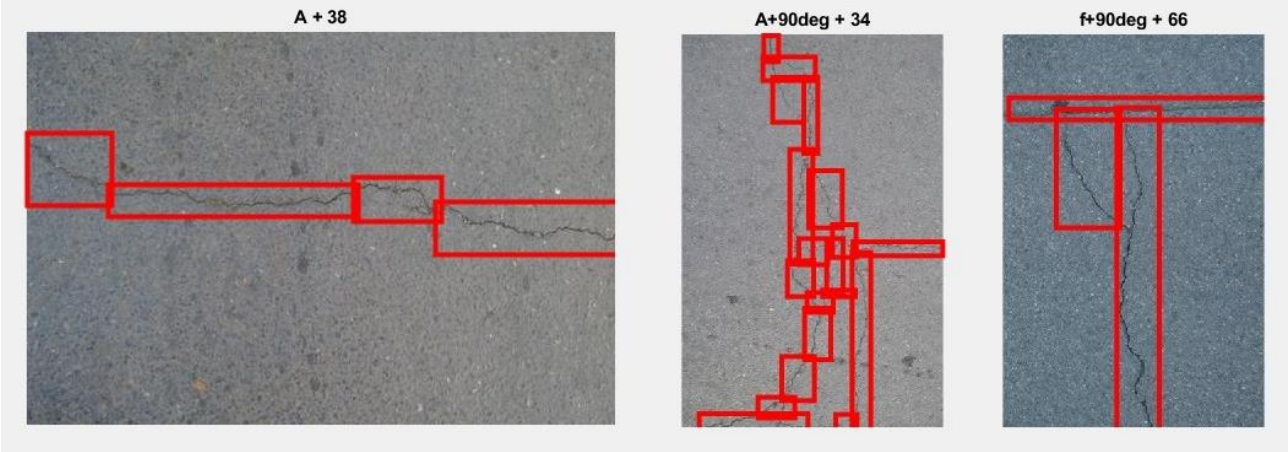
43 pav. Žymos ir originalių paveikslėlių sujungimas

3. Eksperimentinė dalis

3.1. R-CNN tyrimas

Naudojant regioninius giliuosius neuroninius tinklus svarbu tinkamai sužymėti ROI ribas taip, kad į ribinius kvadratėlius patektų kuo mažiau fono, ir kuo daugiau pačio defekto. Taip pat reiktų defektus žymėti kuo įvairiau, kad tinklas turėtų didesnį pasirinkimą ribiniams langeliams brėžti. Žymėjimas buvo atliktas rankiniu būdu. Sužymėjus visus 118 paveikslėlių, šie buvo susukiojami 90, 180 ir 270 laipsnių, bei apsukti veidrodžio efektu, taip padidinus duomenų bazę iki 944 paveikslėlių.

Sužymėti regionai kuriuose yra kelio trūkiai, taip pat buvo atitinkamai susukinėti ir užkrauti į žymas. Buvo atlikta vizualinė patikra programiniame pakete *MATLAB,* kad įvertinti ar sužymėti regionai sutampa su realių trūkių vietomis. Patikros eiga pavaizduota žemiau 44 paveikslėlyje.



44 pav. Defektų sužymėjimas regionais, rankini būdu

Dėl netikslumų žymint defektų regionus, kurie atsirado naudojant „*YoloMark*‘ programinį paketą, prieš užkraunant duomenis mokymui, buvo patikrintos ir pakoreguotos ribinių langelių koordinatės, kadangi šios turi tilpti į 320x480 dimensijos ribas.

Įsitikinus, jog defektai sužymėti teisingai, duomenų bazė pakraunama, ir atsitiktiniu būdu nuotraukos paskirstomos į skirtas mokymui – 70% ir į skirtas testams – 30%.

Mokymas atliekamas su pasirinktais trimis skirtingais, iš anksto apmokytais tinklais: „AlexNet“, „VGG-16“ ir „VGG-19“. Kiekvienas iš tinklų yra naudojamas su trimis skirtingomis parametrų konfigūracijomis: [25]

- Iteracijų skaičius – 4; ‚LearnRateDropFactor‘ – 0,1

- Iteracijų skaičius – 10; ‚LearnRateDropFactor‘ – 0,85
- Iteracijų skaičius – 20; ‚LearnRateDropFactor‘ – 0,95

Atliekant eksperimentus gaunami ir pateikiami visi matavimo matai, tačiau spręsti kuris variantas yra tiksliausias vertėtų tik pagal F1 ir MCC. Pirmenybė teikiama MCC, kadangi MCC apima visų keturių painiavos matricos vertybių vertę ir yra subalansuota priemonė, kuri visapusiškai nusako tinklo tikslumą.

Taip pat, apmokius tinklą, detektorius pateikdamas savo atsakymus kartais gali išeiti už nuotraukos ribų, todėl prieš skaičiuojant tinklo tikslumo vertes yra patikrinamos detektoriaus išduodamos ribinių langelių koordinatės, ir šios yra sutalpinamos į nuotraukos (320x480) ribas, tam, kad tikslumo skaičiavimai būtų atlikti kokybiškai.

Atlikus tris eksperimentus su „AlexNet“ iš anksto apmokytu tinklu ir visomis tinklo konfigūracijomis gauti rezultatai, pateikti pirmoje lentelėje. Iš gautų duomenų matyti, jog didesnis iteracijų skaičius lėmė geresnį tinklo apmokymą. Tačiau kokybės skirtumas yra labai nežymus – iki 3%. Geriausios tinklo konfigūracijos kokybės įverčiai yra: F1 – 0.7057, o MCC – 0.674. Kokybiškiau apmokius tinklą, detektorius tiksliau aptinka kelio trūkius, deja pareikalaudamas didesnių laiko kaštų.

Atlikus tris eksperimentus su „VGG16“ iš anksto apmokytu tinklu ir visomis tinklo konfigūracijomis gauti rezultatai, pateikti pirmoje lentelėje. Iš gautų duomenų matyti, jog ir šiuo atveju didesnis iteracijų skaičius lėmė geresnį tinklo apmokymą. Kokybės skirtumas yra labai nežymus – iki 5%, tačiau kiek didesnis nei „AlexNet“ atveju. Geriausios tinklo konfigūracijos kokybės įverčiai yra: F1 – 0.766, o MCC – 0.736. Kokybiškiau apmokius tinklą, detektorius tiksliau aptinka kelio trūkius, deja pareikalaudamas didesnių laiko kaštų.

Taip pat lyginant „AlexNet“ ir „VGG-16“ geriausias konfigūracijas, matyti, jog „VGG-16“ tiksliau randa kelio trūkius. Skirtumas tarp tinklų tikslumo – iki 6% Tai lėmė sudėtingesnė tinklo architektūra, kadangi „AlexNet“ tinklas turi tik 8 sluoksnius, o „VGG-16“ turi 16 sluoksnių architektūrą.

Atlikus tris eksperimentus su „VGG-19“ iš anksto apmokytu tinklu ir visomis tinklo konfigūracijomis gauti rezultatai, pateikti pirmoje lentelėje. Iš gautų duomenų matyti, jog didesnis iteracijų skaičius lėmė geresnį tinklo apmokymą. Tačiau kokybės skirtumas yra labai nežymus – iki 3%. Geriausios tinklo konfigūracijos kokybės įverčiai yra: F1 – 0.7675, o MCC – 0.7385. Kokybiškiau apmokius tinklą, detektorius tiksliau aptinka kelio trūkius, deja pareikalauja didesnių laiko kaštų..

Palyginus visų trijų iš anksto apmokytų tinklų rezultatus, atsižvelgus į tinklo kokybės matus, galima teigti, jog geriausiai uždavinį išsprendžia „VGG-19“ iš anksto apmokytas tinklas. Tarp geriausių „VGG-19“ ir „AlexNet“ tinklų kokybės įverčių skirtumas lygus tik 6%. Lyginant geriausius „VGG-19“ ir „VGG-16“ tinklų kokybės įverčius, skirtumas siekia tik 0.2%. Galima teigti, jog naudojant vienodų architektūrų tinklus, keli papildomi sluoksniai tinklo tikslumo smarkiai neįtakoja.

lentelė 1 R-CNN eksperimentų rezultatai

		Iteracijų skaičius	Taiklumas	Tikslumas	Atkūrimas	F1	IoU	MCC
Faster R-CNN	ALEXNET	4	0,8924	0,5739	0,8884	0,6817	0,5272	0,6498
		10	0,8816	0,5564	0,9076	0,6762	0,5198	0,6418
		20	**0,9049**	**0,6025**	**0,8877**	**0,7057**	**0,554**	**0,674**
	VGG16	4	0,9102	0,6298	0,9042	0,7279	0,579	0,6997
		10	0,9174	0,6523	0,8969	0,7417	0,5965	0,7174
		20	**0,9291**	**0,702**	**0,8695**	**0,766**	**0,6265**	**0,736**
	VGG19	4	0,8887	0,5701	0,9294	0,6916	0,5394	0,6651
		10	0,9165	0,6305	0,9119	0,7338	0,5876	0,7089
		20	**0,9324**	**0,7066**	**0,8641**	**0,7675**	**0,6291**	**0,7385**

Kalbant apie mokymo laikus, tai nėra svarbus rodiklis, kadangi šis laikas reikalingas tik vieną karta – apmokant tinklą. Vėliau tinklas atpažįsta defektus greitai, šiuo atveju R-CNN tai gali atlikti per maždaug sekundę. Todėl verta pabrėžti, jog regioninis būdas defektams atpažinti negalėtų būti naudojamas esamuoju laiku, kadangi viena sekundė užduočiai atlikti yra per ilgas laiko tarpas, ypač naudojant ne nuotraukas , o filmuotą vaizdo medžiagą.

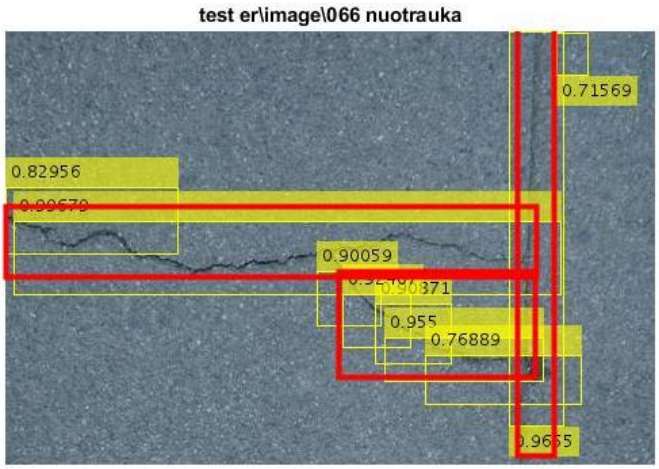
lentelė 2 R-CNN geriausių tinklų mokymo laikai

		Iteracijų skaičius	Mokymo laikas
Faster R-CNN	ALEXNET	4	1:20
		10	2:59
		20	6:02
	VGG16	4	2:01
		10	3:49
		20	7:23
	VGG19	4	2:15
		10	4:08
		20	7:49

Vertinant tinklų veikimo kokybę, reiktų atsižvelgti į tai, jog tinklas išmoksta rasti defektus tokiomis ribinių langelių formomis, kokiomis jis ir apmokomas. Todėl pradžioje buvo akcentuota,

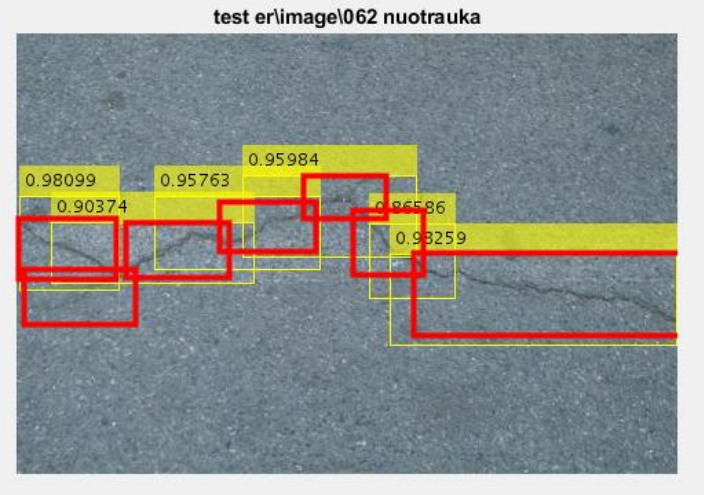
jog rankinis ir įvairus sužymėjimas įtakoja tinklų rezultatus, kadangi kuo didesne įvairove buvo mokamas tinklas, tuo daugiau įvairių formų ribinių langelių šis gali panaudoti, nurodydamas kur randa kelio trūkį. Žiūrint iš kitos pusės, kai tinklas panaudoja daugiau ribinių langelių nei buvo galima, tai buvo dėl tos pačios priežasties, jog šį apmokant, pritrūko ribinių langelių įvairovės, kad pritaikytų tinkamiausią iš išmoktų. Pirmuoju atveju, tinklas kai kuriais atvejais gali sužymėti net geriau už ekspertą, kadangi panaudodamas mažesnius langelius gali apriboti sudėtingą kelio trūkį, paimdamas mažiau fono. O ekspertas sudėtingą kelio trūkį, taupydamas laiką, gali apibrėžti vienu dideliu ribiniu langeliu, ir taip apibrėžti gerokai daugiau fono. Antruoju atveju, kelių ribinių langelių naudojimas pavyzdžiui vienam dideliam, tiesiam trūkiui gali sumažinti tinklo tikslumo kokybės įvertį, nes bus pažymima daugiau fono, nei yra apibrėžta žymose. Todėl didesnė įvairovė ribinių langelių turėtų sumažinti šią problemą, ir padidinto tinklo tikslumą, bei kokybės įverčius.

Pavyzdžiui kur kelio trūkis nėra itin paprastas, tačiau kurį galima pažymėti vienu ribiniu langeliu, tinklas aptinka ir pažymi jį keliais skirtingų dydžių ribiniais langeliais. Pavyzdys pateiktas 45 paveikslėlyje.



45 pav. Faster R-CNN eksperimento pavyzdys nr. 1

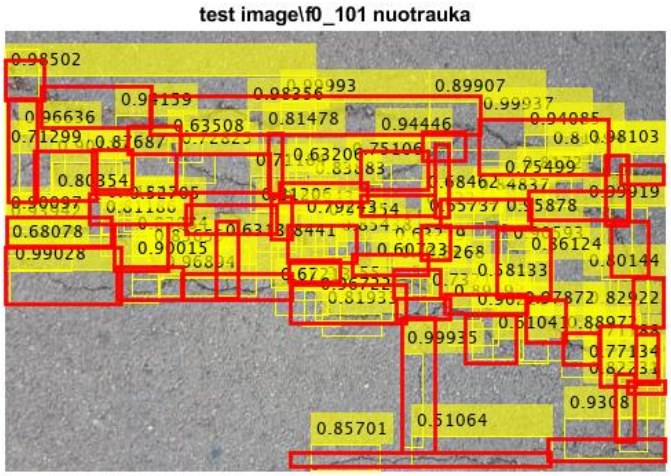
Taip pat pastebėta, jog apmokytas tinklas žymėdamas aptiktus kelio trūkius, net ir pataikydamas dažnai į idealias vietas lyginant su esamomis trūkių žymomis, dažniausiai pasirenka didesnius ribinius langelius ir pažymį didesnį fono plotą, nei sužymėjo ekspertas rankiniu būdu. Pavyzdys pateiktas 46 paveikslėlyje.



46 pav. Faster R-CNN eksperimento pavyzdys nr. 2

Tinklui pateikus sudėtingą nuotrauką, kurioje yra daug kelio trūkių, ir jų pasiskirstymas yra labai išsibarstęs, bei patys trūkiai eina įvairiomis kryptimis, vizualiai akimis sunku vertinti ar detektorius defektus sužymėjo gerai ir ar pasirinko tinkamo dydžio ribinius langelius. Todėl visapusiškai įvertinti tinklo kokybę galima tik iš tinklo tikslumo kokybės įverčių.

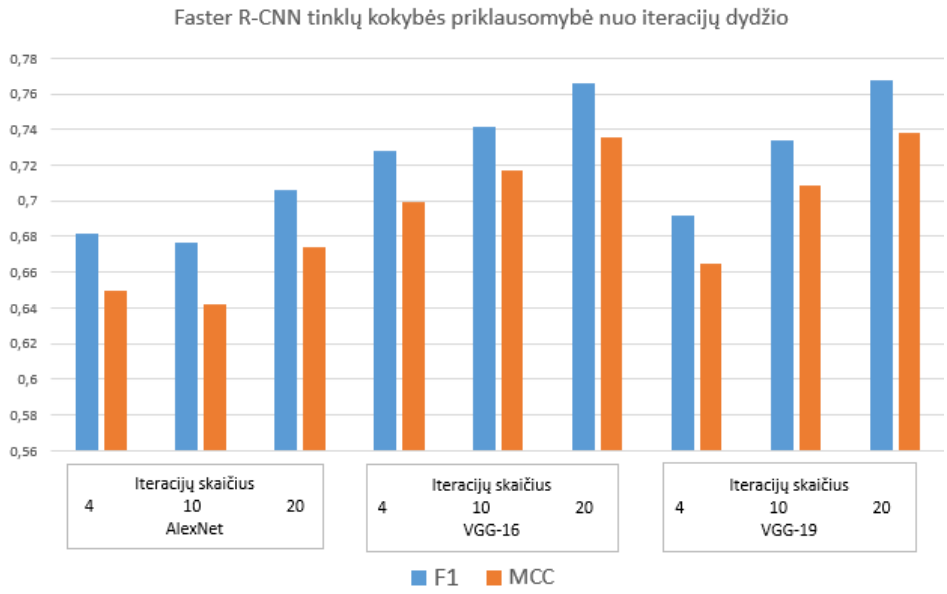
Vizualiniai rezultatai kaip tinklas susidorojo su sudėtinga nuotrauka pateikti 47 paveikslėlyje.



47 pav. Faster R-CNN eksperimento pavyzdys nr. 3

Atvaizdavus pirmoje lentelėje pateiktus duomenis grafiškai, galima lengvai akimi pastebėti, jog tinklų kokybė priklauso nuo iteracijų skaičiaus. Grafikas pateiktas 48 paveikslėlyje. Didinant iteracijų skaičių atitinkamai didėja ir tinklo kokybė, kadangi tinklas gauna daugiau duomenų mokymo procesui, tos pačios nuotraukos yra persimokomos po kelis kartus, ir tinklas geriau prisitaiko atpažinti kelio trūkiams. Tačiau reiktų atkreipti dėmesį į tai, jog padidinus iteracijų

skaičių per daug, tinklo kokybė ims kristi, kadangi gilusis neuroninis tinklas pradės įsiminti nuotraukas, kurios bus paduodamos mokymui vis per naujo ir per naujo, ir nebesugebės atpažinti didesnės įvairovės. Todėl norint pasiekti geresnį tinklo tikslumą, tačiau paržengus iteracijų skaičiaus dydžio ribas, kai tinklas jau ima įsiminti nuotraukas, teisingas sprendimas būtų šiek tiek sumažinti iteracijų skaičių ir geriau didinti nuotraukų kiekį, paduodama mokymui. Taip bus didinama įvairovė tiek pačių defektų, tiek ribinių langeliu, ir tinklas turėtų būti dinamiškesnis, ir geriau atpažinti defektus įvairiose nuotraukose.



48 pav. Faster R-CNN tinklų kokybės priklausomybė nuo iteracijų dydžio

Nors ir naudojant Faster R-CNN būdą tikslumo rodikliai gaunami aukšti, reiktų atkreipti dėmesį, jog šiuo būdu tinklas nurodo regioną kuriame yra kelio trūkis, o ne tikslią trūkio vietą. Geriausi gauti kokybės įverčiai su „VGG-19“ tinklu, t.y. kai F1 – 0.7675, o MCC – 0.7385. Todėl norint nurodyti tikslią kelio trūkio vietą, su regioniniu tinklu praktiškai nėra įmanoma.

Išvados:

1. Geriausi gauti kokybės įverčiai su „VGG-19“ tinklu, t.y. kai F1 – 0.7675, o MCC – 0.7385
2. Tinklo regionų, su kelio trūkiais, žymėjimas labai priklauso nuo duomenų kuriais šis yra apmokomas
3. Didesnė duomenų bazė lemia geresnį tinklo apmokymą
4. Iteracijų skaičius įtakoja tinklo tikslumą, t.y. daugiau iteracijų lemia aukštesnę tinklo kokybę
5. Regioninis tinklas netinkamas naudoti realiojo laiko užduotims įgyvendinti
6. Regioninis tinklas nurodo tik regioną kuriame yra kelio trūkis, bet ne tikslią trūkio vietą

3.2. Semantinės segmentacijos tyrimas

Atliekant semantinę segmentaciją pikselių lygyje, kaip tikrosios defektų žymos yra laikomos duomenų bazėje jau sužymėtos žymos. Kadangi jos nėra visiškai tikslios, reiktų atkreipti dėmesį, kad tai įtakos galutinius rezultatus.

Semantinės segmentacijos atveju, uždaviniui spręsti galime naudoti tik „VGG-16“ ir „VGG-19“ iš anksto apmokytus tinklus, kadangi programinis paketas *MATLAB* negali naudoti „AlexNet“, dėl uždavinio specifikos, t.y. „AlexNet“ netinkamas defektų ieškojimui pikselių lygyje.

Nustatomi keli pagrindiniai parametrai mokymui:

- Iteracijų skaičius – 5
- ‚LearnRateDropFactor‘ – 0.85
- ‚LearnRateDropPeriod‘ – 1
- ‚InitialLearnRate‘ – 0.01

Būtina paminėti, kad didelę reikšmę tinklo tikslumui turi klasių svoriai. *MATLAB* programinis paketas automatiškai gali aptikti pirminius klasių svorius, tačiau siekiant geresnių rezultatų, juos vertėtų koreguoti. Pirminiai klasių svoriai kuriuos pasiūlė *MATLAB* yra [0.5117 : 33].

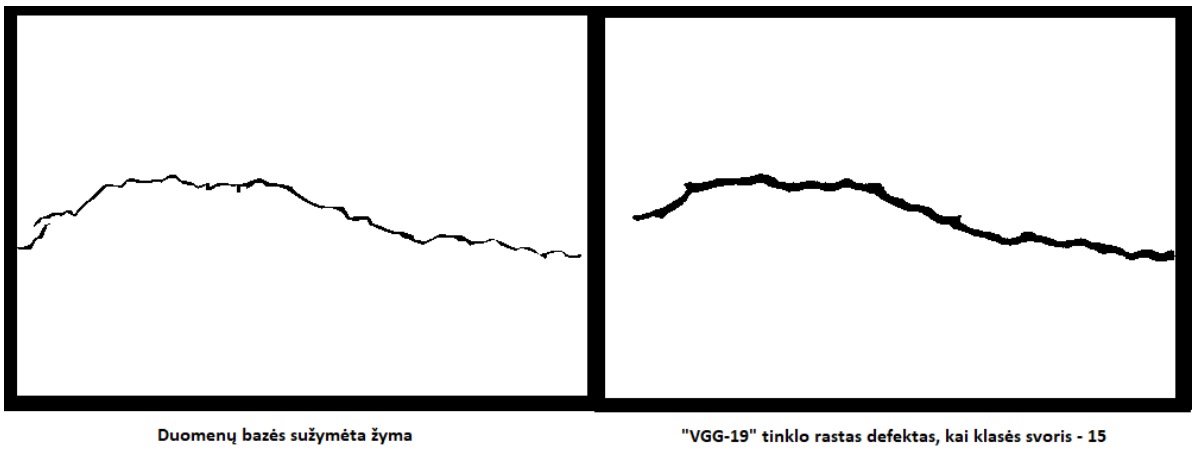
Sudaromi klasių svorio keitimo intervalai, su kuriais bus atliekami eksperimentai.

lentelė 3 Klasės svorio naudojamos variacijos

[0.5117 : 15]	[0.5117 : 24]	[0.5117 : 33]

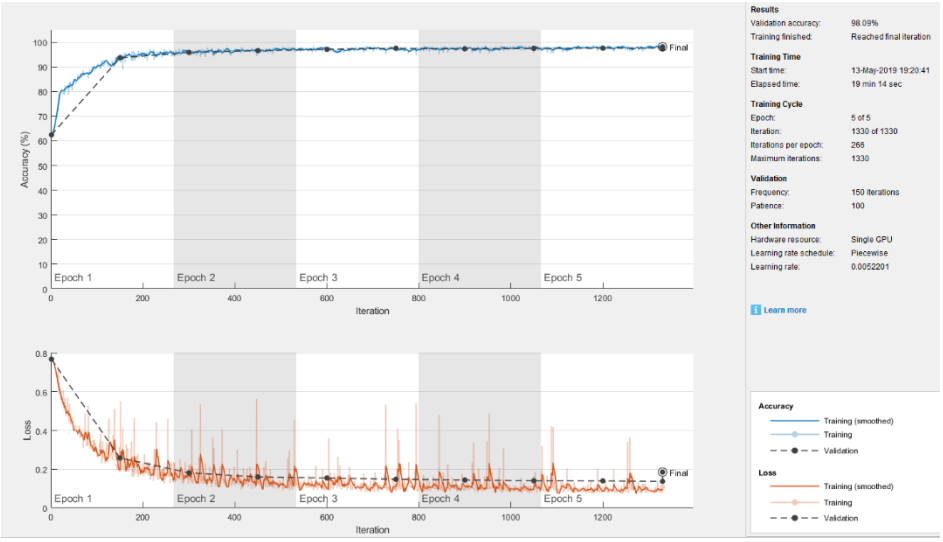
Kaip matyti iš žemiau pateiktų 49, 51 ir 53 paveikslėlių, pikselių kiekis kurį tinklas nurodo kaip rastą defektą, priklauso nuo klasių svorių. Kuo didesnis svoris, tuo daugiau pikselių aplink defektą tinklas sužymės, ir taip bus laikomas didesnis defektas. Tai gali atsiliepti tinklo kokybės įverčiams.

Iš 49 paveikslėlio matyti, jog nustačius klasės svorį – 15, tinklas randa gan panašaus pikselių kiekio defektą, lyginant su tikrąją duomenų bazės žyma. Taip pat galima pastebėti, jog tinklas aptiko ne pilną kelio trūkį. Tai galėjo atsitikti, dėl nepilno tinklo apsimokymo, arba defektas realioje nuotraukoje buvo toks nežymus, kad tinklas palaikė jį kaip foną.



49 pav. Kelio trūkio žymų pokytis, keičiantis svorio klasei

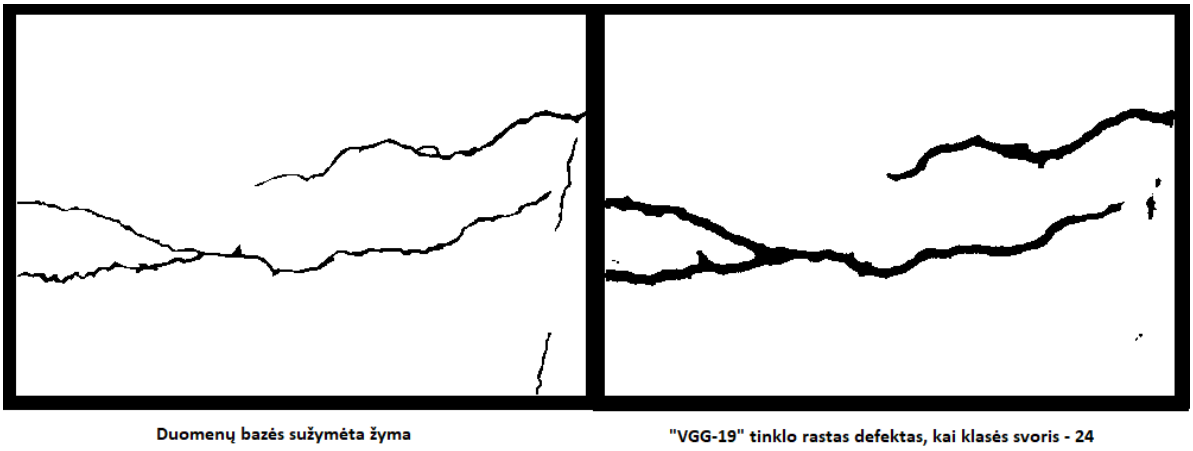
Atliekant semantinę segmentaciją, programinis paketas *MATLAB* pateikia tinklo mokymosi histogramą, kuri pavaizduota 50 paveikslėlyje. Iš histogramos matyti, kaip kito tinklo tikslumas ir duomenų praradimas, vykstant mokymosi procesui. Lyginant validacijos kreivę ir tikslumo kreivę, matyti, jog tikslumo kreivė per pirmas 150 iteracijų yra kiek aukštesnė nei tikėtasi, tačiau po validacijos kreivės sutampa beveik idealiai. Panašiai lyginant validacijos kreivę su duomenų praradimo kreive, duomenų prarandama mažiau nei tikėtasi, tačiau po validacijos, duomenų praradimo skirtumas nuo tikimosi neišnyksta, o tik sumažėja.



50 pav. Tinklo mokymo grafikas, kai klasės svoris - 15

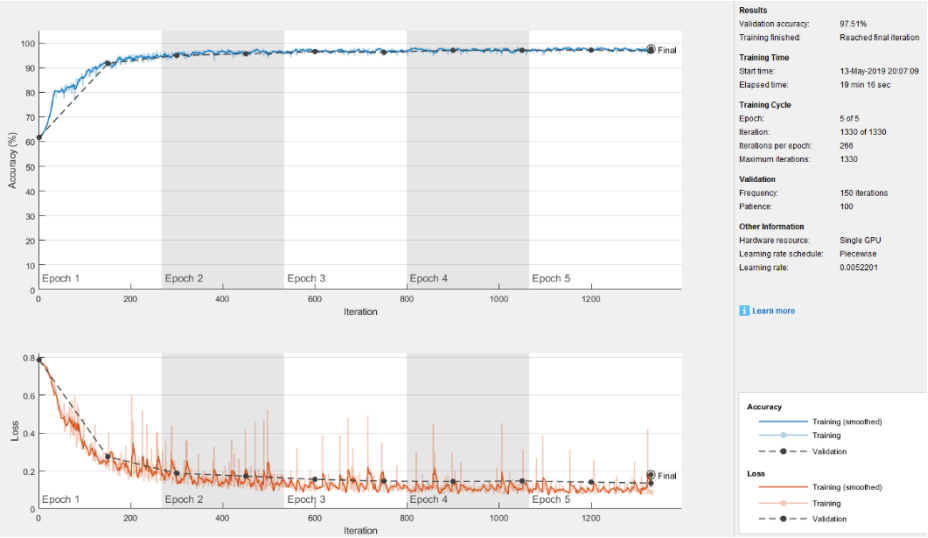
Iš 51 paveikslėlio matyti, jog padidinus klasės svorį iki 24, tinklas randa kelis kartus didesnio pikselių kiekio trūkį, lyginant su tikrąja duomenų bazės žyma. Taip pat galima pastebėti, jog vietomis tinklas randa defektą, nors tikrosiose duomenų bazės žymose jo nėra, o kitomis vietomis

tinklas trūkio išvis neranda. Tai gali atsitikti dėl dviejų priežasčių, pirmoji – duomenų bazėje nepilnai sužymėti trūkiai, antroji – dėl didelio klasės svorio, tinklas sužymi per daug pikselių. Ten kur tinklas nerado kelio trūkio, galima pastebėti, jog trūkis yra įstrižas. Didžioji dalis trūkių duomenų bazėje yra daugiau mažiau horizontalūs arba vertikalūs, įstrižų trūkių yra labai nedaug, todėl galima teigti, kad tinklui pateikus mažai tokių pavyzdžių, šis sunkiai išmoksta atpažinti įstrižus trūkius.



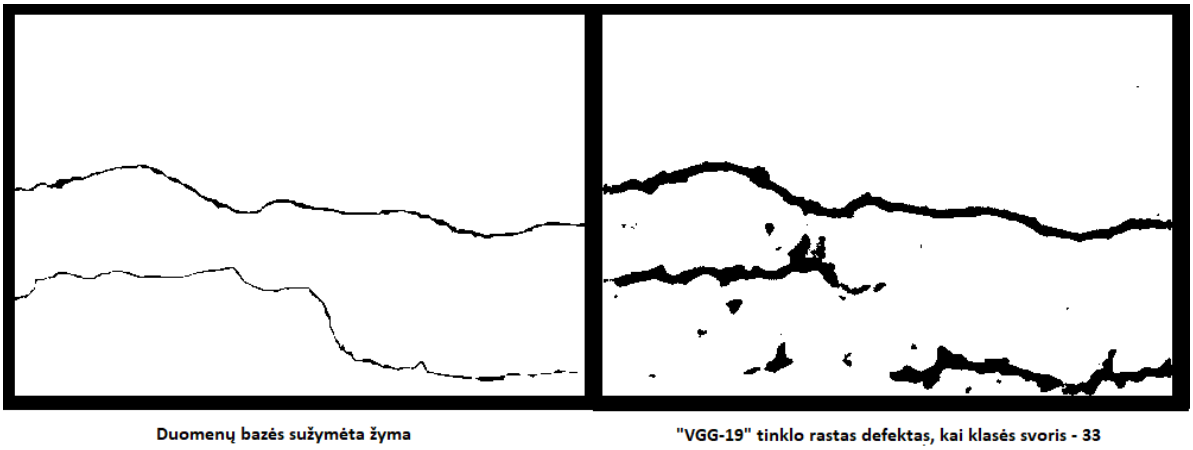
51 pav. Kelio trūkio žymų pokytis, keičiantis svorio klasei

Atliekant mokymą kai klasės svoris – 24, taip pat gaunama mokymo histograma, kuri pavaizduota 52 paveikslėlyje. Iš histogramos matyti, kaip kito tinklo tikslumas ir duomenų praradimas, vykstant mokymosi procesui. Lyginant validacijos kreivę ir tikslumo kreivę, matyti, jog tikslumo kreivė per pirmas 100 iteracijų yra kiek aukštesnė nei tikėtasi, nuo 100 iteracijų skirtumas ėmė mažėti, ir po validacijos kreivės sutampa beveik idealiai. Panaši situacija lyginant validacijos kreivę su duomenų praradimo kreive, duomenų prarandama mažiau nei tikėtasi, tačiau po validacijos, duomenų praradimo skirtumas nuo tikimosi neišnyksta, o tik sumažėja. Taip pat matyti, jog mokymo pradžioje, duomenų praradimas ties 100 iteracijų beveik sutapo su validacijos reikšme, tačiau vėl nukrypo, iki kol įvyko validacija.



52 pav. Tinklo mokymo grafikas, kai klasės svoris – 24

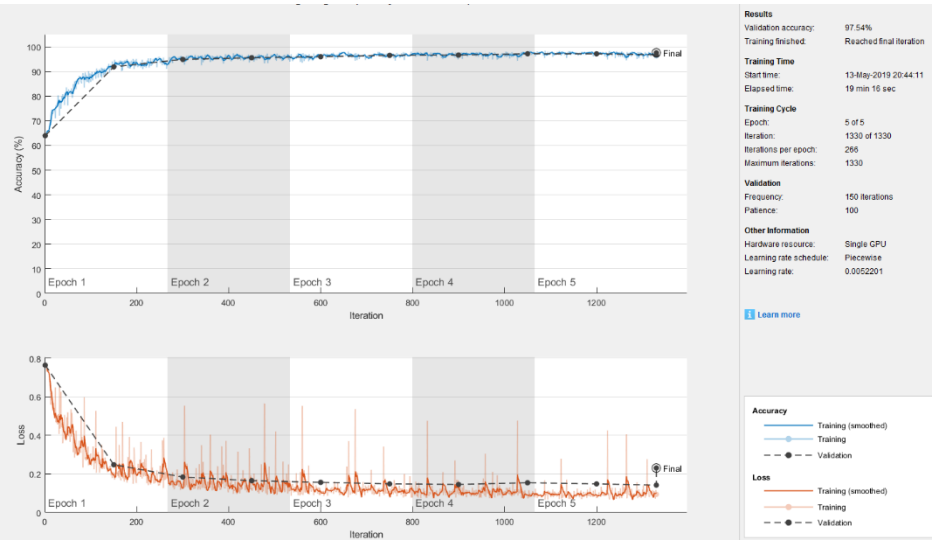
Iš 53 paveikslėlio matyti, jog padidinus klasės svorį iki 33, tinklas randa keletą kartų didesnio pikselių kiekio trūkį, lyginant su tikrąja duomenų bazės žyma. Taip pat galima pastebėti, jog vietomis tinklas randa defektą, nors tikrosiose duomenų bazės žymose jo nėra. Tai gali atsitikti dėl to, kad duomenų bazėje nepilnai sužymėti trūkiai, arba dėl didelio klasės svorio, tinklas sužymi per daug pikselių. Tačiau vietomis tinklas nepastebėjo defekto, nors jis aiškiai buvo pažymėtas duomenų bazės žymoje. Taip atsitiko dėl prastesnio tinklo apsimokymo ir duomenų su įstrižais kelio trūkiais trūkumo, kaip ir pirmuoju atveju, su klasės svoriu – 15.



53 pav. Kelio trūkio žymų pokytis, keičiantis svorio klasei

Atliekant mokymą kai klasės svoris – 33, taip pat gaunama mokymo histograma, kuri pavaizduota 54 paveikslėlyje. Iš histogramos matyti, kaip kito tinklo tikslumas ir duomenų praradimas, vykstant mokymosi procesui. Lyginant validacijos kreivę ir tikslumo kreivę, matyti, jog

tikslumo kreivė per pirmas 150 iteracijų yra kiek aukštesnė nei tikėtasi, ir tik po validacijos kreivės sutampa beveik idealiai. Panaši situacija lyginant validacijos kreivę su duomenų praradimo kreive, duomenų prarandama mažiau nei tikėtasi, tačiau po validacijos, duomenų praradimo skirtumas nuo tikimosi neišnyksta, o tik sumažėja. Nuo 600 iteracijų pastebimas duomenų praradimas mažėjimas, iki pat mokymo galo.



54 pav. Tinklo mokymo grafikas, kai klasės svoris – 33

Atlikus mokymus su „VGG-19“ iš anksto apmokytu tinklu gauti rezultatai pavaizduoti 4 lentelėje. Klasių svoriai buvo keičiami nuo 15 iki 33, žingsniu kas 9. Prasčiausias tinklo tikslumas gautas, kai klasių svoris 33, su tikslumo matais F1=0.446, o MCC=0.5143. Geriausias, kai klasių svoris – 15, gauti tikslumo matai: F1= 0.5657, MCC=0.6035. Taigi mažinant klasių svorį, nuo pradinio, kurį pateikė *MATLAB* programinis paketas, tinklo apmokomas vis geriau. Tačiau pamažinus klasės svorį, dar per vieną žingsnį iki 6, tinklas jau nebeišmoko atpažinti praktiškai nieko, apskaičiuoti tikslumo matai F1=0.00031, o MCC=0.00039, todėl galima teigti, kad tinklas neranda jokių kelio trūkių. Todėl eksperimentas buvo apribotas ties klasių svoriais nuo 15 iki 33.

Pakartojus eksperimentinius mokymus su „VGG-16“ iš anksto apmokytu tinklu gauti rezultatai pavaizduoti 4 lentelėje. Prasčiausias tinklo tikslumas gautas, kai klasių svoris 33, su tikslumo matais F1=0.485, o MCC=0.5339. Geriausias, kai klasių svoris – 15, gauti tikslumo matai: F1= 0.5323, MCC=0.5692. Taigi tendencija išlieka ta pati, jog mažinant klasių svorį, nuo pradinio, kurį pateikė *MATLAB* programinis paketas, tinklo apmokomas vis geriau. Pamažinus klasės svorį, dar per vieną žingsnį iki 6, ir šis tinklas nebeišmoko atpažinti praktiškai nieko, apskaičiuoti tikslumo

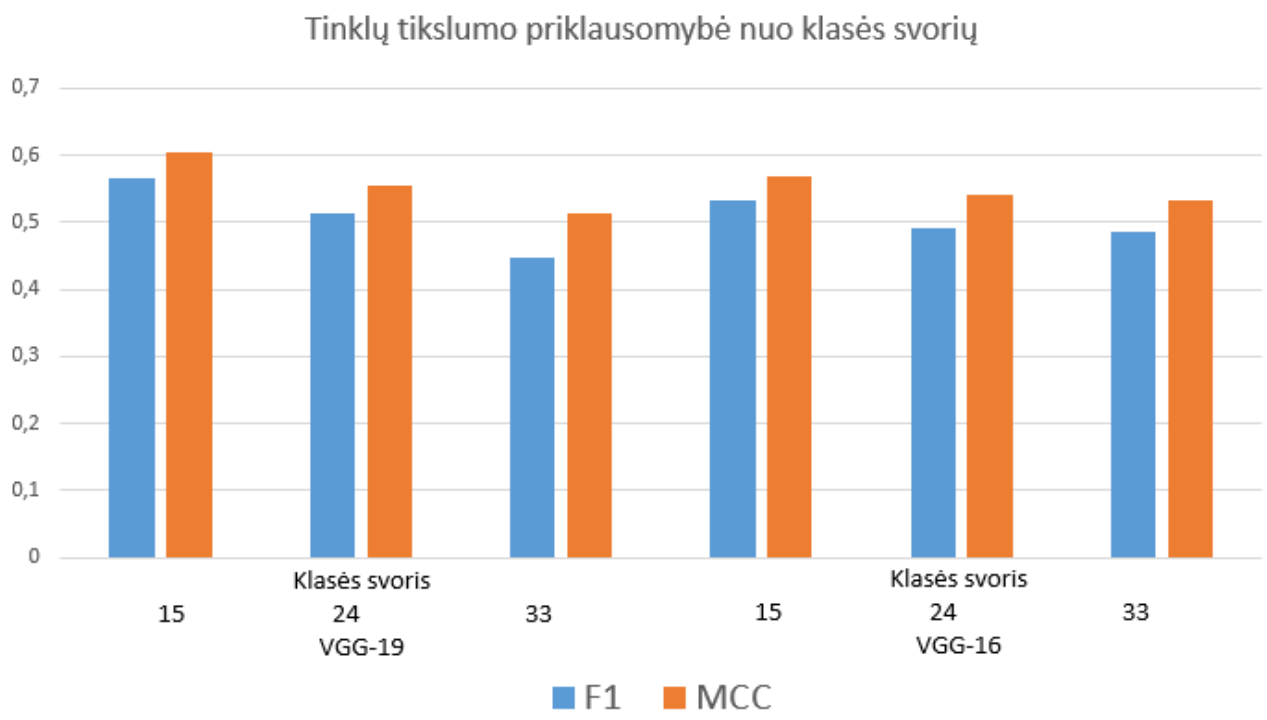
matai F1=0.00027, o MCC=0.00034, todėl galima teigti, kad tinklai neranda jokių kelio trūkių. Eksperimentas buvo apribotas ties klasių svoriais nuo 15 iki 33.

Reiktų pabrėžti tai, jog įvertinus grafinius eksperimento duomenis, pastebima, jog tinklas trūkius randa gana panašiai, neatsižvelgiant į klasių svorius, tačiau būtent šie nusako, kokiu pločio diapazonu neuroninis tinklas juos sužymės, ir būtent šios žymos, įtakoja galutines tikslumo matų apskaičiuojamas reikšmes. Kadangi rastų kelio trūkių lokacija išlieka ta pati, tačiau keičiasi kiekis žymimų pikselių, kurie laikomi kaip defektas. Tačiau, kai klasės svoris yra aukštas, tinklas prasčiau filtruoja trukdžius, todėl dažnai nuotraukoje, vietose kuriose nėra jokio kelio trūkio, tinklas pažymi trukdžius, kaip kelio trūkius, pažymėdamas lopus pikselių, kur triukšmas didesnis. Todėl, net apibendrinus visus galimus variantus, ir atsižvelgiant į tinklų veikimo tendencijas, būtent su mažesniu klasės svoriu, šiuo atveju 15, tinklai ir yra tiksliausi, tiek pagal tikslumo matus, tiek vizualiai įvertinus nuotraukas, kuriose tinklas sužymi randamas kelio trūkius.

lentelė 4 Semantinės segmentacijos eksperimentų rezultatai

	Defekto klasės svoris	**Taiklumas**	**Tikslumas**	**Atkūrimas**	**F1**	**IoU**	**MCC**
VGG19	**15**	**0,9813**	**0,4243**	**0,8961**	**0,5657**	**0,3994**	**0,6035**
	24	0,9772	0,3817	0,8647	0,5124	0,3497	0,556
	33	0,9638	0,2995	0,947	0,446	0,2931	0,5143
VGG16	**15**	**0,9802**	**0,4192**	**0,8293**	**0,5323**	**0,3689**	**0,5692**
	24	0,9747	0,3533	0,8715	0,4916	0,3277	0,5421
	33	0,9744	0,3651	0,8573	0,485	0,3264	0,5339

Atvaizdavus ketvirtoje lentelėje pateiktus duomenis grafiškai, galima lengvai akimi pastebėti, jog tinklų kokybė priklauso nuo klasių svorio. Grafikas pateiktas 55 paveikslėlyje. Didinant klasių svorius atitinkamai mažėja ir tinklo kokybė. „VGG-16“ tinklas beveik visais atvejais eksperimentuose pasirodė blogiau, nei „VGG-19“, tačiau kai klasės svoris buvo 33, „VGG-16“ pasirodė šiek tiek geriau. Kai klasės svoris 15, „VGG-19“ tinklo tikslumas 3% aukštesnis, kai klasės svoris nustatytas 24, „VGG-19“ tikslumas 2% aukštesnis, o kai klasės svoris buvo 33, „VGG-19“ tinklo tikslumas 2% žemesnis už „VGG-16“ tinklo.



55 pav. Tinklų tikslumo priklausomybė nuo klasės svorių

Išvados:

1. „VGG-19“ iš anksto apmokytas tinklas veikia tiksliau, nei „VGG-16“
2. Aukščiausi tikslumo rezultatų duomenys gauti, kai klasės svoris yra 15. „VGG-19“: F1=0.446, o MCC=0.5143. „VGG-16“: F1= 0.5323, MCC=0.5692. Skirtumas 3%.
3. Klasių svoriai stipriai įtakoja eksperimentų galutinius rezultatus ir tinklų tikslumą
4. Nustačius didelį klasės svorį, tinklas pažymi per daug fono pikseliu, ir jautriai reaguoja į triukšmus
5. Su mažu klasės svoriu tinklas veikia gan stabiliai, mažai reaguoja į triukšmus, ir gaunami geri tikslumo matai
6. Duomenų bazės netikslumai įtakojo galutinius eksperimentų rezultatus, sumažindami tinklų tikslumą.

3.3. U-Net tinklo tyrimas

Naudojant U-Net tinklą kelio defektų aptikimo uždaviniui spręsti, reikia atkreipti dėmesį į du pagrindinius parametrus, kuriuos keičiant, gali ženkliai skirtis gaunami rezultatai. Tai yra enkoderio gylis (angl. Encoder depth) ir filtro dydis (ang. Filter size). Enkoderio gylis nustato kokio gilumo U-Net tinklas bus, o filtro dydis nustato kokiu žingsniu tinklas eis per savybių žemėlapius. Šie du parametrai nustato tinklo sudėtingumą.

Šių parametrų keitimo ribos priklauso nuo įrangos, naudojamos eksperimentams atlikti. Giliųjų neuroninių tinklų apmokymas greičiausiai vyksta naudojant GPU resursus, o ne CPU. Šiuo atveju eksperimentai atliekami turint „GeForce GTX 1060 3GB“.

Pagal galimybes turint šią įrangą, nuspręsta enkoderio gylį keisti nuo 2 iki 4, o filtro dydį nuo 3 iki 7, kadangi filtro dydis turi būti nelyginis skaičius. U-Net tinklo galimų konfigūracijų naudojimui lentelė pateikta žemiau. UD* - skaičius po UD nurodo enkoderio gylį, o F* - skaičius po F nurodo filtro dydį.

lentelė 5 Galimos U-Net tinklo konfigūracijos

		Filtro dydis		
		3	5	7
U-Net enkoderio gylis	2	UD2 F3	UD2 F5	UD2 F7
	3	UD3 F3	UD3 F5	UD3 F7
	4	UD4 F3	UD4 F5	UD4 F7

U-Net tinklo apmokymui, be ankščiau paminėtų parametrų, taip pat parenkami šie, nekeičiami parametrai:

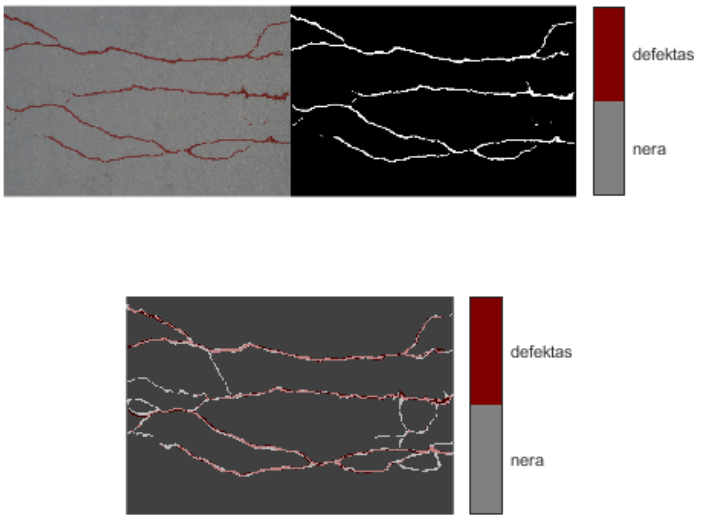
- Iteracijų skaičius – 25
- ‚LearnRateDropFactor‘ – 0.85
- ‚LearnRateDropPeriod‘ – 4
- ‚InitialLearnRate‘ – 0.0001

Atliekant eksperimentus gaunami ir pateikiami visi matavimo matai, tačiau spręsti kuris variantas yra tiksliausias vertėtų tik pagal F1 ir MCC. Pirmenybė teikiama MCC, kadangi MCC apima visų keturių painiavos matricos vertybių vertę ir yra subalansuota priemonė, kuri visapusiškai nusako tinklo tikslumą.

Duomenų bazėje yra 444 nuotraukos, kurios skirstomos atsitiktine tvarka į skirtas mokymui – 70% ir skirtas testui – 30%. Kiekviena nuotrauka yra skirtingo sudėtingumo ir tinklas skirtingu tikslumu šias apdoroja, todėl net atlikus eksperimentą su tais pačiais parametrais gaunami skirtingi rezultatai. Kokybiškesniam įverčiui gauti, atliekami 3 eksperimentai su tais pačiais parametrais, ir išvedamas gautų duomenų vidurkis.

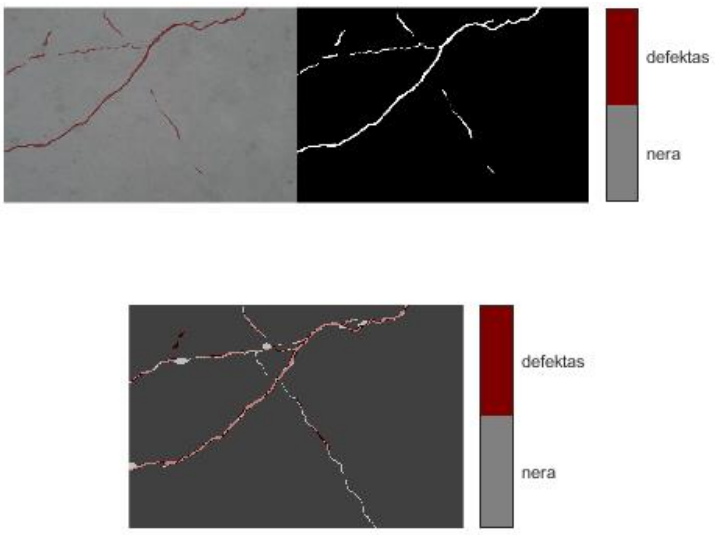
Beveik visais atvejais, nežiūrint į tinklo konfigūraciją, sunkiausiai tinklai susidorojo su nuotrauka, pavaizduota 56 paveikslėlyje. Duomenų bazėje žymos buvo sužymėta nekokybiškai, vietomis apskritai nesužymėta, todėl nors ir tinklas pamatė visus defektus, skaičiuojant jo tikslumą,

buvo lyginamos esamos žymos su detektoriaus gaunamais rezultatais. Todėl įverčiai gavosi mažesni, dėl prastos duomenų bazės kokybės.



56 pav. U-Net tinklo prasčiausias testas

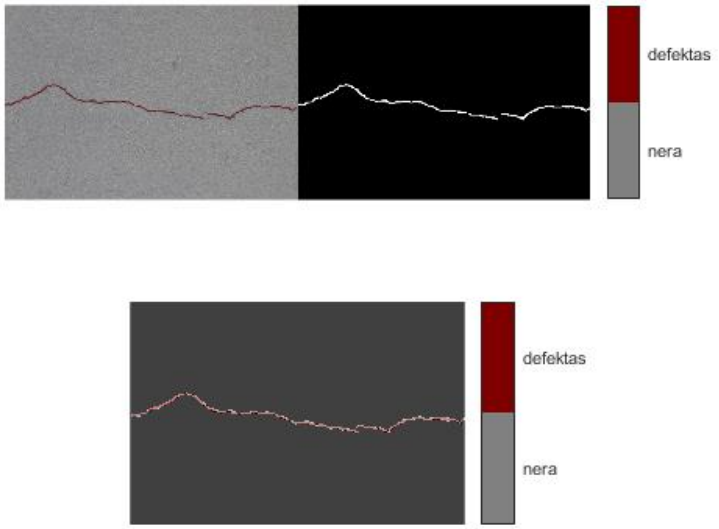
Taip pat svarbu paminėti, jog iš 57 paveikslėlio matyti, kad kelio trūkiai kurių tinklas nepastebėjo yra įstriži. Didžioji dalis duomenų bazėje esančių nuotraukų, turi kelio trūkius kurie yra horizontalūs arba vertikalūs, kitaip sakant įstrižų kelio trūkių pavyzdžių turima mažai, dėl šios priežasties tinklas neturėjo pakankamai duomenų išmokti rasti įstrižus kelio trūkius.



57 pav. U-Net tinklo pavyzdys, su įstrižais kelio trūkiais

Duomenų bazėje yra ir gerai sužymėtų žymų, kurias palyginus su originaliu paveikslėliu matyti, jog defektai sužymėti kokybiškai. Tokiais atvejais, kai tinklas puikiai surado visus trūkius, gauti

rezultatai buvo lyginami su kokybiškai atliktu žymėjimu, todėl ir tinklo kokybės įverčiai buvo gaunami tikslūs ir aukšti. Puikaus atvejo pavyzdys pateiktas 58 paveikslėlyje. Taip pat šiame pavyzdyje matyti, jog kelio trūkis yra praktiškai ištisas ir horizontalus, nėra aštrių įstrižų trūkių, todėl tinklui tai buvo labai paprasta užduotis.



58 pav. U-Net tinklo geriausias testas

Atlikus pirmąjį eksperimentą su nustatytais pastoviais parametrais, tačiau keičiant enkoderio gylį ir filtro dydį, pagal penktoje lentelėje pavaizduotus parametrus, gauti rezultatai iš kurių matyti, jog geriausias apmokyto tinklo tikslumas gaunamas su „UD2 F7“ parametrų kombinacija, kai enkoderio gylis yra - 2 o filtro dydis - 7. Šiuo atveju tikslumo matų vertės gaunamos tokios: F1 = 0.6890, o MCC = 0.6906.

Atlikus antrą eksperimentą, gauti rezultatai iš kurių matyti, jog geriausias apmokyto tinklo tikslumas gaunamas su „UD2 F5“ parametrų kombinacija, kai enkoderio gylis yra - 2 o filtro dydis - 5. Šiuo atveju tikslumo matų vertės gaunamos tokios: F1 = 0.6831, o MCC = 0.6831.

Atlikus trečiąjį eksperimentą, gauti rezultatai iš kurių matyti, jog geriausias apmokyto tinklo tikslumas gaunamas su „UD3 F7“ parametrų kombinacija, kai enkoderio gylis yra - 3 o filtro dydis - 7. Šiuo atveju tikslumo matų vertės gaunamos tokios F1 = 0.6816, o MCC = 0.6815.

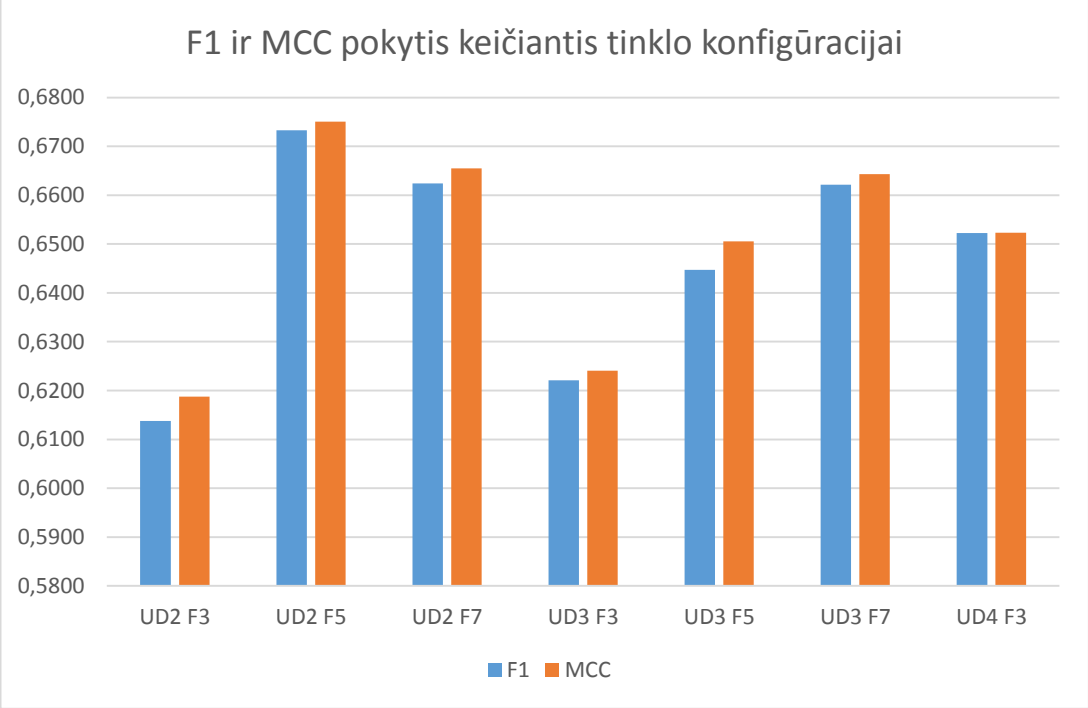
Apskaičiavus vidurkius visų trijų eksperimentų, gauti rezultatai, kurie pateikti penktoje lentelėje. Iš duomenų matyti, jog geriausias apmokyto tinklo tikslumas gaunamas su „UD2 F5“ parametrų kombinacija, kai enkoderio gylis yra - 2 o filtro dydis - 5. Šiuo atveju tikslumo matų vertos gaunamos tokios: F1 = 0.6733, o MCC = 0.6750. Toks tinklas vidutiniškai apmokomas per 3 valandas ir 23 minutes. Prasčiausias U-Net tinklo tikslumas gaunamas su „UD2 F3“ parametrų

kombinacija, kai tikslumo vertės: F1 = 0.6138, o MCC = 0.6187. Prasčiausias tinklas buvo apmokytas per 22 minutes. Skirtumas tarp geriausio ir blogiausio U-Net tinklų siekia 6%. Laiko skirto apmokymui skirtumas viena valanda.

lentelė 6 U-Net visų eksperimentų vidurkiai

		Taiklumas	Tikslumas	Atkūrimas	F1	IoU	MCC	Mokymo laikas
Vidurkis	UD2 F3	0,9892	0,6692	0,6002	0,6138	0,4526	0,6187	00:22
	UD2 F5	**0,9902**	**0,6694**	**0,7064**	**0,6733**	**0,5153**	**0,6750**	**03:23**
	UD2 F7	0,9906	0,7313	0,6254	0,6624	0,5041	0,6655	02:38
	UD3 F3	0,9880	0,5806	0,6955	0,6221	0,4588	0,6241	00:32
	UD3 F5	0,9905	0,7394	0,5934	0,6447	0,4854	0,6506	03:53
	UD3 F7	0,9905	0,6967	0,6555	0,6622	0,5039	0,6643	04:52
	UD4 F3	0,9894	0,6437	0,6823	0,6523	0,4907	0,6523	00:52

Gautus duomenis atvaizdavus grafiškai, nematyti jokio nuoseklumo duomenyse, todėl negalima teigti, jog tinklo tikslumas vienodai priklauso tik nuo enkoderio gylio, ar tik nuo filtro dydžio. Lyginant UD2 F3, UD3 F3 ir UD4F3, pastebėta, kad didėjant enkoderio gyliui, didėja ir tikslumas, tačiau palyginimui imant kitą enkoderio gylį, tokio pačio nuoseklumo nėra, todėl sunku teigti, ar UD4 F5 ir UD4 F7 būtų buvę dar tikslesni. Šiems eksperimentams atlikti įrangos galimybės nebuvo pakankamos.



59 pav. F1 ir MCC pokytis keičiantis tinklo konfigūracijai

Išvados:

1. Atlikus 3 eksperimentus ir išvedus gautų rezultatų vidurkį, geriausios kokybės tinklas gautas su UD2 F5 konfigūracija, t.y. kai enkoderio gylis yra 2, filtro dydis 5.
2. Tinklas vertikalius ir horizontalius kelio trūkius randa labai tiksliai, tačiau įstrižus aptikti jam sunkiau, dėl tokių trūkių duomenų trūkumo duomenų bazėje.
3. Eksperimentų kokybei didelę įtaką turėjo duomenų bazė, t.y. dėl ekspertų prastai sužymėtų defektų, pikselių lygyje, sumažėjo tinklo tikslumas.
4. Nepastebėta nuoseklių priklausomybių tinklo kokybės pokyčiui, keičiant parametrus.
5. Dėl nedidelės rezoliucijos nuotraukų, ir specifinės užduoties, geriems rezultatams gauti, užtenka ir nesudėtingo U-Net tinklo.

Išvados

1. Atlikus literatūros analizę buvo apžvelgti ir parinkti trys dažniausiai naudojami giliųjų neuroninių tinklų panaudojimo būdai vaizdo analizei atlikti. Kelio trūkiams aptikti pasirinkti šie būdai: regioninis gilusis neuroninis tinklas (Faster R-CNN), semantinė segmentacija ir U-Net tinklas.

2. Regioniniams giliesiems neuroniniams tinklams panaudoti esamos duomenų bazės duomenų nebuvo galimybės, kadangi defektai sužymėti pikselių lygyje, todėl kelio trūkiai buvo sužymėti rankinių būdu. Atlikus eksperimentus su iš anksto apmokytais tinklais, geriausias rezultatas gautas su „VGG-19“ tinklu, kai tikslumo matai yra tokie: : F1 – 0.7675, o MCC – 0.7385. Nors ir kokybės įverčiai aukšti, tinklas nėra tikslus dėl savo pateikiamų prognozių, kadangi nurodo tik regioną, kuriame yra kelio trūkis, o ne tikslią trūkio vietą.

3. Semantinei segmentacijai buvo pilnai naudojama pasirinkta duomenų bazė. Kadangi šis būdas skirtas dirbti su dideliais pikselių kiekiais, o kelio trūkis užima tik iki 1% visos nuotraukos pikselių, teko išsverti klasių svorius. Klasių svoriai buvo keičiami nuo 15 iki 33. Nustačius didelį klasės svorį, kokybės įverčiai krenta, kadangi tinklas trūkiui pažymėti naudoja daug aplinkinių pikselių, kurie nėra kelio trūkis. Mažinant defekto klasės svorį kokybės įverčiai gerėja, tačiau sumažinus klasės svorį per daug, tinklas nebeišmoksta atpažinti kelio trūkių. Aukščiausias kokybės įvertis gautas, kai klasės svoris yra 15. Gauti rezultatai F1= 0.5657, MCC=0.6035.

4. U-Net tinklas yra savitas tuo, kad jis buvo kuriamas naujas nuo pagrindų, ir nebuvo naudojami iš anksto apmokyti tinklai. Pagal naudojamos įrangos galimybes, buvo parinktos galimos U-Net tinklo konfigūracijos, kai tinklo gylis keičiamas nuo 2 iki 4, o filtro dydis nuo 3 iki 5. Atlikus eksperimentus su visomis tinklo konfigūracijomis, geriausias tinklas gautas, kai enkoderio gylis yra 2, o filtro dydis 5. Gauti kokybės įverčiai F1 = 0.6733, o MCC = 0.6750. Tinklas vertikalius ir horizontalius kelio trūkius randa labai tiksliai, tačiau įstrižus aptikti jam sunkiau, dėl tokių trūkių duomenų trūkumo duomenų bazėje.

Literatūros sąrašas

1. https://adeshpande3.github.io/The-9-Deep-Learning-Papers-You-Need-To-Know-About.html (žiūrėta 2018-10-15)
2. Nielsen M. A."Neural Networks and Deep Learning", Determination Press, 2015
3. Zeiler M.D. and Fergus R. „Visualizing and Understanding Convolutional Networks“, Dept. of Computer Science, New York, 2014
4. Girshick R.„Fast R-CNN“ Microsoft Research, 2012
5. Girshick R., Donahue J., Darrell T., Malik J.“Region-based Convolutional Networks for Accurate Object Detection and Segmentation” Fellow, 2015
6. Girshick R., Ren S., He K., Sun J. „Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks“, 2015
7. https://www.jeremyjordan.me/semantic-segmentation/ (žiūrėta 2019-02-17)
8. Long J., Shelhamer E., Darrel T., „Fully Convolutional Networks for Semantic Segmentation”, UC Berkeley
9. Pinheiro P.O., Collobert R. „From Image-level to Pixel-level Labeling with Convolutional Networks” Idiap Research Institute, Martigny, Switzerland, 2015
10. https://towardsdatascience.com/u-net-b229b32b4a71 (žiūrėta 2019-02-26)
11. Ronneberger O., Fischer P., Brox T. „U-Net: Convolutional Networks for Biomedical Image Segmentation“ University of Freiburg, Germany, 2015
12. https://github.com/zhixuhao/unet (žiūrėta 2019-02-26)
13. Zhang Z., Liu Q., Wang Y. „Road Extraction by Deep Residual U-Net", 2018
14. https://cv-tricks.com/cnn/understand-resnet-alexnet-vgg-inception/ (žiūrėta 2019-10-05)
15. Krizhevsky A., Suskever I., Hinton G.E. „ImageNet Classification with Deep Convolutional Neural Networks“, 2012
16. https://medium.com/@sidereal/cnns-architectures-lenet-alexnet-vgg-googlenet-resnet-and-more-666091488df5 (žiūrėta 2019-10-05)
17. Simonyan K., Zisserman A. „Very Deep Convolutional Networks for Large-Scale Image Recognition“, 2015
18. https://cv-tricks.com/cnn/understand-resnet-alexnet-vgg-inception/ (žiūrėta 2019-10-05)
19. David M. W. „Evaluation: From Precision, Recall and F-Measure To Roc, Informedness, Markedness & Correlation“ Powers, 2011
20. Everingham M., Winn J. „The PASCAL Visual Object Classes Challenge“, 2012
21. https://stats.stackexchange.com/questions/1773/what-are-correct-values-for-precision-and-recall-in-edge-cases (žiūrėta 2019-10-07)

22. https://lettier.github.io/posts/2016-08-05-matthews-correlation-coefficient.html (žiūrėta 2019-10-07)
23. Shi Y., Cui L., Qi Z., Meng F., Chen Z. „Automatic Road Crack Detection Using Random Structured Forests“, 2016
24. https://github.com/AlexeyAB/Yolo_mark (žiūrėta 2019-10-02)
25. https://se.mathworks.com/help/deeplearning/ref/trainingoptions.html (žiūrėta 2019-10-04)