

VILNIAUS UNIVERSITETAS
INFORMATIKOS INSTITUTAS
PROGRAMŲ SISTEMŲ KATEDRA

**Mažos duomenų imties problemos poveikis
klasifikacijos tikslumui naudojant dirbtinius
neuroninius tinklus**

**The Effect of a Small Dataset Problem on Classification
Accuracy Using Artificial Neural Networks**

Bakalauro baigiamasis darbas

Atliko:	Miglė Vaitulevičiūtė	(parašas)
Darbo vadovas:	asist. dr. Vytautas Valaitis	(parašas)
Darbo recenzentas:	j. asist. Linas Petkevičius	(parašas)

Vilnius – 2019

TURINYS

SANTRAUKA	3
SUMMARY	4
IVADAS	5
1. DIRBTINIS NEURONINIS TINKLAS	7
1.1. Dirbtinio neuroninio tinklo sudėtis	7
1.2. Dirbtinio neuroninio tinklo veikimas.....	7
1.3. Aktyvavimo funkcijos.....	8
1.4. Nuostolio funkcijos	9
1.5. Optimizavimo funkcijos	9
1.6. Hiperparametrai	11
2. KONVOLIUCINIS NEURONINIS TINKLAS	12
2.1. Konvoliucija	12
2.2. Konvoliucinio neuroninio tinklo sluoksniai.....	13
2.2.1. Konvoliucinis sluoksnis	13
2.2.2. Sujungimo sluoksnis	14
2.2.3. Pilno sujungimo sluoksnis	14
2.3. AutoML.....	14
2.4. Architektūros paieškos algoritmas	15
2.4.1. Paieškos erdvė	15
2.4.2. Paieškos strategija	15
2.4.3. Vykdyto vertinimo strategija	16
2.5. Architektūros	17
2.6. Modelio derinimas	19
2.7. Klaidos matrica	19
2.7.1. Tikslumas	20
2.7.2. Atšaukimas	20
2.7.3. Precizija.....	21
3. TECHNOLOGIJOS	22
3.1. ImageNet	22
3.2. Keras	22
3.3. TensorFlow	23
4. MODELIŲ DERINIMAS SU MAŽU DUOMENŲ RINKINIU	24
4.1. Tinklelio paieška	24
4.2. Programos veikimas.....	24
4.3. Modelių derinimas	25
4.3.1. Negilaus modelio derinimas	25
4.3.2. Gilaus modelio derinimas.....	27
REZULTATAI	28
IŠVADOS	29
LITERATŪRA	30
SAVOKŲ APIBRĖŽIMAI IR SANTRUMPOS	35
PRIEDAI	35
1 priedas. Tinklelio paieška	36

Santrauka

Šį darbą sudaro teorinė ir eksperimentinė dalys. Teorinėje dalyje pirmiausia yra aprašomas dirbtinis neuroninis tinklas, jo sudėtis ir veikimas. Toliau yra apibrėžiami konvoliuciniai neuroniniai tinklai, jų sudėtis bei esamos architektūros. Taip pat yra apibrėžiamas AutoML ir architektūros paieškos algoritmas, kadangi eksperimentinėje dalyje yra naudojamas NASNet modelis. Šis modelis buvo pasirinktas dėl savo inovatoriškumo bei AutoML panaudojimo. Eksperimentinėje dalyje du NASNet modeliai (NASNet Large ir NASNet Mobile) yra suderinami su dviem skirtingais mažo kiekio duomenų rinkiniais bei keičiant mokomų sluoksnių kiekį. Naudojami du NASNet modeliai, nes jų gylyai skiriasi ir taip galima pamatyti kokią įtaką modelio gylis daro tikslumui. Gauti bandymų rezultatai lyginami, pateikiamos išvados.

Raktiniai žodžiai: NASNet, AutoML, derinimas, konvoliuciniai neuroniniai tinklai, neuroninės architektūros paieškos algoritmas, tikslumas, mažos duomenų imties problema

Summary

This work consists of theoretical and experimental parts. The theoretical part firstly describes the artificial neural network, its structure and functioning. Next, convolutional neural networks, their composition and existing architecture are defined. Also, the AutoML and architecture search algorithm are defined, as the NASNet model is used in the experimental part. This model has been chosen for its innovation and the use of AutoML. In the experimental part, two NASNet models (NASNet Large and NASNet Mobile) are fine-tuned with two different small datasets while changing the number of training layers. Two NASNet models are used, because they differ in depth and can be used to see the impact of model depth on accuracy. The obtained test results are compared and conclusions are given.

Keywords: NASNet, AutoML, fine-tune, convolutional neural networks, neural architecture search algorithm, accuracy, small dataset problem

Ivadas

Vienas iš dirbtinių neuroninių tinklų tipų yra konvoliuciniai neuroniniai tinklai, kurių viena iš sprendžiamų problemų yra klasifikacijos uždavinys [Fuk80; LHB⁺99]. Tai yra procesas, kurio metu yra ieškoma panašių požymių (angl. feature) tarp skirtingų objektų, pavyzdžiui, paveiksliukų, ir pagal tai jie yra skirstomi į atitinkamas klases [Pau01]. Klasifikacija yra labai aktuali, kadangi ji yra naudojama nuo medicinos iki savivaldžių automobilių.

Konvoliuciniai neuroniniai tinklai gali būti naudojami:

- Veido atpažinimui - identifikuoti arba verifikuoti asmenį. Pavyzdžiui, „DeepFace“ sistema sukurta „Facebook“ [TL14], kuri atpažįsta žmonių veidus nuotraukose, arba „Face ID“ sistema sukurta „Apple“, kuri yra skirta identifikuoti asmenį, kuris bando atrakinti telefoną.
- Medicinoje - širdies, plaučių, prostatos, krūties vėžių [GVR⁺10], akių ligų diagnozavimui [Gau15].
- Žmonių elgesio analizė realiu laiku – „DeepGlint“ nustato žmones nuotraukose ir nuspėja jų elgesį [BX16].
- Vertimas – „Google Translate“ gali versti tekstą iš paveiksliukų realiu laiku [Ras15].

Tam kad neuroninius tinklus būtų galima panaudoti kažkokioje sferoje reikia turėti neuroninių tinklų ekspertą bei tinkamai sužymėtą ir didelį duomenų rinkinį (angl. dataset). Mokslininkai ir komercinės įmonės neturinčios žinių neuroninių tinklų ar mašininio mokymosi srityse negali pasinaudoti jų teikiamais privalumais, nes egzistuoja per daug kintamųjų, pavyzdžiui, architektūros pasirinkimas, hiperparametrų optimizavimas, kurie lemia ar tinklas gerai atliks jam paskirtą užduotį [HKV19; LMH⁺19]. Taip pat visiems neuroniniams tinklams reikia didelio kiekio duomenų, kadanagi jų tikslumas didėja logaritmiškai [CLS⁺15; SSS⁺17]. Tačiau realiaame pasaulyje duomenų kiekis ir žmogiškieji bei laiko resursai yra riboti, todėl yra siekiama keičiant dirbtinio tinklo architektūrą bei jo parametrus gauti kuo didesnę tikslumą. Tad, architektūros paieškos ir parametrų nustatymo automatizavimas turėtų leisti pigiau ir greičiau sukurti efektyvų bei tikslų neuroninį tinklą negu kad tai darant rankiniu būdu [RMS⁺17]. Tokiam procesui atlikti yra naudojamas automatizuotas mašininis mokymasis (angl. Automated Machine Learning (toliau - AutoML)).

2017 metais Google komanda naudodami AutoML sukūrė naują architektūrą NASNet [ZVS⁺17], kuri yra sudaryta iš dviejų tipų blokų, kurie yra sudėti vienas ant kito. Šiai architektūrai sukurti buvo naudojamas neuroninės architektūros paieškos (angl. neural architecture search) algoritmas. Jis automatiškai keisdamas vidinius parametrus ieško geriausios architektūros specifiniai duomenų imčiai. Taip pat NASNet modelio tikslumas yra 1,2% didesnis negu bet kokios kitos

žmogaus sukurtos architektūros [ZVS⁺17].

Bakalauro darbo tikslas yra palyginti skirtingų gylių NASNet modelių veikimą su mažu duomenų rinkiniu.

Uždaviniai:

1. Apžvelgti dirbtinių neuroninių tinklų ir konvoliucinių neuroninių tinklų sudėtį bei NAS algoritmo veikimą.
2. Rasti geriausius hiperparametrus skirtingų gylių modeliams su pasirinktu mažu duomenų rinkiniu.
3. Suderinti (angl. fine-tune) egzistuojančius skirtingų gylių modelius su pasirinktu mažu duomenų rinkiniu.
4. Palyginti ir įvertinti skirtingų gylių modelius pagal gautą derinimo informaciją.

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 sinapsėmis. Per šiuos neuronus sklinda elektroniniai impulsai, perduodantys informaciją. Tokiu būdu žmonės gali atpažinti objektus, garsus ir t.t. Dirbtiniai neuroniniai tinklai veikia panašiai. Jie turi daug besijungiančių neuronų, kurie gauna informaciją ir pagal tą informaciją gali nuspręsti koks tai objektas. Tačiau ties tuo ir baigiasi žmogaus smegenų ir dirbtinių neuroninių tinklų panašumas, kadangi dirbtiniai neuroniniai tinklai yra matematinis algoritmas su aritmetiniais kintamaisiais. Šis algoritmas yra suvokiamas tik žmogui, kuris suprogramavo dirbtinį neuroninį tinklą, pačiam tinklui algoritmas nieko nereiškia, nuovokos nesuteikia.

1.1. Dirbtinio neuroninio tinklo sudėtis

Dirbtinis neuroninis tinklas yra sluoksnių rinkinys - neuronų grupė sudaro sluoksnį, kuris yra sujungtas tarpusavyje su kitais sluoksniais [ZGD03]. Vienas iš sluoksnių privalo būti įvesties sluoksnis, kuris atitinkamai pagal užduotį gali gauti įvairios formos informaciją - paveikslukai, vaizdo medžiaga, garsas ir t.t. Ši informacija yra reikalinga tam, kad tinklas galėtų ją išanalizuoti ir išmokyti, kad vėliau gavęs panašią informaciją galėtų ją atpažinti - tam reikalingas išvesties sluoksnis. Jis yra priešingame dirbtinio neuroninio tinklo gale negu įvesties sluoksnis. Tarp anksčiau apibūdintų sluoksnių yra įvairaus dydžio vidinė sluoksnių sistema, kuri atlieka pagrindinį darbą [Woo18].

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 bei pridedama slenksčio reikšmė (angl. bias). Taip iš vektoriaus gaunamas vienas rezultatas ir jei šis rezultatas tinka aktyvavimo funkcijai, jis yra perduodamas tolimesniems neuronams [Shi12]. Tokio tipo veikimo projektavimas yra vadinamas tiesioginio sklidimo (angl. feedforward) tinklu.

Tačiau jungčių svoriai nėra pastovūs. Kai dirbtinis neuroninis tinklas mokosi, galutinis rezultatas yra lyginamas su tikėtinu teisingu rezultatu (daugiau informacijos „Nuostolio funkcija“),

jei šie rezultatai skiriasi, slenksčio reikšmės ir svoriai yra keičiami atitinkamai [RS17], tai vadinama sklaidimo atgal algoritmu (angl. backpropagation). Mokymo metu duomenys neuroniniu tinklu keliauja į priekį - nuo įvesties į išvesties sluoksnį. Kai išvesties sluoksnis yra pasiekiamas, gautas rezultatas yra palyginamas su norimu rezultatu bei apskaičiuojama nuostolio funkcija - kaip stipriai skiriasi gautas ir norimas rezultatai. Pagal šią reikšmę matoma, kaip reiktų keisti gautą rezultatą, kad nuostolio funkcijos reikšmė pasiektų lokalų minimumą. Tačiau siekiant aukštesnio tikslumo reikia keisti viso neuroninio tinklo parametrus - svorius, slenksčio reikšmes. Taigi, iš išvesties rezultatų galima matyti, kaip reikia pakeisti - didinti arba mažinti - prieš tai buvusio sluoksnio parametrus, kad būtų gautas geriausias tikslumas. Šis procesas yra iteratyviai kartojamas kiekvienam neuronui su prieš jį einančiu sluoksniu bei jį galima įvardinti kaip funkciją (1):

$$\frac{\partial C_0}{\partial w^L} = \frac{\partial z^L}{\partial w^L} \frac{\partial a^L}{\partial z^L} \frac{\partial C_0}{\partial a^L}. \quad (1)$$

Šioje funkcijoje vienas sluoksnis turi vieną neuroną, priklausomai nuo neuronų ir sluoksnių skaičiaus prie funkcijos parametrų prisidėtų atitinkami indeksai. Funkcija parodo dalinės nuostolio funkcijos išvestinės ir dalinės svorio (arba slenksčio reikšmės) išvestinės santykį, kur w^L yra svoris, kurį galima pakeisti į b^L (slenksčio reikšmė), C_0 yra nuostolio funkcijos reikšmė, $z^L = w^L a^{L-1} + b^L$ ir $a^L = \sigma(z^L)$. Šitos funkcijos tikslas yra nustatyti kokį efektą svorio reikšmės pakeitimai turės nuostolio funkcijos reikšmei.

1.3. Aktyvavimo funkcijos

Aktyvavimo funkcijų (angl. activation function) yra įvairių, todėl specifinės problemos gali reikalauti vienos ar daugiau konkrečių aktyvavimo funkcijų [VK11]. Aktyvavimo funkcija yra skirta tam, kad nustatytų ar neuronui reikia būti aktyvuotam ar ne. Tai yra nusprendžiama pagal duomenis, kuriuos neuronas gauna, jeigu jie yra aktualūs, neuronas yra aktyvuojamas, jeigu ne - ignoruojamas. Šią funkciją galima aprašyti žemiau pateikta formule (2):

$$Y = A(\Sigma(w * d) + b). \quad (2)$$

Formulėje (2) pateikta raidė A reiškia bet kokia pasirinkta aktyvavimo funkcija, o jos parametrai w yra svoris, d yra įvesties duomenys ir b yra slenksčio reikšmė. Taigi, ar neuronas bus aktyvuotas priklauso nuo prieš jį buvusio sluoksnio jungčių dydžio, kurios parodo kiek svarbi yra jungtis tarp neuronų, kadangi kuo didesnis svoris tuo didesnis rezultatas gaunamas svorį sudau-

ginus su įvesties duomenimis. Taip pat slenksčio reikšmė parodo ar reikia sustiprinti ar susiplinti gaunamą rezultatą. Y reikšmė priklauso nuo pasirinktos aktyvavimo funkcijos išvesties intervalo. Žemiau yra pateiktos kelios aktyvavimo funkcijos su išvesties intervalais.

Kelios aktyvavimo funkcijos:

- Sigmoidinė (angl. sigmoid function) - išvesties intervale $[0; 1]$.
- Hiperbolinio tangento (angl. hyperbolic tangent) - išvesties intervale $[-1; 1]$.
- Minkštojo maksimumo (angl. softmax function) - sunormuoja išvesties vektorių į 1.
- ReLU - išvesties intervale $[0; \text{begalybė}]$.

1.4. Nuostolio funkcijos

Mokantis dirbtiniam neuroniniam tinklui 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 kvadratinio nukrypčio (angl. mean squared error). Ši funkcija apskaičiuoja kvadratinį skirtumą tarp tikėtino ir gauto rezultatų. Tačiau šios funkcijos vienas iš didesnių trūkumų - neproporcingas išskyrimas didelių rezultatų. Kadangi funkcija didėja kvadratiškai, o ne tiesiškai, tai gaunamas rezultatas tolsta nuo tikėtino rezultato.

Priklausomai nuo to kokią problemą yra bandoma išspręsti yra naudojamos skirtingos funkcijos. Viena iš problemų yra klasifikacijos - dažniausiai išvesties rezultatas yra tikimybės vertė $f(x)$. Bendrai, funkcijos reikšmės dydis parodo gauto rezultato tikslumą.

Kelios klasifikacijos nuostolio funkcijos:

- Binarinė kryžiaus entropija (angl. binary cross entropy).
- Neigiama registravimo tikimybė (angl. negative log likelihood).
- Maržos klasifikatorius (angl. margin classifier).
- Minkštų maržų klasifikatorius (angl. soft margin classifier).

1.5. Optimizavimo funkcijos

Optimizavimo funkcijos naudojamos vidinių tinklo parametrų atnaujinimui, kad sumažinti gaunamų rezultatų netikslumą [Nik16]. Visos optimizavimo funkcijos gali būti suskirstytos į du tipus - nuolatinio mokymosi greičio ir prisitaikančio mokymosi. Lentelė 1 buvo parengta remiantis [Rud16] straipsniu. Lentelėje išvardintos visos populiariausios optimizavimo funkcijos.

1 lentelė. Optimizavimo funkcijos

Pavadinimas	Tipas	Privalumai	Trūkumai	Veikimas
SGD	Nuolatinio mokymosi greičio	Parametrų atnaujinimai turi aukštą dispersiją, kas leidžia lengviau rasti lokalų minimumą.	Didelis svyravimas trukdo konverguoti.	Parametrų atnaujinimas vykdomas kiekvienai mokymo iteracijai.
Adam	Prisitaikančio mokymosi	Greitai konverguoja ir modelio mokymosi greitis yra didelis bei efektyvus.	Praleidžia mažą lokalų minimumą.	Suskaičiuoja mokymosi greitį kiekvienam parametrui bei saugo eksponentiškai nykstantį prieš tai buvusį kvadratinio gradiento vidurkį ir eksponentiškai mažėjantį prieš tai buvusį gradiento vidurkį, panašų į pagreitį.
Adagrad	Prisitaikančio mokymosi	Nereikia rankiniu būdu derinti mokymosi greičio.	Mokymosi greitis visada yra mažėjantis ir nykstantis, kas lėtina konvergavimą.	Leidžia mokymosi greičiui priklausyti nuo parametrų. Dideli atnaujinimai nedažniems parametrams, maži atnaujinimai dažniems parametrams.
RMSprop	Prisitaikančio mokymosi	Greitai konverguoja.	Pagreitis nedidina funkcijos efektyvumo.	Dalija mokymosi greitį iš eksponentiškai nykstančio kvadratinio gradiento vidurkio.

Skyriuje „Dirbtinio neuroninio tinklo veikimas“ minėta, kad sklidimo atgal algoritmas pagal gauto ir norimo rezultatų skirtumą keičia vidinius neuroninio tinklo parametrus. Vidinių paramet-

rų atnaujinimui yra naudojama optimizavimo funkcija, kuri apskaičiuoja gradientą. Svariai yra keičiami pagal priešingą apskaičiuoto gradiento kryptį - bandoma leisti į gradiento minimumą.

Optimizavimo funkcijos turi parametą - mokymosi greitį (angl. learning rate). 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 [Leo98]. Optimizavimo funkcijos tikslas yra surasti lokalų minimumą, o to pasiekti galima gradientu judant į žemiausią jo vietą, tačiau pasirinkus per didelį mokymosi greitį yra galimybė, kad žemiausia vieta bus peršokta ir bus tolstama nuo jos.

1.6. Hiperparametrai

Mašininame mokymesi terminas hiperparametras (angl. hyperparameter) yra naudojamas atskirti parametrus, kurie nėra išmokstami iš duomenų, kurie yra naudojami mokant modelį. Hiperparametrai apima kintamuosius skirtus reguliuoti neuroninį tinklą. Jie turi labai didelį poveikį neuroninio tinklo tikslumui, tačiau pasirinkti jų vertes gali būti sudėtinga, nes įtaką daro architektūra, duomenų rinkinys. Optimalių hiperparametrų radimui yra sugalvota daug metodų, kurie yra aprašyti „Paieškos strategija“ skyriuje. Dalis hiperparametrų yra: mokymosi greitis, epochų ir partijų (angl. batch) vertės, vidinių sluoksnių kiekis, išmetimo sluoksnio reikšmė, aktyvacijos, optimizacijos ir nuostolio funkcijos.

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 neuroninio tinklo įvesties sluoksnius priima duomenis, kurie gali būti konvertuojami į 2D matricą, pavyzdžiui, paveiksliukai, 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 [YK18].

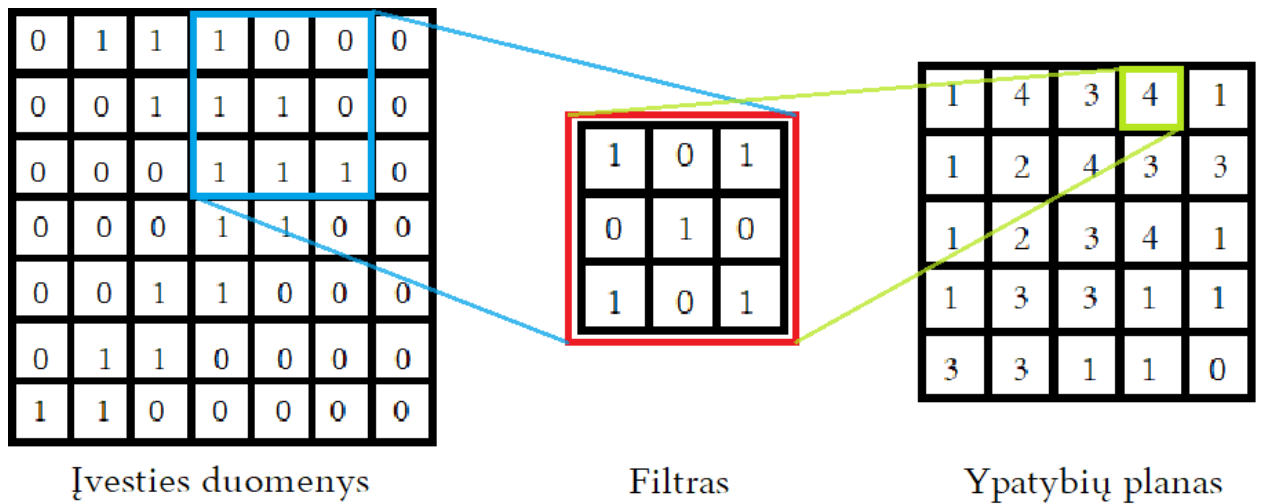
Objektų atpažinimas paveiksliukuose yra sudėtingas dėl šių iššūkių:

- 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 skirtingus dizainus.
- Žvilgsnio taškas - keičiant vietą iš kurios yra žiūrima gali keistis objekto forma, informacija šokinėja per įvesties sluoksnio dimensiją (t.y. pikselius).

2.1. Konvoliucija

Konvoliucija yra matematinė operacija, kuri apibūdina taisyklę, kuri parodo kaip reikia sujungti du informacijos rinkinius [PG17].

Pagal paveiksliuką (1 pav.) matyti, kad įvesties duomenys ir filtras, kuris yra sudarytas iš svorių, yra pateikti 2D matrica. Filtras juda nuo duomenų matricos kairės viršutinės dalies į dešinę, tada yra nuleidžiamas žemiau per vieną eilutę ir taip filtras juda per visą duomenų matricą, kol su visais jos duomenimis filtras yra sudauginamas ir užpildo naują matricą, kuri yra vadinama požymių planu (angl. feature map). Tačiau konvoliuciniai tinklai turi daug filtrų, kurie pereina per vieną paveiksliuką, kiekvienas išskirdamas skirtingą paveiksliuko ypatybę [Epp17]. Pirmuose sluoksniuose šiuos filtrus galima apibūdinti kaip horizontalių, vertikalinių ar įstrižių linijų filtrus, kurie sukuria paveikslėlio kraštų planą.



1 pav. Konvoliucijos veikimas

2.2. Konvoliucinio neuroninio tinklo sluoksniai

Konvoliuciniai neuroniniai tinklai tai yra sluoksnių rinkinys, kuris turi įvesties, vidinius ir išvesties 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 [RPA⁺18].

Nepagrindinių sluoksnių paaiškinimai:

- Plokštinimo sluoksnis (angl. flatten layer) - skirtas tam, kad įeinančius duomenis suploti į atitinkamą sluoksnių skaičių, jeigu sluoksnio parametras nenustatytas suplojama į vieną sluoksnį.
- Išmetimo sluoksnis (angl. dropout layer) - sluoksnyje atsitiktinai yra išjungiami tam tikri neuronai su Bernulio pasiskirstymo tikimybe, kuri priima dvi reikšmes 1 (sėkmė) ir 0 (nesėkmė) bei šių reikšmių tikimybe p ir $1 - p$. Dažniausiai yra nustatytas 50 procentų.

2.2.1. Konvoliucinis sluoksnis

Konvoliucinis sluoksnis (angl. convolutional layer) yra pagrindinis konvoliucinio neuroninio tinklo sluoksnis, kuris nustato visas paveiksluko ypatybes. Kadangi įvesties informacija (paveikslukas) yra didelės dimensijos, neefektyvu visų neuronų sujungti vienus su kitais, todėl neuronai yra sujungiami su lokaliu informacijos kiekiu, kuris yra lygus filtro dydžiui ir vadinamas erdvinio mastu (angl. receptive field) [Li15].

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

- Gylis (angl. depth) - atitinka filtrų skaičių.
- Žingsnis (angl. stride) - pikselių kiekis, kuris parodo per kiek reikia slinkti filtro matrica per

įvesties informacijos matricą.

- Nulių pamušalas (angl. zero-padding) - įvesties informacijos matricos kraštus užpildyti nuliais.

2.2.2. Sujungimo sluoksnis

Periodiškai sujungimo sluoksnis (angl. pooling layer) yra įterpiamas tarp konvoliucinių. Pagrindinis sluoksnio tikslas yra laipsniškai mažinti erdvinį filtruojamo paveiksluko mastą. Šis veiksmas yra atliekamas tam, kad sumažinti parametrų ir skaičiavimų kiekį. Maksimumo sujungimo (angl. max pooling) sluoksnis, nepriklausomai nuo kiekvieno sluoksnio gylio, yra erdviškai (ilgis ir plotis) keičiamas ir rezultatas gaunamas naudojant MAX operaciją [LGT18]. Dažnai šis sluoksnis yra naudojamas su 2x2 dydžio filtru - įvesties duomenys yra suskaidomi į keturias lygias dalis ir iš kiekvienos dalies paimama didžiausia tos dalies reikšmė, iš šių reikšmių sudaroma nauja matrica. Egzistuoja ne tik maksimumo sujungimo sluoksniai, bet ir vidurkio sujungimo (angl. average pooling) - jame yra randama ne didžiausia matricos dalies reikšmė, o suskaičiuojamas vidurkis.

2.2.3. Pilno sujungimo sluoksnis

Pilno sujungimo sluoksnis (angl. fully connected layer) 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 įvesties paveikslukas pagal mokymo informacijos imtį, kai neuroninio tinklo problema yra klasifikacija [Kar16]. Šiam sluoksniui yra priskiriama aktyvacijos funkcija, kuri neprivalo būti tokia pati kaip vidiniuose sluoksniuose naudota aktyvacijos funkcija.

2.3. AutoML

Automatizuotas mašininis mokymasis (angl. automated machine learning (toliau - AutoML)) siekia pilnai automatizuoti mašininio mokymosi pritaikymą realaus pasaulio problemoms spręsti. Siekiamybė yra kad vartotojui tereiks pateikti duomenų rinkinį ir AutoML sistema automatiškai nustatys koks būdas yra tinkamiausias pateiktam duomenų rinkiniui. AutoML gali pagerinti našumą ir sutaupyti didelį kiekį laiko ir pinigų, nes nereikės samdyti mašininio mokymo ekspertų [HKV19].

AutoML tobulinimui spartinti yra rengiamas konkursas - ChaLearn AutoML. Jis pirma kartą surengtas buvo 2014 metais. Konkursas susideda iš kelių etapų - kodo pateikimo, jo vertinimo (validavimo) su viešais duomenų rinkiniais ir atsiliepimų gavimo bei kodo modifikavimo, o finaliniame

etape paskutinis kodo pateikimas yra patikrinamas su penkiais skirtingais privačiais duomenų rinkiniais. Pagal gautus rezultatus dalyviai yra reitinguojami.

2.4. Architektūros paieškos algoritmas

Neuroninės architektūros paieškos (angl. neural architecture search (toliau - NAS)) algoritmas ieško geriausios neuroninio tinklo architektūros naudodamas specifinį duomenų rinkinį [WZL⁺19]. NAS gali būti matomas, kaip kad AutoML polaukis [HKV19]. NAS metodai yra kategorizuojami pagal tris dimensijas: paieškos erdvę (angl. search space), paieškos strategiją (angl. search strategy) ir įvykdymo vertinimo strategiją (angl. performance estimation strategy) [EMH18].

2.4.1. Paieškos erdvė

Paieškos erdvė apibrėžia kokia architektūra gali būti pateikta panaudojant žinias apie būdingas savybes esamų architektūrų, kurios yra tinkamos specifiniai užduočiai spręsti, gali stipriai sumažinti paieškos erdvės dydį ir supaprastinti pačią paiešką. Tačiau pradinių savybių nustatymas priklauso nuo žmogiškumo faktoriaus, kuris gali neleisti atrasti naujų architektūros sudedamųjų blokų. Dažniausios paieškos erdvės:

- Grandinės struktūros neuroninio tinklo (angl. chain-structured neural network) - gali būti aprašyta kaip n sluoksnių seka, kur i -tasis sluoksnis L_i gauna duomenis iš $i - 1$ sluoksnio ir juos toliau siunčia $i + 1$ sluoksniui. Pati erdvė yra parametrizuota pagal maksimalų sluoksnių skaičių, operacijos tipą, kurią vykdo kiekvienas sluoksnis, ir hiperparametrus, kurie asocijuoti su sluoksnių operacijomis.
- Daugiašaknio tinklo (angl. multi-branch network) - naudoja peršokimo jungtis (angl. skip connection) bei duomenys keliauja per visas paralelias šaknis, po kurių gauti duomenys yra sujungti ir perduoti toliau esančiam sluoksniui [AT17].
- Celėmis pagrįsta (angl. cell-based) - naudoja normalią (angl. normal) ir mažinimo (angl. reduction) celes. Pirmoji iš celių išlaiko įeinančių duomenų erdvines dimensijas, o kita - mažina. Galutinė architektūra yra sudaroma iš celių, kurios yra sudedamos nustatytu būdu.

2.4.2. Paieškos strategija

Paieškos strategija aprašo kaip reikia tirti paieškos erdvę - surasti tinkamiausius architektūros hiperparametrus naudojant specifinį duomenų rinkinį. Paieškos erdvei tyrinėti yra naudojami

įvairūs metodai, kurie yra taip pat skirti hiperparametrų optimizacijai (angl. hyperparameter optimization). Dėl paieškos strategijos dimensijos NAS turi didelį persidengimą su hiperparametrų optimizavimu [FH18]. Skirtingos paieškos strategijos:

- Tinklelio paieška (angl. grid search) - išsamiai išnagrinėja rankiniu būdu nustatytą hiperparametrų konfigūraciją iki priimtino modelio tikslumo.
- Atsitiktinė paieška (angl. random search) - bandomos atsitiktinės hiperparametrų kombinacijos, kad rasti geriausią modelį. Ši strategija randa geresnius modelius net jei yra ieškoma didesnė ir mažiau perspektyvi paieškos erdvė [BB12].
- Evoliuciniai metodai (angl. evolutionary methods) - vysto modelių populiaciją (tinklų rinkinį). Kiekviename evoliucijos žingsnyje bent vienas modelis iš populiacijos yra atrenkamas ir tampa pagrindiniu (tėviniu), kad jam taikant mutacijas būtų sugeneruoti palikuonys. Mutacijos NAS kontekste yra lokalsios operacijos, kaip kad pridėjimas ar atėmimas sluoksnio, pridėjimas peršokimo jungties, hiperparametrų keitimas. Apmokius palikuonis jų tikslumas yra patikrinamas su valdiacijos duomenų rinkiniu, tada jie yra pridunami prie populiacijos. Tuomet populiacija yra išrūšiuojama pagal tikslumą ir blogiausių pusė yra išimama iš populiacijos. Šie žingsniai vykdomi iki tol kol baigiasi nustatyti resursai [vWB18].
- Bajeso optimizacija (angl. Bayesian optimization) - efektyviai randa nežinomos funkcijos globalųjį maksimumą apibrėžtoje paieškos erdvėje. Ši strategija susideda iš dviejų dalių. Pirmoji yra surogatinis modelis, kuris susideda iš ankstesnio pasiskirstymo, kuris užfiksuoja įsitikinimus apie nežinomos objektyvios funkcijos elgesį, ir stebėjimo modelio, apibūdinančio duomenų generavimo mechanizmą. O antroji yra nuostolio funkcija, kuri aprašo užklausų sekos optimalumą [SSW⁺16].
- Skatinamasis mokymasis (angl. reinforcement learning) - generavimas architektūros gali būti laikomas agento veiksmu, kai veiksmų erdvė yra identiška paieškos erdvei. Agento apdovanojimas yra pagrįstas apmokytos architektūros veikimo tikslumu su nematytais duomenimis. Agentui nėra apibrėžiama kokius veiksmus reikia daryti, bet jis pats privalo atrasti veiksmus, kurie duoda daugiausia apdovanojimų, juos bandydamas [SB98].

2.4.3. Vykdomo vertinimo strategija

NAS tikslas yra surasti architektūrą, kuri gali įgyti aukštą nuspėjimo nustatymą (tikslumą) su nematytais duomenimis. Vykdomo vertinimas reikalingas įvertinti šį nustatymo procesą. Paprasčiausias būdas yra apmokyti ir validuoti architektūrą, bet tai daryti yra brangu ir limituoja skaičių

architektūrų, kurias būtų galima ištirti. Tokiam architektūrų įvertinimui atlikti yra naudojami metodai, kurie sumažintų vykdymo vertinimo kainą:

- Nepilno veikimo tikslumo vertinimas (angl. lower fidelity estimates) - veikimas gali būti vertinamas remiantis tikslumu po pilno mokymo naudojant tikrojo veikimo mažesnią dalį - mažesnę mokymo laiko intervalą, dalį mokymo duomenų rinkinio, mažesnę paveiksliukų rezoliuciją ar mažesnę filtrų kiekį per sluoksnį ir mažiau celių.
- Mokymosi kreivės ekstrapoliacija (angl. learning curve extrapolation) - mokymo laikas gali būti sumažinamas, kadangi veikimas gali būti ekstrapoliuotas po kelių mokymo epochų.
- Svorio paveldėjimas arba tinklo morfizmai (angl. weight inheritance or network morphisms) - naujos architektūros svorius nustatyti pagal jau apmokytos architektūros svorius, dėl to nereikia mokinti architektūros nuo pradžių, nes svoriai yra paveldimi. Tinklo morfizmas leidžia modifikuoti architektūrą paliekant funkciją, kuri yra reprezentuojama to tinklo, nepakeistą.
- Vieno bandymo modeliai arba svorio dalinimasis (angl. one-shot models or weight sharing) - tik vienas modelis turi būti apmokytas, nes jo svoriai yra padalijami per visas skirtingas architektūras, kurios yra pografiai pradinio modelio.

2.5. Architektūros

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

2 lentelė. Konvoliucinių neuroninių tinklų architektūros

Pavadinimas	Metai	Parametrų kiekis	Veikimas	ILSVRC vieta
LeNet	1998	60 000	Geriausiai atpažįsta ranka parašytus skaičius. Susideda iš sluoksnių - kelių pasikartojančių konvoliucijos ir sujungimo bei pasibaigia dviem pilno sujungimo sluoksniais [LJB ⁺ 95].	-
AlexNet	2012	60 000 000	Veikimu panašus į LeNet, tačiau turi daug daugiau parametrų ir filtrų bei sudėtus konvoliucinius sluoksnius [ATY ⁺ 18].	pirma

2 lentelė. Konvoliucinių neuroninių tinklų architektūros

Pavadinimas	Metai	Parametrų kiekis	Veikimas	ILSVRC vieta
GoogLeNet	2014	23 800 000	Vidiniai sluoksniai sudėti paraleliai, naudojami „Inception“ moduliai. Vienas modulis savyje turi 1x1, 3x3 ir 5x5 dydžių konvoliucijos filtrų bei vidurkio sudėjimo sluoksnius [SLJ ⁺ 14].	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 [SZ15].	antra
ResNet	2015	25 000 000	Turi labai daug sluoksnių, sudėtų vienas po kito, kurie turi liekamąjį (angl. residual) bloką, kuris įvesties informaciją perduoda tolimesniam sluoksniui ją pridėdamas ir taip sumažina konvoliucijos ir aktyvavimo funkcijų kiekį [TAL16].	pirma
CUIImage	2016	-	Dvipusis dvikryptis tinklas, kuris perduoda žinutes tarp skirtingų paramos regionų [PWR ⁺ 18].	pirma
SENet	2017	145 800 000	Panašus veikimas į ResNet, pridėtas SE blokas, kuris sujungia ypatybių planus ir valdo išėjimus iš kanalo [HSS17].	pirma
NASNet	2017	88 900 000	Google naudodami AutoML ir NAS algoritmą rado geriausią architektūrą ImageNet duomenų bazei. NASNet yra sudaryta iš dviejų tipų celių - normalių ir mažinimo [ZVS ⁺ 17].	-

2.6. Modelio derinimas

Apmokius neuroninį tinklą ir nustačius vidinių parametrų reikšmes gaunamas neuroninio tinklo architektūros modelis. Tokį jau egzistuojantį modelį galima derinti (angl. fine-tune) ir pritaikyti specifiniam užduočiai spręsti. Modelį derinti galima nustačius kelis paskutinius sluoksnius kaip mokomus (angl. trainable) ir su mažiau duomenų galima modelį suderinti.

Toks suderinimas yra galimas, nes pirmuosiuose sluoksniuose neuroniniai tinklai išmoka ypatybių panašių į Gaboro filtrą (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].

2.7. Klaidos matrica

Klaidos matrica (angl. confusion matrix) yra lentelė, kuri yra naudojama parodyti klasifikacijos modelio veikimą su validacijos duomenimis, kai yra žinomos teisingos reikšmės. Matrica leidžia vizualizuoti modelio veikimą bei lengvai parodo maišymasi tarp specifinių klasių.

Klaidų matricoje yra naudojamos šios sąvokos:

- TT (tikri teigiami) - spėjimas yra teigiamas ir jis yra teisingas.
- TN (tikri neigiami) - spėjimas yra neigiamas ir jis yra teisingas.
- NT (netikri teigiami) - spėjimas yra teigiamas, tačiau jis yra neteisingas.
- NN (netikri neigiami) - spėjimas yra neigiamas, tačiau jis yra neteisingas.

Šios sąvokos yra savo sutrumpinimais pavaizduotos 1 pav.

Klaidų matrica

Tikras	Klasė 1	TT	NN
	Klasė 2	NT	TN
		Klasė 1	Klasė 2
		Spėtas	

2 pav. Klaidų matricos pavyzdys

Taigi, jeigu modelis įvardina dalį klasės 1 paveiksliukų kaip klasę 1 tai tų paveiksliukų kiekis yra įrašomas į TT matricos laukelį. Tačiau jei modelis klasės 1 paveiksliukus įvardina kaip klasę 2, tai tas paveiksliukų skaičius yra įrašomas į NT laukelį. Tas pats vyksta su klasės 2 paveiksliukais, jeigu jie yra įvardinami kaip klasės 2 paveiksliukai, tas kiekis įrašomas į TN laukelį, tačiau jei kažkiek paveiksliukų įvardinama kaip klasės 1, tas skaičius įrašomas į NN laukelį.

Pagal klaidų matricos duomenis galima apskaičiuoti kelias modelio metrikas - tikslumas, atšaukimas ir precizija.

2.7.1. Tikslumas

Tikslumas (angl. accuracy) yra matavimas, kuris parodo santykį tarp teisingai spėjimų (angl. prediction) ir iš viso darytų spėjimų.

$$tikslumas = \frac{TT + TN}{TT + TN + NT + NN}. \quad (3)$$

Formulės (3) parametras TT yra skaičius tikrų teigiamų spėjimų, TN yra skaičius tikrų neigiamų spėjimų ir NT yra skaičius netikrų teigiamų spėjimų bei NN yra skaičius netikrų neigiamų spėjimų. Turinti pagal klases nesubalansuotą duomenų rinkinį (pvz. viena klasė turi daug daugiau duomenų negu likusios), tikslumo metrika nepilnai atskleidžia modelio gerumą. Pavyzdžiui, kačių nuotraukų yra 9, o šunų yra 91, modelis teisingai atpažįsta 1 katę ir 90 šunų, tikslumas bus 0.91, nors tik viena katė buvo atpažinta.

2.7.2. Atšaukimas

Atšaukimas (angl. recall) yra metrika, kuri parodo santykį tarp visų teisingai atpažintų spėjimų ir visų esamų teigiamų spėjimų.

$$ataukimas = \frac{TT}{TT + NN}. \quad (4)$$

Formulės (4) parametras TT yra skaičius tikrų teigiamų spėjimų, o NN yra skaičius netikrų neigiamų spėjimų. Sudėjus TT ir NN yra gaunama visos teigiamos reikšmės. Aukštas atšaukimas parodo, kad klasė buvo teisingai atpažinta (mažas NN).

2.7.3. Precizija

Precizija (angl. precision) parodo kiek iš visų teigiamų spėjimų, jų buvo iš tiesų teisingi.

$$precizija = \frac{TT}{TT + NT}. \quad (5)$$

Formulės (5) parametras TT yra skaičius tikrų teigiamų spėjimų, o NT yra skaičius netikrų teigiamų spėjimų. Sudėjus TT ir NT yra gaunama visi daryti teigiami spėjimai. Aukšta precizija reiškia, kad reikšmė įvardinta kaip teigiama iš ties yra teigiama (mažas NT).

3. Technologijos

Naudojamų technologijų išsirinkimas yra pradinis žingsnis siekiant įvykdyti bakalauro darbe išsikeltas užduotis. Šiame skyriuje pateiktos populiariausios šių laikų technologijos bei glaustai apibrėžti jų pagrindiniai funkcionalumai.

3.1. ImageNet

Projektas ImageNet buvo sugalvotas profesorės Li Fei-Fei 2009 metais. Projekto tikslas buvo sukurti didelę sukatégorizuotų paveiksliukų ir jų etikečių duomenų bazę, kuri būtų skirta vizualinio objekto atpažinimo programinės įrangos tyrimams. Ši duomenų bazė yra suorganizuota pagal WorldNet hierarchiją - 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+).

ImageNet projektas kiekvienais metais daro konkursą vadinamą „ImageNet Large Scale Visual Recognition Challenge“ (trumpinys ILSVRC). Konkurso užduotis yra išmokinti modelį, kuris galėtų įvesties 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 eksperimentavimą, kad sugalvojus idėją pasiekti rezultato būtų galima su kiek įmanoma mažiau uždelsimo.

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

3.3. TensorFlow

TensorFlow yra atviros programinės įrangos biblioteka skirta aukšto našumo skaitiniams 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. Modelių derinimas su mažu duomenų rinkiniu

Šio eksperimento tikslas yra išanalizuoti skirtingų gylių neuroninius tinklus pagal metrikas gautas po mokymo ir validavimo naudojant mažą duomenų rinkinį. Poskyriuje „Architektūros“ yra trumpai apibūdinti pagrindiniai konvoliucinių neuroninių tinklų tipai. Iš jų buvo išsirinktas NASNet, kadangi jis yra inovatoriškiausias ir geriausių tikslumą turintis. Negilusis modelis buvo NASNetMobile, o gilusis - NASNetLarge.

Nuspręsta daryti paprastą binarinę paveikslėlių klasifikaciją. Buvo surastos ir naudotos dvi skirtingos duomenų imtys - kačių ir šunų bei virtuvės ir gyvenamojo kambario. Jos abi turėjo 1400 paveikslėlių - 1 300 mokymosi tikslui ir 100 validacijos.

4.1. Tinklelio paieška

Taikant paieškos strategiją - tinklelio paiešką - buvo ieškoma geriausių hiperparametrų NASNetMobile ir NASNetLarge modeliams su dviem skirtingais duomenų rinkiniais ir keičiant mokomų sluoksnių skaičių.

Pagal naudojamo kompiuterio pajėgumą buvo nuspręsta tyrinėti:

- Išmetimo sluoksnio reikšmę - 0.25, 0.5 ir 0.75.
- Optimizavimo funkciją - RMSprop, Adam, Adagrad.
- Mokymosi greitį - 0.0005, 0.0001 ir 0.001.
- Mokomų sluoksnių skaičius - 5, 10, 20.

Iš visų šių hiperparametrų susidarė 27 kombinacijos bei jos buvo analizuojamos 12 kartų - su NASNetMobile ir dviem skirtingais duomenų rinkiniais bei keičiant mokomų sluoksnių skaičių, su NASNetLarge ir tais pačiais duomenų rinkiniais bei keičiant mokomų sluoksnių skaičių. Pagrindinis rezultatų rikiavimo kriterijus buvo tikslumas, antrinis - atšaukimas, o paskutinis buvo precizija. Visi rezultatai pateikti Prieduose esančioje 3 lentelėje. Pagal gautus rezultatus buvo nuspręsta išmetimo sluoksnio reikšmę nustatyti 0.25, optimizavimo funkciją kaip Adagrad su 0.0001 mokymosi greičiu.

4.2. Programos veikimas

Ankstesniame skyriuje „Technologijos“ yra išvardintos visos technologijos, kurios buvo naudotos šiam eksperimentui.

Eksperimentui įvykdyti reikėjo paruošti kompiuterį darbui - įrašyti „Python“ programavimo

įrankius, paruošti „Anaconda“ komandinę eilutę, „NVIDIA CUDA“ įrankius, Keras ir TensorFlow. Naudojamas kompiuteris privalo turėti galingą procesorių ar grafinį procesoriaus bloką - kompiuteris, kuris buvo naudotas eksperimentui, turėjo Nvidia 1080 Ti.

Keras pateikia modelius, apmokytus su ImageNet, ir jų svorius, todėl reikėjo importuoti tinkamą modelį - šiuo atveju NASNet. Importavimo metu padaryti nustatymai - pilnai sujungtas sluoksnis nepridedamas, kadangi tada galima parinkti kokių dimensijų paveikslukai naudojami ir kiek spalvų sluoksnių jie turi (standartiniai paveikslukai turi 3). Tuomet reikėjo nustatyti kiek importuoto modelio sluoksnių bus mokoma norima - 5, 10 ir 20. Po šių egzistuojančio modelio paruošimų reikėjo sukurti naują Kero modelį, prie kurio reikėjo pridėti paruoštą egzistuojantį modelį bei pridėti kelis kitus sluoksnius tam tikru išsidėstymu - plokštinimo (angl. flatten), tankumo (angl. dense), išmetimo (angl. dropout) ir vėl tankumo. Pirmasis tankumo sluoksnis turi aktyvacijos funkciją ReLU, o paskutinis - sigmoidinę. Po visų sluoksnių pridėjimo negilus modelis iš viso turėjo 4 405 141 parametrų, o gilusis modelis turėjo 85 916 818.

Po modelio paruošimo, reikėjo nustatyti kokio dydžio paveikslėlių partijomis (angl. batch) bus mokomas ir validuojamas modelis. Geriausias partijos dydis naudotam kompiuteriui buvo mokymosi partijai 130 paveikslėlių, o validacijos - 10. Tuomet nustatomas paveikslėlių aplankalo kelias, jų ir partijos dydis bei nustatomas klasės režimas į binarinį, nes duomenų imtis susideda iš dviejų klasių - kačių ir šunų arba virtuvės ir gyvenamojo kambario.

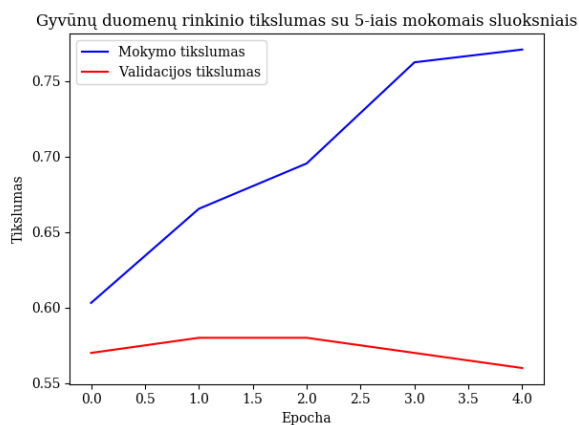
Po modelio ir paveikslėlių rinkinio paruošimo buvo pradėta mokinti ir validuoti modelį. Tačiau, modelio kompiliavimo metode nuostolio funkcija buvo nustatyta binarine kryžiaus entropija, o optimizavimo funkcija - Adagrad. Po modelio sukompiliavimo buvo paleidžiamas mokymo metodas, kuriam turi būti pateikta - paruošti paveikslukai, žingsnis per epochą, epochų kiekis, paruošti validacijos paveikslukai ir jų žingsnis. Epochų kiekis šiame eksperimente buvo nustatyta 5.

4.3. Modelių derinimas

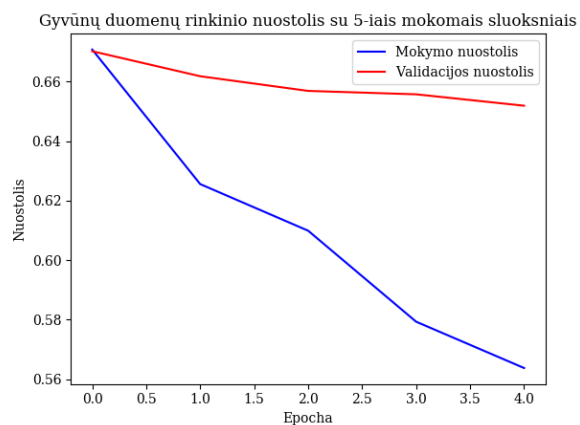
Buvo pradėta derinti ir validuoti modelius su skirtingais duomenų rinkiniais ir keičiant mokomų sluoksnių skaičių.

4.3.1. Negilaus modelio derinimas

Eksperimentas buvo pradėtas nuo negilaus modelio derinimo. Pirmiausia, mokymo sluoksnių skaičius buvo nustatytas 5-iems ir modelis buvo derinamas su kačių ir šunų duomenų rinkiniu. Gauti tikslumo ir nuostolio grafikai - 3 ir 4 pav.

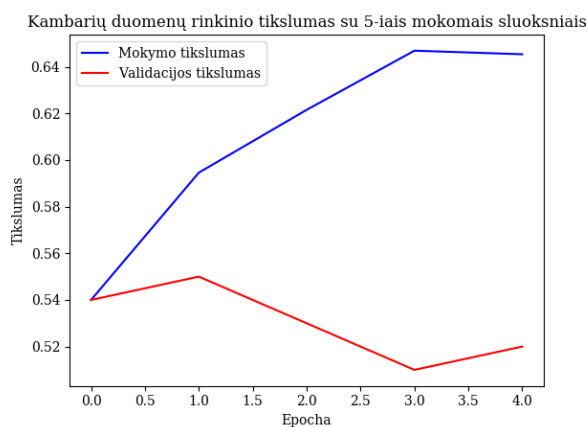


3 pav. Derinimo tikslumas

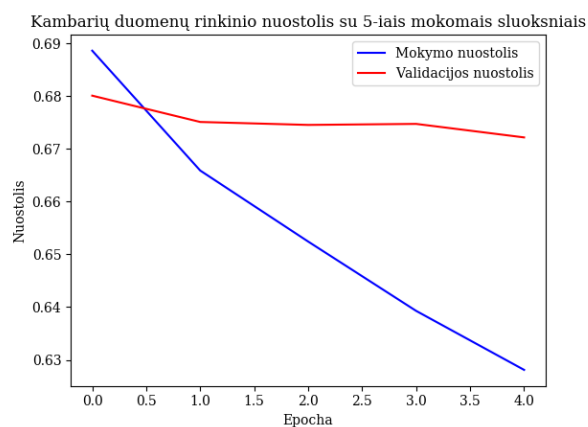


4 pav. Derinimo nuostolis

Nekeičiant mokomų sluoksnių skaičiaus, modelis iš naujo buvo derinamas su virtuvės ir gyvenamojo kambario duomenų rinkiniu. Gauti grafikai - 5 ir 6 pav.

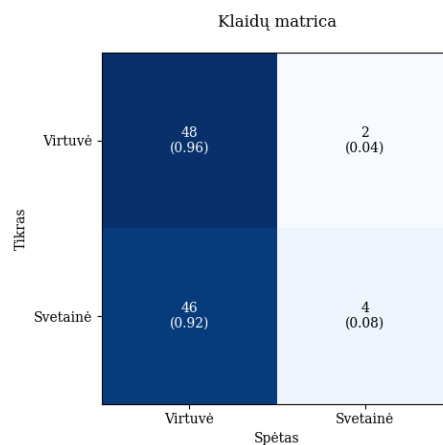
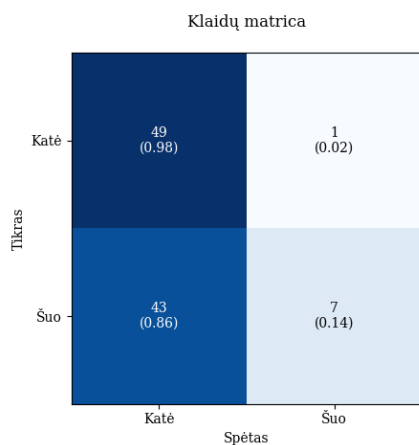


5 pav. Derinimo tikslumas



6 pav. Derinimo nuostolis

Paveikslėliuose (7 ir 8) pateiktos validacijos klaidų matricos skirtingų duomenų rinkinių.



7 pav. Klaidų matrica gauta su gyvūnų rinkiniu 8 pav. Klaidų matrica gauta su kambarių rinkiniu

4.3.2. Gilaus modelio derinimas

Eksperimentas buvo pratęstas naudojant gilų modelį. Pirmiausia, mokomų sluoksnių skaičius buvo nustatytas 5-iems ir modelis buvo derinamas su kačių ir šunų duomenų rinkiniu. Gauti tikslumo ir nuostolio grafikai - .

Su tokiu pačiu mokomų sluoksnių skaičiumi modelis buvo derinamas su virtuvės ir gyvenamojo kambario duomenų rinkiniu. Gauti grafikai - .

paveikslukuose parodytos validacijos klaidų matricos.

Rezultatai

Šio darbo metu buvo tirta dirbtinių neuroninių tinklų ir konvoliucinių neuroninių tinklų veikimas ir jų sudedamosios dalys bei surinkta informacija iš literatūros šaltinių. Atliktas eksperimentas, kurio tikslas buvo pamatyti kokį efektą turi konvoliucinio neuroninio tinklo gylis ir jo suderinimui naudojamas mažas duomenų rinkinys neuroninio tinklo metrikoms - tikslumui, atšaukimui, precizijai.

Darbo rezultatai:

1. Buvo atlikta bendra dirbtinių neuroninių tinklų ir konvoliucinių neuroninių tinklų bei NAS algoritmo analizė.
2. Buvo rasti geriausi hiperparametrai (optimizavimo funkcija, mokymosi greitis, išmetimo sluoksnio reikšmė) NASNetLarge ir NASNetMobile modeliams naudojant du skirtingus duomenų rinkinius bei keičiant mokomų sluoksnių skaičių.
3. Egzistuojantys NASNetLarge ir NASNetMobile modeliai buvo modifikuoti binariniai klasifikacijai ir suderinti naudojant du skirtingus duomenų rinkinius - kačių ir šunų bei virtuvės ir gyvenamojo kambario.
4. Gauti skirtingų gylių suderintų modelių mokymo ir validacijos tikslumo grafikai bei klaidų matricos.

Išvados

Literatūra

- [AT17] Karim Ahmed ir Lorenzo Torresani. Connectivity learning in multi-branch networks. *CoRR*, abs/1709.09582, 2017. arXiv: 1709.09582. URL: <http://arxiv.org/abs/1709.09582>.
- [ATY⁺18] Md. Zahangir Alom, Tarek M. Taha, Christopher Yakopcic, Stefan Westberg, Mahmudul Hasan, Brian C. Van Esesn, Abdul A. S. Awwal ir Vijayan K. Asari. The history began from alexnet: A comprehensive survey on deep learning approaches. *CoRR*, abs/1803.01164, 2018. arXiv: 1803.01164. URL: <http://arxiv.org/abs/1803.01164>.
- [BB12] James Bergstra ir Yoshua Bengio. Random search for hyper-parameter optimization. *J. Mach. Learn. Res.*, 13(1):281–305, 2012-02. ISSN: 1532-4435. URL: <http://dl.acm.org/citation.cfm?id=2503308.2188395>.
- [BX16] Su Chang Bian Ziyang Xu Tingfa ir Luo Xuan. Human abnormal behavior detection based on rgbd video’s skeleton information entropy. *Proceedings of the 2015 International Conference on Communications, Signal Processing, and Systems*, p. 715–723. Springer Berlin Heidelberg, 2016. ISBN: 978-3-662-49831-6.
- [CLS⁺15] Junghwan Cho, Kyewook Lee, Ellie Shin, Garry Choy ir Synho Do. Medical image deep learning with hospital PACS dataset. *CoRR*, abs/1511.06348, 2015. arXiv: 1511.06348. URL: <http://arxiv.org/abs/1511.06348>.
- [Dav15] Cameron Davidson-Pilon. *Bayesian Methods for Hackers*. Addison-Wesley Professional, 2015.
- [EMH18] Thomas Elsken, Jan Hendrik Metzen ir Frank Hutter. Neural architecture search: a survey. *arXiv preprint arXiv:1808.05377*, 2018.
- [Epp17] Sagi Eppel. Setting an attention region for convolutional neural networks using region selective features, for recognition of materials within glass vessels. *CoRR*, abs/1708.08711, 2017. arXiv: 1708.08711. URL: <http://arxiv.org/abs/1708.08711>.
- [FH18] Matthias Feurer ir Frank Hutter. Hyperparameter optimization. Frank Hutter, Lars Kotthoff ir Joaquin Vanschoren, redaktoriai, *AutoML: Methods, Sytems, Challenges*, skyr. 1, p. 3–37. Springer, 2018. To appear.

- [Fuk80] Kunihiro Fukushima. Neocognitron: a self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. 36:193–202, 1980-02.
- [Gau15] Ranjana Raut Gauri Borkhade. Application of neural network for diagnosing eye disease. *International Journal of Electronics, Communication and Soft Computing Science and Engineering (IJECSCE)*, 4:174–176, 2015. Published By International Journal of Electronics, Communication and Soft Computing Science and Engineering.
- [GBC16] Ian Goodfellow, Yoshua Bengio ir Aaron Courville. *Deep Learning*. MIT Press, 2016. <http://www.deeplearningbook.org>.
- [GVR⁺10] N. Ganesan, K. Venkatesh, M. A. Rama ir A. Malathi Palani. Article:application of neural networks in diagnosing cancer disease using demographic data. *International Journal of Computer Applications*, 1(26):76–85, 2010-02. Published By Foundation of Computer Science.
- [HKV19] Frank Hutter, Lars Kotthoff ir J. Vanschoren. *Automatic machine learning: methods, systems, challenges*. English. Challenges in Machine Learning. Springer, Germany, 2019. ISBN: 978-3-030-05317-8. DOI: 10.1007/978-3-030-05318-5.
- [HSS17] Jie Hu, Li Shen ir Gang Sun. Squeeze-and-excitation networks. *CoRR*, abs/1709.01507, 2017. arXiv: 1709.01507. URL: <http://arxiv.org/abs/1709.01507>.
- [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>.
- [YK18] Adam JATOWT Yihong ZHANG ir Yukiko KAWAI. Picture or words: predicting twitter image post popularity with deep learning, 2018. URL: <http://db-event.jpn.org/deim2018/data/papers/365.pdf>.
- [Kar16] Ujjwal Karn. An intuitive explanation of convolutional neural networks. <https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/>, 2016.
- [Leo98] C.T. Leondes. *Image Processing and Pattern Recognition*. Neural Network Systems Techniques and Applications. Elsevier Science, 1998, p. 323–324. ISBN: 9780080551449. URL: <https://books.google.lt/books?id=oDewAeVxr-4C>.

- [LGT18] C. Y. Lee, P. Gallagher ir Z. Tu. Generalizing pooling functions in cnns: mixed, gated, and tree. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(4):863–875, 2018-04. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2017.2703082.
- [LHB⁺99] Yann LeCun, Patrick Haffner, Léon Bottou ir Yoshua Bengio. Object recognition with gradient-based learning. *Shape, Contour and Grouping in Computer Vision*, p. 319–, London, UK, UK. Springer-Verlag, 1999. ISBN: 3-540-66722-9. URL: <http://dl.acm.org/citation.cfm?id=646469.691875>.
- [Li15] Fei-Fei Li. Convolutional neural networks. <https://cs231n.github.io/convolutional-networks/>, 2015.
- [LJB⁺95] Yann LeCun, LD Jackel, Leon Bottou, A Brunot ir k.t. Comparison of learning algorithms for handwritten digit recognition. *International conference on artificial neural networks*, tom. 60, p. 53–60. Perth, Australia, 1995.
- [LMH⁺19] Jason Zhi Liang, Elliot Meyerson, Babak Hodjat, Daniel Fink, Karl Mutch ir Risto Miikkulainen. Evolutionary neural automl for deep learning. *CoRR*, abs/1902.06827, 2019. arXiv: 1902.06827. URL: <http://arxiv.org/abs/1902.06827>.
- [Nik16] Mina Niknafs. Neural network optimization. 2016.
- [Pau01] Colm O’ Riordan Paul O’ Dea Josephine Griffith. Combining feature selection and neural networks for solving classification problems. *Intelligent Exploration of the Web*:389–401, 2001-07.
- [PG17] Josh Patterson ir Adam Gibson. *Deep Learning*. O’Reilly Media, Inc., 2017.
- [PWR⁺18] Zhanglin Peng, Lingyun Wu, Jiamin Ren, Ruimao Zhang ir Ping Luo. Cuimage: a neverending learning platform on a convolutional knowledge graph of billion web images. *2018 IEEE International Conference on Big Data (Big Data)*:1787–1796, 2018.
- [Ras15] Sebastian Raschka. *Python Machine Learning*. Packt Publishing, 2015, p. 342–343. ISBN: 1783555130, 9781783555130.
- [RMS⁺17] Esteban Real, Sherry Moore, Andrew Selle, Saurabh Saxena, Yutaka Leon Suematsu, Quoc V. Le ir Alex Kurakin. Large-scale evolution of image classifiers. *CoRR*, abs/1703.01041, 2017. arXiv: 1703.01041. URL: <http://arxiv.org/abs/1703.01041>.

- [RPA⁺18] Reza Reiazi, Reza Paydar, Ali Abbasian Ardakani ir Maryam Etedadialiabadi. Mam-mography lesion detection using faster r-cnn detector, 2018-01.
- [RS17] Thaqif Rajab ir Roselina Salleh. Classification of diabetes disease using backpropa-gation and radial basis function network. 2:3–4, 2017.
- [Rud16] Sebastian Ruder. An overview of gradient descent optimization algorithms. *CoRR*, abs/1609.04747, 2016. arXiv: 1609.04747. URL: <http://arxiv.org/abs/1609.04747>.
- [SB98] Richard S. Sutton ir Andrew G. Barto. *Introduction to Reinforcement Learning*. MIT Press, Cambridge, MA, USA, 1st leid., 1998. ISBN: 0262193981.
- [Shi12] D. Shiffman. *The Nature of Code*. D. Shiffman, 2012. Skyr. 10. ISBN: 9780985930806. URL: <https://books.google.lt/books?id=hoK6lgEACAAJ>.
- [SLJ⁺14] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke ir Andrew Rabinovich. Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014. arXiv: 1409.4842. URL: <http://arxiv.org/abs/1409.4842>.
- [SSS⁺17] Chen Sun, Abhinav Shrivastava, Saurabh Singh ir Abhinav Gupta. Revisiting unreason-able effectiveness of data in deep learning era. *CoRR*, abs/1707.02968, 2017. arXiv: 1707.02968. URL: <http://arxiv.org/abs/1707.02968>.
- [SSW⁺16] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams ir N. de Freitas. Taking the human out of the loop: a review of bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2016-01. ISSN: 0018-9219. DOI: 10.1109/JPROC.2015.2494218.
- [SZ15] Karen Simonyan ir Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2015.
- [TAL16] Sasha Targ, Diogo Almeida ir Kevin Lyman. Resnet in resnet: generalizing residual ar-chitectures. *CoRR*, abs/1603.08029, 2016. arXiv: 1603.08029. URL: <http://arxiv.org/abs/1603.08029>.
- [TL14] Ranzato Marc’Aurelio Taigman Yaniv Yang Ming ir Wolf Lior. Deepface: closing the gap to human-level performance in face verification. *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition*, p. 1701–1708. IEEE Com-puter Society, 2014. ISBN: 978-1-4799-5118-5. DOI: 10.1109/CVPR.2014.220. URL: <https://doi.org/10.1109/CVPR.2014.220>.

- [VK11] A Vehbi Olgac ir Bekir Karlik. Performance analysis of various activation functions in generalized mlp architectures of neural networks. 1:111–122, 2011-02.
- [vWB18] Gerard Jacques van Wyk ir Anna Sergeevna Bosman. Evolutionary neural architecture search for image restoration. *CoRR*, abs/1812.05866, 2018. arXiv: 1812.05866. URL: <http://arxiv.org/abs/1812.05866>.
- [Woo18] C. Woodford. Neural networks. <https://www.explainthatstuff.com/introduction-to-neural-networks.html>, 2018.
- [WZL⁺19] Y. Weng, T. Zhou, L. Liu ir C. Xia. Automatic convolutional neural architecture search for image classification under different scenes. *IEEE Access*, 7:38495–38506, 2019. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2019.2906369.
- [ZGD03] Qi-Jun Zhang, K. C. Gupta ir V. K. Devabhaktuni. Artificial neural networks for rf and microwave design - from theory to practice. *IEEE Transactions on Microwave Theory and Techniques*, 51(4):1339–1350, 2003-04. ISSN: 0018-9480. DOI: 10.1109/TMTT.2003.809179.
- [ZVS⁺17] Barret Zoph, Vijay Vasudevan, Jonathon Shlens ir Quoc V. Le. Learning transferable architectures for scalable image recognition. *CoRR*, abs/1707.07012, 2017. arXiv: 1707.07012. URL: <http://arxiv.org/abs/1707.07012>.

Sąvokų apibrėžimai ir santrumpos

AutoML - (angl. automated machine learning) automatizuotas mašininis mokymasis.

Hiperparametrai - (angl. hyperparameters) vidiniai neuroninio tinklo parametrai.

Derinimas - (angl. fine-tune) apmokyto modelio dalinis mokymas su kitu duomenų rinkiniu.

Pagreitis - (angl. momentum) dydis, kuris pirmiausiai leidžia greičiau judėti tam kad peršokti lokalius minimumus. Tačiau artėjant link konvergavimo šis dydis mažėja tam kad nebūtų praleistas globalus minimumas.

Priedas nr. 1

Tinklelio paieška

Hiperparametrų stulpelyje yra pateiktos išmetimo sluoksnių, optimizacijos funkcijos, mokymosi greičio reikšmės. Jų tyrinėjimas buvo atliktas su NASNetLarge ir NASNetMobile modeliais naudojant gyvūnų (kačių ir šunų) ir kambarių (virtuvės ir gyvenamojo kambario) duomenų rinkinius bei keičiant mokomų sluoksnių skaičių (5, 10 ir 20). Kiekviename rezultatų langelyje pateiktos tikslumo, atšaukimo ir precizijos reikšmės. Paryškintos reikšmės parodo geriausius rezultatus tame stulpelyje.

3 lentelė. Rezultatai

Hiperpara- metrai	NASNetLarge						NASNetMobile					
	Gyvūnai			Kambariai			Gyvūnai			Kambariai		
	5	10	20	5	10	20	5	10	20	5	10	20
0.25, RMSprop, 0.0005	0.96; 0.94; 0.98	0.89; 0.78; 1.0	0.96; 0.92; 1.0	0.89; 0.84; 0.93	0.88; 0.8; 0.95	0.86; 0.74; 0.97	0.67; 0.34; 1.0	0.69; 0.38; 1.0	0.58; 0.16; 1.0	0.5; 0.0; 0.0	0.53; 0.06; 1.0	0.54; 0.08; 1.0
0.25, RMSprop, 0.0001	0.92; 0.84; 1.0	0.95; 0.9; 1.0	0.98; 0.96; 1.0	0.88; 0.84; 0.91	0.91; 0.88; 0.94	0.88; 0.82; 0.93	0.6; 0.2; 1.0	0.7; 0.44; 0.92	0.61; 0.22; 1.0	0.51; 0.02; 1.0	0.58; 0.24; 0.75	0.58; 0.16; 1.0
0.25, RMSprop, 0.001	0.93; 0.86; 1.0	0.92; 0.84; 1.0	0.92; 0.84; 1.0	0.85; 0.72; 0.97	0.85; 0.86; 0.84	0.91; 0.9; 0.92	0.63; 0.26; 1.0	0.62; 0.24; 1.0	0.58; 0.16; 1.0	0.5; 0.0; 0.0	0.5; 0.0; 0.0	0.66; 0.32; 1.0
0.25, Adam, 0.0005	0.91; 0.82; 1.0	0.91; 0.82; 1.0	0.93; 0.86; 1.0	0.89; 0.82; 0.95	0.88; 0.8; 0.95	0.89; 0.82; 0.95	0.56; 0.12; 1.0	0.67; 0.34; 1.0	0.62; 0.24; 1.0	0.5; 0.0; 0.0	0.5; 0.0; 0.0	0.52; 0.04; 1.0
0.25, Adam, 0.0001	0.95; 0.9; 1.0	0.94; 0.88; 1.0	0.93; 0.86; 1.0	0.88; 0.8; 0.95	0.9; 0.86; 0.93	0.89; 0.84; 0.93	0.62; 0.24; 1.0	0.69; 0.4; 0.95	0.61; 0.22; 1.0	0.54; 0.16; 0.67	0.48; 0.02; 0.25	0.51; 0.02; 1.0
0.25, Adam, 0.001	0.93; 0.86; 1.0	0.92; 0.84; 1.0	0.94; 0.88; 1.0	0.89; 0.82; 0.95	0.88; 0.8; 0.95	0.88; 0.8; 0.95	0.63; 0.26; 1.0	0.65; 0.3; 1.0	0.63; 0.26; 1.0	0.5; 0.0; 0.0	0.5; 0.0; 0.0	0.66; 0.32; 1.0
0.25, Adagrad, 0.0005	0.94; 0.88; 1.0	0.93; 0.86; 1.0	0.95; 0.9; 1.0	0.86; 0.78; 0.93	0.89; 0.84; 0.93	0.9; 0.86; 0.93	0.6; 0.2; 1.0	0.62; 0.24; 1.0	0.63; 0.26; 1.0	0.52; 0.04; 1.0	0.63; 0.36; 0.78	0.49; 0.0; 0.0
0.25, Adagrad, 0.0001	0.98; 0.96; 1.0	0.96; 0.92; 1.0	0.96; 0.92; 1.0	0.85; 0.78; 0.91	0.87; 0.94; 0.82	0.9; 0.88; 0.92	0.56; 0.18; 0.75	0.55; 0.1; 1.0	0.61; 0.22; 1.0	0.54; 0.4; 0.56	0.63; 0.56; 0.65	0.59; 0.26; 0.76

3 lentelė. Rezultatai

Hiperpara- metrai	NASNetLarge						NASNetMobile					
	Gyvūnai			Kambariai			Gyvūnai			Kambariai		
	5	10	20	5	10	20	5	10	20	5	10	20
0.25, Adagrad, 0.001	0.93; 0.86; 1.0	0.92; 0.84; 1.0	0.96; 0.92; 1.0	0.88; 0.82; 0.93	0.85; 0.72; 0.97	0.88; 0.78; 0.97	0.67; 0.34; 1.0	0.65; 0.3; 1.0	0.67; 0.34; 1.0	0.5; 0.0; 0.0	0.52; 0.04; 1.0	0.52; 0.04; 1.0
0.5, RM-Sprop, 0.0005	0.9; 0.8; 1.0	0.97; 0.94; 1.0	0.89; 0.78; 1.0	0.9; 0.82; 0.98	0.88; 0.86; 0.9	0.88; 0.78; 0.97	0.65; 0.3; 1.0	0.59; 0.18; 1.0	0.66; 0.32; 1.0	0.5; 0.0; 0.0	0.51; 0.02; 1.0	0.58; 0.16; 1.0
0.5, RM-Sprop, 0.0001	0.95; 0.9; 1.0	0.98; 0.96; 1.0	0.95; 0.9; 1.0	0.89; 0.84; 0.93	0.9; 0.88; 0.92	0.88; 0.82; 0.93	0.58; 0.16; 1.0	0.59; 0.18; 1.0	0.72; 0.44; 1.0	0.55; 0.1; 1.0	0.54; 0.14; 0.7	0.52; 0.04; 1.0
0.5, RM-Sprop, 0.001	0.95; 0.9; 1.0	0.89; 0.78; 1.0	0.93; 0.86; 1.0	0.88; 0.8; 0.95	0.89; 0.9; 0.88	0.85; 0.72; 0.97	0.63; 0.26; 1.0	0.55; 0.1; 1.0	0.62; 0.24; 1.0	0.5; 0.0; 0.0	0.52; 0.04; 1.0	0.57; 0.14; 1.0
0.5, Adam, 0.0005	0.91; 0.82; 1.0	0.91; 0.82; 1.0	0.95; 0.9; 1.0	0.86; 0.74; 0.97	0.88; 0.8; 0.95	0.89; 0.8; 0.98	0.63; 0.26; 1.0	0.68; 0.4; 0.91	0.67; 0.34; 1.0	0.51; 0.02; 1.0	0.51; 0.02; 1.0	0.52; 0.04; 1.0
0.5, Adam, 0.0001	0.95; 0.9; 1.0	0.93; 0.86; 1.0	0.93; 0.86; 1.0	0.89; 0.84; 0.93	0.89; 0.82; 0.95	0.88; 0.82; 0.93	0.76; 0.54; 0.96	0.6; 0.2; 1.0	0.65; 0.3; 1.0	0.63; 0.3; 0.88	0.54; 0.16; 0.67	0.52; 0.04; 1.0
0.5, Adam, 0.001	0.88; 0.76; 1.0	0.93; 0.86; 1.0	0.95; 0.9; 1.0	0.86; 0.74; 0.97	0.86; 0.74; 0.97	0.88; 0.78; 0.97	0.55; 0.1; 1.0	0.65; 0.3; 1.0	0.61; 0.22; 1.0	0.5; 0.0; 0.0	0.51; 0.02; 1.0	0.63; 0.26; 1.0
0.5, Adagrad, 0.0005	0.93; 0.86; 1.0	0.94; 0.88; 1.0	0.95; 0.9; 1.0	0.88; 0.84; 0.91	0.89; 0.82; 0.95	0.87; 0.8; 0.93	0.59; 0.18; 1.0	0.77; 0.66; 0.85	0.71; 0.42; 1.0	0.64; 0.32; 0.89	0.51; 0.04; 0.67	0.51; 0.02; 1.0
0.5, Adagrad, 0.0001	0.95; 0.9; 1.0	0.93; 0.86; 1.0	0.94; 0.88; 1.0	0.8; 0.9; 0.75	0.88; 0.84; 0.91	0.88; 0.9; 0.87	0.59; 0.34; 0.68	0.66; 0.9; 0.61	0.57; 0.14; 1.0	0.45; 0.18; 0.39	0.51; 0.02; 1.0	0.48; 0.2; 0.45
0.5, Adagrad, 0.001	0.93; 0.86; 1.0	0.93; 0.86; 1.0	0.96; 0.92; 1.0	0.88; 0.84; 0.91	0.89; 0.82; 0.95	0.9; 0.82; 0.98	0.54; 0.08; 1.0	0.67; 0.34; 1.0	0.68; 0.36; 1.0	0.5; 0.0; 0.0	0.56; 0.14; 0.88	0.5; 0.0; 0.0
0.7, RM-Sprop, 0.0005	0.95; 0.9; 1.0	0.94; 0.88; 1.0	0.96; 0.92; 1.0	0.87; 0.76; 0.97	0.9; 0.88; 0.92	0.89; 0.84; 0.93	0.66; 0.32; 1.0	0.56; 0.12; 1.0	0.63; 0.26; 1.0	0.5; 0.0; 0.0	0.5; 0.0; 0.0	0.54; 0.08; 1.0

3 lentelė. Rezultatai

Hiperpara- metrai	NASNetLarge						NASNetMobile					
	Gyvūnai			Kambariai			Gyvūnai			Kambariai		
	5	10	20	5	10	20	5	10	20	5	10	20
0.7, RM-Sprop, 0.0001	0.95; 0.9; 1.0	0.94; 0.88; 1.0	0.98; 0.96; 1.0	0.9; 0.86; 0.93	0.89; 0.84; 0.93	0.89; 0.84; 0.93	0.63; 0.28; 0.93	0.68; 0.38; 0.95	0.7; 0.4; 1.0	0.5; 0.0; 0.0	0.55; 0.1; 1.0	0.51; 0.02; 1.0
0.7, RM-Sprop, 0.001	0.91; 0.82; 1.0	0.97; 0.94; 1.0	0.93; 0.86; 1.0	0.88; 0.78; 0.97	0.9; 0.86; 0.93	0.88; 0.78; 0.97	0.7; 0.4; 1.0	0.62; 0.24; 1.0	0.64; 0.28; 1.0	0.5; 0.0; 0.0	0.5; 0.0; 0.0	0.51; 0.02; 1.0
0.7, Adam, 0.0005	0.92; 0.84; 1.0	0.92; 0.84; 1.0	0.94; 0.88; 1.0	0.88; 0.78; 0.97	0.88; 0.82; 0.93	0.89; 0.82; 0.95	0.63; 0.26; 1.0	0.64; 0.28; 1.0	0.63; 0.26; 1.0	0.5; 0.0; 0.0	0.5; 0.0; 0.0	0.57; 0.14; 1.0
0.7, Adam, 0.0001	0.93; 0.86; 1.0	0.94; 0.88; 1.0	0.93; 0.86; 1.0	0.9; 0.86; 0.93	0.89; 0.88; 0.9	0.89; 0.84; 0.93	0.75; 0.58; 0.88	0.71; 0.62; 0.76	0.68; 0.36; 1.0	0.59; 0.26; 0.76	0.5; 0.02; 0.5	0.52; 0.04; 1.0
0.7, Adam, 0.001	0.94; 0.88; 1.0	0.94; 0.88; 1.0	0.92; 0.84; 1.0	0.88; 0.78; 0.97	0.87; 0.78; 0.95	0.87; 0.78; 0.95	0.66; 0.32; 1.0	0.64; 0.28; 1.0	0.61; 0.22; 1.0	0.5; 0.0; 0.0	0.56; 0.12; 1.0	0.58; 0.16; 1.0
0.7, Adagrad, 0.0005	0.96; 0.92; 1.0	0.94; 0.88; 1.0	0.97; 0.94; 1.0	0.9; 0.88; 0.92	0.88; 0.86; 0.9	0.89; 0.82; 0.95	0.73; 0.46; 1.0	0.68; 0.46; 0.82	0.72; 0.46; 0.96	0.52; 0.04; 1.0	0.52; 0.04; 1.0	0.51; 0.04; 0.67
0.7, Adagrad, 0.0001	0.96; 0.94; 0.98	0.92; 0.9; 0.94	0.95; 0.9; 1.0	0.87; 0.84; 0.89	0.84; 0.92; 0.79	0.87; 0.86; 0.88	0.61; 0.28; 0.82	0.55; 0.36; 0.58	0.65; 0.4; 0.8	0.58; 0.92; 0.55	0.58; 0.32; 0.67	0.7; 0.68; 0.71
0.7, Adagrad, 0.001	0.93; 0.86; 1.0	0.95; 0.9; 1.0	0.96; 0.92; 1.0	0.9; 0.86; 0.93	0.86; 0.76; 0.95	0.87; 0.78; 0.95	0.69; 0.38; 1.0	0.65; 0.3; 1.0	0.64; 0.28; 1.0	0.5; 0.0; 0.0	0.51; 0.02; 1.0	0.5; 0.0; 0.0