VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
PROGRAMŲ SISTEMŲ KATEDRA

Labai panašių neuroninių klasių atskyrimas dirbtiniais neuroniniais tinklais

(Separation of very similar neuronal classes by artificial neural networks)

Kursinis darbas

Atliko: 3 kurso 5 grupės studentai
Miglė Vaitulevičiūtė (parašas)

Darbo vadovas: dr. Vytautas Valaitis (parašas)

Vilnius – 2018

TURINYS

Įvadas

Darbo tikslas: patikrinti mokymosi tikslumo kokybę su skirtinų gylių dirbtiniais neuroniniais tinklais, kai mokymui yra naudojamas mažas paveikslėlių rinkinys.

Užduotys:

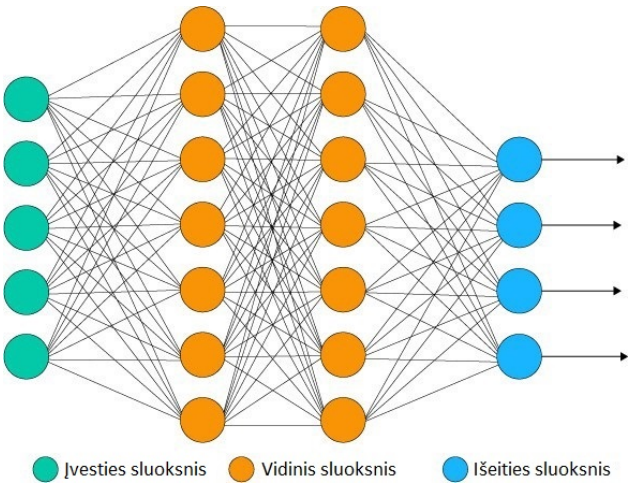
1. Pasirinkti analizuojamą dirbtinį neuroninį tinklą.
2. Pritaikyti ir modifikuoti neuronį tinklą išsikeltam darbo tikslui.
3. Gautą neuronį tinklą su skirtingais gyliais apmokyti.
4. Surasti ir naudoti geriausią optimizavimo funkciją.

1. Dirbtinis neuroninis tinklas

Pagal apibendrintą žmogaus smegenų veikimą buvo sugalvoti dirbtiniai neuroniniai tinklai [GBC16]. Bendrai žmogaus smegenys turi šimtus milijardų neuronų, kurie yra sujungti sinapsėsmis. Per tuos neuronus sklinda elektroniniai impulsai perduodantys informaciją - taip žmonės gali atpažinti objektus, garsus ir t.t. Dirbtiniai neuroniniai tinklai veikia panašiai. Jie irgi turi daug besijungiančių neuronų, kurie gauna informaciją ir pagal tą informaciją gali nuspręsti koks objektas yra paveikslėlyje. Tačiau ties tuo ir baigiasi žmogaus smegenų ir dirbtinių neuroninių tinklų panašumas, kadangi dirbtiniai neuroniniai tinklai yra kompiuterinė simuliacija - matematinis algoritmas su aritmetiniais kintamaisiais. Ši simuliacija yra suvokiama tik žmogui, kuris suprogramavo dirbtinį neuroninį tinklą, pačiam tinklui simuliacija nieko nereiškia, nuovokos nesuteikia.

1.1. Dirbtinio neuroninio tinklo sudėtis

Dirbtinis neuroninis tinklas yra sluoksnių rinkinys (1 pav.) - neuronų grupė sudaro sluoksnį, kuris yra sujungtas tarpusavyje su kitais sluoksniais. Vienas iš sluoksnių privalo būti įvesties sluoksnis, kuris atitinkamai pagal užduoti gali gauti įvairios formos informaciją - paveiksliukai, vaizdo medžiaga, garsas ir t.t. Ši informacija yra reikalinga tam, kad tinklas galėtų ją išanalizuoti ir išmokti. Tuo tikslu, kad vėliau gavęs panašią informaciją galėtų ją atpažinti - tam reikalingas išeities sluoksnis. Jis yra priešingame dirbtinio neuroninio tinklo gale negu įeities sluoksnis. Tarp anksčiau apibūdintų sluoksnių yra įvairaus dydžio sluoksnių sistema, kuri atlieka pagrindinį darbą [Woo18].



1 pav. Sluoksnių rinkinys

1.2. Dirbtinio neuroninio tinklo veikimas

Jungtys tarp neuronų yra pateiktos skaitine išraiška ir vadinamos svoriu. Kuo didesnis šis svoris tuo didesnę įtaką turi vienas neuronas kitam. Vienam neuronui yra pateikiama visų prieš jį buvusių neuronų informacija ir jungčių svoriai. Kiekvieno neurono informacija yra sudauginama su jo svoriu ir visi šie duomenys yra sudedami tarpusavyje. Taip iš vektoriaus gaunamas vienas

rezultatas ir jei šis rezultatas tinka aktyvavimo funkcijai, jis yra perduodamas tolimesniems neuronams. Tokio tipo veikimo dizainas yra vadinamas „feedforward" tinklu.

Tačiau jungčių svoriai nėra pastovūs. Kai dirbtinis neuroninis tinklas mokosi, galutinis rezultatas yra lyginamas su tikėtinu teisingu rezultatu, jei šie rezultatai skiriasi, svoriai yra keičiami atitinkamai, tai vadinama „backpropagation". Tokiu būdu yra gerinamas rezultatas ir mažinamas skirtumas tarp tikėtino ir gauto atsakymų.

1.3. Aktyvavimo funkcijos

Aktyvavimo funkcijų yra įvairių, kadangi sprendžiant tam tikrą problemą yra geriau naudoti vienas funkcijas, o kitas problemas - kitas funkcijas. Pagrinde yra dviejų tipų aktyvavimo funkcijos - tiesinės ir netiesinės. Tiesinės nėra tokios populiarios, kadangi jos neleidžia įvesčiai būti lanksčiai. Nors tiesinė funkcija labai dažnai naudojama išeities sluoksnyje. Netiesinės funkcijos dažniausiai naudojamos vidiniuose sluoksniuose. Šiuo metu labiausiai naudojama yra ReLU, kadangi naudojant šią funkciją mokymo rezultatai nuolatos gerėja, tačiau ReLU funkcijos spraustumas nesuteikia efektyvumo tinklui [XWC+15].

Aktyvavimo funkcijos yra skirstomos į:

- Tiesinė:
  - Žingsninė (binarinė) - išėjimas yra 0 arba 1.
- Netiesinė:
  - Sigmoidinė - išėjimas intervale [0; 1].
  - Hiperbolinio tangento - išėjimas intervale [-1; 1].
  - Minkštojo maksimumo - sunormuoja išėjimo vektorių į 1.
  - ReLU - išėjimas intervale [0; begalybė].

1.4. Nuostolio funkcijos

Kai dirbtinis neuroninis tinklas mokosi, jo gaunami rezultatai gali labai skirtis nuo tikėtinų rezultatų. Todėl nuostolio funkcija apskaičiuoja kaip stipriai skiriasi gautas rezultatas nuo tikėtino. Kuo didesnis nuostolis tuo toliau nuo teisingo atsakymo yra dirbtinis neuroninis tinklas [Dav15]. Paprasčiausia ir dažniausiai naudojama nuostolio funkcija yra vidutinio kvadrato klaida. Ši funkcija apskaičiuoja kvadratinį skirtumą tarp tikėtino ir gauto rezultatų. Tačiau šios funkcijos vienas iš didesnių trūkumų - neproporcingas išskyrimas didelių rezultatų. Kadangi funkcija didėja kvadratiniai, o ne tiesiniai, kai gaunamas rezultatas tolsta nuo tikėtino rezultato.

Priklausomai nuo to kokią problemą yra bandoma išspręsti yra naudojamos skirtingos funkcijos. Viena iš problemų yra klasifikacijos - dažniausiai išeities rezultatas yra tikimybės vertė f(x). Bendrai, funkcijos reikšmės dydis parodo gauto rezultato tikslumą. Dauguma klasifikacijos nuostolių funkcijos stengiasi maksimaliai padidinti tikslumą [Agr17].

Kelios klasifikacijos nuostolio funkcijos:

- Binarinė kryžiaus entropija.
- Neigiama registravimo tikimybė.

- Maržos klasifikatorius.
- Minkštų maržų klasifikatorius.

1.5. Optimizavimo funkcijos

Optimizavimo funkcijos naudojamos vidinių tinklo parametrų atnaujinimui, kad sumažinti gaunamų rezultatų netikslumą. Visos optimizavimo funkcijos gali būti suskirtytos į du tipus - nuolatinio mokymosi greičio ir prisitaikančio mokymosi.

Nuolatinio mokymosi greičio funkcijos turi hiperparametrą - mokymosi greitį. Jis privalo būti nustatytas, tačiau pasirinkti tinkamą mokymosi greitį gali būti sudėtinga - pasirinkus per mažą vidiniai parametrai gali labai lėtai konverguoti, o pasirinkus per didelį parametrams gali trukdyti konverguoti ir priversti nuostolio funkciją svyruoti apie minimumą arba diverguoti. Šio tipo funkcijos turi panašų hiperparametrą - momentą - kuris didina mokymosi greitį, kai jis artėja prie minimumo.

Vienos iš pagrindinių problemų nuolationio mokymosi greičio funkicijų, kad jos privalo turėti nustatytus hiperparametrus iš anksto ir jie labai stipriai priklauso nuo modelio ir sprendžiamos problemos. Dar vienas trūkumas, kad toks pats mokymosi greitis yra pritaikomas visiems vidinių parametrų atnaujinimams.

Prisitaikančio mokymosi funkcijos turi atskirus kiekvieno parametro mokymosi greičio metodus, kurie teikia euristikos metodą, nereikalaujant brangaus darbo rankiniu būdu nustatant hiperparametrus mokymosi greičiui. Tačiau šios funkcijos generalizuoja blogiau negu nuolatinio mokymosi greičio funkcijos, nors ir mokymosi metu pasirodo geriau [WRS+17].

2. Konvoliucinis neuroninis tinklas

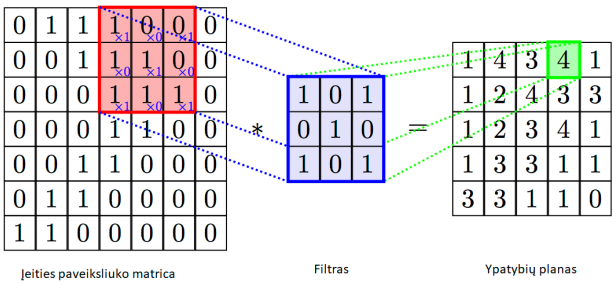
Konvoliuciniai neuroniniai tinklai yra labai panašūs į paprastus dirbtinius neuroninius tinklus (daugiau informacijos skyriuje „Dirbtinis neuroninis tinklas"). Tačiau pagrindinis skirtumas tarp šių tinklų yra, kad konvoliucinio įeities sluoksnis priima tik tai paveiksliukus, kurie jei padaryti su standartine skaitmenine kamera, turi tris komponentus - raudoną, žalią ir mėlyną. Šiuos komponentus galima įsivaizduoti kaip tris 2D matricas sudėtas viena ant kitos. Kiekvienos matricos i-osios eilutės ir j-ojo stulpelio elementas atitinka nuotraukos pikselį, kurio reikšmė yra intervale nuo 0 iki 255. Kadangi naudojamos informacijos tipas yra specifinis, tai labai sumažina tinklo parametrų kiekį ir tinklą padaro efektyvesnį.

Objektų atpažinimas paveiksliukuose yra sudėtingas dėl šių iššukių:

- Segmentavimas - paveiksliukai gali atvaizduoti įvairias scenas, kuriose gali būti pavaizduota daug objektų, kurie vienas kita gali dalinai uždengti.
- Šviesa - pikselių intensyvumas gali būti paveiktas šviesos šaltinio ar pačio objekto.
- Deformacija - objektai gali būti deformuoti įvairiais būdais, pavyzdžiui, kiekvieno žmogaus ranka parašyti skaičiai skiriasi.
- Galimybės - objektų klasės dažnai nustatomos pagal tai kaip patys objektai yra naudojami, pavyzdžiui, kėdės yra objektai sukurti sėdėti, tačiau jos gali turėti įvairų dizainą.
- Žvilgsnio taškas - keičiant vietą iš kurios yra žiūrima gali keistis objekto forma, informacija šokinėja per įeities sluoksnio dimensiją (t.y. pikselius).

2.1. Konvoliucija

Konvoliucija yra matematinė operacija, kuri apibūdina taisyklę, kuri parodo kaip reikia sujungti du informacijos rinkinius [PG17]. Paveiksliukų analizėje, statinė ir pagrindinė funckija yra įeities paveiksliukas, kuris yra analizuojamas, o antroji, judanti funkcija, žinoma kaip filtras, nes ji išskiria paveiksliuko ypatybę. Abi funkcijos yra susietos daugyba (2 pav.).



2 pav. Konvoliucijos veikimas

Tačiau konvoliuciniai tinklai turi daug filtrų, kurie pereina per vieną paveiksliuką, kiekvienas išskirdamas skirtingą paveiksliuko ypatybę. Pirmuose sluoksniuose šiuos filtrus galima įsivaizduoti kaip horizontalių, vertikalių ar įstrižų linijų filtrus, kurie sukuria paveikslėlio kraštų planą. Tinklas paima visus filtrus, gabaliukus paveiksliukų ypatybių vietų, ir juos sudeda į planą, kuris parodo ypatybės vietą. Mokydamasis skirtingų proporcijų ypatybių, tinklas leidžia lengvai kurti greitą ypatybių atpažinimą.

2.2. Konvoliucinio neuroninio tinklo sluoksniai

Konvoliuciniai neuroniniai tinklai tai yra sluoksnių rinkinys, kuris turi įeities, vidinius ir išeities sluoksnius. Tačiau priklausomai kokio tipo konvoliucinis neuroninis tinklas vidiniai sluoksniai gali skirtis. Konvoliuciniai neuroniniai tinklai turi tris pagrindinius sluoksnių tipus, kurie sudaro vidinį sluoksnį. Šie tipai yra konvoliucinis, sujungimo ir pilno sujungimo sluoksniai.

2.2.1. Konvoliucinis sluoksnis

Šis sluoksnis yra pagrindinis konvoliucinio neuroninio tinklo sluoksnis, kuris atlieka daugiausia skaičiavimų, nustato visas paveiksliuko ypatybes. Kadangi, įeities informacija (paveiksliukas) yra didelės dimencijos neefektyvu visų neuronų sujungti vienus su kitais. Todėl neuronai yra sujungiami su lokaliu informacijos kiekiu, kuris yra lygus filtro dydžiui ir vadinamas erdviniu mastu [Li15].

Neuronų kiekis po konvoliucijos (ypatybių plano dydis) yra nustatomas trimis parametrais:

- Gylis - atitinka filtrų skaičių.
- Žingsnis - pikselių kiekis, kuris parodo per kiek reikia slinkti filtro matrica per įeities informacijos matricą.
- Nulių pamušalas - įeities informacijos matrcios kraštus užpilduti nuliais.

2.2.2. Sujungimo sluoksnis

Periodiškai sujungimo sluoksnis yra įterpiamas tarp konvoliucinių. Pagrindinis sluoksnio tikslas yra laipsniškai mažinti erdvinį filtruojamo paveiskliuko mąstą. Šis veikslas yra atliekams tam, kad sumažinti parametrų ir skaičiavimų kiekį bei kontroliuoti perjungimą. Sujungimo sluoksnis veikia nepriklausomai nuo kiekvieno gabalėlio gylio ir keičia jo dydį erdviškai, naudodamas MAX operaciją. Dažniausiai šis sluoksnis yra naudojamas su 2x2 dydžio filtru - kas antras po konvoliucijos gauto gabaliuko kiekvienas gylio sluoksnis yra mažinamas per pusę ties ilgiu ir pločiu, taip yra atsikratoma 75 procentų aktyvacijų. Po šios operacijos gabaliuko gylis nepasikeičia.

Dažniausiai yra naudojamos dvi šio sluoksnio variacijos. Pirmasis yra vadinamas persidengiantis sujungimas, kur filtro dydis yra lygus 3 ir žingsnis yra lygus 2. O kitas dažniau naudojamas turi filtro dydį lygų 2 ir žingsnį taip pat 2. Sujungimo sluoksniai su labai dideliais parametrais yra labai desktruktyvūs.

2.2.3. Pilno sujungimo sluoksnis

Konvoliucinio ir sujungimo sluoksnių išeitys yra aukšto lygio ypatybės, kurios yra gautos iš įeities paveiksliuko. Pilno sujungimo sluoksnis yra sujungtas su visais neuronais iš sluoksnio buvusio prieš jį. Šio sluoksnio tikslas yra panaudojant tas ypatybes, kurios yra gautos iš prieš tai buvusių sluoksnių, nustatyti kokioms klasėms priklauso įeities paveiksliukas pagal mokymo informacijos rinkinį, kai nueroninio tinklo problema yra klasifikacija [Kar16]. Jei šiam sluoksniui yra naudojama minkštojo maksimumo funkcija tuomet sudėtis visų gautų galimybių turi būti lygi 1. Minkštojo maksimumo funkcija priima vektorių įvertinimų ir jį suspaudžia į vektorių, kuriame

yra klasių tikimybių įvertinimai intervale nuo 0 iki 1, kur tikimybė arčiausiai vieneto reiškia, kad labiausiai užtikrintas dėl tos klasės.

2.3. Architektūros

Konvoliuciniai neuroniniai tinklai turi keletą skirtingų architektūrų, kurios yra naudojamos atinkmai pagal sprendžiamą problemą. 1 lentelėje pateikta infromaciją apie įvairias architektūras.

Pavadinimas	Metai	Parametrų kiekis	Veikimas	ILSVRC vieta
LeNet	1998	60 000	Geriausiai atpažysta ranka paraštytus skaičius. Susideda iš sluoksnių - kelių pasikartojančių konvoliucijos ir sujungimo bei pasibaigia dviem pilno sujungimo sluoksniais.	-
AlexNet	2012	60 000 000	Veikimu panašus į LeNet, tačiau turi daug daugiau parametrų ir filtrų bei sudėtus konvoliucinius sluoksnius.	pirma
GoogLeNet/Inception	2014	4 000 000	Vidiniai sluoksniai sudėti paraleliai, naudojami Inception moduliai. Vienas modulis savyje turi 1x1, 3x3 ir 5x5 dydžių konvoliucijos filtrų bei vidurkio sudėjimo sluoksnius.	pirma
VGGNet	2014	138 000 000	Panašus veikimas į AlexNet, tačiau daug gilesnis. Naudojamų filtrų dydis yra 3x3 ir jie yra sudėti vienas po kito.	antra
ResNet	2015	25 000 000	Turi labai daug sluoksnių, sudėtų vienas po kito, kurie turi liekamąjį bloką, kuris įeities informaciją perduoda tolimesniam sluoksniui ją pridėdamas ir taip sumažina konvoliucijos ir aktyvavimo funkcijų kiekį.	pirma

1 lentelė. Konvoliucinių neuroninių tinklų architektūros

2.4. Modelio reguliavimas

Pilnas konvoliucinio neuroninio tinklo apmokymas gali užtrukti labai ilgą laiką ir išnaudoti daug resursų. Todėl yra kai kurios įstaigos arba žmonės, kurie apmoko savo tinklą ir jo svorius bei reikšmes, vadinamą modeliu, pateikia visuomenei, tačiau šis modelis yra nepritaikytas individuoliai žmogaus užduočiai. Modelį reikia reguliuoti - iš naujo apmokyti paskutinius sluoksnius su individuolios užduoties parametrais.

Daugelis konvoliucinių neuroninių tinklų apmokytų su natūraliais paveiksliukais turi fenomeną. Pirmuosiuose sluoksniuose jie išmoksta ypatybių panašių į Gaboro filtą (tiesinis filtras naudojamas tekstūroms analizuoti) ir spalvų dėmes. Šios pirmojo sluoksnio ypatybės nepriklauso nuo duomenų rinkinio, bet yra bendros ir tinkamos daugeliui duomenų rinkinių ir užduočių [YCB+14]. Dėl šio fenomeno galima naudoti modelius neapmokytus su specifiniu duomenų rinkiniu, bet jį minimaliai modifiktą, kitoms užduotims spręsti, kas leidžia sutaupyti resursų bei turėti mažesnį duomenų rinkinį.

3. Technologijos

Naudojamų technologijų išsirinkimas yra pradinis žingsnis siekiant įvykdyti išsikeltas užduotis. Šiame skyriuje pateiktos populiariausios šių laikų technologijos bei trumpai papasakota apie jas.

3.1. ImageNet

ImageNet yra projektas sugalvotas profesorės Li Fei-Fei 2009 metais. Projekto tikslas buvo sukurti didelę sukategorizuotų paveiksliukų ir jų etikečių duomenų bazę, kuri butų skirta vizualinio objekto atpažinimo programinės įrangos tyrimams. Ši duomenų bazė yra suorganizuota pagal WorldNet hierarchija - anglų kalbos žodžiai yra grupuojami į sinonimų rinkinius, kurie turi apibūdinimus ir naudojimo pavyzdžius bei saugo ryšių kiekį tarp sinonimų arba jų narių. ImageNet turi daugiau nei 100 000 sinonimų rinkinių, kur didžioji dalis yra daiktavardžiai (80 000+).

Taip pat šis projektas kiekvienais metais daro konkursą vadinamą „ImageNet Large Scale Visual Recognition Challenge" (trumpinys ILSVRC). Konkurso užduotis yra išmokinti modelį, kuris galėtų įeities paveiksliuką teisingai klasifikuoti į 1000 skirtingų objektų klasių, kurios atitinka realius daiktus, gyvūnus ir t.t. Modeliai yra apmokomi su apie 1.2 milijonų paveiksliukų ir dar 50 000 paveiksliukų yra naudojami validacijai mokymo metu bei 100 000 paveiksliukų yra panaudojami galutiniam modelio testavimui. Šis konkursas yra paveiksliukų klasifikacijos algoritmų etalonas.

3.2. Keras

Keras yra aukšto lygio programų sąsaja skirta neuroniniams tinklams. Sąsaja parašyta su „Python" programavimo kalba ir vidinėje pusėje galinti veikti su „TensorFlow" ir kitomis bibliotekomis. Keras buvo sukurtas tikintis suteikti greitą ekspermentavimą, kad sugalvojus idėją pasiekti rezultato būtų galima su kiek įmanoma mažiau uždėlsimo.

Ši sąsaja savyje turi visus pagrindinius neuroninio tinklo kūrimo blokus, pavyzdžiui, sluoskniai, aktyvavimo ir optimizavimo funkcijos. Taip pat Keras suteikia modelius, kurie yra apmokyti naudojant ImageNet duomenų bazę. Šiuos modelius galima reguliuoti, pridėti papildomų sluoksnių, pasirinkti esamus sluoksnius bei juos iš naujo apmokyti.

3.3. TensorFlow

TensorFlow yra atviros programinės įrangos biblioteka skirta aukšto našumo skaitinimas skaičiavimams. Jo lanksti architektūra leidžia lengvai diegti skaičiavimus įvairiose platformose - procesoriuose, grafikos procesoriuose. Sukurtas „Google" dirbtinio intelekto skyriaus, tad yra labai palaikomas automatinis ir gilusis mokymasis, tačiau dėl bibliotekos ir skaičiavimų lankstumo yra naudojamas įvairiose mokslinėse srityse.

4. Ekspermentas

Šio ekspermento tikslas yra išanalizuoti mokymosi tikslumą su skirtingų gylių neuroninais tinklais, kai mokymui yra naudojamas mažas paveiksėlių rinkinys. Taigi, pirmas šio ekspermento žingsinis - išsirinkti konvoliucinį neuroninį tinklą. Poskyriuje „Architektūros" yra trumpai apibūdinti pagrindiniai konvolicinių neuroninių tinklų tipai. Iš jų buvo išsirinktas VGG, kadangi jo veikimas ir sluoksnių išsidėstymas yra tinkamiausi išsikeltam tikslui įgyvendiniti. Buvo nuspręsta daryti paprastą, binarinę paveikslėlių klasifikaciją - nustatymas ar katė, ar šuo pavaizduotas paveikslėlyje.

4.1. Duomenų rinkinys

Šiais laikais konvoliuciniai neuroniniai tinklai pranoksta prieš tai buvusias naujausias technologijas naudotas paveiksliukų klasifikacijai. Viena iš pagrindinių priežasčių yra dideli ir gerai aprašyti duomenų rinkiniai, kaip kad ImageNet duomenų bazė. Geriausiai tinklas pasirodo, kai yra apmokomas su bendrinėm ypatybėm ir dar efektyviau pasirodo kai yra sureguliuoti su specifiniu duomenų rinkiniu [**http://adas.cvc.uab.es/task-cv2016/papers/0002.pdf**]. Optimalus duomenų kiekis, kad gerai sureguliuoti modelį yra apie 10 000 paveiksliukų.

Tačiau realybėje visų klasių objektų paveiksliukų nėra be galo daug, kadangi reikia skirti labai daug laiko ir žmogiškųjų resursų paveiksliukų žymėjimui bei turėti tiek daug paveiksliukų. Todėl tenka tinklus apmokyti su limituotais duomenų rinkiniais. Pagal šią realaus pasaulio problemą buvo įvykdyta viena iš ekspermento tikslo sąlygų - mažas duomenų rinkinys. Buvo surastas ir naudotas duomenų rinkinys, kuris turi 10 000 paveikslėlių - 8 000 mokymosi tikslui ir 2 000 validacijos.

4.2. Programos veikimas

Skyrije „Technologijos" yra išvardintos visos technologijos, kurios buvo naudotos šio ekspermento programai parašyti.

Pirmiausiai reikia paruošti kompiuterį darbui - įrašyti „Python" programavimo įrankius, paruošti „Anaconda" komandinę eilutę, „NVIDIA CUDA" įrankius, Keras ir TensorFlow. Tačiau kompiuteris privalo turėti tinkamą procesorių ar grafinį procesroiaus bloką - kompiuteris, kurį naudojau turėjo Nvidia 1080 Ti. Programa, kuri reguliuoja modelį ir modifikuoja, nelabai skiriasi, tad dėstysiu modelio reguliavimo eigą, kuri bus vėliau pritaikyta ir modifikavimo daliai.

Modelio reguliavimui pirmiausia reikia turėti modelį, kurį galima reguliuoti. Keras pateikia modelius, apmokytus su ImageNet, ir jų svorius, tad reikia tik importuoti tinkamą modelį - šiuo atveju VGG16. Importavimo metu reikia nustatyti, kad pilnai sujungtas sluoksnis nebūtų pridėtas, kadangi tada galime pasrinkti kokių dimensijų paveiksliukus naudosime ir kiek spalvų sluoksnių jie turės (standartiniai paveiksliukai turi 3). Tuomet reikia nustatyti kokius importuoto modelio sluoksnius norime mokinti ir kuriuos - ne. Iš visų esamų sluoksnių mokinimui nustačiau tik paskutinius 4, kadangi jie yra atsakingi už specifinių ypatybių išmokimą. Po šių egzistuojančio modelio paruošimų reikia sukurti naują Kero modelį, prie kurio reikia pridėti paruoštą egzistuojantį modelį

bei pridėti kelis kitus sluoksnius tam tikru išsidėstymu - plokštinimo, tankumo, išmetimo ir tankumo. Pirmasis išmetimo sluoksnis turi aktyvacijos funkciją ReLU, o paskutinis - sigmoidinę. Po visų sluoksnių pridėjimo modelis iš viso turėjo 40 406 849 parametrų, iš kurių mokinami buvo 32 771 585, o tik 7 635 264 buvo nekeičiami.

Naudotų sluoksnių paaiškinimai:

- Plokštinimo sluoksnis - skirtas tam, kad įeinančius duomenis suploti į atitinkamą sluoksnių skaičių, jeigu sluoksnio prametras nenustatytas suplojama į vieną sluoksnį.
- Tankumo sluoksnis - atlieka tokią pačią funkciją kaip kad pilno sujungimo sluoksnis (daugiau informacijos poskyriuje „Pilno sujungimo sluoksnis").
- Išmetimo sluoksnis - sluoksnyje atsitiktinai yra išjungiami tam tikri neuronai su Bernulio pasiskirstymo tikimybe. Dažniausiai nustatytas 50 procentų.

Po modelio paruošimo, reikia nustatyti kokio dydžio paveikslėlių paketais bus mokomas ir validuojamas modelis. Kadangi partijų dydį reikia nustatyti pagal kompiuterio atmintį, po keletos bandymų nustačiau, kad geriausi dydžiai yra mokymosi partijai 200 paveikslėlių, o validacijos - 50. Tuomet nustatomas paveikslėlių aplankalo kelias, jų ir partijos dydis bei nustatomas klasės režimas į binarinį, nes duomenų rinkinys susideda iš dviejų klasių - kačių ir šunų.

Paruošę modelį bei paveikslėlių rinkinį, galima pradėti mokinti ir validuoti modelį. Taigi, nustatomas modelio kompiliavimo metodas, kur turi būti - nuostolio ir optimizavimo funkcijos. Daugiau apie nuostolio ir optimizavimo funckcijas galima skaityti skyriuose „Nuostolio funckcijos" ir „Optimizavimo funckcijos". Tačiau nuotolio funkcija šiame modelyje yra binarinė kryžiaus entropija bei ji yra nekeičiama viso ekspermento metu. Šio ekspermento metu bus naudojamos keturios optimizavimo funkcijos - SGD, Adam, Adagrad ir RMSprop. Po modelio sukompiliavimo yra aprašomas mokymo metodas, kuriam turi būti pateikta - paruošti paveiksliukai, žingsinis per epochą, epochų kiekis, paruošti validacijos paveiksliukai ir jų žingsnis. Taigi, šiame ekspermente epocha yra nustayta 40, kadangi tokia išeina pagal visų turimų paveiksliukų skaičių ir paketo dydį.

Modelio modifikavimo veikimas yra toks pats išskyrus, kad pridedami ne 4 sluoksniai, o 14 sluoksnių, kurie visi yra tankumo sluoksniai su aktyvacijos funkcija ReLu. Jie yra pridedami tarp plokštinimo ir kito tankumo sluoksnio. Po visų sluoksnių pridėjimo iš viso buvo 51 952 449 parametrų, iš kurių buvo 44 317 185 mokinami, o tiek pat buvo nemokinami kaip ir reguliuojant modelį. Taigi, modifikavimo metu prisidėjo papildomi 11 545 600 parametrų.

4.3. Modelio bandymai?????

- CNN issirinkimas - Duomenu radimas - CNN reguliavimo programos parasymas - bei programos pritaikymas modifikavimui Skirtingu optimizavimo funkciju pritaikymas ir rezultatai

Rezultaitai ir išvados

Darbo rezultatai:

- Pasirinktas tiriamas konvoliucinis neuroninis tinklas - VGG.
- Rasta geriausia ir tinkamiausia optimizavimo funkicja - XX.
- VGG buvo pritaikytas ir modifikuotas išsikeltam tikslui.
- Konvoliucinis neuroninis tinklas buvo apmokytas.

Darbo išvados:

-
-

Literatūra

[Agr17] Apoorva Agrawal. Loss functions and optimization algorithms. https://medium.com/data-science-group-iitr/loss-functions-and-optimization-algorithms-demystified-bb92daff331c. 2017.

[Dav15] Cameron Davidson-Pilon. *Bayesian methods for hackers*. Addison-Wesley Professional, 2015.

[GBC16] Ian Goodfellow, Yoshua Bengio ir Aaron Courville. *Deep learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[YCB+14] Jason Yosinski, Jeff Clune, Yoshua Bengio ir Hod Lipson. How transferable are features in deep neural networks? *Corr*, abs/1411.1792, 2014. arXiv: 1411.1792. URL: http://arxiv.org/abs/1411.1792.

[Kar16] Ujjwal Karn. An intuitive explanation of convolutional neural networks. https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/. 2016.

[Li15] Fei-Fei Li. Convolutional neural networks. https://cs231n.github.io/convolutional-networks/. 2015.

[PG17] Josh Patterson ir Adam Gibson. *Deep learning*. O'Reilly Media, Inc., 2017.

[Woo18] C. Woodford. Neural networks. https://www.explainthatstuff.com/introduction-to-neural-networks.html. 2018.

[WRS+17] A. C. Wilson, R. Roelofs, M. Stern, N. Srebro ir B. Recht. The Marginal Value of Adaptive Gradient Methods in Machine Learning. *Arxiv e-prints*, 2017. eprint: 1705.08292 (stat.ML).

[XWC+15] Bing Xu, Naiyan Wang, Tianqi Chen ir Mu Li. Empirical evaluation of rectified activations in convolutional network. *Corr*, abs/1505.00853, 2015. arXiv: 1505.00853. URL: http://arxiv.org/abs/1505.00853.