VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
STUDIJŲ PROGRAMA: INFORMATIKA

Nepilotuojamų transporto priemonių navigacija, naudojantis kapsulių neuroninio tinklo detaliu ortofotografinių vaizdų palyginimu

Unmanned Aerial Vehicles Navigation using Capsule Neural Network for Fine-Grained Similarity of Aerial Images

Baigiamasis magistro darbas

Atliko: Tautvydas Stakėnas
VU el. p.: tautvydas.stakenas@mif.stud.vu.lt

Vadovas: Asist. Dr. Vytautas Valaitis
Recenzentas: Asist. Dr. Valdas Dičiūnas

Vilnius
2022

Santrauka

Šiame darbe yra sprendžiama drono lokalizacijos problema, pasikliaunant neuroniniais tinklais. Dronas be GPS turi turėti galimybes kitais būdais nustatyti savo lokaciją. Dronai, skrisdami ilgus atstumus, negali pasikliauti tiesine odometrija su ranka kurtais algoritmais dėl įvairių nukrypimų. Žemėlapiais paremta sistema su neuroniniais tinklais gali padėti dronui naviguoti ilgus atstumus be didesnių nukrypimų ir nustatyti pradinę buvimo vietą. Darbe tiriami kapsulių neuroniniai tinklai su ortofotografinėmis nuotraukomis. Baziniai vaizdai yra pasukami (nuo -10 iki +10 laipsnių) simuliuoti drono pasisukimą ir kompaso paklaidą. Duomenų rinkinius sudaro trejetai: baziniai vaizdai, teigiami vaizdai ir neigiami vaizdai. Bandymuose naudojama trejetų nuostolių funkcija su atstumo koeficientu 0,2 norint apskaičiuoti skirtumams tarp trejetų ir apmokyti neuroninį tinklą. Galų gale gautas modifikuotas kapsulių neuroninis tinklas, kuris mokosi greičiau ir pasiekia didesnį tikslumą nei originalus Hinton et. al. pateiktas modelis ir ResNet modelis.

Raktažodžiai:

Kapsulės, CNN, neuroninis tinklas, ortofotografiniai vaizdai, trejetai, kampo transformacija.

Summary

In this paper, we are looking at how to solve the drone localization problem using neural networks. A drone without GPS must have other ways to learn its position. Drone traveling long distances can‘t rely on linear odometry with hand-crafted solutions because of various inaccuracies. A map-based system that uses neural networks can help drones navigate long distances without significant inaccuracies and allow them to learn their initial position. In this work, we research capsule networks with aerial images. Anchor images are transformed by angle (from -10 to +10 degrees) to simulate drone direction and compass error. Data sets consist of triplets: anchor image, positive image and negative image. In our experiments, we use triplet loss with a margin value of 0.2 to calculate distances between triplets and be able to start the learning process. The trained modified capsule network model improved learning speed and accuracy compared to the original Hinton et al. model and ResNet model.

Key words:

Capsules, CNN, neural network, aerial images, triplets, angle transformation.

Turinys

1. Įvadas

Vizualinė odometrija tampa vis labiau reikalinga nepilotuojamų skraidančių transporto priemonių (angl. unmanned aerial vehicles - toliau UAV) navigacijoje. UAV paprastai pasikliauna GPS (angl. Global Position Signal) nustatant savo lokaciją, tačiau gali nutikti taip, kad GPS signalo gali nebūti arba jis yra silpnas ar blokuojamas. Be GPS šios transporto priemonės tampa neveiksnios, kadangi nebegali nustatyti savo lokacijos. Šiai problemai spręsti yra naudojamos sistemos, kurios jutiklių ir kompiuterių pagalba padeda nustatyti lokaciją nenaudojant GPS. Dažnai naudojami jutikliai ar jų kombinacijos: kamera, lazeris, sonaras, akselerometras, kampinio greičio jutiklis, Pito vamzdis, kompasas. Sistemos, paremtos šiais jutikliais, atlieka matematinius skaičiavimus, kurie leidžia su tam tikra paklaida nustatyti drono būvimo vietą. Trumpiems atstumams yra naudojamos map-less (be žemėlapių) sistemos kartu su map-based (su žemėlapiais) sistemomis, tačiau dėl mažo aukščio ortografinių nuotraukų darymas nėra taikomas. Ilgiems atstumams daromos ortografinės nuotraukos gali padėti išgauti didesnį tikslumą ir padėti rasti pradinę drono buvimo vietą. Lokalizacijos technikos su žemėlapių pagalba analizuoja vaizdo medžiagą, perduotą iš jutiklių ir lygina su jau turimais žemėlapių duomenimis. Paprasti algoritmai su nedideliais pasikeitimais praranda tikslumą, dėl to yra bandoma pritaikyti neuroninius tinklus šiai problemai spręsti.

Tam, kad kompiuteriai galėtų lengviau palyginti vaizdus, uždavinys yra skaidomas į du etapus: savybių išgavimą ir klasifikavimą [WSLR+14, SLJS+15, SFH17]. Vienintelė kompiuteriui žinoma informacija yra paveiksliukų pikselių duomenys, iš kurių reikia išgauti aukštesnio lygio informaciją [Kou17]. Šiems uždaviniams atlikti yra naudojami gilieji dirbtiniai neuroniniai tinklai (angl. deep neural networks, trump. DNN). Konvoliucinis neuroninis tinklas (angl. convolutional neural network, trump. CNN) yra plačiausiai naudojamas DNN, kadangi jis labai gerai susidoroja su šia užduotimi. CNN yra paprasčiausiai sistema, sudaryta iš didelio kiekio neuronų grupių [KSH12]. Būtų labai sudėtinga apjungti (angl. map) kiekvieną paveikslėlio pikselį vienu metu, nes tai pareikalautų labai daug kompiuterio resursų ir gaunamas mažesnis tikslumas nežymiai pasikeitus duomenims. Tad CNN yra DNN tipas, kuris padeda supaprastinti ir optimizuoti skaičiavimus, neprarandant informacijos [KSH12, RHW85].

Detalus vaizdų palyginimas (angl. fine-grained image similarity) yra dar sudėtingesnė užduotis už vaizdų klasifikavimą, kadangi yra didelė variacija (didelis panašumas) tarp klasių ir nedidelė variacija tarp poklasių. Detalus vaizdų palyginimas yra tikslinis vaizdų klasifikavimas, kuris siekia atpažinti poklasius pagal tam tikras savybes [XY17]. Vaizdų atpažinimo atveju juodas, baltas ir pilkas automobilis yra automobilis (klasė), tuo tarpu detaliam vaizdų palyginimui, jeigu užklausa yra juodas automobilis, tai reitingavimas pilko automobilio turėtų būti aukštesnis nei balto automobilio (poklasiai) [WSLR+14]. Šiame darbe bus pritaikytas trejeto (angl. triplet) modelis, kuris pasiekė aukštą tikslumą veidų atpažinime [SFKP15]. Trejeto modelis yra mašininiu mokymusi (angl. machine learning) paremtas modelis, kuris leidžia apmokyti neuroninį tinklą skirti paveiksliukus tarp poklasių, išskiriant tam tikras savybes, naudojant užklausos (angl. query) teigiamą ir neigiamą paveiksliukų pavyzdžius. Teigiamas paveiksliukas yra labiau panašus į užklausos paveiksliuką negu neigiamas paveiksliukas [WSLR+14]. Šio modelio esmė ta, kad apmokytas tinklas sugeba atskirti panašius ir nepanašius paveiksliukus pagal užklausos paveiksliuko pavyzdį. Tinklo sugebėjimas atpažinti vienodus poklasius tiesiogiai priklauso nuo į tinklo įvestį paduotų apmokymo (angl. training) duomenų (poklasiai pagal objektų savybes, pvz.: automobilio forma, spalva ir pan.) [WSLR+14, SFKP15].

Apie 2006 m. žinomas mokslininkas Geoffrey Hinton, kartu su kitais, parašė mokslinį darbą apie konvoliucinius neuroninius tinklus. 2012 metais du Hintono studentai laimėjo ImageNet konkursą, pasiekę dvigubai didesnį tikslumą nei antroje vietoje esantis varžovas. Šis neuroninis tinklas yra dar žinomas kaip AlexNet [KSH12, Kou17]. Po to, buvo sukurta daug CNN vis gerinančių tikslumą, tačiau ResNet, pristatytas 2015 metais, smarkiai patobulino standartinį CNN modelį. Šis tinklas atsisakė „DropOut“ sluoksnių, taip išspręsdamas „vanishing gradient problem“ ir, tuo pačiu, sumažino parametrų kiekį, bei mokymosi trukmę [SLJS+15, Hin17, Kou17]. Vietoj to, tinklas susidarė iš atliekamų (angl. residual) sluoksnių. ResNet tapo smarkiai paplitusiu neuroniniu tinklu dėl savo supaprastinto modelio ir sumažinto parametrų kiekio. Kadangi šis tinklas pasiekia aukštą tikslumą su dideliais kiekiais duomenų ir sugeba gerai išmokti savybes įvairiuose gyliuose, jį panaudosime eksperimentuose lyginant su CapsNet neuroniniu tinklu.

Funkciškai atpažįstami pakitimai (angl. equivariance) yra vis dar neišspręsta problema neuroniniuose tinkluose. Šiai problemai spręsti buvo pateiktas modelis, kuris grupuoja kelis CNN, kurie, priklausomai nuo transformuotų duomenų, tiksliau atliktų spėjimus. Taip varžantis

modelio darbą veikti tik su tam tikromis transformacijomis. Šios technikos yra taikomos atpažinti pasukimo bei slenkamojo judėjimo transformacijoms, tačiau dažniausiai nesugeba generalizuoti transformacijų su daugiau kombinacijų. Apie CapsNet Geoffrey Hinton et. al. išleido pirmąjį straipsnį 2017 metais, kaip tinklą, kuris turėtų padėti funkciškai atpažinti pakitimus. CapsNet architektūra nemažai skiriasi nuo įprastų neuroninių tinklų, todėl jo galimybės ir pritaikymo būdai gali taip pat skirtis. Šis sumažina standartinių konvoliucinių sluoksnių kiekį, vietoj to jis sluoksnius įkomponuoja (angl. nest) į kapsulių sluoksnį. Šios kapsulės grąžina vektorių sekantiems sluoksniams ir taip išlaiko daugiau informacijos apie įvestį. Taip pat, kaip ir ResNet, šis tinklas atsisako atgalinio klaidos sklidimo metodo ir „DropOut“ sluoksnių, dėl to, jis nepraranda 3D informacijos ir sąlyginai sumažėja parametrų skaičius [SFH17].

Darbe aprašomas CapsNet neuroninio tinklo pritaikymas detaliam vaizdų palyginimui. Kadangi CapsNet yra ganėtinai naujas CNN modelis, dar nėra visiškai aiškios jo galimybės. Žinant, kad CapsNet išlaiko informaciją apie objektų padėtį ir tekstūrą, galima tikėtis gerų rezultatų [SFH17]. Darbe formuojamiems duomenims atliksime pasukimo transformacijas. Tai turėtų pademonstruoti ar CapsNet geriau už CNN išmoksta atskirti savybes nepriklausomai nuo jų transformacijų.

1.1. Tikslai ir uždaviniai

Darbo tikslas - Ištirti CapsNet neuroninio tinklo tinkamumą detaliam vaizdų palyginimui su žemėlapių nuotraukomis, palyginant su CNN alternatyva.

Uždaviniai:

1. Atlikti CNN, CapsNet konvoliucinių neuroninių tinklų analizę;
2. Reikalingų duomenų tinklų apmokymui ir testavimui paruošimas;
3. Realizuoti CapsNet metodu veikiantį neuroninį tinklą vaizdų palyginimo problemai spręsti;
4. Realizuoti skirtingus CapsNet modelius (atlikti modifikacijas) ir palyginti jų veikimą;

5. Išmokyti CNN ir CapsNet konvoliucinius neuroninius tinklus trejetų tinkle su paruoštais duomenimis;

6. Patikrinti CapsNet ir CNN gebėjimą prisitaikyti prie nuotraukos pasukimo transformacijos ir palyginti gautą tikslumą iš mokymo, validavimo ir testavimo duomenų rinkinių.

2. Literatūrinė analizė

Šiame skyriuje išnagrinėti skirtingi būdai išspręsti dronų navigacijos problemą nenaudojant GPS. Aprašyti vizualinės odometrijos ir neuroninių tinklų metodai bei eksperimentai naudoti išspręsti navigacijos problemai. Pateikiama neuroninių tinklų, konvoliucinių tinklų, trejetų ir kapsulių neuroninių tinklų analizė, kuria remiantis bus atliekami eksperimentai navigacijos problemai spręsti.

2.1. Vizualinė odometrija

Dar gerokai prieš neuroninius tinklus odometrija dominavo robotikoje. Tik 2006 metais, kai Hintonas pristatė pirmąjį CNN tinklo modelį, kuris įveikė prieš tai naudotus algoritmus, neuroniniai tinklai pradėjo susilaukti daugiau dėmesio ir tyrimų [ZDZ15]. Vėliau, 2012 metais, kai buvo pradėta naudoti GPU (angl. graphics processing unit) neuroninių tinklų apmokymui, neuroniniai tinklai pradėjo įgauti populiarumą. Šiame skyriuje išanalizuoti atvejai, kur vizualinė odometrija pasiteisino ir kodėl neuroniniai tinklai gali padėti išspręsti esamas problemas.

2.1.1. Vizualinės odometrijos pritaikymas navigacijoje Marse

Vaizdinės odometrijos (trump. VO) „map-less“ techniką NASA pritaikė „Mars Exploration Rovers“ (trump. MER) projekte. Ši technika pasiteisino, nors tai buvo pirmas kartas, kai ji buvo panaudota kitoje planetoje - Marse. Ši technika leido dronui tiksliai nustatyti savo poziciją, nes dronas sugebėjo automatiškai nustatyti ir kompensuoti bet kokius nuslydimus kelionės metu dėl akmenuoto ir slidaus reljefo. Sistemą sudarė pačiame roveryje esantis kompiuteris su paruošta programine įranga, bei dvi 45 laipsnių matymo lauko (FOV) navigacijos stereo kameros (NAVCAMs). Sistema stebėjo judėjimą automatiškai parinktų reljefo savybių iš dviejų porų 256x256 pikselių stereo nuotraukų, algoritmas turi apskaičiuoti neatitikimus 6 laipsnių paklaida ir atnaujinti roverio poziciją (x, y, z, roll, pitch, yaw). Sistema pademonstravo labai gerus rezultatus, aukštas sėkmingų konvergencijų dažnis (95-97%), sėkmingai aptinka nuslydimus iki 125%, bei pakitimus iki 2 mm, net ir važiuojant ant aukštų įkalnių - apie 31 laipsnio statumo. Tikslas buvo, kad MER sugebėtų įvertinti poziciją 10% tikslumu 100-ui metrų. Pakitimai pozicijai (roll, pitch, yaw) buvo matuojami Inertial

Measurement Unit (IMU), kuris turi 3 ašių akselerometrus ir 3 ašių kampinio greičio jutiklius. Pakitimai pozicijai buvo kombinuojami kartu su pakitimais, kiek pasisuko roverio ratai (ratų odometrija). Pajudėjus nedidelį atstumą ant slidaus paviršiaus, roveris dažnai duodavo komandą sistemai, jog panaudotų kameromis paremtą VO algoritmą, ir kad ištaisytų galimas klaidas pradiniams ratų odometrija padengtiems skaičiavimams, kas ypač nutinka, kai ratai praslysta judant stačiomis įkalnėmis ar akmenuotu paviršiumi. Ši VO sistema apskaičiuoja atnaujintą roverio poziciją (x, y, z, roll, pitch, yaw), sekdama judėjimą „įdomių“ reljefo savybių tarp dviejų porų stereo nuotraukų, tiek 2D pikselių koordinatėmis, tiek 3D pasaulio koordinatėmis. Didžiausias tikėtinas įvertinimas, naudojantis apskaičiuota 3D kaita, (angl. offset) priimamas, kaip galutinis judėjimo įvertinimas. Tačiau, jeigu bet koks pastovumo patikrinimas atmeta ar per mažai reljefo savybių buvo užregistruota, ar neišeina sukonvertuoti įvertinimų, tokiu atveju joks judėjimo įvertinimo atnaujinimas nebus priimamas ir bus naudojamas pradinis įvertinimas (dažniausiai padengtas ratų odometrija ir IMU) [MCM07].

Jau 1980 metais Moravec išleido darbą apie roboto judėjimo tikrinimą naudojant stereo kameras [M80]. Matthies et al. laikė judėjimo tikrinimą kaip statistinę problemą ir sukūrė nuoseklius metodus transporto priemonės judėjimo tikrinimui ir vietovės modelio atnaujinimui. Ši sistema pasiekė 98% tikslumą per 5.5 metrus ir 55 stereo nuotraukų poras [LS87, L89], su patobulintu pastovaus lygio tikslumu 2003 metais [CLMW03]. Panašūs darbai buvo išleisti ir kitur [ZON88, SARL99, N04]. Nist’er et al. pateikė sėkmingą VO realaus laiko modelį [NNB04, NNB06]. Šis modelis susideda iš dviejų judėjimo testavimo schemų: stereo schemos, kuri yra iteratyvi pozicijos nustatymo schema, ir monoculiarinė schema, kuri buvo paremta 5-taškų algoritmo sistmos [N04] ir išskirtinumų išgryninimo schemos [N05]; geri rezultatai buvo nustatyti su labai ilgomis vaizdų sekomis. Yra ir daugiau darbų prie VO schemų. Pavyzdžiui, McCarthy ir Barnes pateikė darbą apie optinio tiekimo efektyvumą bandant nustatyti judėjimą [MB04]; Vassallo ir Gluckman et al. sukūrė „ego-motion“ tikrinimo schemą su įvairiakryptėmis nuotraukomis [VSS02, GN98].

Pozicijos tikrinimo esmė yra surasti tam tikrus išskirtinumus stereo nuotraukų poroje ir juos sekti iš vieno kadro į kitą. Pagrindinė idėja MER modelyje yra nustatyti dviejų stereo nuotraukų porų pasikeitimus pozicijoje ir nuotolyje atsirandant neapibrėžtumų 3D pozicijos nustatymo formulavime naudojant maksimaliai tikėtiną rezultatą [MCM07].

Bruožų radimas

Pirmiausia pasirenkami bruožai, kurie gali būti paprastai palyginami ir sekami tarp vieno judesio žingsnio stereo porų nuotraukų. Kampų aptikimo įdomumo rodiklis nustatomas (pvz.: Forstner or Harris) ir pritaikomas nuotraukų porai, tada pasirenkami pikseliai su didžiausiais įdomumo rodikliais. Tam, kad sumažinti skaičiavimų kiekį, sudaroma lentelė su langeliais, mažesniais negu iš anksto apibrėžtas minimalus atstumas tarp bruožų, ir uždedama ant kairės nuotraukos. Bruožas su stipriausiu įdomumo rodikliu yra pasirenkamas kaip tinkamas kandidatas kiekvienam lentelės langeliui. Tokiu būdu dėl fiksuoto pasirinktų bruožų skaičiaus yra užtikrinama, kad jie būtų išsidėstę per visą nuotrauką [MCM07].

Bruožais paremtas porų atitikmuo

Kiekvienam pasirinktam bruožui yra skaičiuojamas porų atitikmuo ir taip nustatoma 3D pozicija. Kadangi stereo kameros yra gerai sukalibruotos, porų atitikimas yra atliekamas griežtai per epipolinę liniją, tik su kelių pikselių paklaidos buferiu. RAV naudoja pseudo normalizuotą koreliaciją nustatyti geriausiems atitikmenims. Kad pavyktų pasiekti pikselio tikslumą, daugianaris turi tilpti į kaimyno 3x3 koreliacijos balus ir didžiausias polinomas yra pasirenkamas, kaip koreliacijos viršūnė [MCM07], tokiu būdu sumažinama klaidos tikimybė, kad buvo pasirinktos klaidingos poros.

3D pozicijos šių pasirinktų bruožų yra nustatomos susikertančių spindulių projektuojamų per kamerų modelius. Tobulomis sąlygomis, spinduliai to paties bruožo kairėje ir dešinėje nuotraukoje susikirstų erdvės viename taške. Tačiau, dėl vaizdo triukšmo, kameros modeliai yra nepatikimi ir atitikmuo gali būti klaidingas, todėl jie nebūtinai susikerta. Trumpiausias atstumas tarp tarpo tarp dviejų spindulių indikuoja kokios kokybės buvo porų atitikmuo: bruožai su dideliais atstumais yra atmetami [MCM07].

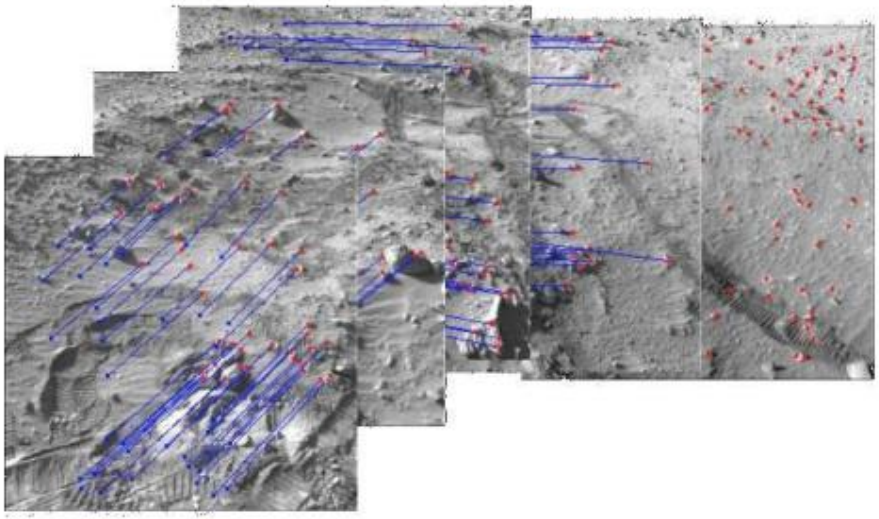
Bruožų sekimas

Kai roveris pajuda trumpą atstumą, gaunama antra pora stereo nuotraukų. Bruožai, pasirinkti iš prieš tai buvusios nuotraukos, yra projektuojami į antrą porą, naudojant apytikslų judėjimą, gautą iš esančios ratų odometrijos (angl. wheel odometry). Tada koreliacija, paremta paieška, iš naujo nustato tikslias 2D pozicijas antroje nuotraukų poroje. Sekamų bruožų porų derinimas nustato jų naujas 3D pozicijas. Kadangi 3D pozicijos sekamų bruožų buvo žinomos nuo praėjusio žingsnio, porų derinimo paieškos diapazonas yra gerokai sumažinamas.

Patikimumo testas (lyginant reliatyvius 3D atstumus atitinkamų taškų) yra atliekamas, kaip pirminis atmetimo žingsnis [MCM07].

Patikimas judesio įvertinimas

Jei pirminis judėjimas yra tikslus, skirtumas tarp dviejų rastų 3D bruožų pozicijų turėtų tilpti į klaidos elipsę. Tačiau, kai pirminis judėjimas yra netikslus, skirtumas tarp dviejų rastų 3D taškų pozicijų atspindi klaidingą judėjimą ir tai gali būti panaudojama nustatant roverio pozicijos pasikeitimą [MCM07].



1 Pav. Marso drono vaizdinės odometrijos nuotraukos ir savybių atstumų nustatymas

Šis VO pritaikymas, daugiausiai, pasikliauna ratų odometrijos skaičiavimais ir naudoja VO, kaip pagalbinę priemonę. Dronai naudoja kitus sensorius kartu su VO. Roverio atveju netikslumai gali atsirasti dėl slidaus ar nelygaus reljefo [MCM07]. Drono atveju reljefas neturi įtakos, tačiau netikslumai gali atsirasti dėl įvairių oro sąlygų. Vėjo pasipriešinimas gali suklaidinti drono skaičiavimus, dėl to reikia sugebėti vizualiai arba GPS pagalba nustatyti drono vietą ir ištaisyti įmanomas klaidas. Taip pat, dronai juda daug greičiau, ir dėl to reikia daug dažniau daryti nuotraukas. Tada atsiranda didesnė paklaida ir gali apskritai nepavykti padaryti skaičiavimų dėl greito judėjimo. Taip pat, skirtumas yra atstumas nuo žemės, kadangi dronai, kitaip nei roveriai, gali būti skirtinguose aukščiuose, todėl reikia skaičiuoti atitinkamai nuo esamo aukščio 3D erdvėje. Sekantis skyrelis plačiau paaiškins, kaip VO gali būti pritaikytas dronuose.

2.1.2. Vizualinės odometrijos pritaikymas dronuose

Per paskutinį dešimtmetį, vertikalaus pakilimo ir nusileidimo (angl. vertical taking-off and landing, VTOL) mikro orlaiviai (angl. micro aerial vehicles, MAVs), kurie naudoja priešingo sukimosi rotorius sugeneruoti traukos ir sukimosi jėgas, gavo daug dėmesio moksliniuose tyrimuose ir industrijoje. Dėl naujų pritaikymo būdų, kaip struktūrų inspekcija, aerofotografija, kinematografija ir aplinkos stebėjimas, prasiplėtė paruoštų skrydžiui komercinių platformų pasirinkimas, kurių našumas pastoviai įvairiai gerėja: skrydžio laikas, apkrova ir saugumu susijusios „smart“ funkcijos, leidžiančios stabilesnį ir paprastesnį piloto manevravimą [SKBB+18]. Tačiau, pagrindinis iššūkis yra komercinių platformų adaptavimas užduotims, reikalaujančioms tikslių dinaminių modelių, aukšto pajėgimo kontrolerių, tikslumo, mažo vėlavimo būsenos atnaujinimo įvertinimo, kaip kliūčių išvengimas ir kelio planavimas, nusileidimas ant judančių platformų, objektų paėmimas ir tikslusis žemės ūkis [LPLK12].

„Visual-inertial“ (trump. VI) sensorius yra palankus pasirinkimas orinėje robotikoje (angl. aerial robotics) dėl jo mažo svorio, mažo galios suvartojimo ir galimybė atgaminti nežinomus mastus (monocular camera). Šis sensoriaus derinys gali suteikti laiko sinchronizuotą plačią matymo lauko (angl. field of view, FoV) nuotrauką ($\approx 133^{\circ}$) ir judėjimo matavimus. Jau yra sukurtų ir tyrimams paruoštų VI sensorių [SKBB+18].

VI odometrija yra aktyvi tyrimų tema per paskutinį dešimtmetį, atsižvelgiant į microelektromechaninių sistemų (trump. MEMS), IMU (angl. intertial measurement unit) ir vaizdo technologijų pažangą. Tai yra vienas geriausių pasirinkimų ribotos apkrovos platformoms, kaip VTOL MAV, dėl lengvo svorio ir gana nedidelių skaičiavimo reikalavimų. Visos sensorių būsenos yra sujungtos tarpusavyje pasitelkiant filtravimo ar netiesinius optimizavimo algoritmus. Šie algoritmai nustato roboto judėjimą stebėdami vizualinius ir inercinius matavimus, kartu ir minimizuoja duomenis ir paruošia juos tolimesniam naudojimui pagerinant nuoseklumą. Filtravimu paremti metodai dažnai naudoja Kalman Filter [SKBB+18, BOHS15], dėl to kenčia nuo nuklydimų ir turi ribotą supratimą apie tikrąją aplinką. Tačiau, atlikus lokalizavimo darbus, tikslumas yra pakankamas kontrolės stabilizavimui ir mažo atstumo užduočių atlikimui [SKBB+18].

Atliktame bandyme [SKBB+18] buvo atlikti 9 eksperimentai, tiek uždaroje patalpoje, tiek atviroje erdvėje, kad būtų galima geriau nustatyti kontrolės ir būsenos nustatymą įvairiomis aplinkybėmis. Atliktos 3 užduotys: sklandymas, žingsnio atgalinis ryšys ir trajektorijos

sekimas. Pademonstruoti kontrolerio prisitaikymą yra generuojami vėjo trikdžiai uždaroje patalpoje naudojantis ventiliatoriumi su 260W ir 300 m^3/min oro srautu. Pamatavus anemometru, jis pagamina 11-11.5 m/s oro srauto trikdžius sklandančioje pozicijoje. Šie trikdžiai padeda simuliuoti vėjo gūsius, kurie gali pasitaikyti realiame scenarijuje.

Nustatyti kontrolės darbą uždaroje patalpoje, suskaičiuojamas RMS (angl. root-mean-square) paklaida tarp tikros ir numanomos pozicijų ir orientacijų, kas gaunama iš judėjimo nustatymo įrenginių, su Euklido atstumu, naudojamu suskaičiuoti skirtumus tarp 3D pozicijų. Atviroje erdvėje, buvo vertinama tik pozicijos klaida su pagrįsta tiesa iš lazerio sekimo prietaisų. Panašiai, kaip būsenos nustatymo kokybė yra apskaičiuojama naudojantis RMS paklaida tarp esamos pozicijos ir jos įvertinimo. Iš visų atliktų bandymų [SKBB+18] galima pastebėti, kad paklaida (RMS error) yra šiek tiek didesnė atvirose vietovėse dėl nepastovaus vėjo. Dėl to galima teigti, kad užduotims, kurios yra mažoje erdvėje, šis sprendimas gali būti tinkamas, bet navigacijai didesniems atstumams reikia ieškoti kitų alternatyvų.

2.2. Neuroniniai tinklai

VO ir lokalizacija ir kartografavimas (angl. Simultaneous localization and mapping, trump. SLAM) technikos parodė labai gerus rezultatus skrydžiuose uždaroje patalpoje ir žemiau 100 metrų aukščio atviroje erdvėje. Kadangi šie algoritmai nenaudoja žemėlapių, jie nėra tinkami skrydžiams, ilgesniems nei ~1 km dėl įvairių nukrypimų. Tikslumas taip pat kenčia skrydžiams virš 100 metrų aukščio dėl įvairių vibracijų ir drono pasisukimų. Taip pat, GPS nebuvimas (angl. jamming and spoofing) tapo aktualus per paskutinius kelerius metus [VMJ20]. Kitas SLAM minusas, kad jis reikalauja žinoti pradinę poziciją (kam dažniausiai naudojami įvesties parametrai ar GPS), kad galėtų toliau sekti nukeliautą kelią. Navigacijos problemai be GPS spręsti gali padėti neuroniniai tinklai. Vietovių nuotraukomis apmokytas neuroninis tinklas gali palyginti padarytas žemės nuotraukas su turimų nuotraukų duomenų baze ir rasti esamą lokaciją trejetų arba siamo neuroninių tinklų pagalba. Šie neuroniniai tinklai turi vieną užduotį - atlikti nuotraukų palyginimą. Šį palyginimą galima atlikti ir kitais būdais, pvz. skaičiuoti koreliaciją tarp pikselių, tačiau darome prielaidą, kad neuroniniai tinklai gali efektyviau ir tiksliau atlikti vaizdų palyginimą. Trejetų arba siamo neuroninių tinklų architektūros naudoja konvoliucinius neuroninius tinklus (angl. convolutional neural networks, trump. CNN). CNN dažnai naudojami dirbant su nuotraukų apdorojimu, kaip plačiai žinomi ResNet, DensNet ar VGG16 [VMJ19]. Problema ta, kad šie tinklai reikalauja tikslių nuotraukų

ir daug duomenų apmokymui išmokti praktiškai visoms variacijoms (gali būti mažesnis tikslumas įvairesniems egzemplioriams), dėl to aptarsime, kaip CapsNet neuroninis tinklas gali padėti išspręsti šias problemas.

2.2.1. Tenzoriai ir vektorizacija

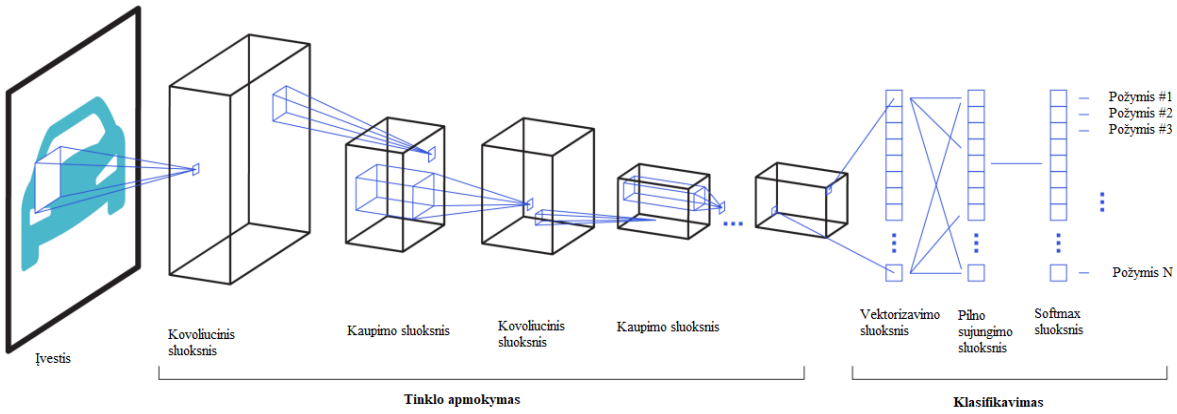
Tarkime x žymės vektorių, pavyzdžiui, $x \in R^D$ yra stulpelio vektorius su D elementų arba matrica su 1 stulpeliu ir D eilučių; X žymės matricą, tai $X \in R^{H \times W}$ yra matrica su H eilučių ir W stulpelių. Šie konceptai gali būti generalizuojami kaip aukštesnio laipsnio matricos, arba tiksliau - tenzoriai. Pavyzdžiui, $x \in R^{H \times W \times D}$ yra 3-čio laipsnio (angl. third order) tenzorius. Jam priklauso HWD elementai, ir kiekvienas iš jų gali būti indeksuotas trejetu (i, j, d), kai $0 \leq i \leq H, 0 \leq j \leq W, 0 \leq d \leq D$. Trečio laipsnio tenzorius dar gali būti traktuojamas, kaip tenzorius, turintis D kanalų. Kiekvienas kanalas yra matrica su dydžiu $H \times W$. Pirmas kanalas turi visus tenzoriaus skaičius, kurie indeksuojami $(i, j, 0)$. Kai $D = 1$, tai trečio laipsnio tenzorius sumažėja iki matricos. Kaip pavyzdys, spalvotas paveiksliukas yra trečio laipsnio tenzorius. Vaizdas su H eilučių ir W stulpelių yra tenzorius su dydžiu $H \times W \times 3$. Jeigu vaizdas yra išsaugotas RGB formatu, tai jis turi 3 kanalus, kiekvienam kanalui $H \times W$ matricą, kuri turi R (arba G, arba B) reikšmes kiekvienam pikseliui [W17].

Yra naudinga reprezentuoti vaizdus (ar kitokio tipo informaciją), kaip tenzorius. Ankstyvajame kompiuterių vaizdų ir atsikartojimų (angl. pattern) atpažinime, spalvotas vaizdo failas dažnai būdavo konvertuojamas į juodai baltą (angl. greyscale) vaizdo versiją. Tokius failus būdavo paprasčiau apdoroti ir buvo daugiau patvirtintų algoritmų dirbant su matricomis nei su tenzoriais. Failo konvertavimas neišlaikydavo informacijos apie spalvas. Spalvos yra labai svarbi dalis vaizdų palyginimui ar kitoms problemoms spręsti ir vaizdo spalvų informacija bus apdorojama naudojantis CNN. Tenzoriai yra esminis CNN komponentas. Įvestis, tarpinė reprezentacija ir parametrai CNN tinkle yra visi tenzoriai. CNN dažnai naudoja aukštesnio nei 3-čio laipsnio tenzorius, pavyzdžiui konvoliucinė sumavimo matrica (angl. kernel) konvoliuciniame sluoksnyje suformuoja 4-to laipsnio tenzorių. Paėmus bet kokį tenzorių, jį galima perskirstyti į ilgą vektorių, kuris seka numatytą eiliškumą, tai vadinama vektorizavimu [W17].

2.2.2. Standartinė CNN architektūra

Kaip įvestį CNN dažniausiai priima trečios eilės tenzorių, pavyzdžiui vaizdo failo tenzorių su H eilučių, W stulpelių ir 3-ejais kanalais (R, G, B spalvų kanalai). Įvestis tada eina tada per seriją apdorojimų (skaičiavimų). Vienas iš apdorojimo žingsnių dažniausiai vadinamas sluoksniu (angl. layer), kuris gali būti konvoliucinis sluoksnis, kaupimo sluoksnis (angl. pooling layer), normalizavimo sluoksnis, pilnai prijungtas sluoksnis, nuostolių sluoksnis (angl. loss layer) ir t.t.

$$x^1 \rightarrow \boxed{w^1} \rightarrow x^2 \rightarrow \cdots \rightarrow x^{L-1} \rightarrow \boxed{w^{L-1}} \rightarrow x^L \rightarrow \boxed{w^L} \rightarrow z \qquad (1)$$



2 Pav. Konvoliucinio neuronio tinklo pavyzdys

Formulė (1) parodo, kaip CNN pateikti duomenys apdorojami pereinant per sluoksnius. Įvestis žymima kaip x^1, dažniausiai vaizdo failas (trečios eilės tenzorius). Įvestis paduodama apdorojimui pirmajam sluoksniui (žr. 1 pav.). Parametrai naudoti pirmo sluoksnio skaičiavimuose bendrai pažymėti, kaip tenzorius w^1. Pirmojo sluoksnio išvestis yra x^2, sekančio sluoksnio įvestis [W17].

Skaičiavimai vyksta tol, kol praeinami visi CNN sluoksniai, kas sukuria išvestį x^L. Tinklo išvestis naudojama nustatyti apskaičiuoti nuostolių (angl. loss) funkcijos reikšmę ir palyginti ją su jau išmokta nuostolių funkcijos reikšme. Mažesnė nuostolių funkcijos reikšmė leidžia atnaujinti neuronų parametrų reikšmes, tam naudojamas atgalinės sklaidos metodas (angl. backpropagation). Toks būdas vadinamas mokymasis su instruktoriu (angl. supervised learning). Konvoliucinių sluoksnių skaičiavimų rezultatai būna vektorizuojami (arba dar

vadinama angl. flattening). Sekantis žingsnis ilgo vektoriaus parametrus perduoti į pilnai sujungtą sluoksnį (angl. fully connected layer arba dense layer). Taip gautos savybės iš konvoliucinių sluoksnių dar kartą apdorojamos ir išmokstama, kurios savybės yra naudingesnės. Tarkime, sprendžiama klasifikavimo problema su C klasių. Dažniausiai naudojama strategija yra išvesti x^L kaip C dimensijų vektorių, kurios i-tasis elementas nusako tikimybę ar atitinka klasę ar ne. Kad paversti x^L į tikimybės svorio funkciją (angl. probability mass function), $L-1$ sluoksnis dažniausiai naudoja softmax arba sigmoid aktyvavimo funkcijas (žr. 1 pav.). Reikia atkreipti dėmesį, kad kai kurie sluoksniai neturi jokių parametrų, tai yra, w_i gali būti tuščias kai kuriems i, softmax sluoksnis yra vienas iš pavyzdžių [W17].

Paskutinis sluoksnis yra nuostolių sluoksnis (angl. loss layer). Jei x^L yra C dimensijų vektorius, kur kiekvienas elementas nusako tikimybę, tai nuostolių sluoksnio esmė yra suskaičiuoti kiek neuroninis tinklas suklydo lyginant su pagrįsta tiesa (angl. ground truth). Tarkime t yra atitinkama reikšmė (pagrįsta tiesa, reikšmė lygi 1 toms klasėms, kurios atitinka įvestį, 0 kitoms klasėms) įvesčiai x^1, tada kaina ar nuostolio funkcija gali būti naudojama išmatuoti neatitikimus tarp CNN spėjimo ir reikšmės t. Ši funkcija padeda suprasti kokio tikslumo yra neuroninis tinklas ir leidžia jam kiekvieną epochą atnaujinti savo svorius (mokytis). Pavyzdžiui, paprasta nuostolio funkcija gali atrodyti taip:

$$z = \frac{1}{2}\|t - x^L\|^2 \tag{2}$$

Formulėje (2) nustatomas kvadratinio atstumo vidurkis tarp t ir x^L, tačiau dažnai sudėtingesnės atstumo funkcijos yra naudojamos. Klasifikavimo problemai spręsti dažnai naudojama kryžminė entropija (angl. cross entropy). Kategorinis kintamasis t pirma sukonvertuojamas į C erdvinį (angl. dimensional) vektorių t'. Tada tiek t', tiek x^L yra tikimybės svorio funkcijos ir kryžminės entropijos nuostolio funkcija gali išmatuoti atstumą tarp jų. Dėl to galima naudojantis gradientinio nusileidimo metodu (angl. gradient descent) atnaujinti w_i reikšmes ir taip minimizuoti kryžminę entropiją [W17, N18]. Dronų navigacijos problemai naudosime trejetų nuostolių funkciją, apie kurią daugiau kitame skyriuje.

2.2.3. Paketų normalizavimas

Neuroniniai tinklai besimokydami gali keisti vidinių mazgų dispersijas (angl. internal covariate shift). Straipsnis [IS15] pristatė paketų normalizavimą (angl. batch normalization),

kuris sumažina tinklo mazgų dispersiją ir taip paspartina tinklo mokymąsi. Jis tai pasiekia per normalizavimo žingsnį, kuris mažina dispersiją ir vidurkį sekančio sluoksnio įvesčiai. Paketų normalizavimas, taip pat, gerina tinklo gradientą, sumažindamas priklausomybę nuo parametrų tarpusavio santykio ar pradinių jų reikšmių. Tai leidžia turėti didesnį mokymosi greitį be nukrypimo rizikos [IS15]. Taip pat, paketų normalizavimas subendrina (angl. regularizes) modelį ir padaro Dropout sluoksnį nebereikalingą [SNHG+14]. Aukščiausią tikslumą pasiekęs tinklas [ALPH+20] su MNIST duomenimis naudojo paprastus konvoliucinius sluoksnius su paketų normalizavimu (pasiekė 99.91% tikslumą). Tyrimai parodė, kad geriausi rezultatai buvo pasiekti naudojant paketų normalizavimą po kiekvienu konvoliuciniu sluoksniu ir pilnai sujungtu sluoksniu.

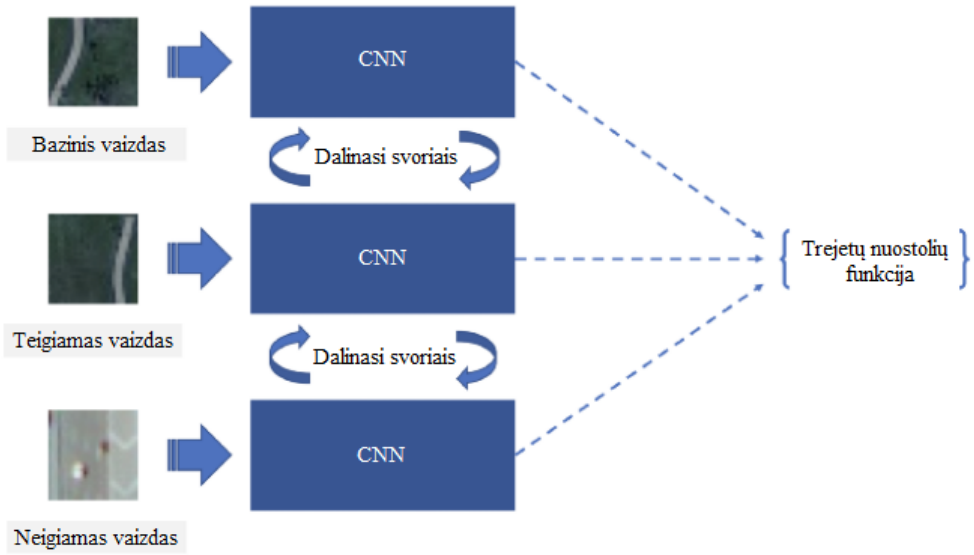
2.2.4. Trejetų nuostolių funkcijos modelio architektūra

Pramonėje sutinkamos veidų atpažinimo, objektų atpažinimo, robotų navigacijos problemos yra traktuojamos kaip daugiaklasės klasifikavimo problemos. Standartiniems klasifikavimo neuroniniams tinklams klasių kiekis yra per didelis, nes yra per daug išvesties taškų. Idėja yra išmokti paskirstytų duomenų taškų reprezentaciją taip, kad panašūs duomenų taškai būtų projektuojami į arti esantį regioną, tuo tarpu nepanašūs duomenų taškai yra projektuojami toli vienas nuo kito. Trejetų neuroninio tinklo architektūra padeda išmokti paskirstytų duomenų taškų reprezentaciją naudojant panašumo ir nepanašumo notaciją. Tai yra neuroninių tinklų architektūra, kur keli paraleliai egzistuojantys neuroniniai tinklai mokosi ir dalinasi savo svoriais tuo pačiu metu. Spėjimo metu įvesties duomenys yra duomenys yra apdorojami kiekvieno tinklo ir gaunama įterpinio (angl. embedding) išvestis [D19, SKFP15, DS18].

Trejetų tinklui reikia atitinkamai paruošti duomenis. Duomenys turi susidaryti iš trejetų (bazinis, teigiamas, neigiamas) susidarančių iš bazinio vaizdo, teigiamo vaizdo (kuris panašus į bazinį vaizdą) ir neigiamo vaizdo (kuris nepanašus į bazinį vaizdą). Jeigu duomenų rinkinys turi daug klasių, tada vaizdai tos pačios klasės gali būti laikomi panašiais ir vaizdai iš skirtingų klasių gali būti laikomi ne panašiais [D19, SKFP15, DS18]. Šio darbo atveju klasių nėra, o yra daug žemėlapio nuotraukų kurios tarpusavyje panašios ir nepanašios. Tai mažina savybių skirtumus ir didina problemos kompleksiškumą.

$$L(a,p,n) = max(0, D(a,p) - D(a,n) + margin) \quad (3)$$

Trejetų nuostolių (angl. loss) funkcija, apibrėžta (3), kur $D(x, y)$, atstumas tarp išmokto vektoriaus x ir y reprezentacijos. Gali būti naudojama atstumo metrika „L2 atstumas“ arba „(1 - cosine similarity)“[D19]. Šios funkcijos tikslas yra nustatyti atstumą tarp bazinio ir teigiamo vaizdo tokį, kad jis būtų mažesnis už atstumą tarp bazinio ir neigiamo vaizdo. Trejetų nuostolių funkcija paremtas modelis susideda iš trijų identiškų neuroninių tinklų, kurie dalinasi savo svoriais. Paskutinis sluoksnis turi D skaičių neuronų, kad išmoktų D dimensijų vektorių reprezentaciją. Teigiamų ir neigiamų atstumų vektorių reprezentacios yra atimamamos viena iš kitos, taip gaunant atstumą tarp panašių ir nepanašių vaizdų *np*. Tada prie šio *np* atstumo pridedamas vaizdų atstumo matas *margin*. Šis matas reikalingas padidinti atstumus tarp panašių ir nepanašių vaizdų mokymosi metu. Tai veikia, nes padidinama galutinė nuostolių funkcijos reikšmė leidžianti *np* atstumui būti neigiamam (nuostolių funkcija visada turi būti virš nulio norint apskaičiuoti gradientą). Teigiami, neigiami ir baziniai vaizdai paduodami jiems skirtiems neuroniniams tinklams. Pabaigoje, atgalinės sklaidos metu, svorių vektoriai yra atnaujinami jeigu gautos nuostolių funkcijos reikšmės yra mažesnės nei paskutinės gautos nuostolių reikšmės [D19, SKFP15, DS18].



3 Pav. Trejetų nuostolių funkcijos architektūra

2.2.5. Kapsulių neuroninio tinklo modelis

Straipsnis, kuris pristatė CapsNet architektūrą, vadinasi „Dynamic Routing Between Capsules“ [SFH17]. Šiame straipsnyje pristatytas naujas neuroninių tinklų modelis vaizdams klasifikuoti, kuris iš principo veikia kitaip negu standartiniai konvoliuciniai neuroniniai tinklai. Pagrindinis šio modelio privalumas - kad jis gali išlaikyti hierarchinį erdvinį santykį; teoriškai, ši architektūra gali greičiau mokytis ir naudoti mažiau mokymosi duomenų per klasę. Taip pat, straipsnis pateikia išsamų mokomą CapsNet modelį ir modelio rezultatus. Ant MNIST duomenų rinkinio buvo pasiektas 0,25% klasifikavimo klaidos dažnis, 5,2% and MultiMNIST, 10,6% ant CIFAR10, 2,7% ant smallNORB.

Kapsulių neuroninio tinklo idėja yra suskaidyti tinklo sluoksnius į kapsules kiekvienam bruožui įsiminti ir pagal paduotą įvestį nuspėti, kurios kapsulės turi ją apdoroti naudojantis tinkle esančiu „dynamic capsule routing“ (trump. DCR) mechanizmu. Neuronų darbas aktyvioje kapsulėje reprezentuoja įvairias vaizde esančio objekto savybes. Šios savybės gali susidėti iš daugybės skirtingų momentinio parametro tipų, kaip poza (pozicija, dydis, orientacija), deformacija, greitis, albedas, atspalvis, tekstūra ir panašiai. Viso tinklo sluoksniai yra išskaidyti į kapsules, kurios grąžina vektorių (atliekama vektorizacija). Kadangi kapsulės grąžina vektorių, tai leidžia DCR mechanizmui užtikrinti, kad kapsulės išvestis bus nusiųsta tinkamai tėvinei kapsulei viršutiniame sluoksnyje. Jeigu kapsulės išvestis būtų skaliaras, tai nebūtų galima nustatyti tėvinės klasės dėl per mažo informacijos kiekio (tokiu atveju kiti metodai yra naudojami, pvz.: atgalinės sklaidos metodas). Pradžioje, išvestis yra nuvedama visoms įmanomoms tėvinėms kapsulėms, bet išvesties mastas yra sumažinamas naudojantis poravimo koeficientais, kurių suma yra 1. Kapsulė apskaičiuoja „spėjimo vektorių“ kiekvienam tėvui sudauginant savo išvestį su svorių matrica (5). Jeigu spėjimo vektorius turi didelį skaliarinį produktą su numanomo tėvo išvestimi, tai sukuriamas iš viršaus į apačią atgalinis ryšys, kuris padidina poravimo koeficientą tam tėvui ir sumažina kitiems tėvams. Toks „routing-by-agreement“ tipas turėtų būti daug efektyvesnis nei primityvi nukreipimo forma naudojama „max-pooling“, kuri leidžia neuronams viename sluoksnyje ignoruoti visus, bet labiausiai aktyvų savybių detektorių apatiniame sluoksnyje [SFH17, MC18].

Ekvivalentiškumas

CNN dažnai gerai aptinka tam tikras savybes, tačiau jis yra mažiau efektyvus atpažinti erdvės ir atstumo santykius tarp savybių. CNN modelis gali išgauti savybes namų, gatvių,

medžių, tačiau aktyvuoti ne tą neuroną ir lokacijos nustatymas bus neteisingas. Kapsulių neuroninių tinklų neuronai turi informaciją ne tik tikimybės, bet ir savybių parametrų. Tai gali būti vektorius turintis tikimybę, orientaciją, dydį. Su šia informacija galima tiksliau nuspėti kokiai klasei priklauso vaizdas. Konceptualiai, CNN modelis naudoja daug neuronų ir sluoksnių, kad pavyktų išmokti skirtingas savybių variacijas. Kapsulių neuroninis tinklas įvairioms savybėms dalinasi ta pačia kapsule, kad sugebėtų atskirti tos pačios savybės skirtingus variantus. Ekvivalentiškumas (angl. equivariance) yra tokių objektų atpažinimas, kurie gali būti transformuojami vienas į kitą. Kapsulė atpažįsta ar vaizdas yra pasuktas 20° į kairę vietoj to, kad nuspręstų jog vaizdas sutampa su pasuktu variantu. Specialiai apmokant modelį atpažinti skirtingas transformacijas gali ekstrapoliuoti įmanomus variantus efektyviau su mažiau apmokymo duomenų [SFH17, D19].

Kapsulės išvesties skaičiavimai

Tarkime, kapsulės įvestis yra u_i ir išvestis yra v_j, abu vektoriai. Atliekama transformacija W_{ij} matricos kapsulės išvesčiai u_i prieš tai buvusio sluoksnio. Pavyzdžiui, su $p \times k$ matrica transformacija u_i į $\hat{u}_{i|j}$ (5) iš k dimensijos į p dimensiją (4). Tada apskaičiuojama pasverta suma s_j su svoriais c_{ij}. c_{ij} yra poravimo koeficientai, kurie yra skaičiuojami iteruojant dinaminio suvedimo procesą ir $\sum_j^i c_{ij}$ vėliau su bus susumuotas. Konceptualiai, c_{ij} nurodo kokia tikimybė, kad kapsulė i gali aktyvuoti kapsulę j. Vietoj standartinės ReLU funkcijos, paskaičiuojama suspaudimo (angl. squashing) funkcija s_j (6), taip, kad kapsulės paskutinio vektoriaus išvestis v_j (7) turi dydžius tarp 0 ir 1. Ši funkcija suspaudžia mažus vektorius į 0 ir didelius vektorius į normalizuotus vektorius (angl. unit vectors) [SFH17, D19].

$$(p \times k) \times (k \times 1) \Longrightarrow p \times 1 \tag{4}$$

Spėjimo vektorius $\hat{u}_{i|j}$ (angl. prediction vector):

$$\hat{u}_{i|j} = W_{ij} u_i \tag{5}$$

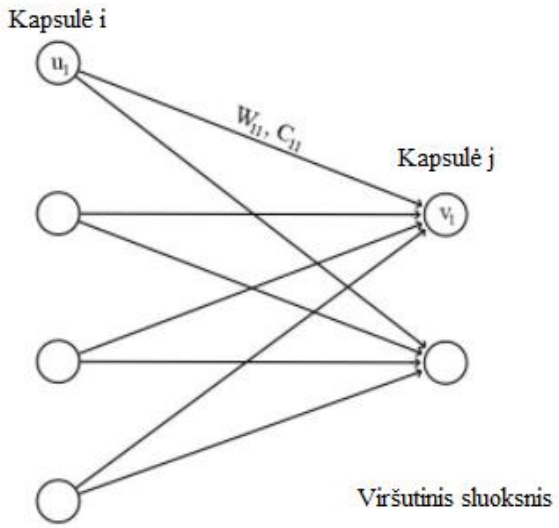
Aktyvumo vektoriaus v_j apskaičiavimas:

$$s_j = \sum_i c_{ij} \hat{u}_{i|j} \tag{6}$$

$$v_j = \frac{\|s_j\|^2}{1 + \|s_j\|^2} \frac{s_j}{\|s_j\|} \tag{7}$$

$v_j \approx \|s_j\| s_j$ kai s_j yra mažas.

$v_j \approx \frac{s_j}{\|s_j\|}$ kai s_j yra didelis.



4 Pav. Kapsulių tarpusavio komunikavimo pavyzdys. Kapsulių įvestis u_i ir išvestis v_j yra vektoriai.

Dinaminio suvedimo iteravimas

Kad būtų galima paskaičiuoti kapsulės išvestį, kapsulėje yra naudojamas dinaminio suvedimo iteravimas gauti poravimo koeficientui c_{ij}. Spėjimo vektorius $\hat{u}_{i|j}$ yra balsavimo vektorius iš kapsulės i dėl išvesties iš kapsulės j viršutiniame sluoksnyje. Jeigu aktyvumo vektorius (7) turi daug panašumų ir yra arti vienas kito, reiškia, kad abi kapsulės yra artimai susiję. Toks panašumas yra matuojamas naudojantis spėjimo ir aktyvumo vektoriaus skaliariniu produktu [SFH17, D19].

$$b_{ij} \leftarrow \hat{u}_{i|j} \cdot v_i \tag{8}$$

Panašumo balas b_{ij} (8) žino tiek tikimybinius, tiek savybinius parametrus, vietoj tiesiog tikimybės neuronuose. Taip pat, b_{ij} išlieka žemas jeigu i kapsulės aktyvacija u_i yra žema, kadangi $\hat{u}_{i|j}$ ilgis yra proporcingas u_i. Tada poravimo koeficientas c_{ij} yra apskaičiuojamas kaip softmax iš b_{ij} (9). Kad b_{ij} būtų tikslesnis, jis yra atnaujinamas iteratyviai, dažniausiai 3-ijomis iteracijomis (10) [SFH17, D19].

$$c_{ij} = \frac{exp\, b_{ij}}{\sum_k exp\, b_{ik}} \tag{9}$$

$$b_{ij} \leftarrow b_{ij} + \hat{u}_{i|j} \cdot v_i \tag{10}$$

Pseudo kodas, aprašantis dinaminį suvedimą (angl. dynamic routing) [SFH17]:

Procedūra 1 Suvedimo algoritmas

procedūra $Routing(\hat{u}_{i|j}, r, l)$
    visoms kapsulėms i sluoksnyje l ir kapsulėms j sluoksnyje $(l+1)$: $b_{ij} \leftarrow 0$.
    for r iterations **do**
        visoms kapsulėms i sluoksnyje l: $c_i \leftarrow softmax(b_i)$
        visoms kapsulėms j sluoksnyje $(l+1)$: $s_j \leftarrow \sum_i c_{ij}\hat{u}_{i|j}$
        visoms kapsulėms j sluoksnyje $(l+1)$: $v_j \leftarrow squash(s_j)$
        visoms kapsulėms i sluoksnyje l ir kapsulėms j sluoksnyje $(l+1)$: $b_{ij} \leftarrow b_{ij} + \hat{u}_{i|j} \cdot v_j$
    return v_j

Dinaminis suvedimas nėra visiškas atgalinės skaidos (angl. backpropagation) pakaitalas. Transformacijos matrica W yra vis tiek apmokoma, naudojantis atgalinės sklaidos metodu su kainos funkcija (angl. cost function). Skiriasi tik tuo, kad yra naudojamas dinaminis suvedimas apskaičiuoti kapsulės išvestį. Apskaičiuojama c_{ij} kad nustatyti ryšį tarp tėvinių ir vaikinių kapsulių. Ši reikšmė yra svarbi, tačiau vienkartinė. Jos reikšmė yra 0 kiekvienai įvesčiai prieš pradedant dinaminio suvedimo skaičiavimus. Kad apskaičiuoti kapsulės išvestį,

apmokymo ir testavimo metu, visada atliekami dinaminio suvedimo skaičiavimai [SFH17, D19].

2.2.6. Vėlesni kapsulių tinklo darbai

Per penkerius metus nuo pirmojo kapsulių tinklo modelio pristatymo buvo sukurta daugybė kitų modelių kurie gavo geresnius rezultatus. Tyrimai realizavo CapsNet ir tyrė skirtingas modelio variacijas, nuo pridedant daugiau sluoksnių iki keičiant hyperparametrus [BJ17]. Straipsniai, kaip [PDS18, MC18, VBB19, HZ20, MSC21] vienas už kitą vis pagerindavo tikslumą su įvairiais duomenimis (MNIST, CIFAR10 ir pan.). Iš jų geriausią tikslumą pasiekęs [MSC21] yra top 4 tinklas pagal pasiektą tikslumą su MNIST duomenų rinkiniu[1]. Šis tinklas ne tik padidino tikslumą, bet ir išsprendė kai kurias problemas su CapsNet. CapsNet tinklas yra aktyviai tiriamas, ir nors gaunamas aukštas tikslumas su MNIST ir panašiais duomenimis, vis dar neprilygsta standartiniams konvoliuciniams tinklams, kaip ResNet. Pagrindinė CapsNet problema, kad jis persimoko ir parametrų kiekis išauga eksponentiškai didinant kapsules ar vaizdų rezoliuciją. Darbas su ilgais vektoriais taip pat reikalauja didelio kiekio atminties. Straipsnyje [MSC21] teigiama, kad jų sukurtas modelis - Efficient-CapsNet - sąlyginai išsprendžia šias problemas. Tinklas naudoja žymiai mažiau parametrų (angl. by order of magnitude), dėl to greičiau mokosi ir naudojamas dinaminis dėmėsio (angl. dynamic attention) algoritmas išsprendžia persimokymo problemą.

2.2.7. Atstumo funkcija

Šimtai įvairių algoritmų vaizdų palyginimui buvo sukurta per paskutinius 30 metų [SEI20]. Viename pirmųjų darbų apie vaizdų panašumų nustatymą „phase residual“ ir „Fourier ring“ koreliacijos ir funkcinės priemonės buvo aprašytos [WH87]. A.J. Baddeley buvo vienas pirmųjų, kuris sistematiškai studijavo nespalvotų vaizdų palyginimą. Vėliau panašumo matavimai buvo aprašyti ir spalvotiems vaizdams [SO95, SEI20]. Dažniausiai naudojama atstumo funkcija vaizdų palyginimui – Euklido atstumas. Atstumo funkcija yra vienas iš esminių dalykų, nulemiančių, kaip neuroninis tinklas gerai išmoksta atskirti klases vieną nuo kitos. Įvairioms problemoms spręsti naudojamos skirtingos atstumo funkcijos [SEI20]. Šiuo atveju, sprendžiant dronų navigacijos problemą, pasitelkiant vaizdų palygimo metodą, reikia

[1] https://paperswithcode.com/sota/image-classification-on-mnist?p=efficient-capsnet-capsule-network-with-self

naudoti atstumo funkcijas, skirtas nustatyti atstumą tarp panašių ir nepanašių vaizdų. Neuroniniam tinklui, kuris skaičiuoja vaizdo savybių vektorius, reikalingos ir metrikos, galinčios pamatuoti, kokiu tikslumu jis tą atlieka. Gali būti naudojamos šios metrikos tam atlikti: vidutinis tikslumas (angl. accuracy), nuostolių funkcijos rezultatas (angl. loss).

Tikslumo metrika

Tikslumui (angl. accuracy) matuoti reikalinga kokia nors atstumo metrika d, pagal kurią bazinis paveikslėlis a yra arčiau teigiamo paveikslėlio p ir toliau nuo neigiamo n. Šioje nelygybėje nuostolių funkcijoje pasirinkta atstumo konstanta $margin$ parodo kiek įterpinių neuroninis tinklas atskyrė per tam tikrą atstumą.

$$d(a,p) + margin < d(a,n) \tag{11}$$

Jeigu rezultatas tenkina tokią sąlygą (11), laikoma, kad tinklas teisingai įvertino trejetą. Taip galime paskaičiuoti vidurkį visam duomenų rinkiniui N. Bendrą tikslumą tinklui galime apskaičiuoti:

$$tikslumas = \sum_{i=1}^{N} \frac{1_{d(a,p) + margin < d(a,n)}}{N} \tag{12}$$

3. Eksperimentai

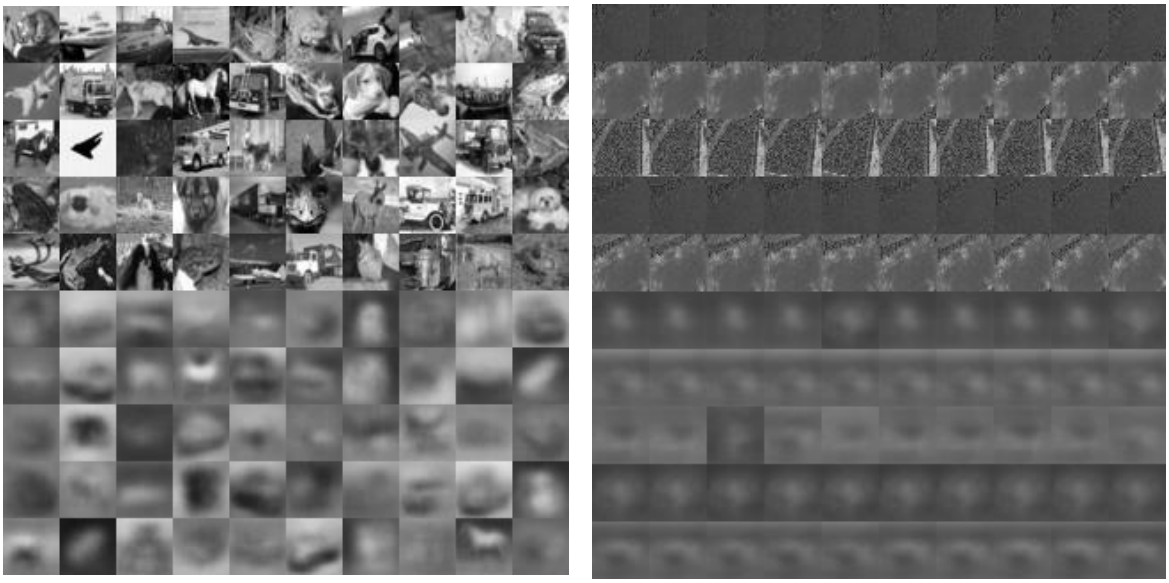
Šiame skyriuje aptarsime, kaip buvo atlikti eksperimentai. Aptarsime technologijas, kurios buvo naudojamos apmokyti neuroninius tinklus, bei pateiksime apmokytų neuroninių tinklų apžvalgą ir rezultatus. Darbo tikslas - parodyti CapsNet tinklo tikslumą vaizdų atpažinime ir palyginti juos su ResNet neuroninio tinklo realizacija. Tam reikia paruošti duomenis, atitinkančius sprendžiamą problemą, ir juos paduoti į apmokytus tinklų modelius, paimti geriausius gautus tinklo svorius iš apmokymo epochų (vertinant pagal mažiausią nuostolio reikšmę) ir įvertinti CapsNet tinklo veikimą lyginant su ResNet.

3.1. Duomenų rinkinių paruošimas

Pirmiausia, norint pradėti naudoti neuroninius tinklus su trejetais, reikia juos paruošti. Neuroniniai tinklai gali turėti milijonus parametrų ir laiko bei atminties kompleksiškumas gali smarkiai išaugti naudojant aukštos rezoliucijos nuotraukas neuroniniuose tinkluose, arba tinklus su daug parametrų (daugiau sluoksnių ir pan.). Šių tinklų apmokymui naudojama sistema turi Geforce RTX 3070ti 8GB vaizdo plokštę ir 16GB RAM atminties. Ši sistema yra labai ribojanti, pavyzdžiui Amazon leidžia išsinuomoti sistemas su 500GB+ atminties ir keliomis vaizdo plokštėmis. Atsižvelgiant į turimus sistemos parametrus, reikia atitinkamai apdoroti duomenis, nes, pavyzdžiui, 3 kanalų 64x64 rezoliucijos nuotraukoms reikia maždaug dvigubai daugiau atminties ir laiko apdoroti ir apmokyti tinkle, nei 32x32 rezoliucijos nuotraukoms. Tam, kad užtektų atminties, paveiksliukus reikia sumažinti prieš paduodant į neuroninį tinklą. Darant išvadas, reikia atsižvelgti į tai, kad buvo naudotos mažos rezoliucijos nuotraukos, nes jos gali padaryti tam tikras savybes sunkiau atpažįstamas.

CapsNet neuroninio tinklo realizacija dar nėra pridėta prie nei vieno tinklams apmokyti skirto karkaso. Kad jis efektyviau veiktų trejetuose, reikia pirma jį paruošti. Šviežiai sukurti neuroniniai tinklai gali turėti nulinius svorius, atsitiktinius svorius arba svorius, kurie buvo gauti apmokant tinklą su kažkokiais duomenimis. Šiuo atveju CapsNet tinklą galima paruošti, naudojantis MNIST arba CIFAR10 duomenų rinkiniais. Šie duomenys turi nemažai nuotraukų ir abu naudoja 10 klasių. Juos lengva naudoti, kadangi Tensorflow Keras turi API juos pakrauti tiesiai į atmintį. Didesnės apimties duomenų rinkiniams, kaip ImageNet, reikėtų naudoti galingesnę sistemą, kadangi jie turi apie 1000 klasių ir daug daugiau nuotraukų. Reikia turėti

omenyje, kad klasių kiekis, taip pat, prideda daugiau dimensijų, nes tinklui reikia išmokti daugiau savybių, kad galėtų efektyviai klasifikuoti. Hinton et al. [SFH17] darbe naudojo MNIST duomenų rinkinį, tačiau šis rinkinys gali duoti prastesnius rezultatus trejetų tinkle nei CIFAR10 dėl dviejų priežasčių: pirma, jis turi per mažai detalių, todėl konvoliucinio sluoksnio filtrai gali netiksliai klasifikuoti; antra, jis turi tik nespalvotas nuotraukas, kas automatiškai nuima dvi papildomas dimensijas. Spalvos gali būti ypač svarbios ortografiniuose vaizduose, pavyzdžiui, norint atskirti lauką nuo vandens ar panašiai. CIFAR10 duomenų rinkinys turi 10 skirtingų klasių, kurių kiekviena turi 6000 nuotraukų su 32x32 rezoliucija. Šitas duomenų rinkinys puikiai tinka paruošti CapsNet tinklą dėl jo nedidelio dimensijų kiekio ir įvairių klasių. Su šiuo rinkiniu CapsNet tinklas turi virš 11 milijonų parametrų.



5 Pav. Kapsulių neuroninio išvestis, rekonstruotos nuotraukos. Kairėje CIFAR10 50 egzempliorių, dešinėje ortografinių vaizdų 50 egzempliorių ir atitinkamai 50 rekonstruotų nuotraukų pavyzdžių

ResNet tinklo realizacija yra pridėta prie Tensorflow Keras karkaso API. Kadangi tai yra labai gilus tinklas; šiame eksperimente naudojamas ResNet50 turintis virš 50 sluoksnių (galimi tinklai su 100 sluoksnių ir daugiau). Tensorflow Keras leidžia naudoti jau apmokytus tinklo svorius. Šiuo atveju bus naudojami svoriai, gauti apmokius su ImageNet duomenų rinkiniu. Kadangi šis duomenų rinkinys turi daug klasių ir aukštos rezoliucijos nuotraukas, filtrai bus daug geresnės kokybės nei CapsNet neuroninio tinklo. Tačiau šie filtrai keisis trejetų tinklo mokymosi metu, tai neturėtų daryti didelės įtakos eksperimentams. ResNet tinklas su 32x32x3 dydžio įvestim turės virš 28 milijonų parametrų.

Pradinis ortografinių duomenų rinkinys susideda iš ~3700 nuotraukų mokymui ir ~200 nuotraukų validacijai. Kadangi šis rinkinys bus naudojamas trejetų tinklui apmokyti (tiek su ResNet, tiek su CapsNet) gausis ~1300 trejetų. Kad geriau ištestuoti ir pritaikyti duomenis problemai spręsti, nuotraukas būtina transformuoti. Šiuo atveju bus naudojama nuotraukos skirtingo laipsnio pasukimo transformacija. Kiekviena bazinė (angl. anchor) nuotrauka turės nuo -10° iki +10° transformacijas. Dėl patogumo nuotraukos transformuojamos į 10 skirtingų (~2.5° atstumai kiekvienam trejetui), gaudami ~13000 skirtingų trejetų apmokymui ir apie ~600 validavimui. Šios transformacijos naudingos problemos sprendimui, kadangi dronai skrisdami ne visada gali tiksliai nustatyti, kur yra šiaurė (~10 laipsnių paklaida), todėl tinklas turi būti paruoštas nepriklausomai nuo transformacijos rasti artimiausią vaizdą. Artimiausio vaizdo radimas, nepriklausomai nuo transformacijos, naudingas, kad dronas galėtų iš visos ar dalinės nuotraukų bazės rasti panašiausią vaizdą nustatyti savo buvimo vietai. Dėl to svarbu, kad atstumai tarp panašių ir nepanašių vaizdų būtų kuo didesni. Turimų duomenų nuotraukos pradinė rezoliucija yra 640x480. Kad būtų galima paduoti į tinklą, užkrovimo metu vykdoma pasukimo transformacija ir sumažinama iki 32x32 rezoliucijos. Tokiu būdu prarandama nemažai savybių kurios gali būti naudingos tinklui, tačiau smarkiai sumažinamas tinklo dydis ir reikiamas atminties kiekis mokymui. Tai yra būtina, kadangi nuotraukų dydis eksponentiškai didina tinklo parametrų kiekį ir mokymo trukmę. Iš pateiktų duomenų nuotraukos (žr. 6 pav.) matosi kad baziniai paveiksliukai yra žymiai tamsesni, kadangi nuotraukos atliktos tamsesniu paros metu. Atlikus kelis bandymus buvo pastabėta, kad didelis nuotraukų spalvų skirtumas gali tiesiogiai įtakoti rezultatus ir apsunkinti tinklo mokymąsi, todėl buvo atliktas duomenų normalizavimas. Formulė duomenų standartizavimui: c – kanalai, x_c – kanalo pikselių matrica, $mean_c$ – kanalo vidurkis, std_c – kanalo standartinis nuokrypis.

$$x'_c = \frac{(x_c - mean_c)}{std_c} \tag{13}$$

Atlikus šiuos skaičiavimus, kad gautos reikšmės $x' \in [0, 1]$, atliekama minimalios-maksimalios (angl. min-max) normalizacija:

$$x' = \frac{(x - x_{min})}{x_{max} - x_{min}} \tag{14}$$

6 Pav. Pradiniai trejetų duomenys prieš normalizavimą ir transformacijas. Pirmas vaizdas yra bazinis, antras teigiamas, trečias neigiamas.

7 Pav. Trejetų pavyzdys atlikus normalizavimą ir transformacijas. Pirmas vaizdas yra bazinis, antras teigiamas, trečias neigiamas.

8 Pav. Trejetų pavyzdys atlikus normalizavimą ir transformacijas. Pirmas vaizdas yra bazinis, antras teigiamas, trečias neigiamas. Bazinis paveiksliukas yra pasuktas ~10 laipsnių nuo originalo.

3.2. Naudojamos technologijos ir įrankiai

Šiame darbe aprašytiems bandymams naudojama:

- Windows 10 operacinė sistema;
- GeForce RTX 3070ti 8GB vaizdo plokštė, 16 GB DDR3 RAM, Intel Core i5-4460 procesorius;
- Visual studio code IDE su Jupyter įskiepiu;
- Python 3.9.4 programavimo kalba, pagrindinės bibliotekos:

- Tensorflow 2.5.0rc2 - karkasas skirtas dirbti su neuroniniais tinklais;
- keras-nightly 2.5.0 - biblioteka leidžianti naudoti įvairius modelius bei sluoksnius;
  - ResNet50 išmokytas su ImageNet duomenų rinkiniu;
  - CIFAR10 duomenų rinkinys;
  - CapsNet-Keras[2] realizacija su MNIST pagal Hintono darbą, pasiekiantis 99,7% tikslumą po 50 epochų. Šio darbo eksperimentams adaptuotas dirbti su CIFAR10 duomenų rinkiniu;
- numpy - biblioteka naudojama darbui su matricomis;
- pyplot - biblioteka naudojama paveiksliukų ir grafikų sudarymui.

3.3. Kapsulių neuroninio tinklo realizacija

Pradinis kapsulių neuroninis tinklas CapsNet-Keras pritaikytas veikti su Keras ir Tensorflow, bei MNIST duomenų rinkiniu. Kad tinklas veiktų su 32x32x3 tenzoriu, užtenka pakeisti tinklo įvesties parametrus. Visos paveikslėlio paduodamos reikšmės turi būti padalintos iš 255, kad gauta reikšmė būtų tarp 0 ir 1. Bandymams realizuotos skirtingos kapsulių neuroninio tinklo realizacijos. Kai kurios realizacijos turėjo papildomų sluoksnių, naudojo paketų normalizavimą ar turėjo skirtingas konvoliucines sumavimo matricas:

- CapsNet-Hinton9 – Hinton et. al. pateiktas modelis (1 priedas);
- CapsNet-Hinton7 – modifikuotas Hinton et. al. modelis naudojantis 7x7 konvoliucines sumavimo matricas (2 priedas);
- CapsNet-Conv – modifikuotas Hinton et. al. modelis su papildomais konvoliuciniais sluoksniais su paketų normalizavimo sluoksniais (3 priedas);

[2] https://github.com/XifengGuo/CapsNet-Keras

- CapsNet-BNorm – modifikuotas Hinton et. al. modelis su papildomais konvoliuciniais sluoksniais su paketų normalizavimo sluoksniais po kiekvienu konvoliuciniu sluoksniu ir pilnai sujungtu sluoksniu (4 priedas);
- CapsNet-Mono – Hinton et. al. pateiktas modelis su nespalvotais vaizdais (5 priedas);
- CapsNet-E – Efficient-CapsNet modelis pagal [MSC21] (6 priedas).

Tinklo mokymo metu pasirinkti šie parametrai:

- Suvedimų kiekis (*routings*) - 3, kiek kartų bus atliekamas suvedimo algoritmas. Su šiuo parametru buvo gauti geriausi rezultatai pagal Hinton et. al.;
- Duomenų paketo dydis (*batch_size*) - 100, šis parametras įtakoja tinklo apmokymo greitį ir kokybę, nes po kiekvieno žingsnio (po kiekvieno paketo) pagal gautą nuostolio (angl. loss) reikšmę atnaujinami tinklo svoriai, kurie toliau naudojami iki kol baigiasi epocha. Tuo pačiu žingsnių kiekis buvo *dataset_length / bach_size*, tokiu būdu visas treniravimo rinkinys pereidavo tinklą per vieną epochą. Epochos pabaigoje svoriai su mažiausia nuostolio reikšme išsaugomi ir naudojami kitoje epochoje;
- Apmokymo greitis (angl. learning rate, *lr*) - 0,001, naudojamas kartu su **Adam** optimizatoriumi. Šiuo atveju tokio apmokymo greičio užteko (pasiektas 90% tikslumas su treniravimo duomenimis per 10-20 epochų), nes tinklo esmė yra pasiruošti filtrus, kurie bus vėliau permokomi trejetų neuroniniam tinkle;
- Rekonstravimo koeficientas (*lam_recon*) - 0.392, šis dydis naudojamas kaip SSE (angl. sum squared error) pakaitalas MSE (angl. mean squared error) skaičiavimams, identiškai Hintono straipsnyje.

Su šiais parametrais tinklo sudėtis atrodė šitaip (žr. pav. 6):

- InputLayer1 (pav. įvesties sluoksnis) - (100, 32, 32, 3) tenzorius, išvertus *(batch_size, image_height, image_width, channels)*;
- Conv2D/1 (pirmas konvoliucinis sluoksnis) - (100, 24, 24, 256) tenzorius; 62 tūks. parametrų; paprastas konvoliucinis sluoksnis turintis 256 filtrus. Skirtingose modelių realizacijose konvoliucinių sluoksnių bei filtrų kiekis ir sumavimo matricų dydžiai

skiriasi (žr. 1-6 priedus). Didesnės sumavimo matricos gali prasčiau generalizuoti ar išmokti mažiau smulkių savybių, dėl to bandymai atlikti su skirtingomis matricų konfigūracijomis;

- Conv2D/2 (kapsulių sluoksnis suformuotas iš konvoliucinio sluoksnio) - (100, 8, 8, 256) tenzorius, kuris yra transformuojamas į kapsules (8 dimensijos) ir gaunamas (100, 2048, 8) tenzorius šiame sluoksnyje naudojama suspaudimo (angl. squash) aktyvavimo funkcija; 5,3 mln. parametrų Hinton CapsNet tinkle ir vos 30 tūkst. CapsNet-E;

- Digitcaps - (100, 10, 16) tenzorius; iš kapsulių gaunami 16 dimensijų vektoriai kiekvienai klasei (10). Šis sluoksnis atlieka kapsulių suvedimo algoritmą;

- InputLayer2 (label įvesties sluoksnis) - (100, 10) tenzorius, priima klasių duomenis (angl. labels), klasė yra 10 dimensijų vektorius, kur viena reikšmė yra 1, visos kitos yra 0;

- Mask - (100, 160) tenzorius, randa didžiausio ilgio įvesties vektorių, visus kitus padaro 0, didžiausio ilgio vektorius turi informaciją arčiausią reikiamo rezultato (pagal Hintono darbą), t.y. kurią klasę tinklas spėja. Priima InputLayer2 ir Digitcaps atrasti kuri klasė buvo atspėta ir atliekamas vektorizavimas (angl. flatten);

- Decoder - šis sluoksnis yra sudarytas iš kelių sluoksnių (naudojamas kaip sluoksnis tačiau yra formuojamas kaip atskiras modelis), šis sluoksnis naudojamas dekoduoti arba, tiksliau, rekonstruoti kapsulių tinklo išvestį (žr. 5 pav.):

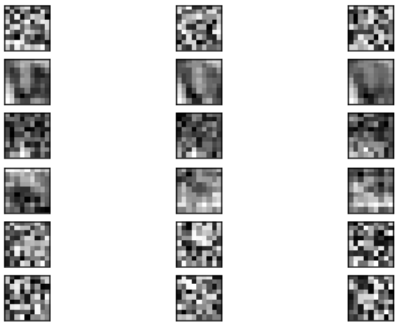
    - Dense (paslėptas sluoksnis) - relu aktyvavimo funkcija, 512 dimensijų

    - Dense (paslėptas sluoksnis) - relu aktyvavimo funkcija, 1024 dimensijų

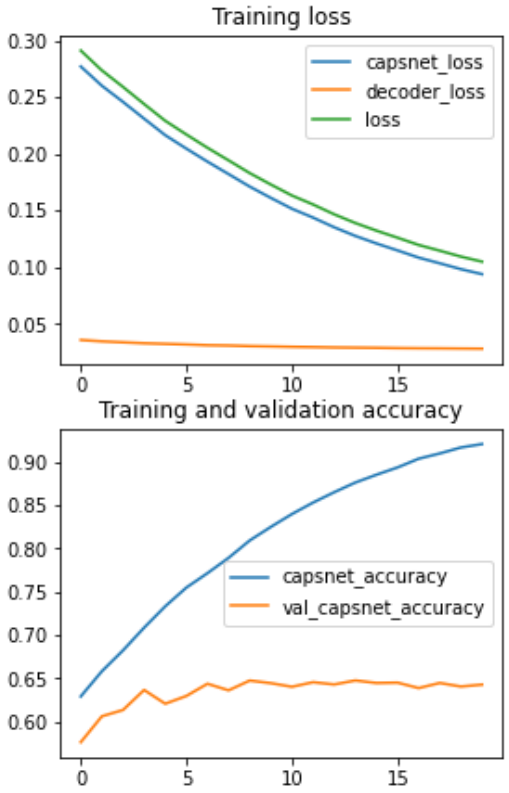
    - Dense (paslėptas sluoksnis) - sigmoid aktyvavimo funkcija, 32*32*3=3072 dimensijų

    - Reshape - praeitas sluoksnis performatuojamas į (32, 32, 3) tenzorių, tokį patį kaip ir tinklo įvesties, tam kad būtų galima atvaizduoti kaip paveiksliuką (rekonstruota praėjus pro tinklą). Šis rekonstruotas paveiksliukas bus naudojamas trejetų tinklo apmokymui.

CapsNet-Hinton9 tinklui pradėjus mokyti reikėjo 30 epochų (žr. pav. 8) pasiekti gana aukštą tikslumą (95%) su CIFAR10 apmokymo duomenų rinkiniu. Hinton et. al. savo darbe pasiekė 90% tikslumą, tačiau jis įskaičiavo ir validavimo duomenų rinkinį ir papildomai atliko karpymo (angl. crop) operacijas prieš nuotraukas paduodamas į tinklą. Šiuo atveju nėra labai svarbu aukštas tikslumas, kadangi tinklas vis tiek mokysis iš naujo trejetų tinkle, todėl tokio tikslumo filtrų turėtų užtekti pradėti tinklą naudoti trejetų tinklo mokymui. Visas apmokymas truko maždaug apie valandą, neskaičiuojant kitų bandymų, kurie davė prastesnius rezultatus. Viso mokymo esmė yra gauti reikiamus filtrus (žr. pav 7), kurie vėliau bus panaudoti apmokant trejetų tinklą. Po mokymo filtrų ir paslėptų sluoksnių svoriai yra išsaugomi, kad juos būtų galima panaudoti trejetų tinkle. Visiems kitiems tinklams apmokyti užtruko panašiai laiko, kaip ir CapsNet-Hinton9, tačiau CapsNet-Conv ir CapsNet-E apsimokė greičiausiai ir pasiekė dar didesnį tikslumą nei kiti modeliai.



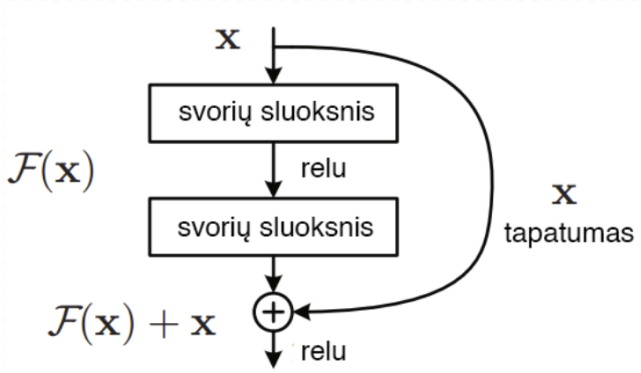
10 Pav. Kapsulių tinklo filtrų pavyzdžiai



9 Pav. Kapsulių tinklo apmokymo parodymai, paskutinės 20 epochų su CIFAR10 duomenimis

3.4. ResNet neuroninio tinklo realizacija

Kadangi ResNet tinklas jau yra realizuotas ir prieinamas naudojant Tensorflow Keras API jo atskirai realizuot nereikia. Šiam eksperimentui naudojamas ResNet50[3] tinklas [HZSS15]. Tai yra mažiausias gilusis ResNet tinklas kokį galima naudoti. Šis tinklas jau turi paruoštus filtrų svorius gautus apmokant su ImageNet duomenų rinkiniu. Kadangi jis buvo apmokytas su 224x224 rezoliucijos nuotraukomis, filtrai yra daug aukštesnės kokybės, nei CapsNet tinklo, apmokyto su CIFAR10. Potencialiai naudojant gilesnius ResNet101 ar ResNet152 galima būtų išgauti geresnius rezultatus, bet jiems reikia daugiau resursų bei duomenų. Šiuo atveju ResNet50 yra pakankamas atlikti palyginimui su CapsNet neuroniniu tinklu, kadangi jis turi 26 mln. parametrų prieš CapsNet 11 mln. parametrų. Ko šitas tinklas neturi, tai dekodavimo (angl. decoder) ar rekonstravimo sluoksnio. Šis sluoksnis bus pridedamas identiškas, kaip ir CapsNet tinklo atveju, tačiau jis apsimokys tik trejetų eksperimento metu. Kadangi tinklas yra labai gilus jo viso grafiko atvaizduoti nepraktiška (7 priedas) , tačiau pagrindinė logika yra, kad jis gali praleisti filtrus, taip išlaikantis geresnį gradientą (žr. 9 pav.).



11 Pav. ResNet tinklo išliekamasis blokas (angl. residual block)

3.5. Trejetų neuroninio tinklo realizacija

Trejetų neuroninius tinklus vadinsime TCapsNet (modifikacijos pavyzdys - TCapsNet-Hinton9) ir TResNet. Tiek TCapsNet, tiek TResNet naudoja identiškus mokymo parametrus ir veikia iš principo taip pat, vienintelis skirtumas yra paduotas modelis su savo svoriais (skiriasi

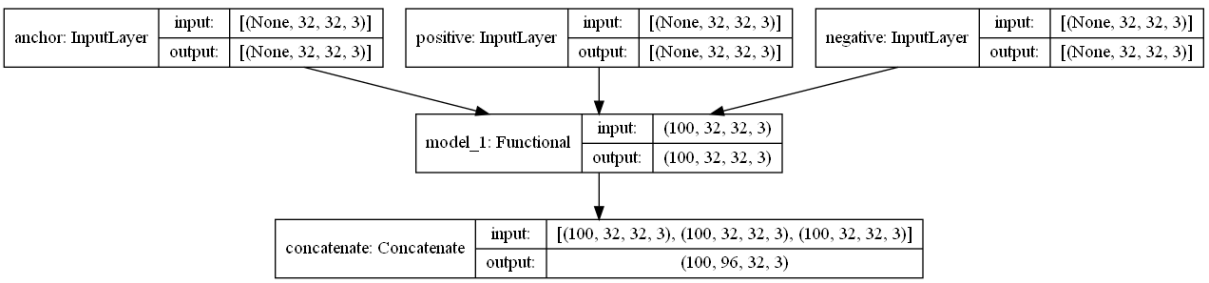
[3] https://keras.io/api/applications/resnet/#resnet50-function

išmokti filtrai). CapsNet ir ResNet yra naudojami, kaip įterpiamasis (angl. embedding) sluoksnis, kurio paskirtis gauti rekonstruotus paveiksliukus. Šiuos paveiksliukus naudosime apskaičiuoti bendrą tinklo trejetų nuostolio reikšmę. Trejetų tinklas naudoja šiuos parametrus:

- learning rate - 0,0006, naudojamas dar mažesnis apmokymo greitis dėl mažų atstumų tarp vaizdų, juos reikia didinti iš lėto. Per didelis mokymosi greitis gali greitai permokyti tinklą;
- batch size - 100, naudojamas tas pats dydis kaip ir CapsNet tinkle, kad atitiktų įvesties tenzorius. CapsNet tinkle tenzoriai atsimena šitą parametrą todėl norint jį pakeisti reiktų permokyti tinklą;
- triplet loss constant - 0,2, konstanta naudojama nuostolio funkcijos reikšmės skaičiavimuose, ta pati reikšmė buvo naudojama ir [SKFP15]. Jos pagrindinė idėja yra padidinti nepanašių vaizdų ir panašių vaizdų skirtumą pridedant nekintamą atstumą. Kadangi loss reikšmės negali būti neigiamos, ši konstanta leidžia išmokti didesnius atstumus tarp nepanašių vaizdų.

Su šiais parametrais gautas neuroninis tinklas (žr. 10 pav.):

- Įvestis susideda iš 3-ijų įvesties tenzorių su tuo pačiu dydžiu. Bazės (anchor), teigiamas (positive), neigiamas (negative) įvesties tenzoriai priima atitinkamai paveiksliukus ir juos paduoda į sekantį sluoksnį, kuris yra CapsNet ar ResNet neuroninis tinklas.
- Antrame sluoksnyje esantis neuroninis tinklas prafiltruoja nuotraukas ir grąžina dekoduotą spėjimo tenzorių, kurio forma (angl. shape) sutampa su įvesties forma, t.y. (32,32,3).
- Paskutiniame sluoksnyje neuroninių tinklų rezultatas sujungiamas į vieną matricą ir paduodamas į trejetų nuostolio funkciją.



12 Pav. Trejetų neuroninio tinklo modelis

Trejetų formavimo metodai gali būti skirtingi. Galima imti duomenis - iš eilės, atsitiktinai, arba parinkti smarkiai skiriančius (angl. hard batch), švelniai skiriančius (angl. soft batch), skirtumo per nuostolių funkcijos atstumo konstantos (angl. semi-hard batch) trejetus. Šiuo atveju, visi elementai viename pakete (batch_size = 100) imti atsitiktiniu būdu, tuo pačiu ir ant bazinio paveiksliuko atlikta atsitiktinė transformacija (nuo -10° iki +10° kampo pasukimas kas 2,5°). Taip kiekvienas paketas turi labai skirtingus duomenis iš viso duomenų rinkinio. Tai yra būtina norint paspartinti mokymąsi ir nepermokyti ant vienodų duomenų. Vienintelis tokio metodo trūkumas, kad pradinė nuostolių funkcija gali smarkiai skirtis tarp bandymų ir gaunami labai įvairūs pradiniai rezultatai, nes pradinis gradientas (kartu ir pradiniai svoriai) gaunasi vis kitoks. Iš principo, tai neturi labai didelės įtakos galutiniam rezultatui, kadangi po 10 epochų rezultatai suvienodėja, skirtumas tik, kad tinklas gali lėčiau mokytis nei su semi-hard batch metodu.

3.6. Trejetų neuroninių tinklų bandymai

Šio darbo tikslas yra parodyti, kad kapsulių tinklas gali pasiekti geresnius rezultatus su duomenų pasukimais trejetų modelyje naudodamas mažiau parametrų. Visi trejetų tinklai naudojo tą pačią konfigūraciją ir tą patį dekodavimo modelį (dekodavimo modelio parametrų kiekis vienodas). Visi tinklai buvo mokomi tol kol per kelias epochas tinklas pradeda nebesimokyti (nebemažėja loss reikšmė). ResNet ir CapsNet įterpiniai atitinkamai buvo gauti iš Keras ImageNet ir apmokant su CIFAR10. Epocha (angl. epoch) kurioje buvo gauti geriausi tinklo rezultatai. Mokymo metu buvo stebimi nuostolių funkcijos (angl. loss), tikslumo (angl. train_acc), validavimo nuostolių funkcijos (lentelėje val_loss) ir validavimo tikslumo (lentelėje val_acc) parodymai. Taip pat gautas tikslumas su testavimo duomenimis (lentelėje test_acc) ir tikslumas (skliausteliuose) pridedant 0.01 atstumą (paklaidos atstumas, *pa*) norint įsitikinti kiek neigiamo vaizdo tinklo išvesties rezultatų yra panašių į bazinį vaizdą:

1 Lentelė. Apmokytų trejetų neuroninių tinklų parodymai

tinklas	epoch	loss	val_loss	train_acc	val_acc	test_acc
TCapsNet-Hinton9	57	0.009	**0.084**	0.89	**0.58**	0.74 (0.71)
TCapsNet-Hinton7	47	0.042	0.13	0.63	0.40	0.66 (0.63)
TCapsNet-Conv	**20**	**0.0007**	0.088	**0.98**	0.57	0.71 (0.70)
TCapsNet-BNorm	**20**	0.0064	0.127	0.91	0.42	0.58 (0.55)
TCapsNet-Mono	30	0.05	0.142	0.58	0.31	0.82 **(0.80)**
TCapsNet-E	60	0.04	0.147	0.69	0.42	0.58 (0.45)
TResNet	30	0.0092	0.1009	0.94	0.47	**0.86** (0.60)

Testavimo duomenų grafikai (9 priedas)

Buvo naudojama 10 skirtingų teigiamų porų (8 priedas), kuri kiekviena turėjo 10 skirtingų neigiamų paveiksliukų. Taip pat, kiekvienas bazinis (angl. anchor) paveiksliukas buvo pasuktas 10 kartų (viso 1000 trejetų). Kiekvienas bazinis paveiksliukas turėjo skirtingas savybes ištestuoti kokius bruožus tinklai išmoksta geriau ar blogiau. Visi testavimo duomenys buvo normalizuoti kaip ir mokymo bei validavimo duomenys.

Kiekvienas iš grafikų turi trejeto pavadinimą, pasiektą tikslumą ir pasiektą tikslumą per paklaidos atstumą (*pa*). Mėlyna spalva žymi teigiamo vaizdo ir bazinio vaizdo skirtumus, oranžinė žymi neigiamo vaizdo ir bazinio vaizdo skirtumus. Skirtumas apskaičiuojamas Euklido atstumo funkcija kiekvienam pikseliui, ir iš jų visų išvedamas vidurkis. Teigiamo vaizdo ir neigiamo vaizdo skirtumai palyginami. Jeigu teigiamo vaizdo skirtumo reikšmės yra mažesnės negu neigiamo vaizdo skirtumo reikšmės, reiškia tinklas gerai identifikavo vaizdų panašumą.

Kiekvienas grafikas turi 100 trejetų (10 grafikų kiekvienam bruožui kiekvienam modeliui). Tuose 100-ame trejetų kas 10 elementų yra skirtingas neigiamas vaizdas, o kiekvienas iš tų 10 yra pasuktas bazinis vaizdas (nuo –10 iki +10 laipsnių). Didelė dispersija grafike identifikuoja, kad tinklas prasčiau identifikuoja pasuktus vaizdus, t.y. tinklo bazinio vaizdo išvesties įterpinys (angl. embedding) yra nepanašus vienas į kitą su skirtingais pasukimais.

Tinklų palyginimas

TCapsNet-Hinton9 modelis lyginant su TResNet modeliu parodė geresnius rezultatus su validavimo duomenimis. Nors iš pirmos pažiūros TResNet tinklas pasiekė aukštesnį tikslumą su testavimo duomenimis galima matyti (9 priedas), kad dauguma tinklo išvesčių yra labai arti viena kitos. Reiškias teigiami įterpiniai ir neigiami įterpiniai pagal tinklo išvestį yra skaitomi panašiais. Pridėjus mažą reikšmę ($pa = 0.01$) prie teigiamų ir neigiamų vaizdų pikselių vidurkio skirtumo ir juos palyginus, galima matyti kad TResNet tikslumas siekia vos 60%, tuo tarpu TCapsNet-Hinton9 toliau siekia 71%. Nors ir TResNet tinklas puikiai atskyrė (išskyrus paskutinį trejetą), kuris vaizdas yra panašus, jis taip pat klaidingai identifikavo neigiamus

vaizdus, kaip panašius. Abu tinklai gerai atskyrė vaizdus su pasukimais (gauti panašūs įterpiniai), kadangi nėra didelės dispersijos tarp pasukimų. Iš gautų rezultatų matosi, kad TCapsNet-Hinton9 geriau generalizuoja vaizdus, ir geriau atskiria panašius nuo nepanašių.

TResNet tinklas turi panašius grafikus su miškais, tuo tarpu visiškai priešingi grafikai gauti (irgi tarpusavyje panašūs) yra visi likę. Vienintelis grafikas su neatpažintais panašiais vaizdais yra priemiesčio grafikas. Toks grafikas parodo, kad kai kuriuose scenarijuose tinklas gali išvesti neteisingus įterpinius tam tikriems pasukimams. Toks tinklas galėtų būti pritaikytas drono skrydžiams, bet dronas turėtų atmesti dalį duomenų rinkinio nuo pradinės pozicijos dėl daug panašiai laikomų trejetų.

TCapsNet-Hinton9 tinklas prasčiausius rezultatus parodė su duomenimis, kuriuose yra daugiau miško, ir visai gerai klasifikavo visus kitus duomenis. Tinklas turėjo daug mažiau artimų teigiamų ir neigiamų vaizdų nei TResNet, dėl to su tokiu tinklu būtų galima tiksliau nustatyti drono pradinę poziciją. Taip pat tinklas turėjo gana stabilią dispersiją tarp paveiksliukų pasukimų, kas leidžia gerai atskirti panašius ir nepanašius vaizdus. Tą taip pat galima išspręsti drono sistemai sugeneravus kelis įterpinius su nedaug pasuktais paveiksliukais ir gavus jų vidurkį identifikuoti ar tai panašus vaizdas. Taip pat bandėme šio tinklo modifikacijas TCapsNet-Hinton7, TCapsNet-Mono, TCapsNet-Conv ir TCapsNet-BNorm.

TCapsNet-Hinton7 (žr. 2 priedas) buvo bandytas norint įsitikinti ar mažesnė konvoliucinė sumavimo matrica (keičiant iš 9x9 į 7x7) galėtų lemti geresnius rezultatus. TCapsNet-Hinton7 parodė prastesnius rezultatus visais atvejais, todėl tolimesniuose eksperimentuose kapsulėse buvo naudojama 9x9 konvoliucinė sumavimo matrica.

TCapsNet-Mono buvo mokomas ir testuojamas su nespalvotais vaizdais (žr. 4 priedas). Šis tinklas neparodė gero tikslumo su mokymo duomenimis (58%) ir validavimo duomenimis

(31%), tačiau pasiekė aukštą tikslumą su testavimo duomenimis (~80%). Iš principo rezultatai labai panašūs, kaip ir TCapsNet-Hinton9 tinklo, t.y. prastai klasifikuoja miškus ir gerai visa kita. Aukštesnis tikslumas gautas, nes tarp panašių trejetų yra didelė dispersija dėl to atrodo, kad yra didesnis tikslumas, nors jis praktiškai nieko neduoda. Iš principo šis tinklas atrodo daug prasčiau dėl mažo mokymo duomenų ir validavimo duomenų tikslumo. Galimai su didesniu testavimo duomenų rinkiniu būtų daug mažesnis tikslumas ir su testavimo duomenimis.

TCapsNet-Conv tinklas turėdamas daugiau konvoliucinių sluoksnių (žr. 3 priedas) pasiekė labai panašius rezultatus, kaip ir TCapsNet-Hinton9. Šio tinklo pranašumas yra jo mokymosi greitis ir geresnis tikslumas (10% geresnis tikslumas nei TCapsNet-Hinton9) su mokymo duomenimis. Tinklas prarado šiek tiek pasukimų informacijos dėl papildomų konvoliucinių sluoksnių, tai matosi, nes ašių dispersija yra šiek tiek pakilusi nuo TCapsNet-Hinton9 tinklo. Kaip ir kiti Hinton modeliai, tinklas prasčiau atpažina miškų panašumus dėl mažo savybių kiekio. Šis tinklas kartu su TCapsNet-Mono turėjo du atvejus, kur sugebėjo tobulai (100% tikslumu) atskirti panašų vaizdą nuo visų kitų nepanašių vaizdų. Galima spręsti, kad gal buvo naudojama per didelė trejetų atstumo nuostolių funkcijos reikšmė, kuri lėmė tinklą persimokyti tarp skirtingų bruožų. Taip pat, galima sugeneruoti papildomų duomenų apsunkinti tinklo mokymąsi, kas taip pat mažintų persimokymą. Atsižvelgiant į visų duomenų rinkinių rezultatus, šis tinklas parodė geriausius rezultatus iš visų bandytų tinklų. Tai parodo, kad papildomi konvoliuciniai sluoksniai, kartu su paketų normalizavimo sluoksniais, gali padėti išgauti didesnį tikslumą ir paspartinti tinklo mokymąsi. Tuo tarpu, testuotas TCapsNet-BNorm tinklas (žr. 5 priedas) su papildomais normalizavimo sluoksniais prie pilnai sujungtų sluoksnių neparodė geresnių rezultatų nei vienu aspektu. TCapsNet-E (žr. 6 priedas), kuris buvo atkurtas pagal [MSC21] straipsnį nepasižymėjo taip kaip Hintono suvedimo algoritmu paremti modeliai, nors ir pasiekė aukštesnį tikslumą su MNIST ir CIFAR10 duomenų rinkiniais. Galimai šis

tinklas turi perspektyvų dėl jo greito mokymosi ir mažo parametrų kiekio, tačiau jam reikia atlikti papildomų modifikacijų šiai problemai spręsti, kad pavyktų išgauti aukštesnį tikslumą.

Apibendrinant, TCapsNet-Conv pasiekė geriausią tikslumą atsižvelgiant į visus duomenų rinkinius, tačiau dėl papildomų konvoliucinių sluoksnių pakilo skirtumų dispersija. Matant labai aukštą tikslumą su mokymo duomenimis ir mažesnį tikslumą su validavimo ir testavimo duomenimis galima spręsti, kad reikia arba daugiau duomenų mokymui arba keisti nuostolių funkcijos atstumo parametro reikšmę (su mažesne reikšme tinklas gali mažiau persimokyti). Taip pat visi paveiksliukai buvo konvertuojami į 32x32 rezoliucijos nuotraukas. Paveiksliukams sumažinti išmetamos tam tikros detalės, kurios gali padėti CapsNet tinklui tiksliau apsispręsti, kuriai kapsulei ta savybė labiau tinka. Reikia atsiminti, kad CapsNet kapsulės apsimoko naudojantis vektoriais, o ne skaliarais, kaip ResNet. Dėl to, mažas erdvinis nukrypimas gali pakeisti vektoriaus reikšmes ir aktyvuotis neteisinga kapsulė. Paveiksliukų rezoliucijos padidinimas leistų geriau atskirti tam tikras savybes (pvz. 1 ir 4 pikseliai gali aktyvuoti skirtingą kapsulę) ir geriau nustatyti jų reliatyvią poziciją.

4. Apibendrinimas

4.1. Rezultatai

1. Paruošti mokymo ir validavimo duomenys su nuo -10° iki +10° laipsnių pasukimo transformacijomis, taip sudarant apie ~13000 skirtingų trejetų mokymui ir ~600 skirtingų trejetų validavimui, 1000 skirtingų trejetų testavimui. Paveiksliukai buvo transformuojami į 32x32 rezoliucijos matricas, dėl darbinės atminties. Visi paveiksliukai buvo normalizuojami (vidurkis ir dispersija);
2. Realizuoti ir apmokyti 6 skirtingi CapsNet neuroniniai tinklai su CIFAR10, gauti svoriai buvo panaudoti, kaip pradiniai svoriai trejetų tinkluose:
    a. CapsNet-9 – 9x9 konvoliucinės sumavimo matricos kapsulėse;
    b. CapsNet-7 – 7x7 konvoliucinės sumavimo matricos kapsulėse;
    c. CapsNet-Mono – 9x9 konvoliucinės sumavimo matricos kapsulėse, apmokytas su nespalvotais paveiksliukais;
    d. CapsNet-Conv – 9x9 konvoliucinės sumavimo matricos kapsulėse, pridėti papildomi konvoliuciniai sluoksniai ir paketų normalizavimo sluoksniai prieš kapsulių sluoksnius;
    e. CapsNet-BNorm – CapsNet-Conv struktūra tik pridėti paketų normalizavimo sluoksniai prieš pilnai sujungtus sluoksnius;
    f. CapsNet-E – Efficient-CapsNet pagal [MSC21] darbe pateiktą sujungimo algoritmą.
3. Realizuoti ir apmokyti 6 skirtingi TCapsNet trejetų neuroniniai tinklai:
    a. CapsNet-9 – pasiektas 89% tikslumas su mokymo duomenimis, 58% tikslumas su validavimo duomenimis, 74% tikslumas su testavimo duomenimis;
    b. CapsNet-7 – pasiektas 63% tikslumas su mokymo duomenimis, 40% tikslumas su validavimo duomenimis, 66% tikslumas su testavimo duomenimis;
    c. CapsNet-Mono – pasiektas 58% tikslumas su mokymo duomenimis, 31% tikslumas su validavimo duomenimis, 82% tikslumas su testavimo duomenimis;
    d. CapsNet-Conv – pasiektas 98% tikslumas su mokymo duomenimis, 57% tikslumas su validavimo duomenimis, 71% tikslumas su testavimo duomenimis;

e. CapsNet-BNorm – pasiektas 91% tikslumas su mokymo duomenimis, 42% tikslumas su validavimo duomenimis, 58% tikslumas su testavimo duomenimis;
f. CapsNet-E – pasiektas 69% tikslumas su mokymo duomenimis, 42% tikslumas su validavimo duomenimis, 58% tikslumas su testavimo duomenimis;

4. Realizuotas ir apmokytas TResNet trejetų neuroninis tinklas, pasinaudojus jau apmokyto ResNet modelio filtrais iš Keras. Pasiektas 94% tikslumas su mokymo duomenimis ir geriausias 47% tikslumas su validavimo duomenimis, 86% tikslumas su testavimo duomenimis (60% įskaičiavus mažo atstumo paklaidą *pa*).

4.2. Išvados

1. TResNet tinklas nors ir gavo aukštą (84%) tikslumą su testavimo duomenimis, turėjo daug rezultatų su panašiomis reikšmėmis. Pridėjus paklaidos reikšmę tikslumui matuoti matosi, kaip tikslumas nukrenta iki 60%. Tai rodo, kad nors tinklas ir sugeba atpažinti panašius vaizdus, gauna daug klaidingų rezultatų. Šį tinklą galima būtų pritaikyti dronuose, tačiau gali reikėti optimizuoti vaizdų atpažinimą, skaidant duomenų rinkinius pagal pradinę drono buvimo vietą.
2. Atlikti 6 skirtingi bandymai su skirtingais kapsulių neuroniniais tinklais. Geriausius rezultatus pavyko gauti žiūrint į visus duomenų rinkinius su Hintono tinklo modifikacija, pridėjus papildomus konvoliucinius sluoksnius ir paketų normalizavimo sluoksnius. Šis tinklas prastesnį tikslumą gauna vaizduose su miškingomis vietomis, t.y. tinklo įterpinių reikšmės tarp teigiamo ir bazinio vaizdo turi didelį atstumą. Tai parodo, kad tinklas geriau veikia su vaizdais, kurie turi daugiau savybių. Pridėti konvoliuciniai sluoksniai taip pat padidina dispersiją tarp pasuktų bazinių vaizdų.
3. Bendrai paėmus, iš rezultatų matosi, kad tinklai su papildomais konvoliuciniais sluoksniais ir paketų normalizavimo sluoksniais apsimokė per mažiau epochų ir pasiekė panašius rezultatus. Papildomi konvoliuciniai sluoksniai sumažina sekančių sluoksnių dimensijų kiekį, ir tuo pačiu sumažina parametrų kiekį. Daugiau sluoksnių leidžia išmokti daugiau savybių per skirtingus filtrus, kas leidžia tinklui išlaikyti aukštą tikslumą.
4. Apmokant ir testuojant tinklą skirtumas tarp spalvų sukuria didelį skirtumą tiek tarp teigiamų, tiek tarp neigiamų paveiksliukų. Spalvų normalizavimas leidžia tinklui daugiau dėmesio kreipti į savybių išgavimą.

Šaltiniai

[ALPH+20] S. An, M. Lee, S. Park, H. Yang, J. So. An ensemble of simple convolutional neural network models for mnist digit recognition. 2020. Nuoroda per internetą:

https://arxiv.org/pdf/2008.10400.pdf

[BJ17] Xi, E., Bing, S., Jin, Y.: Capsule Network Performance on Complex Data. 10707, pp. 1–7. 2017. likelihood parameter identification for MAVs," in IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2016.

[BNOA+16] M. Burri, J. Nikolic, H. Oleynikova, M. W. Achtelik, and R. Siegwart, "Maximum likelihood parameter identification for MAVs," in IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2016.Systems (IROS). pp. 298–304. 2015.

[BOHS15] M. Bloesch, S. Omari, M. Hutter, R. Siegwart. "Robust visual inertial odometry using a direct EKFbased approach," in IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp. 298–304. 2015.

[CAS16] Covington Paul; Adams Jay; Sargin Emre. Deep neural networks for youtube recommendations. In: Proceedings of the 10th ACM Conference on Recommender Systems. ACM, 2016. p. 191-198.

[CLMW03] Olson C. F., Matthies L. H., Schoppers M., Maimone M. W. Rover navigation using stereo ego-motion. Robotics and Autonomous Systems, 43(4):215–229. 2003.

https://towardsdatascience.com/image-similarity-using-triplet-loss-3744c0f67973

[D19] Shibsankar Das. Image similarity using Triplet Loss. 2019. [Žiūrėta 2021/02/01] Nuoroda per internetą:capsule network. 2020. Nuoroda per internetą:

https://towardsdatascience.com/image-similarity-using-triplet-loss-3744c0f67973

[DS18] Xingping Dong, Jianbing Shen. Triplet Loss in Siamese Network for Object Tracking. 2018. [Žiūrėta 2021/02/01] Nuoroda per internetą:

https://openaccess.thecvf.com/content_ECCV_2018/papers/Xingping_Dong_Triplet_Loss_with_ECCV_2018_paper.pdf

[GN98] Gluckman, J., Nayar, S. K. Ego-motion and omnidirectional cameras. In ICCV. 1998. pp. 999–1005.. 2017 balandžio 3. [Žiūrėta 2017/12/14] Nuoroda per internetą:

http://moreisdifferent.com/2017/09/hinton-whats-wrong-with-CNNs

[Hin17] G. E. Hinton. Geoffrey Hinton on what's wrong with CNNs. 2017 balandžio 3. [Žiūrėta 2017/12/14] Nuoroda per internetą: on Artificial Neural Networks. Springer, Berlin, Heidelberg, 2011. p. 44-51.

[HKW11] Geoffrey E. Hinton, Alex Krizhevsky, Sida D. Wang. Transforming auto-encoders. In: International Conference on Artificial Neural Networks.

Springer, Berlin, Heidelberg, 2011. p. 44-51.vision and pattern recognition. 2016. pp. 770-778.

[HZ20] Wenkai Huang, Fobao Zhou. DA-CapsNet: dual attention mechanism capsule network. 2020. Nuoroda per internetą:Object Tracking. 2018. [Žiūrėta 2021/02/01] Nuoroda per internetą:

https://www.nature.com/articles/s41598-020-68453-w.pdf

[HZSS15] He, K., Zhang, X., Ren, S., & Sun, J. Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2016. pp. 770-778.

http://cv-tricks.com/cnn/understand-resnet-alexnet-vgg-inception/

[Kou17] Koustubh. ResNet, AlexNet, VGG, Inception: Understanding various architectures of Convolutional Networks. 2017. [Žiūrėta 2017/12/14] Nuoroda per internetą:

http://cv-tricks.com/cnn/understand-resnet-alexnet-vgg-inception/

https://stats.stackexchange.com/questions/208936/what-is-translation-invariance-in-computer-vision-and-convolutional-neural-netwo

[Kra17] Matt Krause. Invariance explanation. 2016 balandžio 23.Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems. 2012. pp. 1097-1105.

[KSH12] A. Krizhevsky, I. Sutskever, G. E. Hinton. Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems. 2012. pp. 1097-1105.

[L89] Matthies L. Dynamic Stereo Vision. PhD thesis, Carnegie Mellon University Computer Science Department. CMU-CS-89-195. 1989.over. 2017. [Žiūrėta 2017/12/14] Nuoroda per internetą:

https://www.axios.com/ai-pioneer-advocates-starting-over-2485537027.html

[Lev17] Steve LeVine. Artificial intelligence pioneer says we need to start over. 2017. Unmanned aerial vehicles. IEEE Robotics and Automation Magazine, vol. 19, no. 3. pp. 33–45, 2012.

[LPLK12] H. Lim, J. Park, D. Lee, H. J. Kim. Build your own quadrotor: Open-source projects on unmanned aerial vehicles. IEEE Robotics and Automation Magazine, vol. 19, no. 3. pp. 33–45, 2012.

[LS87] Matthies L. and Shafer S. Error modelling in stereo navigation. IEEE Journal of Robotics and Automation, RA-3(3). 1987.rover. Stanford Univ CA Dept of Computer Science, 1980.

[M80] Moravec, Hans P. Obstacle avoidance and navigation in the real world by a seeing robot rover. Stanford Univ CA Dept of Computer Science, 1980.Pp. 5093–5098.

[MB04] McCarthy, C., Barnes, N. Performance of optical flow techniques for indoor navigation with a mobile robot. In International Conference on Robotics and Automation. 2004. Pp. 5093–5098.

https://arxiv.org/ftp/arxiv/papers/1805/1805.11195.pdf

[MC18] Rinat Mukhometzianov, Juan Carrillo. CapsNet comparative performance evaluation for image classification. arXiv preprint arXiv:1805.11195. 2018. [Žiūrėta 2021/02/01] Nuoroda per internetą: exploration rovers.“ Journal of Field Robotics 24.3. 2007. pp. 169-186. [Žiūrėta 2021/02/01] Prieiga per internetą:

https://arxiv.org/ftp/arxiv/papers/1805/1805.11195.pdf

[MCM07] Mark Maimone, Yang Cheng, Larry Matthies. „Two years of visual odometry on the mars exploration rovers.“ Journal of Field Robotics 24.3. 2007. pp. 169-186. [Žiūrėta 2021/02/01] Prieiga per internetą:

[MSC21] Vittorio Mazzia, Francesco Salvetti, Marcello Chiaberge. Efficient-CapsNet: capsule network with self-attention routing. 2021. Nuoroda per internetą:

https://www.nature.com/articles/s41598-021-93977-0.pdf

[N04] Nist´er, D. An efficient solution to the five-point relative pose problem. IEEE Trans. Pattern Anal. Mach. Intell., 26(6):756–777. 2004.

[N05] Nist´er, D. Preemptive ransac for live structure and motion estimation. Machine Vision and Applications, 16(5):321–329. 2005.Series. Vol. 1004. No. 1. IOP Publishing, 2018.

[N18] Cui Nan. Applying gradient descent in convolutional neural networks. Journal of Physics: Conference Series. Vol. 1004. No. 1. IOP Publishing, 2018.

[NNB04] Nist´er, D., Naroditsky, O., Bergen, J. Visual odometry. In IEEE Conference on Computer Vision and Pattern Recognition. IEEE Computer Society Press. 2004. Pp. 652 – 659.

[NNB06] Nist´er, D., Naroditsky, O., Bergen, J. Visual odometry for ground vehicle applications. Journal of Field Robotics, 23(1):3–20. 2006.

https://medium.com/@Synced/hinton-proposes-capsnet-c04e8cec8bea

[PDS18] Pal, Dipan K., and Marios Savvides. "Non-parametric transformation networks." arXiv preprint arXiv:1801.04520. 2018.

[RHW85] Rumelhart, David E.; Hinton, Geoffrey E.; Williams, Ronald J. Learning internal representations by error propagation. California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[SARL99] Lacroix S., Mallet A., Chatila R., Gallo L. Rover self localization inplanetary-like environments. In International Symposium on Artificial Intelligence, Robotics, and Automation for Space (i-SAIRAS), Noordwijk, Netherlands. 1999. Pp. 433–440.Journal of Mathematical and Computer Applications 8.1. 2020. Pp. 76-90. [Žiūrėta 2021/02/01] Nuoroda per internetą:

https://www.researchgate.net/publication/339899775

[SEI20] Starovoytov, V. V., E. E. Eldarova, and Kazizat Takuadinovich Iskakov. Comparative analysis of the SSIM index and the pearson coefficient as a criterion for image similarity. Eurasian Journal of Mathematical and Computer Applications 8.1. 2020. Pp. 76-90. [Žiūrėta 2021/02/01] Nuoroda per internetą:

[SFH17] Sabour, Sara, Nicholas Frosst, and Geoffrey E. Hinton. Dynamic routing between capsules. In: Advances in Neural Information Processing Systems. 2017. pp. 3859-3869.on Computer Vision and Pattern Recognition. 2015. pp. 815-823.

[SFKP15] Schroff, Florian, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2015. pp. 815-823.

https://arxiv.org/pdf/1708.06652.pdf

[SKBB+18] Inkyu Sa, Mina Kamel, Michael Burri, Michael Bloesch, Raghav Khanna, Marija Popovic, Juan Nieto, Roland Siegwart. Build your own visual-inertial odometry aided cost-effective and open-source autonomous drone. 2018. [Žiūrėta 2021/02/01] Prieiga per internetą:

https://arxiv.org/pdf/1708.06652.pdf

[SLJS+15] Christian Szegedy, Wei Liu, Yangqing Jia, Perrie Sermanet ir kiti. Going deeper with convolutions. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2015. pp. 1-9.

[SNHG+14] Srivastava, Nitish, Hinton, Geoggrey, Krizhevsky, Alex, Sutskever, Ilya, Salakhutdinov, Ruslan. Dropout: A simple way to prevent neural networks from overfitting. 2014.

[SO95] Stricker M.A., Orengo M., Similarity of color images, Storage and Retrieval for Image and Video Databases III, SPIE Conf., Vol. 2420, DOI: 10.1117/12.205308. 1995, pp. 381-393.

[SZ15] Karen Simonyan, Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556, 2014.cars. arXiv preprint arXiv:1708.08559. 2017.

[TPJR+17] Yuchi Tian, Kexin Pei, Suman Jana, Baishakhi Ray. DeepTest: Automated testing of deep-neural-network-driven autonomous cars. arXiv preprint arXiv:1708.08559. 2017.

[VBB19] Vanderson Martins Do Rosario, Edson Borin, Mauricio Breternitz. The multi-lane capsule network. 2019. [Žiūrėta 2021/02/01] Nuoroda per internetą: https://arxiv.org/pdf/1902.08431.pdf

[VH87] Van Heel M., Similarity measures between images, Ultramicroscopy, Vol. 21. No. 1. 1987. pp. 95-100.In: Sergeyev Y., Kvasov D. (eds) Numerical Computations: Theory and Algorithms. NUMTA 2019. Lecture Notes in Computer Science, vol 11974. 2020. [Žiūrėta 2021/02/01] Nuoroda per internetą: https://doi.org/10.1007/978-3-030-40616-5_15

[VMJ20] Valaitis V., Marcinkevicius V., Jurevicius R. Learning Aerial Image Similarity Using Triplet Networks. In: Sergeyev Y., Kvasov D. (eds)

Numerical Computations: Theory and Algorithms. NUMTA 2019. Lecture Notes in Computer Science, vol 11974. 2020. [Žiūrėta 2021/02/01] Nuoroda per internetą: https://doi.org/10.1007/978-3-030-40616-5_15

[VSS02] Vassallo, R. F., Santos-Victor, J., Schneebeli, J. A general approach for egomotion estimation with omnidirectional images. 2002. Pp. 97–103.

https://cs.nju.edu.cn/wujx/paper/CNN.pdf

[W17] Jianxin Wu. Introduction to Convolutional Neural Networks. 2017. [Žiūrėta 2021/02/01] Nuoroda per internetą:Chuck Rosenberg, Jingbin Wang, James Philbin, Bo Chen, Ying Wu. Learning fine-grained image similarity with deep ranking. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1386-1393.

[WSLR+14] Jiang Wang, Yang Song, Thomas Leung, Chuck Rosenberg, Jingbin Wang, James Philbin, Bo Chen, Ying Wu. Learning fine-grained image similarity with deep ranking. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1386-1393.

[XY17] Xiangteng He, Yuxin Peng. Fine-graind Image Classification via Combining Vision and Language. arXiv preprint arXiv:1704.02792. 2017.

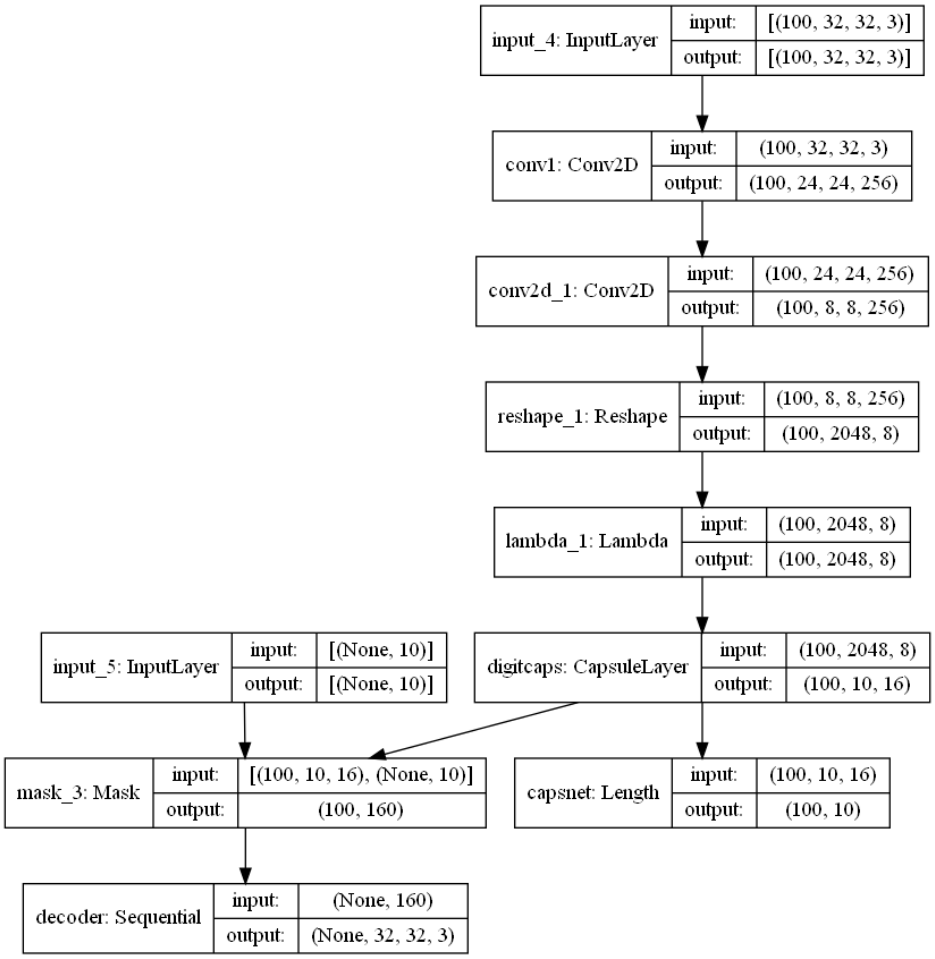
[ZDZ15] Zhu, Rüdiger Dillmann, Dalin Zhou. Intelligent Robotics and Applications. 2015. pp. 447.containing multiple moving objects using rigidity constraints. In International Conference on Computer Vision. Computer Society of the IEEE, IEEE Computer Society Press. 1988. Pp. 177–186.

[ZON88] Zhang Z., Faugeras O., Ayache N. Analysis of a sequence of stereo scenes containing multiple moving objects using rigidity constraints. In International Conference on Computer Vision. Computer Society of the IEEE, IEEE Computer Society Press. 1988. Pp. 177–186.

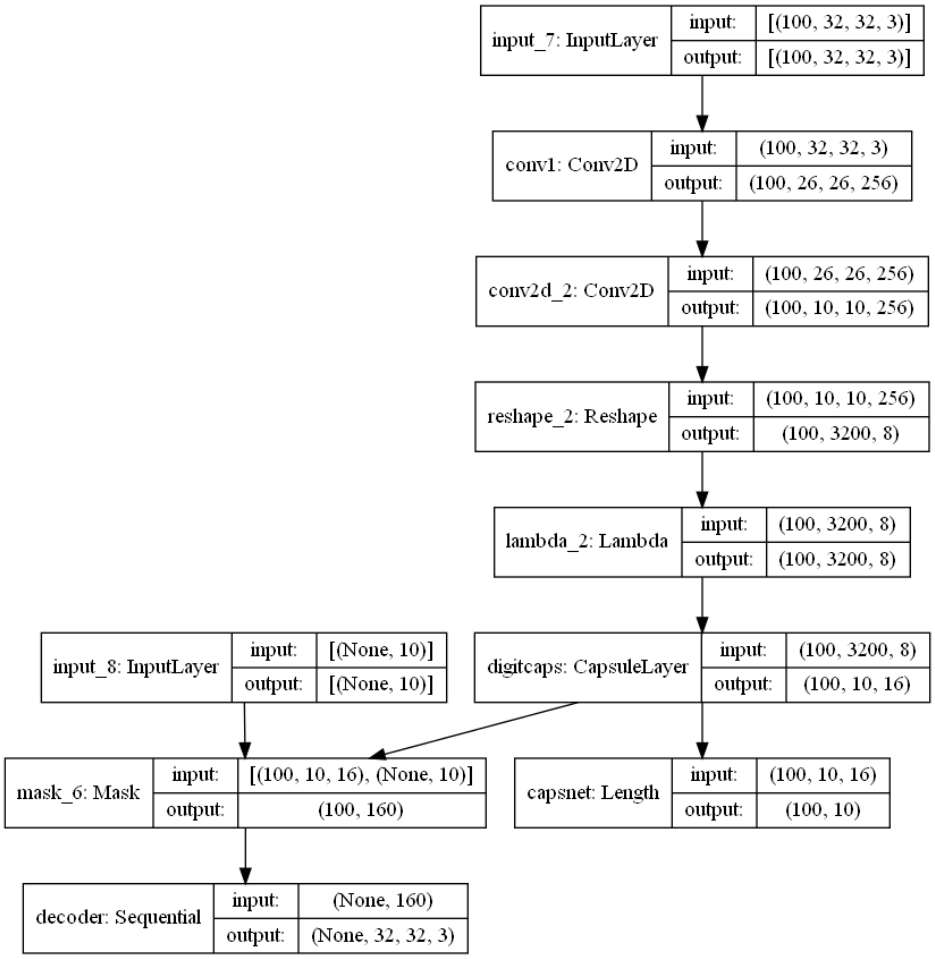
https://arxiv.org/pdf/1502.03167.pdf

Priedai

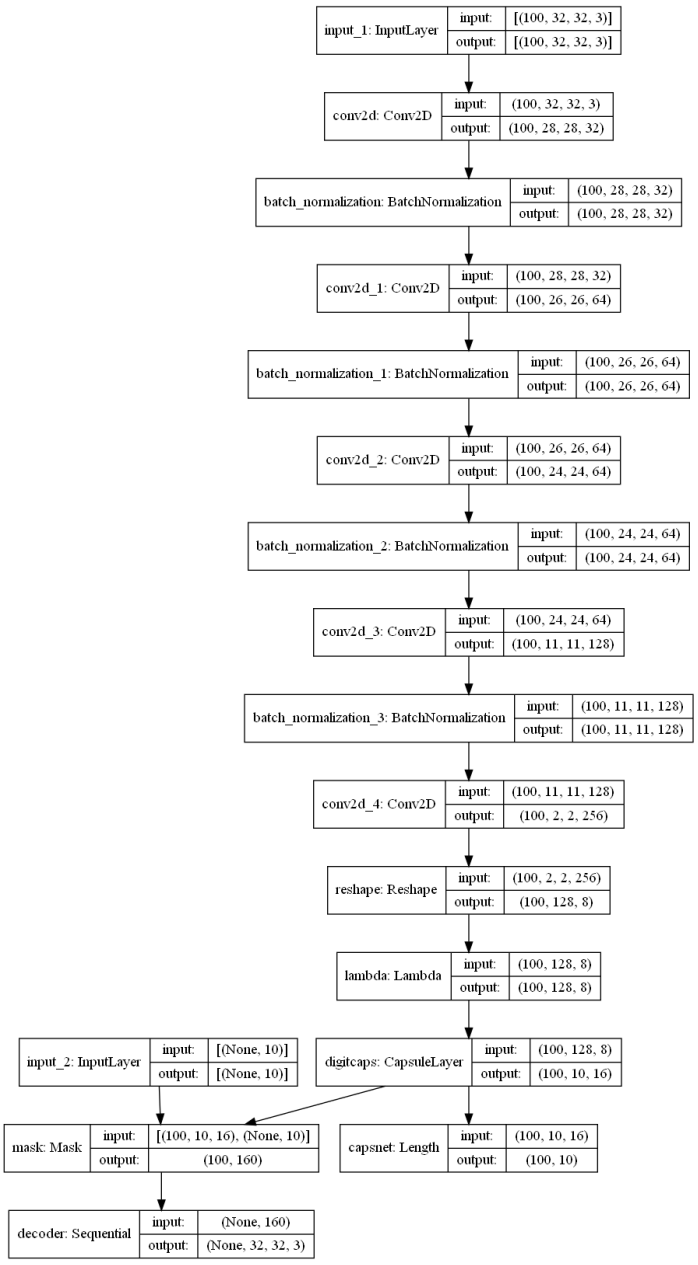
1. CapsNet-Hinton9 tinklo grafikas



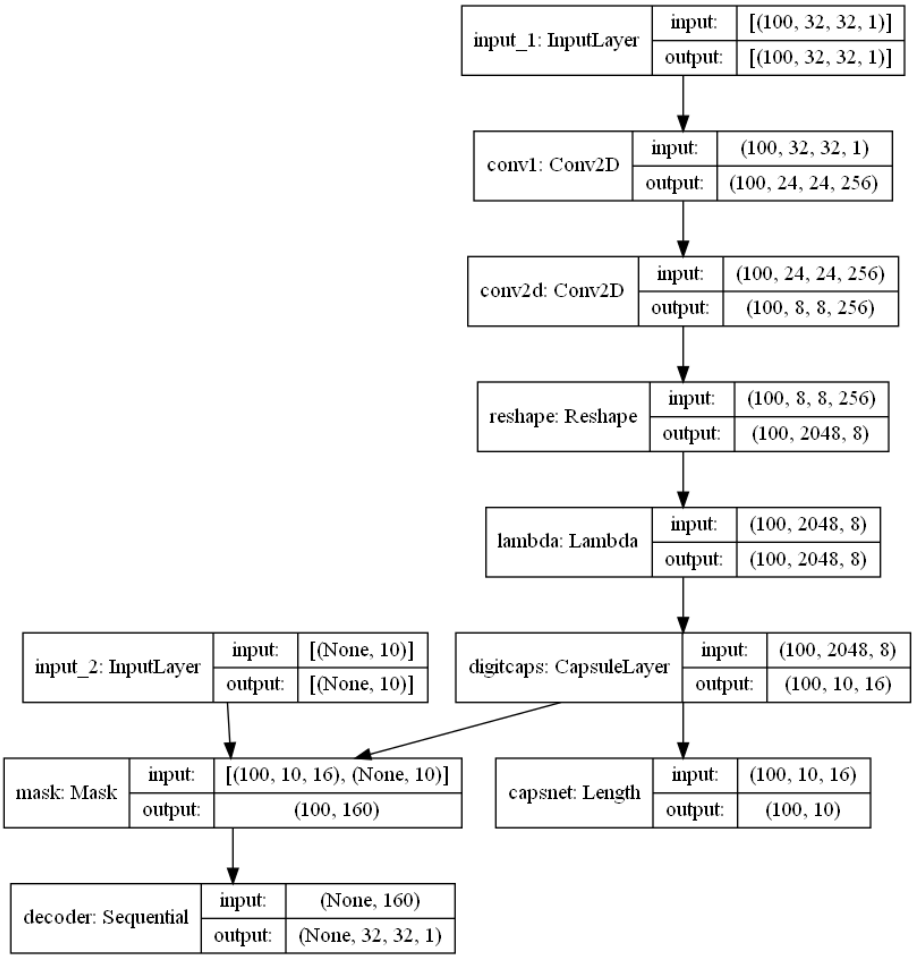
2. CapsNet-Hinton7 tinklo grafikas



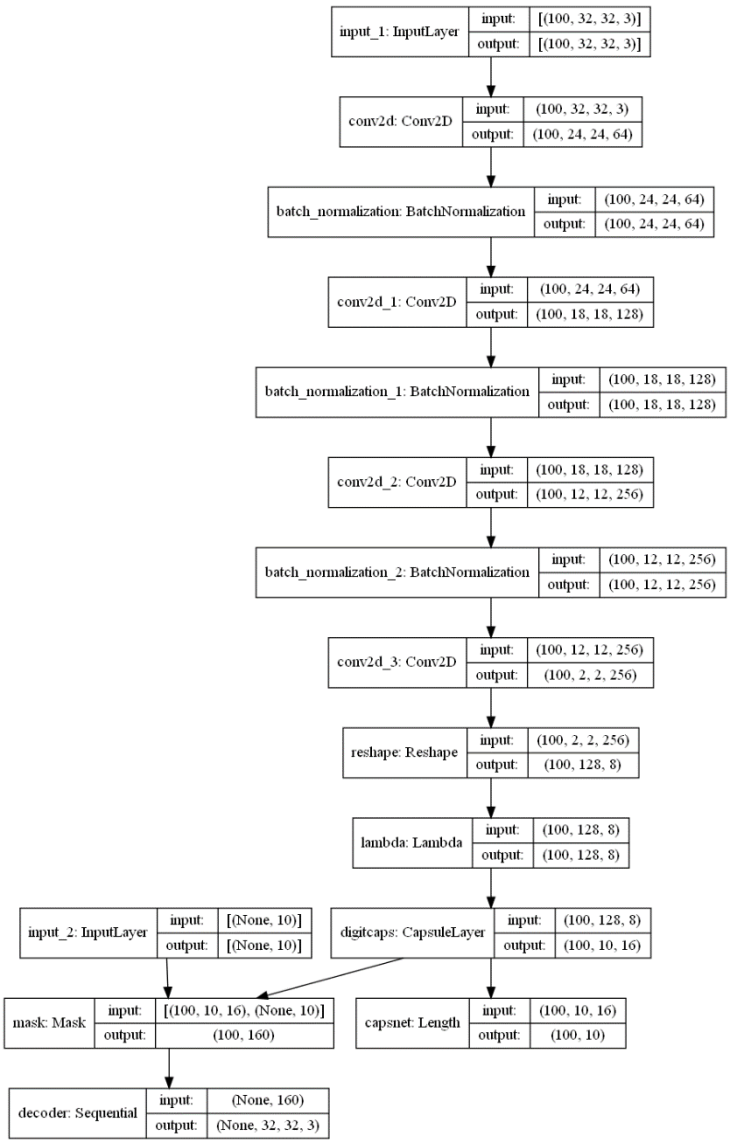
3. CapsNet-Conv tinklo grafikas



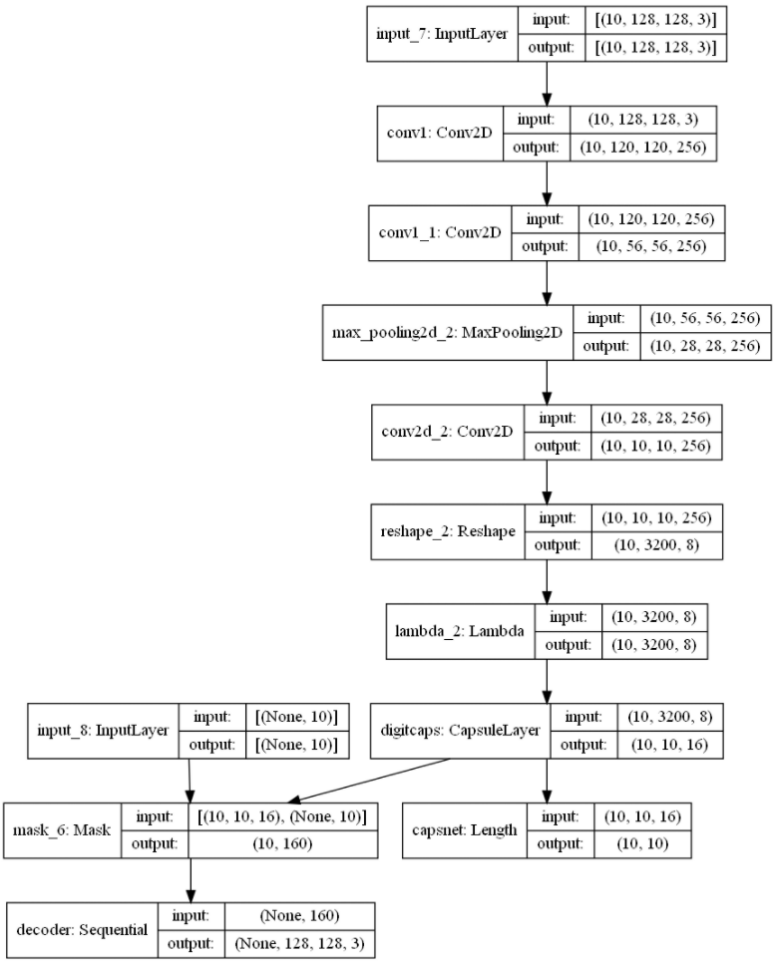
4. CapsNet-Mono tinklo grafikas



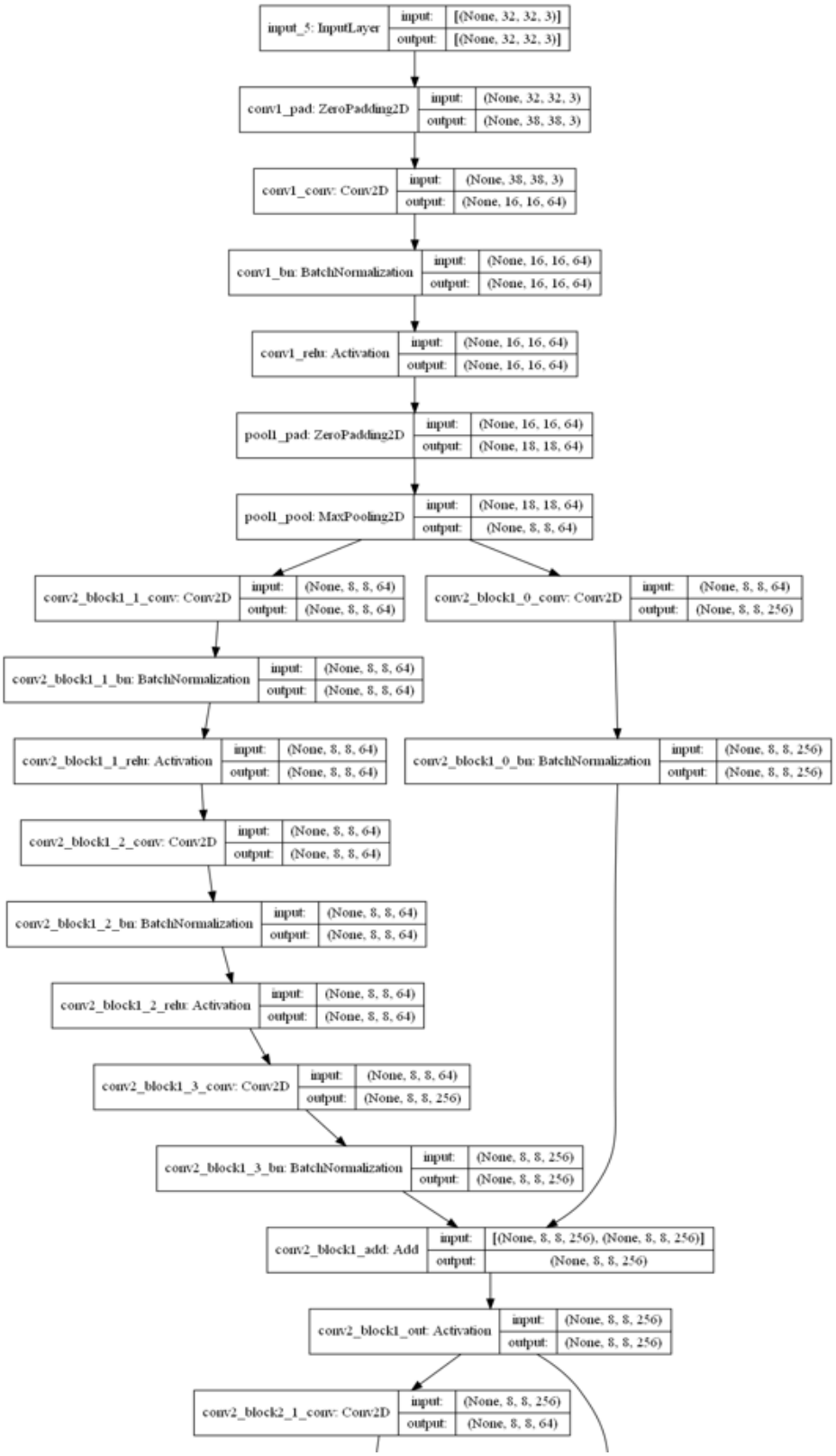
5. CapsNet-BNorm tinklo grafikas



6. CapsNet-E tinklo grafikas



7. Pirmojo išliekamojo bloko ResNet50 tinkle grafikas



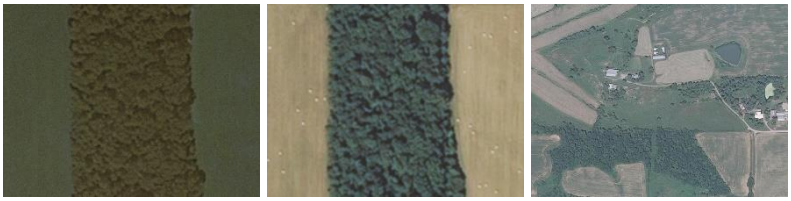
8. Testavimo duomenų rinkinys

Testavimo trejetai. Pirmi du vaizdai panašūs, trečias nepanašus. Visi nepanašūs vaizdai buvo lyginami su visais panašiais.

1. Kelias ir laukas (angl. road and field)

2. Ežeras ir kelias (angl. lake and road)

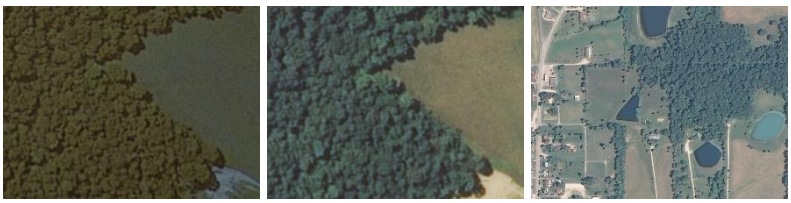
3. Miškas ir laukas (angl. forest and field)



4. 2 Ežerai ir kelias (angl. 2 lakes and road)

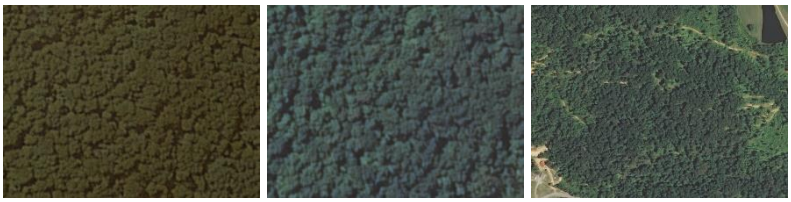
5. Kelias ir miškas (angl. road and forest)

6. Miškas su forma (angl. forest with shape)

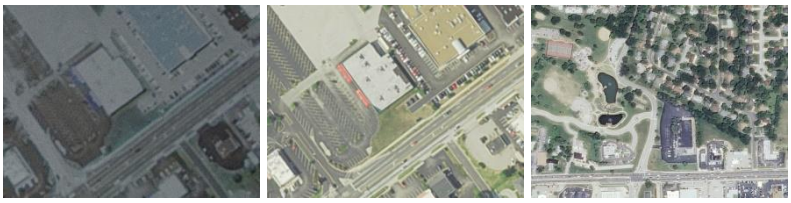


7. 90% miškas (angl. forest)

8. Vien miškas (angl. all forest)



9. Miestas (angl. urban)

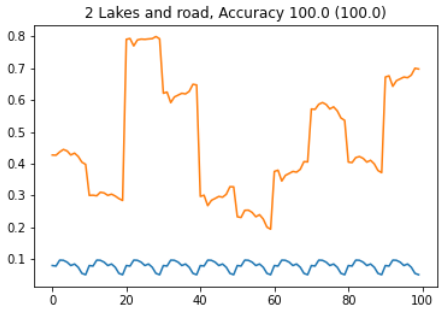


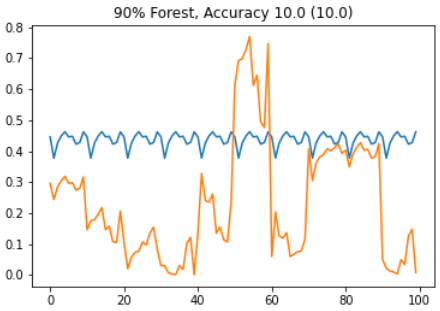
10. Priemestis (angl. suburb)

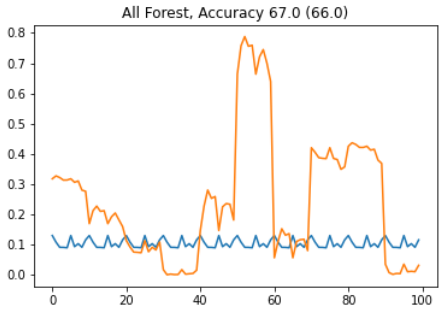
9. Rezultatų grafikai

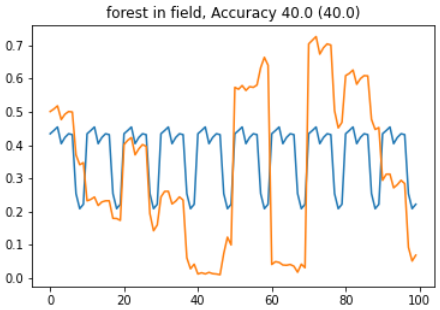
Atstumo reikšmė antrajai tikslumo metrikai apskaičiuoti (*pa*) - 0.01

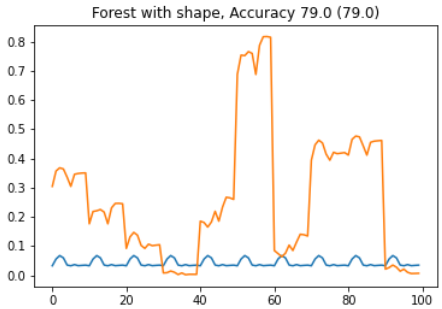
TCapsNet-Hinton9 (tikslumas: 0.742, tikslumas su atstumu: 0.715)

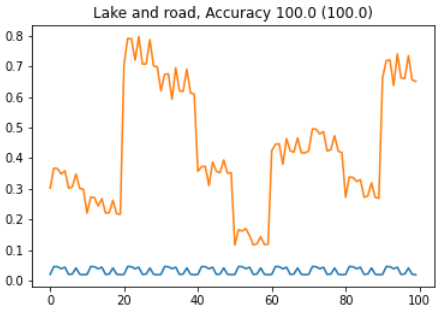


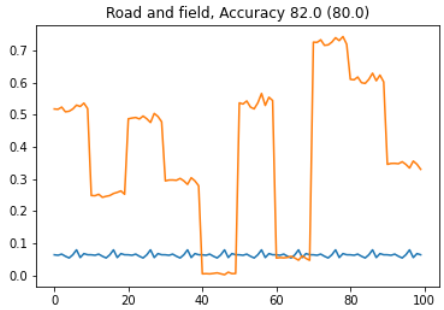


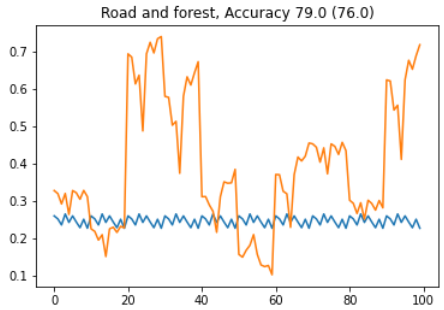


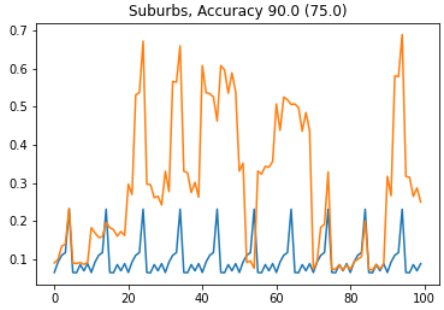


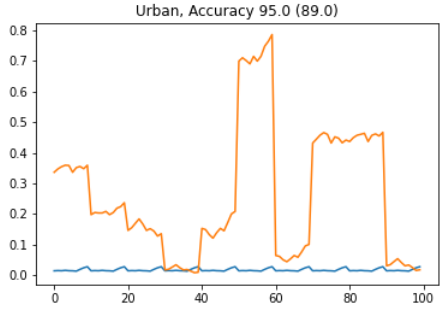




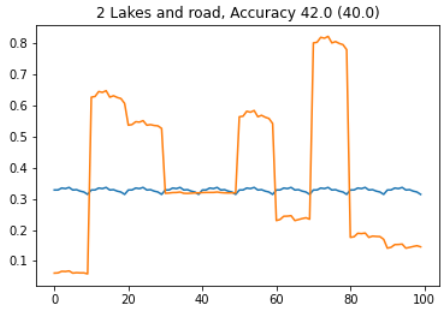


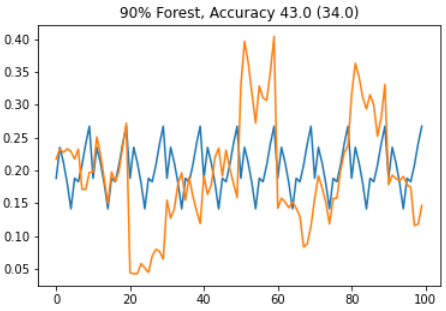


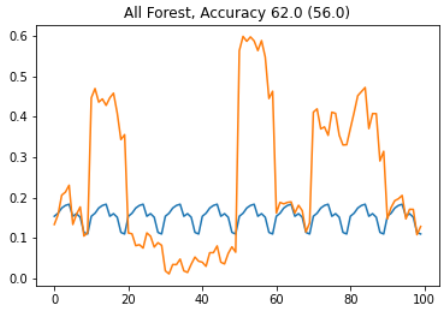


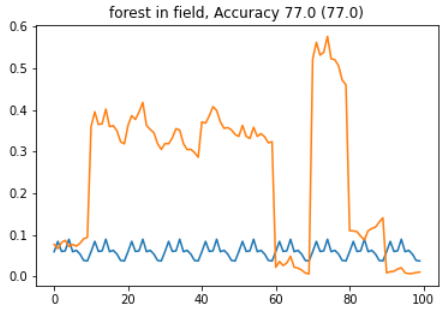


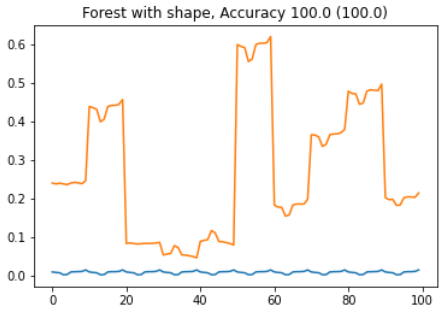
TCapsNet-Hinton7 (tikslumas: 0.669, tikslumas su atstumu: 0.639)

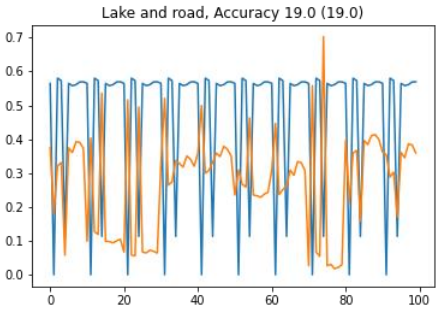


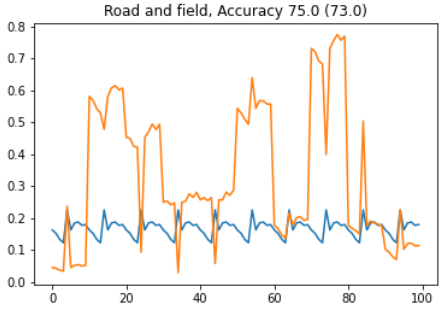


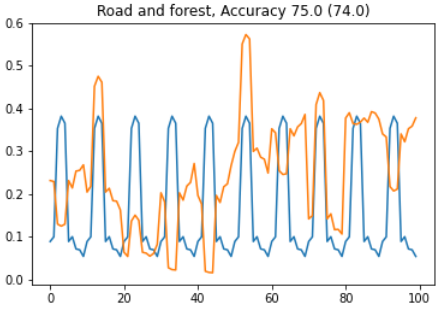


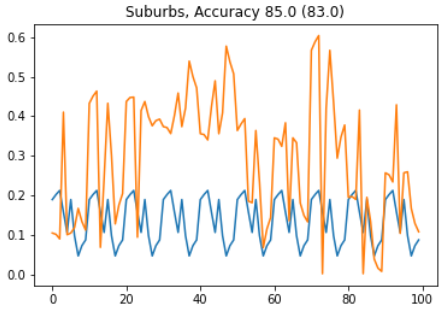


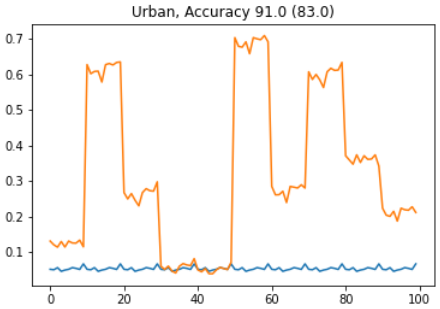

Forest with shape, Accuracy 100.0 (100.0)


Lake and road, Accuracy 19.0 (19.0)

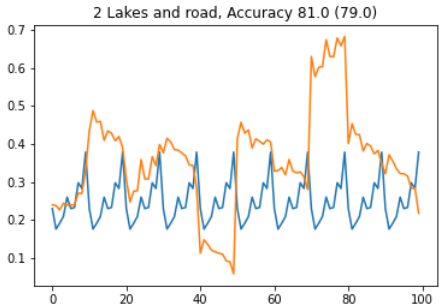

Road and field, Accuracy 75.0 (73.0)

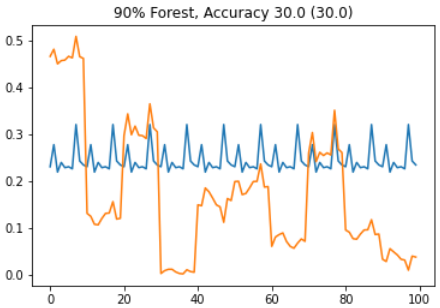

Road and forest, Accuracy 75.0 (74.0)

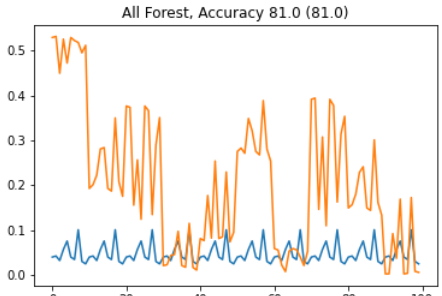

Suburbs, Accuracy 85.0 (83.0)

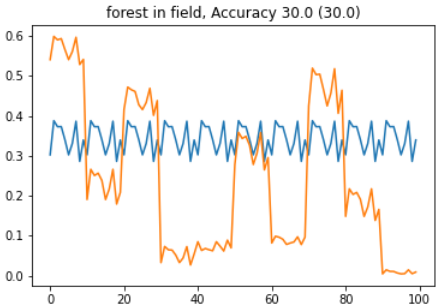

Urban, Accuracy 91.0 (83.0)

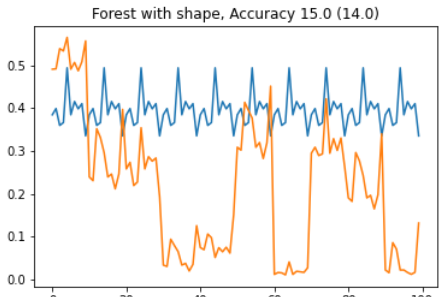
TCapsNet-Conv (tikslumas: 0.717, tikslumas su atstumu: 0.709)

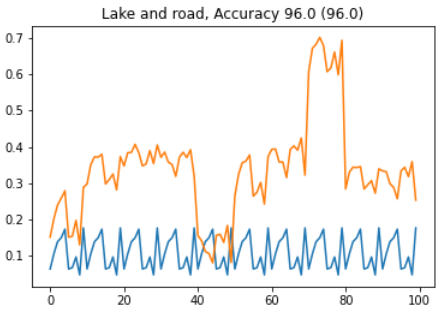


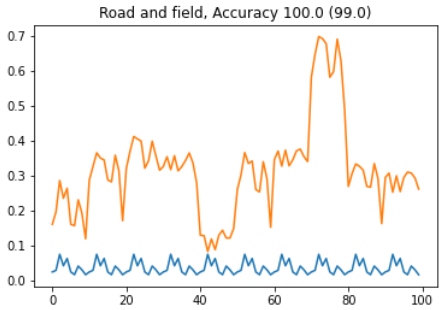


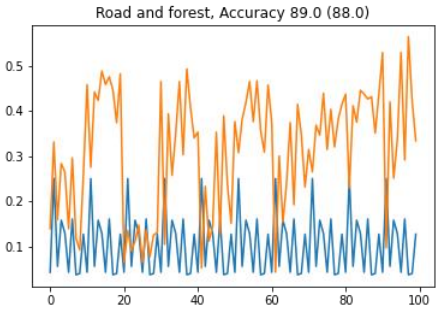


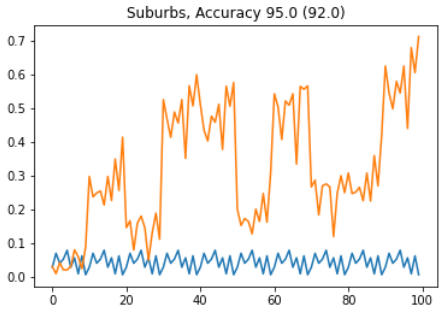


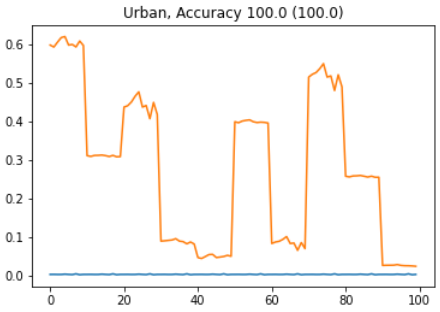




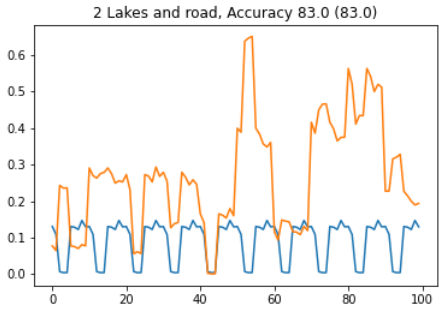


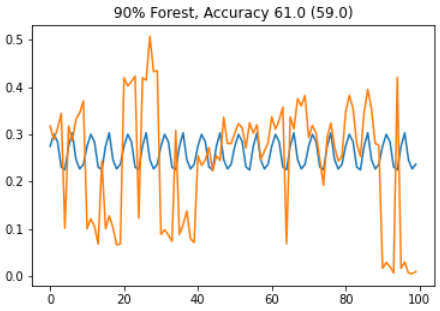


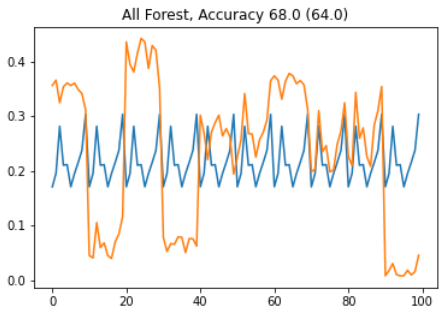


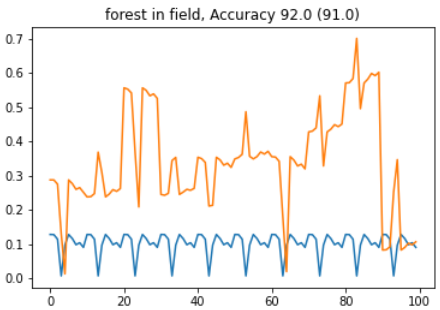


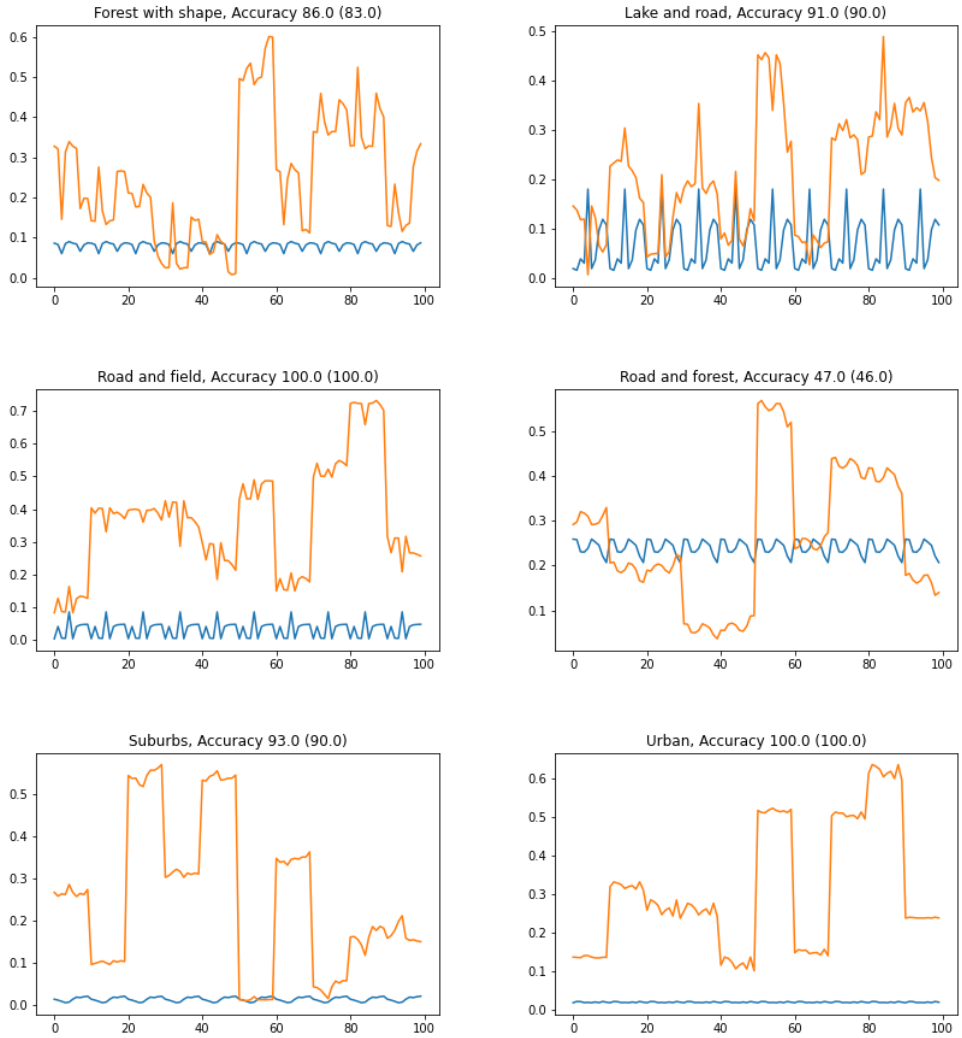
TCapsNet-Mono (tikslumas: 0.821, tikslumas su atstumu: 0.806)



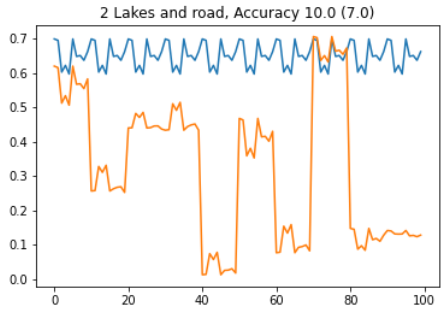


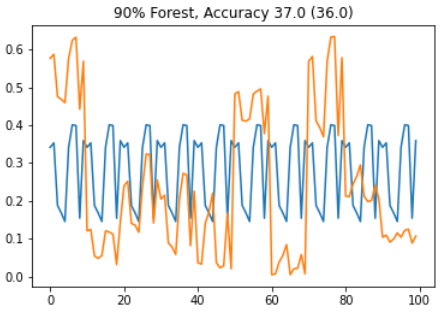


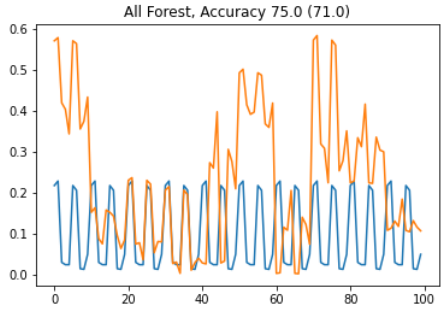


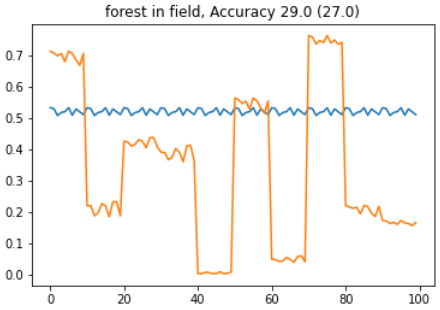


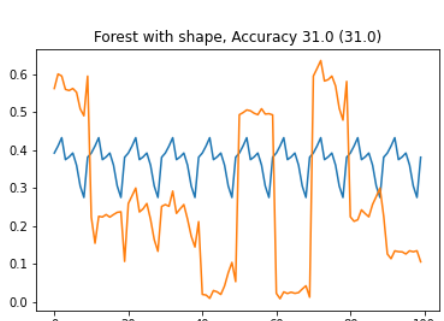
TCapsNet-BNorm (tikslumas: 0.587, tikslumas su atstumu: 0.557)

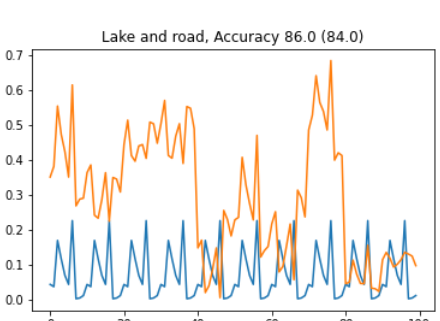

2 Lakes and road, Accuracy 10.0 (7.0)

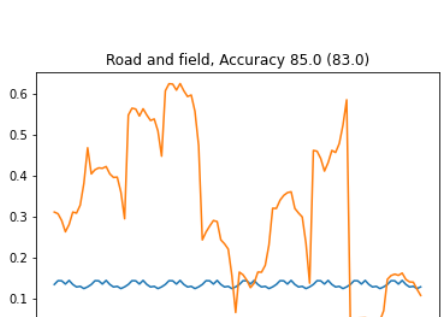

90% Forest, Accuracy 37.0 (36.0)

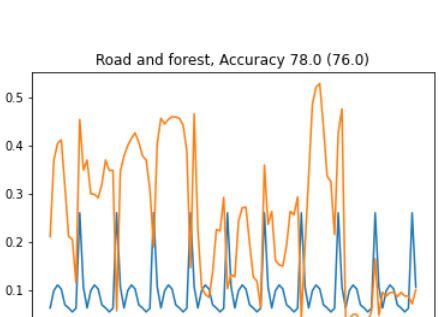

All Forest, Accuracy 75.0 (71.0)

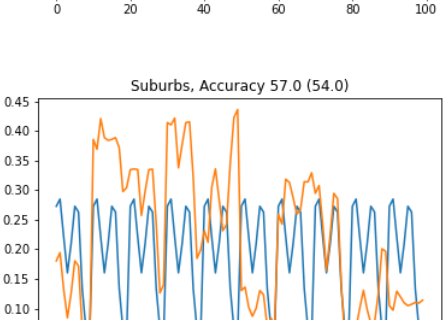

forest in field, Accuracy 29.0 (27.0)

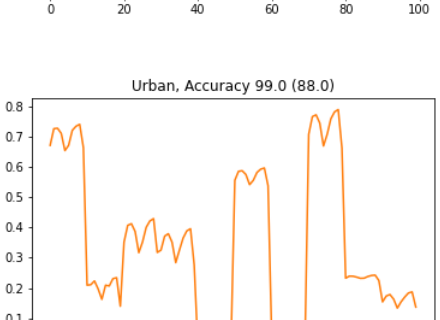

Forest with shape, Accuracy 31.0 (31.0)


Lake and road, Accuracy 86.0 (84.0)

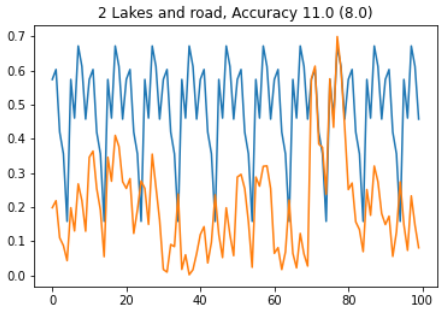

Road and field, Accuracy 85.0 (83.0)

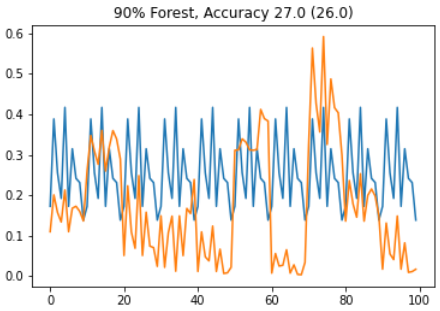

Road and forest, Accuracy 78.0 (76.0)

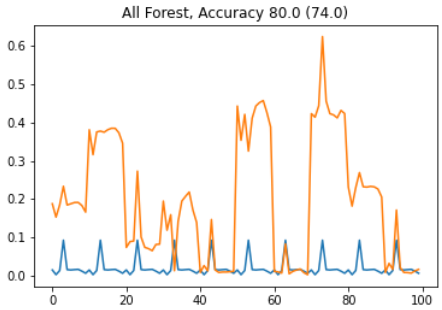

Suburbs, Accuracy 57.0 (54.0)

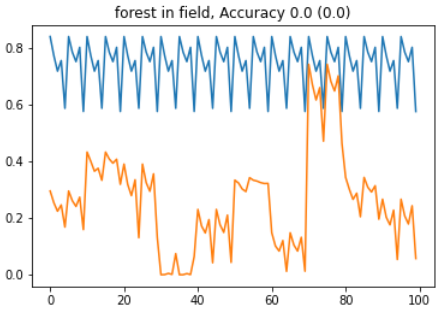

Urban, Accuracy 99.0 (88.0)

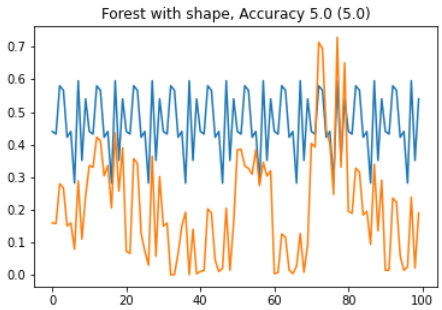
TCapsNet-E (tikslumas: 0.48, tikslumas su atstumu: 0.459)

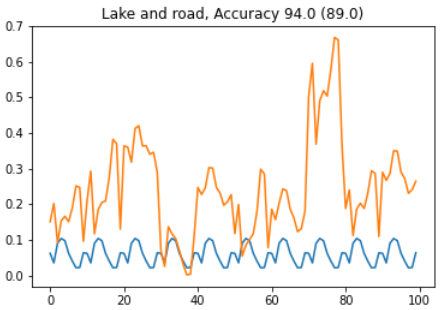


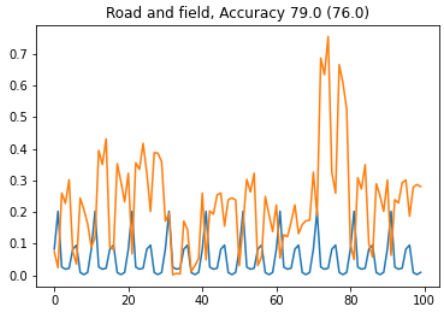


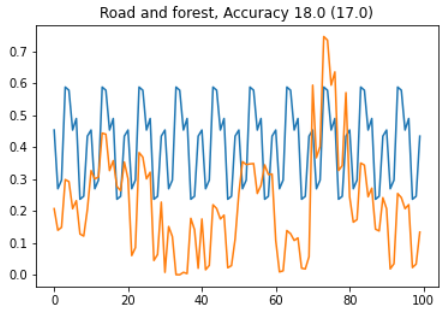


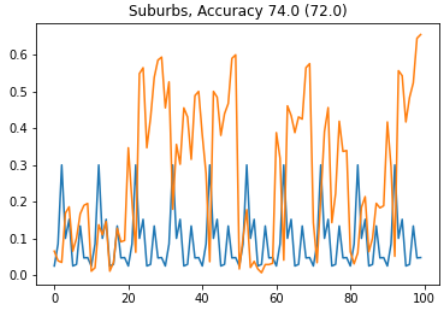


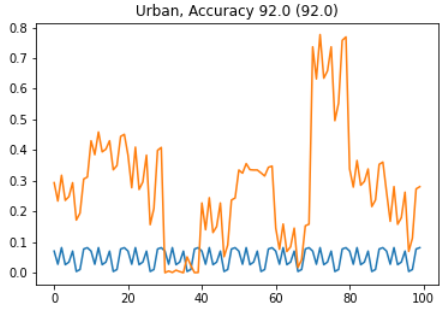




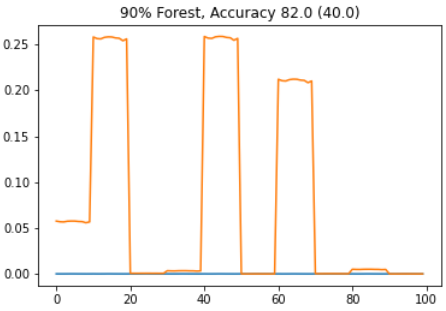


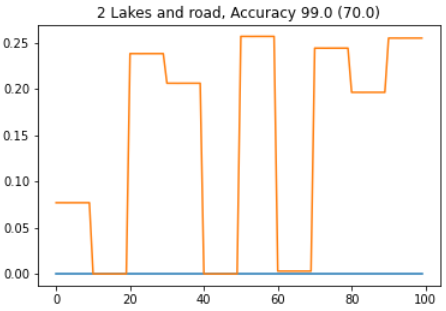


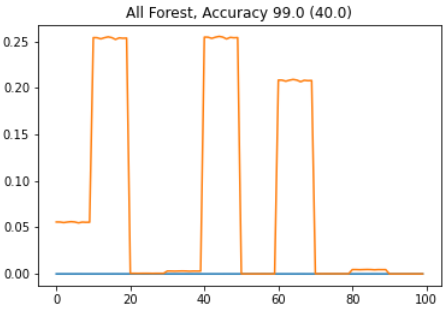


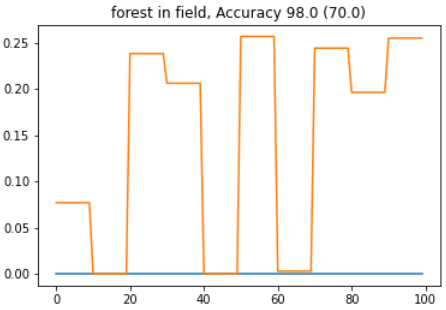


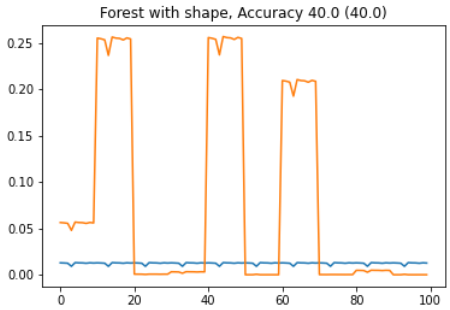
TResNet (tikslumas: 0.864, tikslumas su atstumu: 0.608)

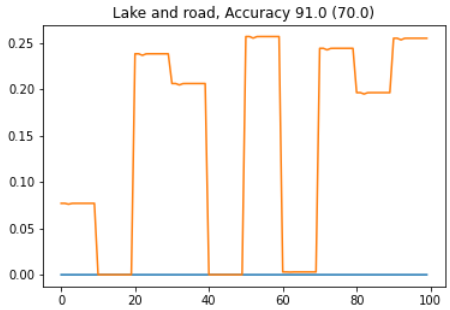


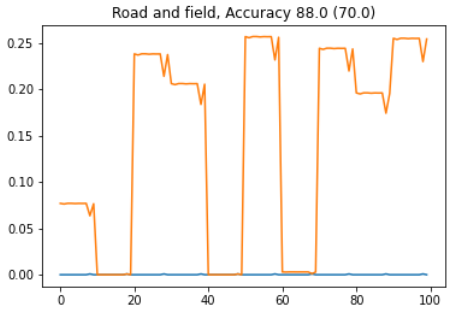


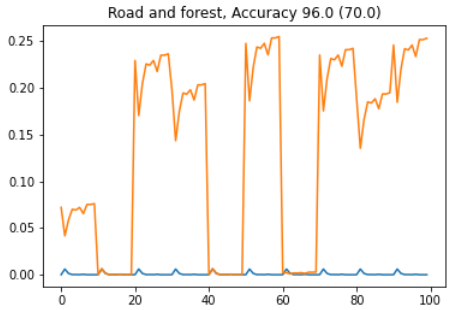


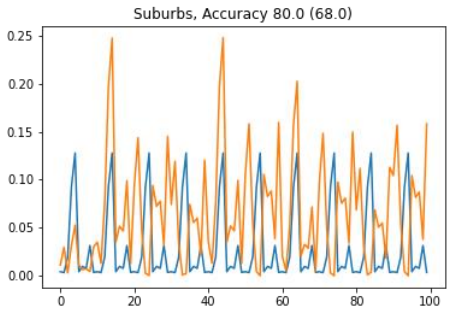


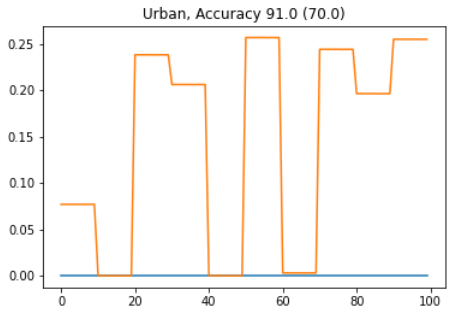












10. Kodas

Darbo kodas prieinamas per internetą:

https://gitlab.com/Cnnamon/masters-capsnet-similarity