Kauno technologijos universitetas

Matematikos ir gamtos mokslų fakultetas

Sąsūkų neuroninių tinklų mokymo metodas, paremtas Bajeso optimizacija modelio parametrų paieškai ir mokymo duomenų filtravimu

Baigiamasis magistro studijų projektas

Ignas Dovydaitis

Projekto autorius

Doc. dr. Paulius Palevičius

Vadovas

Kaunas, 2024

Kauno technologijos universitetas

Matematikos ir gamtos mokslų fakultetas

Sąsūkų neuroninių tinklų mokymo metodas, paremtas Bajeso optimizacija modelio parametrų paieškai ir mokymo duomenų filtravimu

Baigiamasis magistro studijų projektas

Taikomoji matematika (6211AX006)

Ignas Dovydaitis

Projekto autorius

Doc. dr. Paulius Palevičius

Vadovas

Doc.dr. Loreta Saunorienė

Recenzentė

Kaunas, 2024

Kauno technologijos universitetas

Matematikos ir gamtos mokslų fakultetas

Ignas Dovydaitis

Sąsūkų neuroninių tinklų mokymo metodas, paremtas Bajeso optimizacija modelio parametrų paieškai ir mokymo duomenų filtravimu

Akademinio sąžiningumo deklaracija

Patvirtinu, kad:

1. baigiamąjį projektą parengiau savarankiškai ir sąžiningai, nepažeisdama(s) kitų asmenų autoriaus ar kitų teisių, laikydamasi(s) Lietuvos Respublikos autorių teisių ir gretutinių teisių įstatymo nuostatų, Kauno technologijos universiteto (toliau – Universitetas) intelektinės nuosavybės valdymo ir perdavimo nuostatų bei Universiteto akademinės etikos kodekse nustatytų etikos reikalavimų;

2. baigiamajame projekte visi pateikti duomenys ir tyrimų rezultatai yra teisingi ir gauti teisėtai, nei viena šio projekto dalis nėra plagijuota nuo jokių spausdintinių ar elektroninių šaltinių, visos baigiamojo projekto tekste pateiktos citatos ir nuorodos yra nurodytos literatūros sąraše;

3. įstatymų nenumatytų piniginių sumų už baigiamąjį projektą ar jo dalis niekam nesu mokėjęs (-usi);

4. suprantu, kad išaiškėjus nesąžiningumo ar kitų asmenų teisių pažeidimo faktui, man bus taikomos akademinės nuobaudos pagal Universitete galiojančią tvarką ir būsiu pašalinta(s) iš Universiteto, o baigiamasis projektas gali būti pateiktas Akademinės etikos ir procedūrų kontrolieriaus tarnybai nagrinėjant galimą akademinės etikos pažeidimą.

Ignas Dovydaitis

Patvirtinta elektroniniu būdu

Dovydaitis, Ignas. Sąsūkų neuroninių tinklų mokymo metodas, paremtas Bajeso optimizacija modelio parametrų paieškai ir mokymo duomenų filtravimu. Magistro studijų baigiamasis projektas / vadovas Doc. dr. Paulius Palevičius; Kauno technologijos universitetas, Matematikos ir gamtos mokslų fakultetas.

Studijų kryptis ir sritis (studijų krypčių grupė): Taikomoji matematika (Matematikos mokslai).

Reikšminiai žodžiai: Bajeso optimizacija, branduolio duomenų aibė, mašininis mokymas, sąsūkų neuroniniai tinklai, dirbtiniai neuroniniai tinklai, neuroninių architektūrų paieška, hiperparametrų optimizacija.

Kaunas, 2024. 40 p.

Santrauka

Dirbtiniai neuroniniai tinklai žinomi dėl gebėjimo aproksimuoti sudėtingas, netiesines ir sunkiai išreiškiamas funkcijas, ypatingai vaizdų atveju, kur naudojami sąsūkų neuroniniai tinklai (SNT), tačiau jie ypatingai jautrūs hiperparametrų pasirinkimui. Siekiant rasti geriausius parametrus kiekvienai užduočiai spręsti, mokoma aibė tinklų, kas reikalauja ekspertinių žinių ir daugybės kompiuterinio skaičiavimo resursų. Nors modernios neuroninių tinklų architektūros reikšmingai sumažino hiperparametrų kiekį, jie vis tiek turi būti randami bandymų ir klaidų būdu. Yra išrasta daugybė automatinių hiperparametrų paieškos algoritmų, o dažnas jų remiasi Bajeso optimizacijos (BO) principu. Nors tokie metodai ženkliai pagreitina ir palengvina geriausio modelio paiešką, jie vis tiek reikalauja bent keliolikos bandymų, kas yra brangu, ypatingai apmokant naudojant didelės apimties duomenų rinkinius. Šiuo atveju ne kiekvienas duomenų rinkinio elementas yra tiek pat vertingas kaip kitas, tad išfiltravus mažiausiai reikšmingus elementus iš mokymo duomenų rinkinio, galima pagreitinti hiperparametrų parinkimo procesą, taip sumažinant ir mokymo laiką ir kainą. Šiame darbe pristatytas ir analizuojamas SNT mokymo metodas apjungia BO hiperparametrų paiešką ir nereikšmingų duomenų elementų filtravimą siekiant minimizuoti geriausio modelio paieškos kainą be didelių papildomų skaičiavimo resursų. Duomenų filtravimas vykdomas pagal elementų pamiršimo įvykių kiekį, dinaminį egzempliorių sudėtingumo įvertį bei siūlomą naują įvertį - pirmo žvilgsnio sudėtingumą. Metodų efektyvumui parodyti, jie testuojami naudojant du viešai prieinamus duomenų rinkinius. Empirinis tyrimas parodė, kad tam tikrais atvejais treniravimo duomenų kiekį galima sumažinti iki 85% išlaikant modelio tikslumą, taip sumažinant mokymo kainą kelis kartus.

Dovydaitis Ignas. Convolutional Neural Network Teaching Method Based on Bayes Optimization for Model Parameter Search and Core Training Set Filtering. Master's Final Degree Project / supervisor Assoc. Prof. PhD. Paulius Palevičius; Faculty of Mathematics and Natural Sciences, Kaunas University of Technology.

Study field and area (study field group): Applied Mathematics (Mathematical Sciences).

Keywords: Bayesian optimization, neural architecture search, machine learning, convolutional neural networks, artificial neural networks, hyperparameter optimization

Kaunas, 2024. 40.

Summary

Artificial neural networks are known for their ability to approximate complex, nonlinear, and hard-to-express functions, especially in the case of images, where convolutional neural networks (CNN) are used, but they are particularly sensitive to the choice of hyperparameters. To find the best parameters for solving each task, a set of networks is trained, which requires expert knowledge and a lot of computing resources. Although modern neural network architectures have significantly reduced the amount of hyperparameters, they still must be found by trial and error. Many automatic hyperparameter search algorithms have been invented, and many of them are based on the principle of Bayesian optimization (BO). Although such methods significantly speed up and facilitate the search for the best model, they still require at least a dozen trials, which is expensive, especially when working with large datasets. In this case, not every element of the data set is as valuable as the other, so filtering out the least significant elements from the training data set can speed up the hyperparameter selection process, thus reducing both training time and cost. The SNT training method presented and analyzed in this paper combines the search for BO hyperparameters and the filtering of irrelevant data elements in order to minimize the cost of searching for the best model without large additional computing resources. Data filtering is performed according to the amount of element forgetting events, the dynamic estimate of the complexity of the instances and the proposed new estimate - the first look hardness. To demonstrate the effectiveness of the methods, they are tested using two publicly available datasets. Empirical research has shown that in certain cases, the amount of training data can be reduced by up to 85% while maintaining the accuracy of the model, thereby reducing the training cost several times.

Turinys

Lentelių sąrašas

Paveikslų sąrašas

Santrumpų ir terminų sąrašas

Santrumpos:

DNT – Dirbtiniai neuroniniai tinklai.

SNT – Sąsūkų neuroniniai tinklai.

BO – Bajeso optimizacija.

RNT – rekurentinis neuroninis tinklas.

EA – Evoliucinis algoritmas.

NAP – Neuroninės architektūros paieška.

ENAP – Evoliucinė neuroninės architektūros paieška.

KO – Kombinatorinis optimizavimas.

Įvadas

Dirbtiniai neuroniniai tinklai (DNT) per praeitus dešimtmečius padarė milžinišką pažangą kompiuterinės regos srityje [1]. Tačiau neuroninių tinklų architektūros atrinkimas ir treniravimas reikalauja ne tik milžiniškų laiko, kompiuterinių ir gamtos resursų, bet ir daugybės ekspertinių žinių. Tokios metodikos kaip neuroninių architektūrų paieška (NAP), hiperparametrų optimizavimo metodai ir branduolio aibės konstravimo metodai padeda sumažinti žmogiškąsias pastangas ir kompiuterinius kaštus, automatizuotu būdu iš duotos paieškos erdvės atrenkant geriausias architektūras, o jas treniruojant naudojant sumažintą duomenų rinkinį.

Neuroninių tinklų populiarumą lėmė jų gebėjimas automatiškai rasti požymius nestruktūrizuotuose duomenyse, tokiuose kaip paveiksliukai, ir iš jų daryti naudingas išvadas. Tačiau DNT efektyvumas tiesiogiai priklauso nuo jo architektūros ir naudojamų mokymo duomenų. Visų įmanomų architektūrų išbandymas duotai užduočiai dažnai yra neįmanomas, todėl tyrėjai dažniausiai remiasi savo nuovoka ir eksperimentuoja su keliomis ar keliolika jų, priklausomai nuo turimų laiko ir kompiuterinių resursų, iki kol randa optimalią, tačiau jie negali garantuoti kad tai geriausia architektūra iš visų įmanomų. Tuo metu mokymo duomenys yra kaupiami tikintis kad didesnėje duomenų aibėje DNT atras daugiau ir bendresnių bruožų, taip didinant jų efektyvumą, tačiau ir čia galima pastebėti problematiką – kartojant eksperimentą, kai visa DNT inicializacija yra identiška, tik treniravimo duomenis rikiuojant atsitiktine tvarka, keičiasi išmokyto modelio rezultatai. Abiem šioms problemoms spręsti yra sukurtos metodikos.

Neuroninės architektūros paieškos metodai siekia įvairiais būdais automatizuoti ir paspartinti geriausios neuroninės architektūros radimą tam tikrai užduočiai spręsti. Tai gan nauja paradigma dirbtinio intelekto (DI) pasaulyje, pradžioje buvo itin kritikuota dėl milžiniško resursų eikvojimo [2]. Tačiau naujausių tyrimų dėka yra atrasti greiti ir efektyvūs NAP metodai [3], leidžiantys net mažą skaitmeninį biudžetą turintiems naudotojams reikšmingai paspartinti šią paiešką.

DNT mokosi iš daugybės jam paruoštų pavyzdžių, randa juose slypinčius požymius. Bendru atveju yra sakoma, kad kuo daugiau pavyzdžių yra turima, tuo tikslesnį modelį galima išmokyti [4]. Tačiau tai nebūtinai tiesa. Lygiai kaip mokinys gavęs pakankamai sudėties ar atimties pavyzdžių, kad suprastų šios aritmetinės operacijos koncepciją, iš daugiau panašių pavyzdžių nieko naujo neišmoks, taip ir DNT iš nematyto pavyzdžio jokios naujos informacijos neišmoks, jei ji jau išmokta iš kitų pavyzdžių. Ši problema pastebėta kompiuterinėje geometrijoje dar prieš DNT erą ir buvo sukurtas jos sprendimo metodas – branduolio aibių konstravimas. Tokioje aibėje yra minimizuojamas duomenų kiekis siekiant išlaikyti galutinio su šia aibe atrasto sprendinio efektyvumą.

Šias koncepcijas apjungus yra gaunamas efektyvaus sąsūkų neuroninių tinklų (SNT) mokymo metodas. Panašios metodikos yra aprašytos pirmame skyriuje, realizuotas ir analizuojamas metodas detaliai aprašytas antrame skyriuje, rezultatai ir jų analizė pateikti trečiame skyriuje, o darbo gale pateiktos tyrimo išvados ir rekomenduojamos tolesnių tyrimų kryptys.

Darbo tikslas – sukurti naują neuroninių tinklų mokymo metodą paremtą hiperparametrų paieška ir mokymo duomenų filtravimu bei ištirti jo parametrus, efektyvumą ir greitaveiką.

Tyrimo objektas – sąsūkų neuroninių tinklų mokymo metodas.

Uždaviniai:

- Sukurti naują DNT mokymo metodiką paremtą HPS ir duomenų filtravimu.
- Ištirti atradimo funkcijos diversifikaciją ir intensifikaciją balansuojančio parametro įtaką optimizacijos efektyvumui.
- Palyginti siūlomų duomenų filtravimo metodų efektyvumus.
- Ištirti geriausių modelių kiekio parametro įtaką filtravimo metodų efektyvumui.
- Nustatyti greitaveikos pokytį taikant treniravimo duomenų filtravimo metodus.

1. Literatūros apžvalga

Šiame skyriuje yra apžvelgtos SNT architektūros, jų kūrimo ir naudojimo istorija, neuroninės paieškos metodai nuo pirmų iki naujų, pateikta Bajeso optimizacijos istorija bei ją tobulinantys metodai ir branduolio aibės aprašas, konstravimo metodai.

1.1. Sąsūkų neuroninių tinklų architektūros

Bendruoju atveju DNT aproksimacijos tikslumas priklauso nuo dviejų dedamųjų: modelio architektūros ir jai priskirtų svorių. Optimalūs svoriai yra gaunami mokymo iteracijų metu, kai skirtumas tarp modelio spėjimų ir tikrųjų duomenų yra gaunamas iš nuoseklios nuostolių funkcijos, šis skirtumas diferencijuojamas ir gradiento nusileidimo principu modelio svoriai yra atnaujinami siekiant minimizuoti nuostolių funkciją. Ar tai būtų tam tikras tikslumas, ar iteracijų kiekis, pasiekus baigimo sąlygą, modelio treniravimo procesas yra sustabdomas.

Tuo metu dominuojantis optimalios architektūros radimo metodas yra bandymų ir klaidų, kuomet tyrėjas išbando mažą aibę rankiniu būdu atrinktų architektūrų, jas lygindamas ir pasirenkant geriausią. Šis procesas yra ne tik ilgai trunkantis, bet ir reikalaujantis nuolatinio tyrėjo pastangų. Šiam procesui palengvinti yra naudojami neuroninės architektūros paieškos metodai, arba, bendru atveju, kai paieškos erdvėje yra ir tokie parametrai, kaip mokymosi greitis, optimizacijos metodai ar reguliarizacijos koeficientai, hiperparametrų optimizacijos algoritmai.

Tokie algoritmai tapo aktualūs tik nuo 2017 m. [5], kai vis didėjantys kompiuteriniai resursai leido architektūroms tapti gilesnėms, platesnėms, ir sudėtingesnėms. Kai 1950-taisiais buvo sukurtas perceptronas [6] optimizuojamų hiperparametrų beveik nebuvo, o ir mokymo metodas buvo kitoks. Modernus atgalinio sklidimo algoritmas buvo pristatytas tik 1969 m. [7], jis leido jungti perceptronus į sluoksnius, taip pradedant giliųjų DNT erą. Tiesa, dėl ribotų kompiuterinių resursų, gylis buvo tai pat ribotas, todėl jų realizacija nebuvo efektyvi tad šios srities tyrimai sulėtėjo ir dirbtinis intelektas patyrė pirmąją „žiemą“ [8]. Po šios „žiemos“ buvo sukurtas kitas moderniems tinklas būtinas elementas – sąsūkų neuroniniai tinklai [8]. Paraleliai buvo kuriamos ekspertinės sistemos, jos grįstos žmonių atrastomis taisyklėmis ir buvo pirmosios gebančios padėti tyrėjams jų tyrimuose [9]. Antroji DI „žiema“ tęsėsi nuo ankstyvųjų 1980-tų iki 1990-tų vidurio, kai ekspertinės sistemos pasiekė savo ribas ir jų panaudojamumas nebuvo plečiamas. Šią „žiemą“ užbaigė paradigmos pokytis – dėmesys buvo nukreiptas nuo rankomis kurtų požymių sistemų, prie statistika grįstų, tokių kaip paslėpti Markovo modeliai [10]. Taip pat buvo publikuota daugybė viešai prieinamų duomenų rinkinių, tokių kaip MNIST [11] ar TIMIT [12].

Vėliau, didėjantis kompiuterinių resursų prieinamumas leido rinkti vis didesnius duomenų rinkinius ir mokyti didesnius modelius. Didžioji dalis šių duomenų, tokių kaip vaizdo ar garso įrašai, buvo be žymų, tad kilo poreikis dideliems ir sudėtingiems sužymėtiems duomenų rinkiniams. Vienas pirmų tokių buvo 2009 m. sukurtas ImageNet [13], kuris susideda iš milijonų sužymėtų vaizdų. Šis duomenų rinkinys įgavo platų pripažinimą ir buvo pradėtas naudoti kaip etalonas, kuomet 2012 m. Aleksas Križevskis su kolegom pristatė AlexNet [14] – pirmą tiesiogiai su vaizdais apmokytą gilų sąsūkų neuroninį tinklą. Nuo šio tinklo prasidėjo iki šiol nesibaigusi DI „vasara“ ir tyrėjai sukūrė įvairiose srityse naudojamus mašininio mokymosi metodus [8], [15].

Konkrečiai SNT sritis buvo taip pat tobulinama. Po AlexNet buvo sukurtas VGGNet [16], kuris parodė, jog gilesni, mažesnius sąsūkų branduolius turintys tinklai yra pranašesni. Vėliau GoogleNet

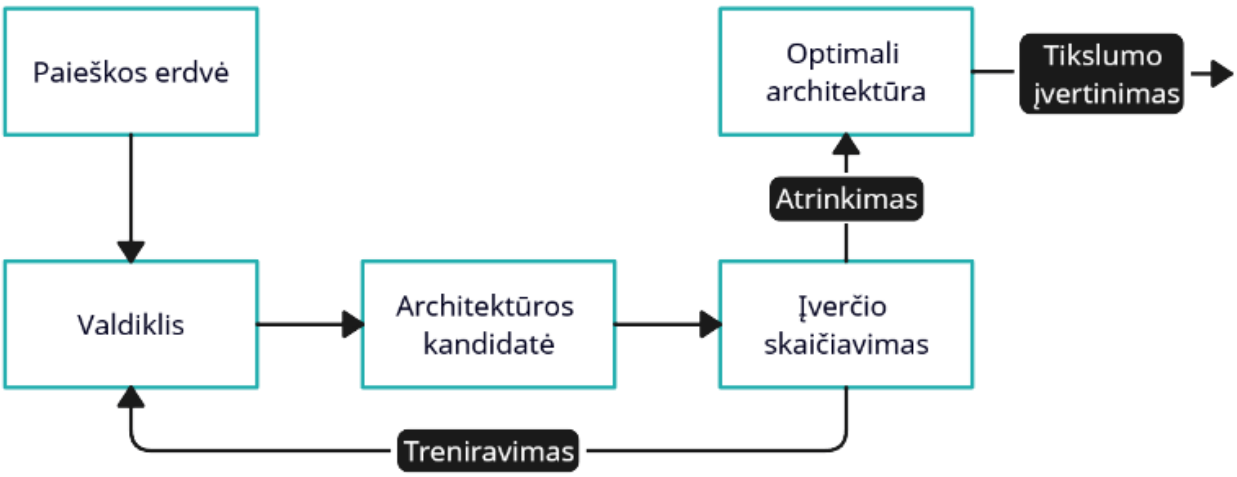
[17] pristatė „Inception“ sąsūkų bloką, kuris pasižymėjo skirtingu branduolio dydžiu ir paraleliomis jungtimis, taip dar padidinant aukščiausią pasiektą tikslumą ImageNet rinkiniui. Bendras šių tinklų bruožas buvo vis didėjantis nuosekliai jungiamų sluoksnių kiekis, tačiau šis kiekis turėjo limitą. Mokant gilius DNT susiduriama su dvejomis didelėmis problemomis – sprogstantys ir išnykstantys gradientai. Jei šių problemų tyrėjas neišsprendžia, modelis negali mokytis. Šios problema kyla kuomet dėl įvairių priežasčių atgalinio sklidimo algoritmu mokomų modelių gradientai yra dauginami su atitinkamai didesniu ar mažesniu nei 1 skaičiumi daugybę kartų, taip jį pakeliant iki begalybės arba sumažinant iki nulio. Nuosekliai sujungti sluoksniai būtent taip ir elgiasi. Tam spręsti ResNet [18] pristatė paralelias jungtis, kurios praleidžia kelis modelio sluoksnius iki kol yra sujungiamos su pagrindine informacijos tėkme, taip leisdamos pratekėti ir gradientams. DenseNet [19] šį principą maksimizavo jungdamas visas sluoksnių išvestis su paskutiniu sluoksniu. MobileNet [20] naudojosi jau sukurtais metodais parametrų skaičiui sumažinti iki prieinamo mobiliesiems telefonams.

Tyrėjams atradus, jog skirtingai sujungti skirtingi sluoksniai pasižymi skirtingais rezultatais, buvo eksperimentuojama su įvairiais jų junginiais. Tačiau žmonių vaizduotė yra limituota, o NAP sritis jau buvo pradėjusi vystytis, tad 2019 m. tyrėjai pristatė NAP metodu atrastą optimalią architektūrą tinkančią daugybei kompiuterinės vizijos užduočių – EfficientNet [21]. Apart unikalios architektūras, minėtame darbe taip pat įrodoma, jog norint padidinti modelio tikslumą užduočiai, jo rezoliucija, gylis ir plotis turi būti didinami kartu.

Visos minėtos architektūros susideda iš pasikartojančių blokų, kadangi juos lengva sukurti ir pagrįsti jų efektyvumą loginėmis operacijomis. Tuo metu tyrėjai atrado, jog pasikartojimų neturinčios architektūros gali būti pranašesnės konkrečioms užduotims spręsti [22]. Žmogui tokias architektūras kurti yra beveik neįmanoma, ar bent teoriškai pagrįsti jų efektyvumą yra ypatingai sudėtinga, tai empirinių tyrimų ir optimizacijos algoritmų rezultatas.

1.2. Neuroninės architektūros paieška

NAP yra siūlomas kaip dirbtinių neuroninių tinklų architektūrų paieškos erdvės paieškos metodas automatiniu būdu randantis konkrečiai užduočiai tinkamiausią DNT architektūrą minimizuojant paieškos resursus [23]. Bendras NAP veikimo principas iliustruotas **1 pav.**. NAP metodai pirmiausia iš paieškos erdvės atsitiktiniu būdu parenka aibę architektūrų, kurios bus apmokytos ir testuojamos naudojant validacijos duomenų aibę tikslumo įverčiui gauti. Tada, priklausomai nuo gautų įverčių, valdiklis parenka naują aibę architektūrų, kurios vėl apmokomos ir testuojamos. Šis ciklas tęsiamas iki kol pasiekiama baigimo sąlyga ir grąžinama geriausia išmokyta architektūra kartu su atitinkamu tikslumo įverčiu [23]. Tačiau šių metodų yra aibės [5], [23], [24] ir ne visi tiesiogiai seka **1 pav.** nurodytą struktūrą, dalis NAP metodų šį standartinį procesą modifikuoja siekiant paspartinti optimalios architektūros radimą.



1 pav. Bendras NAP veikimo principas.

Matematiškai NAP uždavinys gali būti formuluojamas kaip

$$\begin{cases} \arg\min_A \mathcal{L}\left(A, \mathcal{D}_{treniravimo}, \mathcal{D}_{validacijos}\right) \\ toks\ kad\ A \in \mathcal{A} \end{cases} \quad (1)$$

Kur $\mathcal{A}$ žymi paieškos erdvę, t.y. visos architektūrų variacijos, $\mathcal{L}(\cdot)$ žymi su $\mathcal{D}_{treniravimo}$ duomenų rinkiniu apmokytos architektūros A tikslumo įvertį $\mathcal{D}_{validacijos}$ duomenų rinkiniui. Daroma prielaida jog funkcija $\mathcal{L}(\cdot)$ yra neišgaubta ir nediferencijuojama [25]. Dėl to NAP yra sudėtingas optimizacijos uždavinys, kurį sprendžiant yra susiduriama su daugybe problemų, tokių kaip: diskretūs parametrai; modelių svoriai turi būti sveikieji skaičiai; sudėtingi ir ilgai trunkantys skaičiavimai; modelių apmokymas gradientų nusileidimo metodu naudojant dideles duomenų apimtis gali trukti paras, savaites ar mėnesius.

Dėl šių priežasčių nuo 2017 tyrėjai pristatė daugybę euristiniais, meta-euristiniais ir kitais metodais grįstus optimizacijos algoritmus [5], [23], [24]. Kuomet ankstyvieji metodai reikalavo šimtų ar tūkstančių individualiai apmokytų tinklų validacijos rezultatų [26], modernesni metodai kur kas spartesni, sumažina mokymų skaičių iki dešimčių, apmoko nepilnai, o dalis visai nemoko modelių ir remiasi kitais skaičiavimais [27], [28], [29].

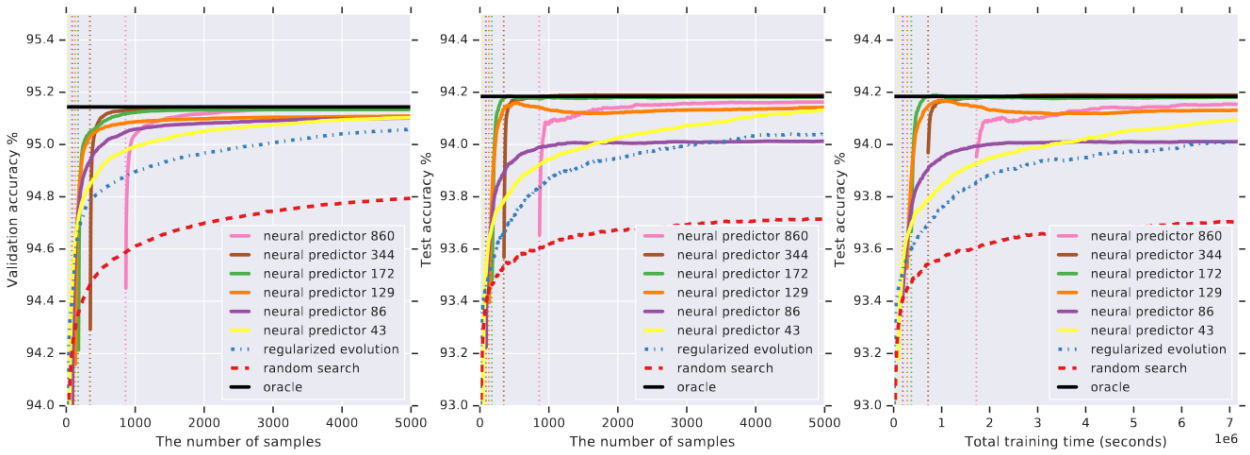
Žemiau pateikta keleto NAP srityje progresą padariusių straipsnių apžvalga.

1.2.1. Kelių mokymų NAP

Ankstyvosios NAP publikacijos iš esmės sekė **1 pav.** nurodytą struktūrą ir buvo intuityvūs [2], [26], [30]. Viena pirmųjų NAP publikacijų [26] kilo iš idėjos, kad DNT architektūrą galima užkoduoti kaip kintančio ilgio simbolių eilutę, kur kiekvienas simbolis nusako sluoksnio tipą ar jo hiperparametrus. Tokią eilutę gali spėti su pastiprinimu apmokytas rekurentinis neuroninis tinklas (RNT). Darbe šis atskiras RNT naudojamas kaip valdiklis, kuris optimizuoja paiešką tarp 6×10^{16} galimų architektūrų. Dalis kitų darbų šią problemą sprendė kaip kombinatorinio optimizavimo (KO) uždavinį [2], [31] ir pasitelkė klasikinį, dar 1948 Alano Turingo aprašytą KO metodą – evoliucinių algoritmą (EA) [32]. Šio algoritmo pagalba generuojamos architektūrų kandidačių populiacijos, jos apmokamos tikslumo įverčiui gauti ir pagal jį yra mutuojamos, kryžminamos ir atrenkamos. Tokie metodai bendrai vadinami evoliuciniai neuroninės architektūros paieškos (ENAP) metodai. Konkrečiau, [2]

darbas minimizuoja žmogaus įsikišimą į paieškos procesą, pateikdamas vos kelis paieškos hiperparametrus, o visą paieškos erdvę leidžia tyrinėti pačiam algoritmui. Paieška vykdoma aprašius galimas mutacijas ir kryžminimo būdus. Kitas ENAP metodas GeNet [31] pristatė DNT architektūros aprašymo būdą naudojant fiksuoto ilgio dvinarę simbolių eilutę, taip ją standartizuojant, o mutaciją aprašant kaip dvinario nario pakeitimą.

Užkodavus architektūrą kaip fiksuoto ilgio eilutę, ją galima apdoroti naudojant klasikinį regresinį neuroninį tinklą. Būtent taip padarė „Google Brains“ tyrėjai [30]. Jie taip pat inicializacijai naudoja atsitiktinę atranką, o apmokę N architektūrų, apmoko grafų sąsūkų tinklą vadinamą neuroniniu prognozuotoju (NP). NP įvestis yra topologiškai užkoduota modelio architektūra, todėl argumentuojama, jog grafų sąsūkų tinklas yra geriausias pasirinkimas tam, o išvestis yra modelio tikslumas validacijos duomenų imčiai. Apmokytam NP liepiama prognozuoti visų kitų įmanomų architektūrų tikslumų įverčius, tada atrenkamos K geriausios, kurios pilnai apmokomos ir iš jų atrenkama geriausia. Metodo efektyvumas įrodytas naudojant ImageNet [13] vaizdų duomenų rinkinį. Atlikę 600 eksperimentų palygino NP metodą su kitais kelių mokymų NAP metodais **2 pav.** Rezultatai rodo, jog NP geba aukštu užtikrintumu rasti optimalią modelio architektūrą kai N=119 ir K=137, kas yra dešimtis sykių mažiau nei kiti tuo metu siūlyti kelių mokymų metodai.



2 pav. N+K dydžių analizė NP metodui, lyginant su kitais kelių mokymų NAP metodais. Rodomas 600 eksperimentų vidurkis. Paimta iš [30].

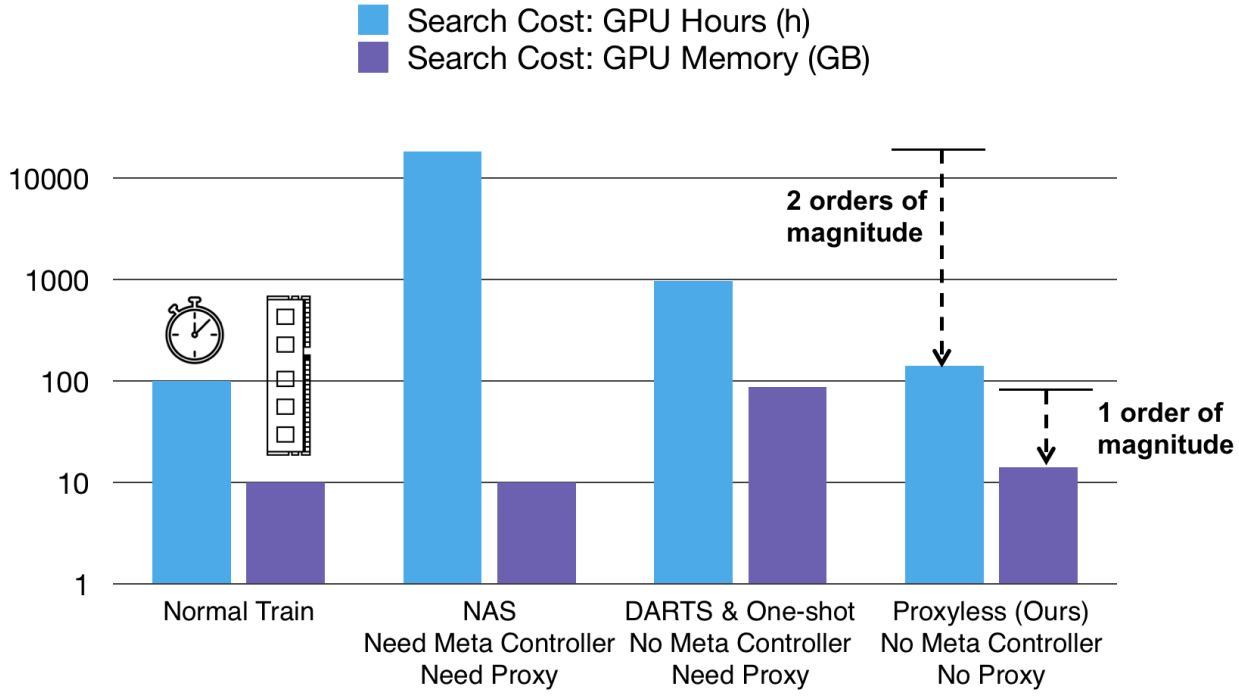
1.2.2. Nepilno mokymo NAP

Ankstyvieji darbai [2], [30] darė prielaidą, kad kiekvienas modelis yra nepriklausomas nuo kito ir mokė juos nuo atsitiktinės inicializacijos iki tikslumo nusistovėjimo ir tik tada vertino juos pagal validacijos duomenų imties tikslumo įvertį. Mokant naują modelį, visų prieš tai atliktų eksperimentų modelių svoriai yra ignoruojami. Kuomet šis procesas duoda maksimaliai tikslų architektūros įvertį [33], jis yra maksimaliai lėtas. Dalis tyrėjų matė tai kaip spragą ir pasiūlė svorių dalinimosi metodą kaip jos užpildymą.

Pirmasis darbas realizavęs ir įrodęs svorių dalinimosi efektyvumą buvo ENAS [27]. Šiame darbe paieškos erdvė yra analizuojama topologiškai ir siūloma į ją žvelgti kaip į didelį kryptingą aciklinį grafą. Toks grafas gali būti skirstomas į pografius, kurių dalis viršūnių, simbolizuojančios sluoksnius ir briaunų, kurios čia interpretuojamos kaip informacijos tėkmė ar sąryšis tarp sluoksnių, persidengia. Kuomet tarp pografių persidengia gretimos viršūnės ir briaunos, tokia atkarpa yra identiška kitai, tad ir svoriai tinka abejiems modeliams, tiek konteksto, tiek dydžio prasme. ENAS būtent ir siūlo

perpanaudoti tokias atkarpas treniravimo greičiui padidinti. Darbe yra argumentuojama, kad toks metodas, nors ir sumažina galutinį įverčio tikslumą, palyginimas tarp tokiu pat būdu mokytų architektūrų yra pilnai galimas, o optimalios architektūros paieškos laikas gali sumažėti net iki 1000 kartų, lyginant su prieš tai atliktais darbais [26], išlaikant ar net pagerinant galutinį tikslumą.

Svorių dalinimosi idėją naudojo ir MIT tyrėjai ProxylessNAS darbe [28]. Čia vienu metu mokomi visi įmanomi tinklo variantai, pasitelkiant super-tinklą su diferencijuojamais dvejetainiais keliais. NAP pagreitinimas naudojant super-tinklus jau buvo pristatytas prieš šį darbą [34], šis metodas reikalavo kur kas daugiau vaizdinės atminties kompiuterinio resurso. Būtent dėl diferencijuojamų dvejetainių kelių algoritmo atminties resurso reikalavimas buvo sumažintas beveik iki paprasto tinklo apmokymo reikalavimo, kaip parodyta **3 pav.** Tokios paieškos pasėkoje gaunama daugybės tinklų rezultatų, iš kurių paprasta atrinkti geriausią.



3 pav. ProxylessNAS [28] laiko ir vaizdinės atminties resursų naudojimas lyginant su kitais NAP metodais. Paimta iš [28].

Kita nepilno mokymo NAP metodika siekia ieškoti optimalios architektūros panašiai, tačiau paprastesnei problemai spręsti. [35] darbe pristatytas NAP metodas ieškojo geriausios generalizuotos architektūros sprendžiant CIFAR-10 [36] užduotį, tada šią architektūrą padidino ir apmokė naudojant kur kas didesnį ir sudėtingesnį ImageNet [14] vaizdų imtį klasifikacijai, taip pagerinant tikslumo rekordą. Prieš tai minėtas ProxylessNAS straipsnis argumentavo prieš šio metodo naudojimą.

Viena iš paprastesnių NAP pagreitinimo strategijų yra modelius mokyti ne iki tikslumo nusistovėjimo, o trumpiau, tik tiek kad būtų galima spėti apie modelio tolimesnį mokymą. [37] tyrėjai naudojo tikimybių teorija grįstą metodą, kuris spėja kokia bus tolimesnė DNT mokymosi kreivė iš kelių pirmų. Jei spėjama jog modelis išsimokys prasčiau nei nustatyta riba, tokio modelio mokymas yra sustabdomas ir mokomas kitas. [38] darbas pratęsė šią idėją, tačiau naudojo Bajeso neuroninį tinklą spėjimams gauti.

1.3. Bajeso Optimizacija

Bajeso optimizacija yra nuosekli hipotezių tikrinimo strategija skirta globaliam juodosios dėžės funkcijų optimizavimui. Dažniausiai pasitelkiama kuomet juodosios dėžės funkcija reikalauja daug kompiuterinių resursų [39]. Juodosios dėžės funkcija čia yra funkcija, kuriai daromos tam tikros prielaidos: ši funkcija neturi griežtai aprašytų konstantų, kintamųjų bei operacijų sujungtų aritmetinėmis operacijomis; neįmanoma ar brangu apskaičiuoti jos išvestines; turi daug lokalių maksimumų bei minimumų; funkcija yra neišgaubta. BO tikslas yra ištyrinėti duotą paieškos erdvę, kaip įmanoma greičiau randant norimą ekstremumą.

Pirmasis BO pristatė Kušneris [40], tačiau šiuo metu naudojamą jos formą sukūrė ir aprašė Jonas Mockus savo straipsniuose aštuntame ir devintame dešimtmetyje [41], [42], [43]. Darbuose pristatytas ne tik veikimo principas, bet ir kelių tikslų, kelių tikslumų optimizavimo metodus, tyrinėjo panaudojimo būdus bei konvergavimo greičius. Nuo to laiko šis metodas buvo pritaikytas daugybėje sričių: robotikoje [44], informacijos išgavime [45], jutiklių tinkluose [46] ir daugybėje kitų, o nuo 2017 m. modelio treniravimui aktualių hiperparametrų paieškai [47], [48], [49].

Mašininio mokymosi ir DNT kontekste BO optimizuoja ne tik architektūrą, bet ir susijusius parametrus, tokius kaip mokymosi greitis, optimizatoriaus tipas ar reguliarizacijos stiprumas [50]. Dėl to paieškos erdvė gali tapti dar didesnė nei NAP metodų, tad ir paieška gali tapti lėtesnė. Tam spręsti tyrėjai sukūrė BO pagreitinimo ir patobulinimo strategijų [47], [50]. Šiame poskyryje yra aptariamos keletas jų.

1.3.1. Diversifikacijos strategijos

Diversifikacija yra nepamainoma strategija siekiant patobulinti optimizacijos algoritmus [50], [51], [52], ji reikšmingai padidina tokių metodų efektyvumą. BO kontekste viena dažniausiai naudojamų diversifikacijos strategijų utilizuoja modelių ansamblį, kuomet pasitelkiama modelių aibė užuot pasitikėjus vienu jų [53]. Tokie modeliai gali būti homogeniniai, kur visi modeliai mokyti vienodu stochastiniu būdu, kiti yra heterogeniniai, kur modeliai turi skirtingus parametrus. Tokie metodai išvengia rizikos susijusios su vieno modelio buvimo neefektyviu. Vienas tokių yra S-Div [50], kuris kiekvienos iteracijos metu naudoja seniausiai naudotą modelį, taip rotuodamas jų selekciją.

Adaptyvios diversifikacijos metodai pratęsia S-Div metodą dinamiškai koreguojant modelių selekciją optimizavimo procese. Vienas tokių yra apribotos rizikos metodas [50]. Šis metodas ties kiekviena iteracija pasirenka modelį su geriausiu per visas prieš tai buvusias iteracijas akumuliuotu efektyvumo įverčiu. Šio metodo tikslas yra naudoti istoriškai geriausius modelius. Kitas yra ϵ-godus diversifikacijos metodas [50]. Priešingai nei deterministiniai S-Div ir apribotos rizikos metodai, šis modeliams priskiria tikimybes būti panaudotiems. Geriausiam priskirta 1- ϵ, o blogiausiam ϵ, taip užtikrindamas kad optimizacijos pradžioje bus naudojama aibė skirtingų modelių, o iteracijų eigoje ϵ vertė yra tiesiškai mažinama, taip didinant intensifikaciją.

1.3.2. Išankstinio nutraukimo strategijos

Ši strategija iš esmės panaši į jau prieš tai aprašytą NAP nepilno mokymo strategiją [37], [38], kur DNT modelių mokymas yra nutraukiamas prieš pasiekiant tikslumo įverčio nusistovėjimą ar maksimalų mokymo iteracijų skaičių. Šie metodai grįsti euristinėmis taisyklėmis [54], tikimybiniu

modeliavimu [37] ar regresija [55]. Kaip įvestį jie gauna ankstyvus modelio efektyvumo įverčius ir pagal juos sprendžia kokia bus tolimesnė modelio mokymosi eiga.

Tačiau tyrimai rodo [50], jog ši strategija yra efektyvi tik priėmus tam tikras prielaidas apie modelio mokymosi eigą. Viena tokių yra kad modelio mokymosi kreivė pradžioje yra stati, o vėliau suplokštėja. Tuo metu tokie metodai kaip atsitiktinio praretinimo transformacija [56] ar rinkinio normalizacija [57] minėtą kreivę gali nenuspėjamai pakeisti, tad naudojant šiuos efektyvius ir patikrintus reguliarizavimo metodus, didėja šansas nutraukti efektyvaus modelio mokymą. Dėl šios priežasties modernūs darbai [47], [50] vengia šios strategijos arba naudoja konservatyvią, pavyzdžiui pritaikant išankstinį nutraukimą individualiems modeliams kai jų tikslumo įvertis nepakilo tam tikrą iteracijų skaičių [58].

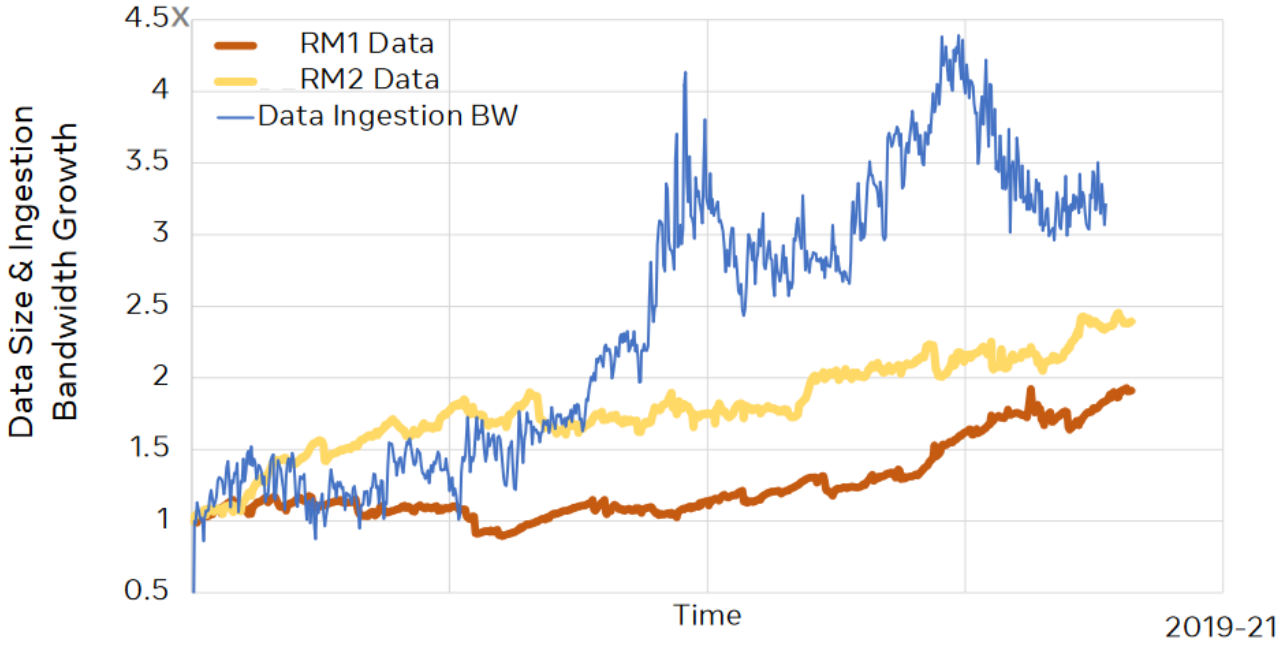
1.3.3. Lygiagretinimas

Iki šiol aprašyti metodai veikia įvertinant vieną modelį po kito, tačiau turint daugiau kompiuterinių resursų juos galima išnaudoti mokant kelias architektūras vienu metu, taip lygiagretinant šį procesą. Tai būtina daryti asinchroniniu būdu, kadangi skirtingų modelių treniravimas gali trukti skirtingą kiekį laiko. Tačiau neturint eksperimento rezultatų kyla rizika parinkti vienodus ar panašius į jau treniravimo eilėje esančius parametrus. [59] tyrimas sprendžia šią problema lygindamas naujus parametrų kandidatus su esančiais eilėje ir pasirenkant nepanašius. Kitas tyrimas [50] įvertina visų eilėje esančių besimokančių modelių negalutinius rezultatus prieš generuojant naują hiperparametrų aibę naujam eksperimentui, o modeliams išsimokius pilnai, BO metodas yra atnaujinamas su galutiniais rezultatais.

1.4. Duomenų filtravimas

Kompiuteriniams resursams tampant vis labiau prieinamiems, duomenų kaupimas tapo vis pigesnis, tad ir jų kiekiai didėjo. Ilgą laiką buvo teigiama kad mašininio mokymosi algoritmams dideli duomenų kiekiai yra būtini ir buvo rekomenduojama naudoti vis didesnius duomenų rinkinius [60]. **4 pav.** iliustruoja Meta įmonės rekomenduojamų duomenų imčių dydžių didėjimą jų rekomendacinėms sistemoms per 2 metų laikotarpį, ko pasėkoje ši įmonė savo mašininio mokymosi ekspertams rekomendavo naudoti duomenų imtis, kurios užima daugiau nei eksabaitą kietojo disko talpos [4]. Tokie duomenų kiekiai yra ne tik brangūs finansiškai, jie reikalauja masyvių duomenų centrų, kurių kiekvienas vartoja milžiniškus kiekius elektros energijos, tad ir gamtos išteklių.

Negana to, DNT algoritmai yra mokomi stochastinio gradientinio nusileidimo metodu, kas reiškia jog kiekvienoje mokymo iteracijoje modeliui leidžiama mokytis tik iš duomenų imties poaibio. Tuo metu jau 1990 m. buvo pastebėta kad DNT mokydamiesi naują informaciją, netikėtai ir staigiai užmiršta seną [61]. Šis fenomenas yra vadinamas katastrofišku užmiršimu. Tad kuo didesnė treniravimo duomenų imtis, tuo didesnis šansas kad prieš tai išmokti požymiai bus užmiršti. Taip atsitinka dėl to, kad vienos mokymo iteracijos metu DNT svoriai yra optimizuojami užduoties A sprendimui (teisingam duoto poaibio spėjimui), kitoje iteracijoje to paties modelio svoriai yra optimizuojami užduočiai B (teisingam kito poaibio spėjimui), tad požymiai išmokti iš užduoties A gali būti pakeičiami naujais, skirtais užduočiai B spręsti [62].



4 pav. Santykinės treniravimo duomenų imties dydžio rekomendacijos Meta įmonėje tarp 2019 m. ir 2021 m. Paimta iš [60].

1.4.1. Branduolio duomenų aibės

Vienas iš įrankių šiai problemai spręsti yra treniravimo duomenų imties mažinimas. [63] darbas tiria branduolio duomenų aibes. Tai sumažintos duomenų aibės, kurias naudojant sprendžiant konkrečią problemą gaunamas toks pat rezultatas kaip sprendžiant su pilna užduočiai skirta duomenų aibe. Šis metodas reikalauja kiekvieno duomenų aibėje esančio elemento svarbos įvertinimo, o įvertinti kaip nesvarbūs ar mažiausiai svarbūs yra išfiltruojami, taip sumažinant duomenų imties dydį nepakenkiant galutinio metodo efektyvumui, kuomet metodo kompiuterinių resursų naudojimas yra sumažinamas.

Branduolio duomenų aibės buvo naudojamos ir prieš DNT erą, tokiose srityse kaip kompiuterinė geometrija [64], K-vidurkių grupavimas [65] ir kitose didelių duomenų imčių reikalaujančiose problemose [66]. Bendru atveju branduolio duomenų aibės konstravimo sąlyga yra aprašoma taip:

$$|cost(X,Q) - cost(C,Q)| \leq \varepsilon * cost(X,Q). \quad (2)$$

Kur X yra pilna duomenų aibe, C – branduolio duomenų aibe, Q – nuostolių funkciją minimizuojantys parametrai, cost – nuostolių funkcija, $\varepsilon > 0$. Ši sąlyga garantuoja minimalią branduolio duomenų aibės įtaką galutiniam rezultatui lyginant su pilnos duomenų aibės naudojimu.

Tačiau tokios aibės sukūrimas nėra savaime suprantamas ir ją kuriant būtina atsižvelgti į jau minėtą katastrofiško užmiršimo fenomeną [62]. Šiame poskyryje apibendrintas keleto branduolio duomenų aibių konstravimo metodų.

1.4.2. Branduolio duomenų aibės konstravimo metodai

Dalis darbų [67], [68] daro prielaidą jog panašūs duomenų elementai, pavyzdžiui arti esančios koordinatės, pasižymi panašiais požymiais, todėl galima išfiltruoti dalį topologiškai panašių duomenų. Vienas tokių yra ganymo metodas [67]. Šis metodas iteratyviai atrenka duomenų elementus iš pilnos aibės siekiant kuo greičiau sumažinti skirstinių skirtumą tarp branduolio ir pilnos duomenų

aibės. Tai daroma įvertinant kiekvieno pilnos aibės elementų atstumą nuo jau sukonstruotos aibės ir pridedant didžiausią atstumą turinčius. Šis metodas minėtą skirtumą mažina greitai ir efektyviai [67].

Neužtikrintumu grįsti metodai teigia, jog duomenų imties elementai su mažesniu užtikrintumu daro didesnę įtaką modelio optimizacijai nei didesnį užtikrintumą turintys. [69] darbas naudoja mažiausią užtikrintumą, entropiją bei maržą kaip neužtikrintumo rodiklius klasifikacijos užduoties duomenų filtravimui. Šie rodikliai skaičiuojami apmokius pradinį modelį ir tada leidžiant jam spėti visų treniravimo imtyje esančių elementų klasių tikimybes. Mažiausias užtikrintumas skaičiuojamas įvertinant mažiausią užtikrintumą turinčios spėtos klasės užtikrintumą, entropijai apskaičiuoti naudojama kryžminės entropijos formulė [70], marža įvertinama pagal mažiausią užtikrintumą skirtą neteisingai klasei.

Paklaida grįsti metodai duomenų imties elementų svarbumą sprendžia pagal nuostolių funkciją naudojamą treniruojamo DNT. Jie daro prielaidą kad elementai gaunantys didesnę nuostolių funkcijos vertę turi didesnį svorį, tad ir svarbumą, tuo metu elementai su maža paklaida gali būti išfiltruoti branduolio duomenų imčiai gauti. Vienas tokių filtravimo būdų yra tiriamas [71] darbe. Šiame skaičiuojami pamiršimo įvykiai. Pamiršimo įvykių skaičius elementui nusako kiek kartų elementas buvo pamirštas DNT mokymo procese. Minėtame darbe pamiršimu yra laikomas toks įvykis kai vieną kartą gerai klasifikuotas elementas mokymo procese buvo klasifikuojamas neteisingai vėlesnėse mokymo iteracijose.

Kitas paklaida grįstas metodas GraNd [72] įvertina kiekvieno duomenų imties elemento įnašą modelio mokymuisi. Tai daroma skaičiuojant modelį treniruojant kelis kartus iki ankstyvos epochos t ir saugant nuostolių funkcijos pokytį priklausomai nuo iteracijoje buvusių treniravimo duomenų imties elementų. Šie išsaugoti pokyčiai yra vidurkinami, taip gaunant kiekvieno elemento svarbos įvertį. Visi mažesnį įvertį nei nustatyta riba gavę elementai yra išfiltruojami, taip sudarant branduolio duomenų aibę. Šio metodo pranašumas yra tas, kad nėra būtina modelį mokyti iki nuostolių vertės nusistovėjimo, o tik iki ankstyvos epochos t.

2. Duomenys ir tyrimo metodai

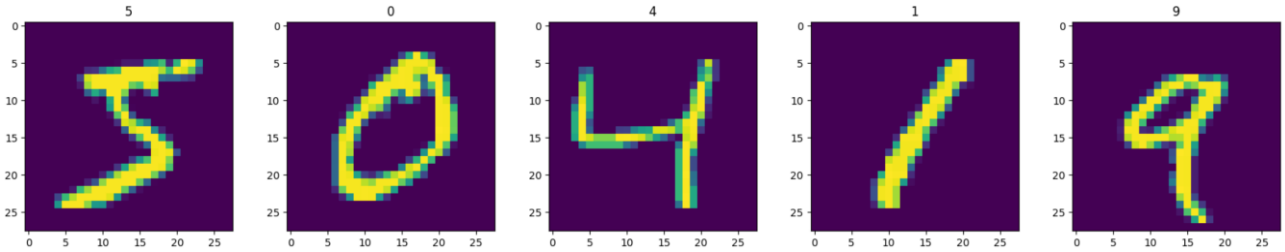
Tyrimui yra reikalingi bent keli klasifikacijai skirtų vaizdinių duomenų rinkiniai metodų veiklai palyginti. Šiame skyriuje jie pateikti ir aprašyti. Šiame skyriuje taip pat aprašytas visas realizuotas hiperparametrų paieškos metodas grįstas Bajeso optimizacija, modelių mokymo algoritmas bei du darbe naudoti duomenų filtravimo metodai.

2.1. Duomenų rinkiniai

Didžioji dalis [24], [73] NAP tyrimų yra atliekami naudojant jau ištreniruotų modelių ir jų tikslumų porų duomenų rinkinius, tokius kaip NAS-Bench-101 [22], ar panašius [74], [75]. Tačiau šiam darbui jie netinka, nes čia aktualus modelių treniravimas su sumažintu duomenų rinkiniu, todėl naudojami bus originalūs vaizdų duomenų rinkiniai. Naudojami duomenų rinkiniai taip pat turi būti naudojami kitose panašiose publikacijose, jų rezoliucija turi būti maža greitiems eksperimentams atlikti, o vaizdų skaičius tenkantis vienai klasei turi siekti bent 1000 [63] efektyviam branduolio aibės filtravimui. Taip pat, rezultatams lyginti yra būtina atkartoti paieškos erdvę. Dėl šių priežasčių tyrimas bus atliktas su [50] publikacijoje nurodytais MNIST [11], CIFAR-10 [36] duomenų rinkiniais ir atitinkamomis paieškos erdvėmis, kurios remiasi LeNet [11] architektūra.

MNIST [11] vaizdų rinkinys buvo sukurti dar 1994 metais JAV nacionalinio standartų ir technologijos instituto tuo metu egzistuojančių gradientiniu nusileidimu grįstų metodų analizei ir buvo publikuotas plačiam naudojimui. Nuo to laiko šį duomenų rinkinį akademinė ir kompiuterinės vizijos tyrėjų bendruomenė naudojo kaip etaloną naujų metodų lyginimui. Kelios to priežastys yra šio rinkinio prieinamumas, dydis ir tiesioginis užkodavimas, leidžiantis MNIST naudoti bet kokioje platformoje naudojant bet kokią programavimo kalbą [76]. MNIST susideda iš 10 klasių, kurie atspindi arabiškuosius skaičius, kiekvienai jų skirta apie 7000 paveikslėlių, o iš viso jų yra 70,000 vienetų, kurių 10,000 skirti testavimui. Šio darbo reikmėms iš treniravimo duomenų imties yra atrinkti 5000 paveikslėlių validacijai, tad treniravimo rinkinys susideda iš 55,000 vaizdų. Kiekvienas vaizdas turi tik vieną spalvos dimensija, kas reiškia, kad jie pilkų atspalvių, o jų rezoliucija yra 28x28, kaip matoma **5 pav.**

Kaip ir [50] publikacijoje, šiam duomenų rinkiniui pritaikytos dvi skirtingos hiperparametrų erdvės, jos detaliai aprašytos **2 lentelė** ir **3 lentelė**.



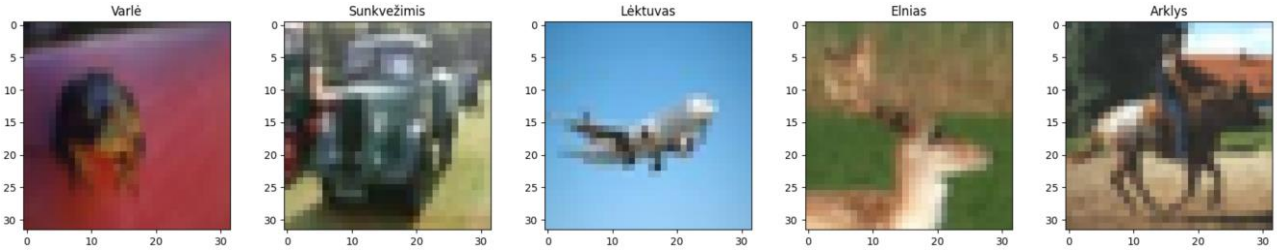
5 pav. MNIST duomenų rinkinio pavyzdžiai.

Kitas vaizdinių duomenų rinkinys tinkamas filtravimo metodams lyginti ir efektyvumui analizuoti yra CIFAR-10 [36]. Šis pristatytas 2009 metais ir tapo etaloninis kompiuterinės vizijos srityje [77]. Apart prieinamumo ir mažo rinkinio dydžio, yra įrodyta kad CIFAR-10 rinkiniui optimalios architektūros gali būti naudojamos ir didesniems bei sudėtingesniems vaizdų rinkiniams, tokiems kaip ImageNet [13], kuomet yra padidinamos [35]. Rinkinys susideda iš 60,000 spalvotų vaizdų,

kurių rezoliucija yra 32x32, o po 10,000 jų yra skirti testavimui, keli jų pavaizduoti **6 pav**. Kaip ir su MNIST rinkiniu, po 5000 vaizdų yra atimta iš treniravimo rinkinio ir priskirti validacijai. CIFAR-10 rinkinys susideda iš 10 klasių, kurių kiekvienai priskirti po 6000 paveikslėlių. Testavimo rinkinyje yra proporcingai mažiau paveikslėlių nei treniravimo. Šiam duomenų rinkiniui yra naudojama **4 lentelė** nurodyta hiperparametrų paieškos erdvė.

Taigi, iš viso tyrimui bus naudojami 2 duomenų rinkiniai. Jų suvestinė pateikta

1 lentelė.



6 pav. CIFAR-10 duomenų rinkinio pavyzdžiai.

1 lentelė Naudojamų vaizdų duomenų rinkinių suvestinė.

Duomenų rinkinio pavadinimas	**Treniravimo imties dydis**	**Validacijos imties dydis**	**Testavimo imties dydis**	**Spalvų kanalų kiekis**	**Rezoliucija**	**Klasių kiekis**
MNIST [11]	55,000	5000	10,000	1	28x28	10
CIFAR-10 [36]	45,000	5000	10,000	3	32x32	10

2 lentelė MNIST LeNet1 paieškos erdvė.

Hiperparametro pavadinimas	**Galimos vertės**	**Vertės tipas**
Pirmo sąsūkų sluoksnio branduolių kiekis	Tarp 1 ir 350	Sveikasis skaičius
Pirmo sutelkimo sluoksnio dydis	2 arba 3	Sveikasis skaičius
Antro sąsūkų sluoksnio branduolių kiekis	Tarp 1 ir 350	Sveikasis skaičius
Antro sutelkimo sluoksnio dydis	2 arba 3	Sveikasis skaičius
Tankaus sluoksnio gylis	Tarp 1 ir 1024	Sveikasis skaičius
Sąsūkų sluoksnių branduolių dydis	Tarp 2 ir 10	Sveikasis skaičius
Mokymosi greitis	Tarp 0.0001 ir 0.4	Slankiojo kablelio skaičius
L2 reguliarizacijos parametras	Tarp 0 ir 1	Slankiojo kablelio skaičius
Atmetimo sluoksnio parametras	Tarp 0 ir 1	Slankiojo kablelio skaičius

3 lentelė MNIST LeNet2 paieškos erdvė.

Hiperparametro pavadinimas	**Galimos vertės**	**Vertės tipas**
Pirmo sąsūkų sluoksnio branduolių kiekis	Tarp 1 ir 350	Sveikasis skaičius
Antro sąsūkų sluoksnio branduolių kiekis	Tarp 1 ir 350	Sveikasis skaičius

Tankaus sluoksnio gylis	Tarp 1 ir 1024	Sveikasis skaičius
Mokymosi greitis	Tarp 0.0001 ir 0.4	Slankiojo kablelio skaičius
L2 reguliarizacijos parametras	Tarp 0 ir 1	Slankiojo kablelio skaičius
Atsitiktinio praretinimo transformacijos parametras	Tarp 0 ir 1	Slankiojo kablelio skaičius
Aktyvacijos funkcijos tipas	ReLU, tanh, sigmoid, eLU, leaky ReLU	Kategorinis
Optimizatoriaus tipas	AdaDelta, AdaGrad, Adam, GD, Momentum, RMSProp	Kategorinis
Paketo normalizacija	Įjungta arba išjungta	Kategorinis

4 lentelė CIFAR-10 LeNet3 paieškos erdvė.

Hiperparametro pavadinimas	**Galimos vertės**	**Vertės tipas**
Pirmo sąsūkų sluoksnio branduolių kiekis	Tarp 8 ir 32	Sveikasis skaičius
Antro sąsūkų sluoksnio branduolių kiekis	Tarp 32 ir 64	Sveikasis skaičius
Trečio sąsūkų sluoksnio branduolių kiekis	Tarp 64 ir 128	Sveikasis skaičius
Ketvirto sąsūkų sluoksnio branduolių kiekis	Tarp 64 ir 128	Sveikasis skaičius
Tankaus sluoksnio gylis	Tarp 1 ir 1024	Sveikasis skaičius
Sąsūkų sluoksnių branduolių dydis	2 arba 3	Sveikasis skaičius
Mokymosi greitis	Tarp 0.0001 ir 0.4	Slankiojo kablelio skaičius
L2 reguliarizacijos parametras	Tarp 0 ir 1	Slankiojo kablelio skaičius
Aktyvacijos funkcijos tipas	ReLU, tanh, eLU	Kategorinis
Optimizatoriaus tipas	AdaDelta, AdaGrad, Adam, GD, Momentum, RMSProp	Kategorinis
Normalizacijos metodo tipas	Išjungta, atsitiktinio praretinimo transformacija, paketo normalizacija	Kategorinis

2.2. Siūlomas metodas

Siūlomas metodas susideda iš trijų pagrindinių dedamųjų: Bajeso optimizacijos, modelių mokymo bei treniravimui skirtų duomenų rinkinio filtravimo. Apie kiekvieną dedamąją detaliai aprašyta šiame poskyryje.

2.2.1. Bajeso Optimizacija

Šiame darbe hiperparametrų paieškai yra naudojama Bajeso optimizacija.

Maksimalios juodosios funkcijos vertės paieškos atveju BO gali būti išreikštas kaip:

$$x^+ = \arg\max_{x \in A} f(x)\,. \tag{2}$$

Kur $f(x)$ yra juodosios dėžės funkcija, A – paieškos erdvė, x – parametrų rinkinys atrinktas iš A [39]. Tuo metu BO veikimo principas yra paremtas Bajeso teorema [78]. Turint įrodymų duomenis E, modelio M aposteriori tikimybė P(M|E) yra proporcinga tikimybei P(E|M) padaugintai su apriori tikimybe P(M):

$$P(M|E) \propto P(E|M)P(M). \tag{3}$$

Formulė (3) yra esminė Bajeso optimizacijai. Ji aprašo funkcijos $f(x)$ apriori skirstinio kombinavimą su jau ištyrinėtais įrodymų taškais aposteriori skirstiniui gauti. Tada šis skirstinys naudojamas kartu su atradimo funkcija u funkcijos $f(x)$ ekstremumui rasti. Funkcijos u uždavinys yra generuoti kitą paieškos tašką įrodymams gauti, balansuojant tarp diversifikacijos (siekiant naršyti aukšto neužtikrintumo plotus) ir intensifikacijos (siekiant naršyti aukštus $f(x)$ plotus), taip minimizuojant reikiamų taškų įvertinimus, ypatingai kai juodosios dėžės funkcija turi bent kelis lokalius ekstremumus.

Apriori skirstiniui išreikšti dažnai naudojamas Gauso procesas (GP) [39] dėl jo lankstumo bei paprastumo. Šių kriterijų realizavimas būtinas norint optimizuoti bet kokią funkciją. Pačio GP veikimo principas paremtas Bajeso mokymosi ir stochastinių procesų teorijomis [79]. Tai toks procesas, kuris teigia jog bet koks baigtinis atsitiktinių kintamųjų poaibis turi daugiamatį Gauso skirstinį. Yra daroma prielaida kad panaši įvestis grąžina panašią išvestį, taip kuriant statistinį funkcijos modelį. Šie Gauso skirstiniai yra aprašomi per vidurkius $m: x \to \mathbb{R}$ ir kovariacijos $k: x \times x \to \mathbb{R}$ funkcijas, tad GP gali būti išreikštas kaip:

$$f(x) \sim GP\big(m(x), k(x, x')\big). \tag{4}$$

Svarbu paminėti skirtumą tarp GP ir Gauso skirstinio. Gauso skirstinio tankio funkcija bet kokiam x yra ne skaliarinis dydis, o normalusis skirstinys visoms įmanomoms tankio funkcijos reikšmėms. Paprastumo dėlei daroma prielaida kad GP vidurkis lygus nuliui, kuomet kovariacijos funkcijai k aprašyti naudojama eksponentinė kvadratinė funkcija:

$$k(x_i, x_j) = \exp\left(-\frac{1}{2}\|x_i - x_j\|^2\right) \tag{5}$$

Kur x_i ir x_j yra i-tasis ir j-tasis imties elementai. Iš (5) galima matyti jog elementams x_i ir x_j esant panašiems, jų koreliacija stipri ir artėja prie 1 ir prie 0 atvirkštiniu atveju.

Aposteriori skirstiniui $f(x)$ apskaičiuoti pirmiausia iš treniravimo imties $D_{1:t} = \{x_n, f_n\}_{n=1}^{t}, f_n = f_n(x_n)$ yra įvertinami t elementų atsižvelgiant į daugiamatį normalų skirstinį $f \sim N(0, \boldsymbol{K})$, kur

$$\boldsymbol{K} = \begin{bmatrix} k(x_1, x_1) & k(x_1, x_2) & \cdots & k(x_1, x_t) \\ \vdots & & \ddots & \vdots \\ k(x_t, x_1) & k(x_t, x_2) & \cdots & k(x_t, x_t) \end{bmatrix}. \tag{6}$$

Tada apskaičiuojamas normalusis skirstinys $f_{t+1} = f(x_{t+1})$ naujame taške x_{t+1}. GP daroma prielaida, teigianti jog iš treniravimo imties gauta $\boldsymbol{f}_{1:t}$ suma su funkcijos reikšme f_{t+1} grąžina t+1 dimensijų normalųjį skirstinį:

$$\begin{bmatrix} \boldsymbol{f}_{1:t} \\ f_{t+1} \end{bmatrix} = N\left(0, \begin{bmatrix} \boldsymbol{K} & \boldsymbol{k} \\ \boldsymbol{k}^T & k(x_{t+1}, x_{t+1}) \end{bmatrix}\right). \tag{7}$$

Kur $\boldsymbol{f}_{1:t} = [f_1, f_2, \dots f_t]^T$, o $\boldsymbol{k} = [k(x_{t+1}, x_1)k(x_{t+1}, x_2) \dots k(x_{t+1}, x_t)]$. f_{t+1} skirstinys, kaip visi normalieji, gali būti apibūdinami kaip $N(\mu_{t+1}, \sigma_{t+1}^2)$. Pasinaudojus sujungtų Gauso skirstinių požymiu, galima išvesti:

$$\mu_{t+1}(x_{t+1}) = -\boldsymbol{k}^T \boldsymbol{K}^{-1} \boldsymbol{k} + k(x_{t+1}, x_{t+1}). \quad (8)$$

Turint pakankamai didelį duomenų rinkinį, šiuo būdu galima aproksimuoti bet kokios funkcijos $f(x)$ skirstinį.

2.2.2. Atradimo funkcija

Atradimo funkcija u yra naudojama priklausomai nuo jau turimų verčių vertingiausiems paieškos taškams generuoti. Pati paprasčiausia tokia funkcija generuoja potencialiai maksimalią aproksimuojamos funkcijos vertę turintį tašką:

$$x^+ = \arg\max_{x \in A} u(x|D)\,. \quad (9)$$

Tačiau toks metodas neatsižvelgia į aukštą neapibrėžtumą turinčias funkcijos vietas. Dėl to tyrėjai yra pasiūlę kitų atradimo funkcijų, tokių kaip pagerėjimo tikimybė, tikėtinas pagerėjimas ir viršutinė pasitikėjimo riba. Šiame darbe naudojama pastaroji. Jos forma yra ši:

$$UCB(x) = \mu(x) + k\sigma(x). \quad (10)$$

Kur $\mu(x)$ - žinomas optimalus taškas; $\sigma(x)$ - aukšto neužtikrintumo taškas; k - parametras balansuojantis tarp $\mu(x)$ ir $\sigma(x)$, arba tarp diversifikacijos ir intensifikacijos.

Bajeso optimizacija ir atradimo funkcija šio darbo reikmėms yra realizuota naudojant viešai prieinamą atvirą kodą [80].

2.2.3. Modelių mokymas

Modelio mokymo metodas yra pradedamas nuo duomenų rinkinio sudarymo ir paruošimo. Naudojant TensorFlow [81] mašininio mokymosi biblioteką, sukuriamas duomenų rinkinio objektas gebantis kiekvienos mokymo ir gradientinio nusileidimo iteracijos metu atrinkti pasirinkto dydžio atsitiktinį duomenų poaibį, konvertuoti jį į tenzorių ir perkelti jį į grafinio procesoriaus atmintį greitam apdorojimui.

Tada sukuriamas modelis. Tai atliekama eksperimentui priklausančiai funkcijai, kurioje implementuoti baziniai sluoksniai, perskaitant Bajeso optimizacijos atradimo funkcijos sugeneruotą hiperparametrų aibę ir įrašant kiekvieną sluoksnio parametrą į atitinkamą funkciją. Pavyzdžiui, pirmo sąsūkų sluoksnio filtrų kiekio hiperparametrą nuskaitant ir įrašant į atitinkamo sluoksnio filtrų kiekio argumentą. Šioje dalyje daugiausiai naudojama TensorFlow bibliotekos funkcijos, kuri leidžia viena kodo eilute sukurti sluoksnį ir priskirti jam poziciją tenzorių tėkmėje bei sukuria reikalingus ryšius tarp sluoksnių. Paskutinis tankus sluoksnis visada turi Softmax [82] aktyvacijos funkciją, kuri skirta klasifikacijai.

Po modelio sukūrimo, inicializuojami nuostolių bei optimizacijos objektai. Nuostolių objektas skaičiuoja kryžminės entropijos nuostolių funkciją lygindamas modelio spėtas klasių tikimybes gautas iš Softmax aktyvacijos sluoksnio ir tikrąsias vaizdų klases. Kuomet optimizavimo funkcija nėra duota, naudojama Adam [83] optimizavimo funkcija, ji inicializuojama.

Paskutiniame žingsnyje prieš mokymo ciklą yra inicializuojamos kelios mokymui skirtos funkcijos.

- Pirmoji jų skirta mokymosi greičio sumažinimui kai stabilizuojasi modelio tikslumas validacijos imčiai. Po 4 epochų nepagerėjus tikslumui, mokymosi greitis yra mažinamas 3 kartus.

- Antroji saugo geriausią išmokytą modelį. Po mokymosi proceso terminavimo geriausi modelio svoriai yra nuskaitomi tikslumo įverčiui testavimo imčiai gauti.

- Trečioji sustabdo mokymąsi prieš pasiekiant paskutinę epochą. Tai daroma po 7 epochų nepagerėjant tikslumui validacijos imčiai.

Modelio mokymo iteracijų metu pirmiausia modeliui leidžiama spėti klasių pasiskirstymą duomenų poaibiui, tada su nuostolių funkcija gaunama nuostolių reikšmė ir naudojant atgalinio sklidimo algoritmą yra atnaujinami modelio svoriai. Visos nuostolių reikšmės yra saugomos kartu su duomenų elementų indeksu. Tai kartojama kol nebeįmanoma sudaryti norimo dydžio poaibį nekartojant duomenų elementų. Tada modeliui duodama spėti validacijos rinkinio klases, spėjimai lyginami su tikraisiais ir apskaičiuojamas tikslumas, naudojamas modelio mokymosi procesui įvertinti, šis taip pat išsaugojamas tolesnei analizei. Gavus įvertį, treniravimo rinkinys vėl išmaišomas ir gradientinio nusileidimo ciklas kartojamas iki 100 kartų arba iki kol 7 epochas modelio tikslumas validacijos imčiai nepagerėja.

Kuomet mokymas nutraukiamas, užkraunami geriausi svoriai, apskaičiuojamas tikslumas testavimo imčiai ir šis grąžinamas Bajeso optimizacijos metodui.

2.2.4. Treniravimo duomenų filtravimas

Modelių treniravimo metu yra saugomos kiekvienos iteracijos elementų indeksas bei gauta nuostolių reikšmė. Po tam tikro nustatyto epochų skaičiaus $m_{geriausi} + 1$ yra pradedama tikrinti ar pavyko išmokyti $m_{geriausi}$ skaičių modelių, kurių testavimo duomenų imties tikslumo įvertis yra didesnis nei atsitiktinio spėjimo slenksčio. Atsitiktinio spėjimo tikimybė atspėti teisingą klasę y_i elementui i yra lygi $\frac{1}{k}$, kur k yra klasių skaičius. Tiek MNIST, tiek CIFAR-10 turi 10 unikalių klasių, tad šiame darbe naudojamas tikslumo slenkstis yra lygus 0.1. Kuomet yra randami $m_{geriausi}$ tokie modeliai, yra atrenkami $m_{geriausi}$ geriausi modeliai ir pagal juos yra atliekamas treniravimo duomenų imties filtravimas, o modelių mokymo iteracija fiksuojama kaip e_1. Šiame darbe yra tiriami du duomenų filtravimo metodai.

Pirmasis jų yra aprašytas [71] darbe ir vadinamas elementų užmiršimo skaičiumi. Šis skaičius yra gaunamas skaičiuojant pamiršimo įvykius ir naudojamas klasifikacijai skirtų DNT modelių mokymo kontekste. Pažymime $p(y_{ik}|x_i;\theta^t)$ kaip iki iteracijos t apmokyto modelio θ^t santykinę tikimybę spėti klasę y_{ik} elementui i su įvestimi x_i. Tada elemento i spėjama klasė įvesčiai x_i ties mokymo iteracija t yra:

$$\hat{y}_i^t = \arg\max_k p(y_{ik}|x_i;\theta^t)\,. \tag{99}$$

Dvejetainis kintamasis įgaunantis reikšmę 1, kuomet ties iteracija t elemento i klasė y_i yra spėta teisingai ir 0, kai neteisingai apibrėžiamas:

$$acc_i^t = 1_{\hat{y}_i^t = y_i}$$

Taigi, elemento i pamiršimo įvykis yra fiksuojamas tada, kai iteracijoje $t < \infty$ šis elementas yra klasifikuojamas teisingai, tačiau iteracijoje $t + 1$ neteisingai, t. y. $acc_i^t > acc_i^{t+1}$. Neužmirštami elementai yra tokie, kurių užmiršimo skaičius modelio mokymo gale yra lygus 0.

Užmiršimo įvykių skaičius tarp atrinktų $m_{geriausi}$ modelių yra agreguojamas naudojant min funkciją ir išfiltruojami neužmirštami elementai. Kitais žodžiais, jei bent vieno iš atrinktų modelio mokymo metu elementui i nebuvo užfiksuotas nei vienas užmiršimo įvykis, toks elementas yra išfiltruojamas ir nebenaudojamas vėliau treniruojamų modelių mokyme.

Kitas treniravimo duomenų imties filtravimo metodas yra pirmo žvilgsnio sudėtingumas. Šis yra naujas, o veikimo būdas yra grįstas kitais paklaida grįstais metodais [71], [72]. Pirmiausia yra nuskaitomi $m_{geriausi}$ geriausių modelių pirmos epochos elementų nuostolių vertės, taip gaunant pirmo žvilgsnio nuostolių funkcijos reikšmes. Šios reikšmės yra agreguojamos naudojant min funkciją, minimaliai pirmo žvilgsnio nuostolių reikšmei gauti. Tada, naudojant slenkstinę vertę yra rūšiuojamos į sudėtingas ir nesudėtingas, priklausomai ar nuostolių funkcijos vertė yra atitinkamai didesnė ar mažesnė už slenkstinę. Čia slenkstinė vertė yra minimali nuostolių funkcijos vertė teisingai klasifikuotam elementui, šiuo atveju naudojant kryžminės entropijos [70] nuostolių funkciją apskaičiuojama šios funkcijos vertė kai visi įvesties elementai turi vienodą tikimybę lygią $\frac{1}{k*(1+\varepsilon)}$, kur k yra klasių skaičius, o ε yra maža konstanta, tyrimo metu lygi 0.001. Tuo metu teisingos klasės tikimybė yra lygi $1 - \frac{k-1}{k*(1+\varepsilon)}$, taip užtikrinant, kad visų tikimybių vektoriaus suma lygi 1. Šio tyrimo metu $k = 10$, tad slenkstinė vertė yra lygi ~0.324. Elementai su mažesne nuostolių verte nei ši slenkstinė vertė yra apžymimi kaip nesudėtingi ir yra išfiltruojami iš vėliau treniruojamų modelių mokymo duomenų imties.

Po pirmo treniravimo duomenų imties filtravimo ties modelių mokymo iteracija e_1, ties kiekviena modelių išmokymo iteracija yra tikrinama ar buvo išmokytas modelis turintis didesnį tikslumą testavimo imčiai nei mažiausias iš prieš tai atrinktų. Jei taip, vėl atrenkami $m_{geriausi}$ aukščiausią tikslumą testavimo imčiai turintys modeliai ir atliekamas toks pat filtravimas.

2.2.5. Naudota programinė ir kompiuterinė įranga

Šiam metodui realizuoti buvo naudojama Python 3 [84] programavimo kalba ir aibė paketų. Bajeso optimizacijai buvo naudota viešai prieinama atviro kodo biblioteka BayesianOptimization [80], modelių mokymui – biblioteka TensorFlow [81], o duomenų filtravimo metodai buvo implementuoti naudojant bazines Python kalbos funkcijas bei duomenų analizei ir statistikai skirtų duomenų struktūrų biblioteką Pandas [85].

Eksperimentams ir skaičiavimams atlikti buvo naudojamas vienas Windows 10 [86] operacinę sistemą naudojantis asmeninis kompiuteris. Į jį yra įdiegta 32 gigabaitai darbinės atminties, NVIDIA GeForce RTX 3070 grafikos procesorius bei AMD Ryzen 7 3700X centrinis procesorius.

3. Tyrimų rezultatai ir jų aptarimas

Šiame skyriuje pateikti 4842 eksperimentų rezultatai atlikti MNIST ir CIFAR-10 duomenų imtims naudojant elementų pamiršimo kiekio ir pirmo žvilgsnio sudėtingumo duomenų filtravimo metodus leidžiant Bajeso optimizacijai ieškoti geriausių hiperparametrų rinkinio 100 iteracijų ir gautų rezultatų analizė.

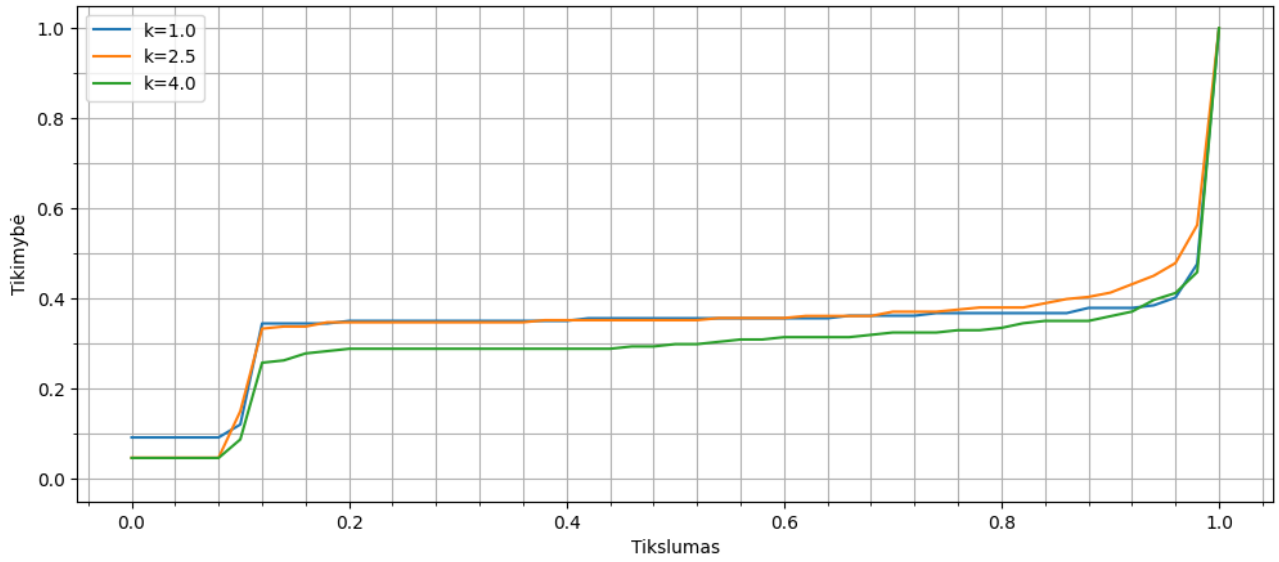
3.1. Atradimo funkcijos parametro k analizė

Eksperimentai buvo atlikti naudojant viršutinės pasitikėjimo ribos atradimo funkciją, kuriai yra priskirtas vienas kintamasis – k. Nuo šio parametro priklauso Bajeso optimizacijos algoritmo diversifikacijos ir intensifikacijos kompromiso riba. Kuo didesnis šis parametras, tuo algoritmas linkęs diversifikuoti. Tyrimui buvo naudojamos trys skirtingos šio parametro reikšmės: 1, 2.5 ir 4. 2.5 yra rekomenduojama vertė [87], tačiau priklausomai nuo užduoties gali skirtis, todėl buvo parinktos kitos reikšmingai besiskiriančios vertės.

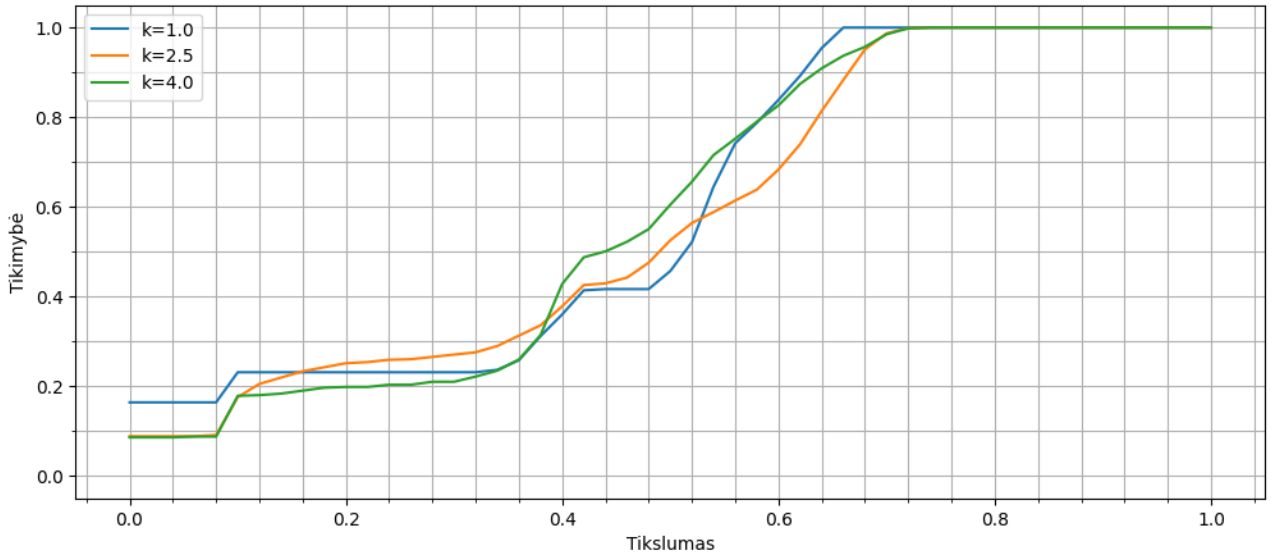
Kaip matoma iš MNIST LeNet1 ir LeNet2 tikslumo pasiskirstymo grafiko **7 pav.** , kai k=1, išbandoma daugiausiai mažesnį nei 10% tikslumą turinčių architektūrų, kas reiškia kad modeliai nesugebėjo apsimokyti su duotais hiperparametrais. Tarp 10% ir 80% mažiausiai architektūrų išmokyta kai k=4, kuomet tarp k reikšmių 1 ir 2.5 skirtumo beveik nėra. Tačiau virš 90% rėžyje aiškiai matyti, jog kai k=1 sugeneruojama apie 10% daugiau architektūrų nepasiekiančių 100% tikslumo, nei su k=2.5 ar k=4 parametrais. Tikslumo vidurkis aukščiausias kai k=4: 70%, kas yra maždaug 5% daugiau nei kitų k parametrų rezultatai.

Atliekant CIFAR-10 LeNet3 eksperimentus buvo aiškiai pastebėta, jog k=1 vertė nėra tinkama, kadangi aukščiausias pasiektas tikslumas su šia parametro reikšme yra 65.7%, kas yra 7% mažiau nei nustačius kitas parametro vertes. Tuo metu vidurkis tarp k=1 ir k=4 eksperimentų beveik nesiskiria, kuomet pastarojo parametro eksperimentai pasiekė didžiausią tikslumą iš visų: 72.7%. Taip yra dėl to kad su maža diversifikacijos reikšme atradimo funkcija generuoja panašius į geriausią tikslumą gavusius parametrus. Stebint **8 pav.** grafiką, matoma jog parametro k=2.5 kreivė siekia geriausią balansą iki 50% tikslumo, o tarp šio ir maksimalaus pasiskirstymas yra kur kas arčiau maksimalaus nei kitų parametrų kreivių.

Taigi, galima teigti, jog optimali parametro k vertė priklauso nuo atliekamo eksperimento, tačiau yra didesnė nei 1.



7 pav. MNIST LeNet1 eksperimentų tikslumo pasiskirstymo grafikas priklausomai nuo k vertės.



8 pav. CIFAR-10 LeNet3 eksperimentų tikslumo pasiskirstymo grafikas priklausomai nuo k vertės.

3.2. Treniravimo duomenų imties filtravimo metodų analizė

Šiai analizei pasitelktos sklaidos diagramos lyginančios kiekvieno išmokyto modelio tikslumą testavimo duomenų imčiai ir naudotą treniravimo duomenų imties dydį **9 pav.**, **10 pav.** , **11 pav.** Jos geriausiai atspindi modelio klasifikacijos efektyvumo priklausomybę nuo išfiltruoto duomenų kiekio. Kiekvienas taškas grafikuose atspindi vieno pilnai ištreniruoto modelio geriausią pasiektą tikslumą. Punktyrinės linijos žymi didžiausią pasiektą tikslumą naudojant nurodytą ar mažesnę duomenų imtį.

Abejais filtravimo metodais naudojant abi treniravimo duomenų imtis daugiausia modelių buvo ištreniruota su pilna duomenų imtimi. Taip yra dėl to, kad kiekvienos Bajeso optimizacijos pradžioje yra naudojama pilna duomenų imtis, o filtravimas vykdomas po bent 3 mokymų iteracijų.

Naudojant užmiršimo įvykių kiekio filtravimo metodą MNIST treniravimo duomenų imčiai ir LeNet1 paieškos erdvę **9 pav.** galima pastebėti, jog modelių tikslumas gali siekti maksimalų (žalia ir mėlyna punktyrinės linijos) iki kol šioje duomenų imtyje lieka tik ~6000 vaizdų elementų, kas yra ~90%

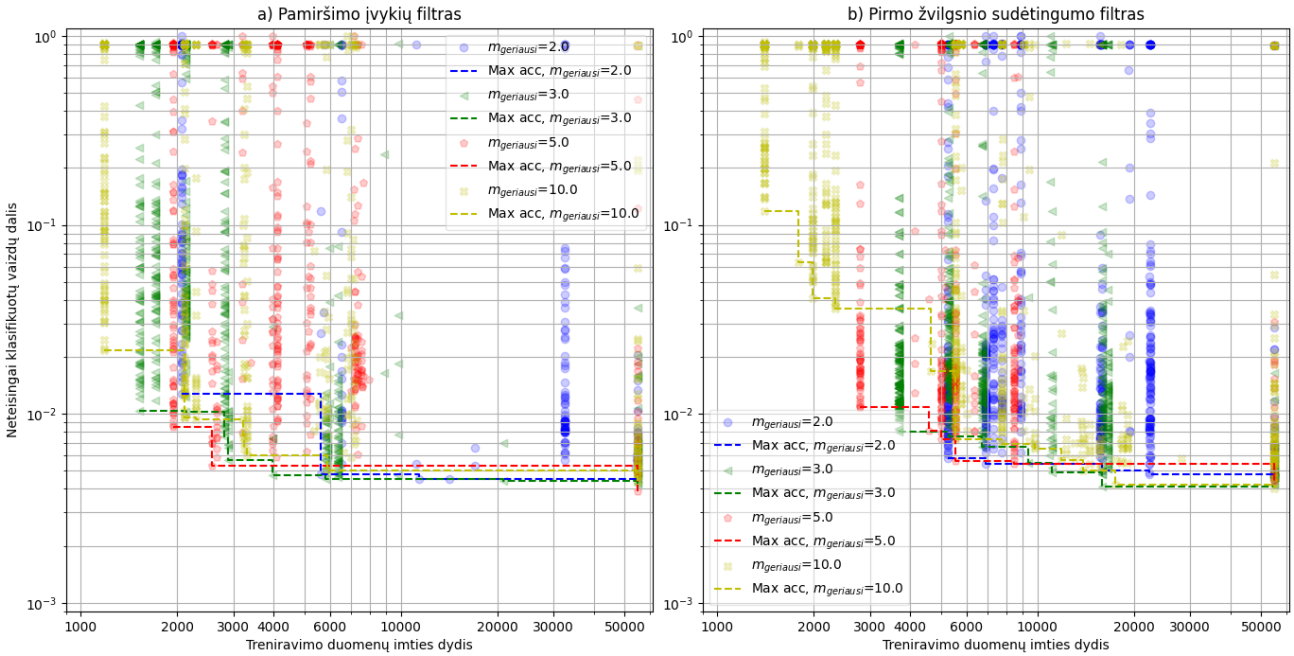
pilnos duomenų imties. Filtruojant daugiau, matoma, jog modelių tikslumas tendencingai mažėja. Tai reiškia, jog branduolio duomenų aibė gauta naudojant šį metodą MNIST duomenų imčiai sudaro tik ~10% pilnos duomenų imties, kuomet išfiltruoti duomenys nėra reikšmingi modelio efektyvumui. Tai patvirtina ir žemas Pirsono koreliacijos koeficientas lygus 0.168.

Tam pačiam eksperimentui naudojant pirmo žvilgsnio sudėtingumo duomenų filtravimo metodą **9 pav.11 pav.**, branduolio duomenų aibės dydis tesiekia ~30% pilnos duomenų aibės dydį (žalia punktyrinė linija). Šiuo metodu filtruojant didesnę dalį duomenų modelių tikslumas yra mažesnis. Tai rodo ir didesnis Pirsono koreliacijos koeficientas lygus 0.206, grafikas **9 pav.11 pav.** rodo, jog šis metodas MNIST duomenų imčiai yra mažiau efektyvus nei užmiršimo įvykių kiekio filtravimo metodas.

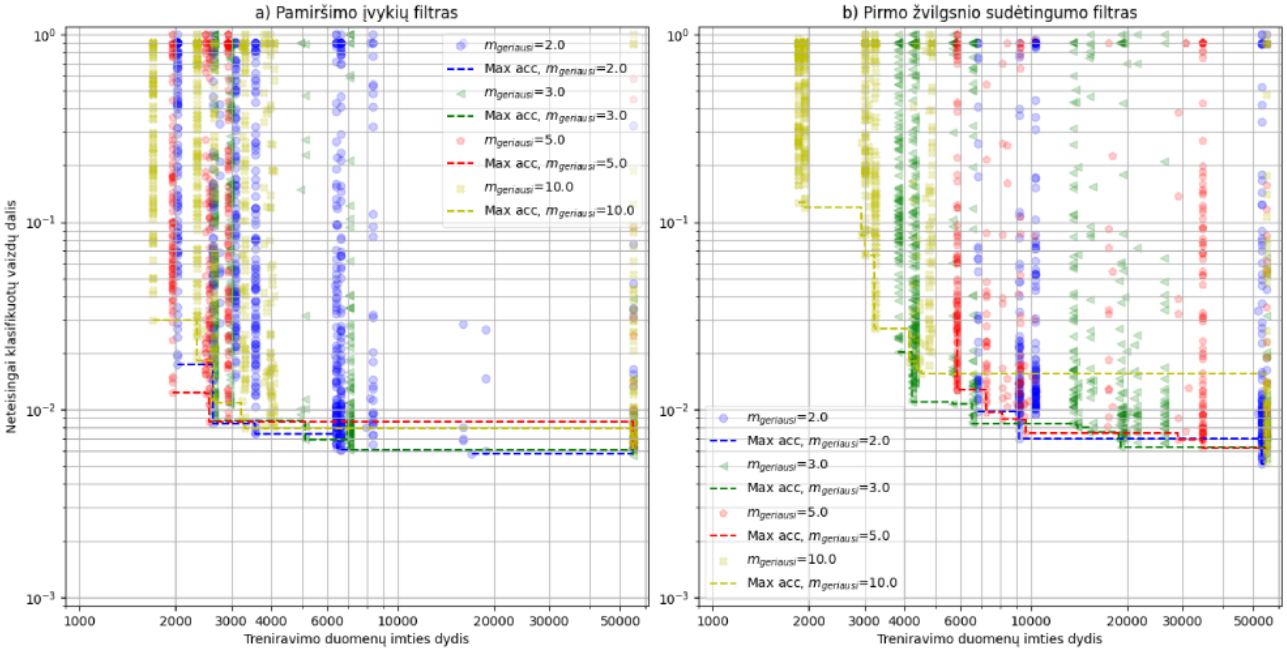
Tai pačiai MNIST duomenų imčiai taikant LeNet2 paieškos erdvę, gaunami rezultatai panašūs **10 pav.** Pamiršimo įvykių filtras išfiltruoja panašų kiekį elementų nepakenkiant tikslumo įverčiui, o pirmo žvilgsnio metodas išfiltruoja tik apie 65% duomenų iki kol pradedamas mažinti aukščiausias įmanomas tikslumas.

CIFAR-10 duomenų aibės rezultatai rodo tą pačią efektyvumų tendenciją **11 pav.** Užmiršimo įvykių kiekio filtravimo metodas geba išfiltruoti 35% duomenų nepakeičiant maksimalaus modelių tikslumo (žalia punktyrinė linija), kuomet filtruojant pirmo žvilgsnio sudėtingumo metodu maksimalus tikslumas mažėja išfiltravus vos ~4000 elementų arba ~9% iš pilnos duomenų imties.

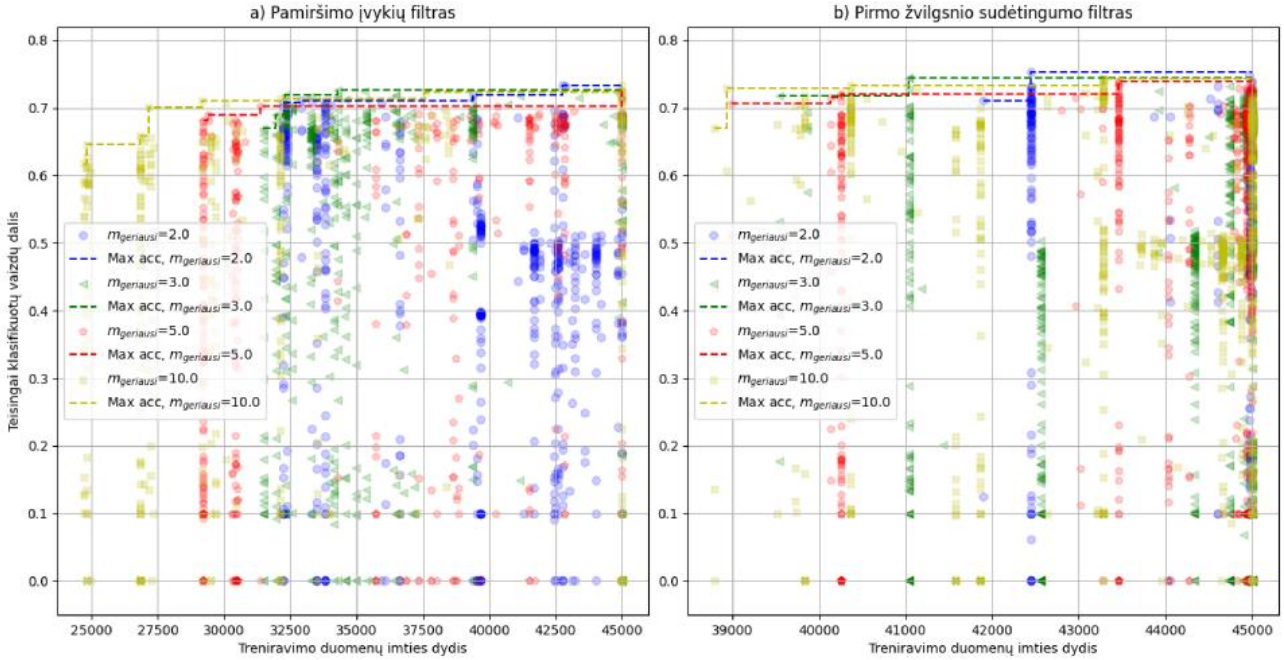
Taigi, naudojant užmiršimo įvykių kiekio filtravimo metodą gaunama mažesnė branduolio duomenų aibė nei naudojant pirmo žvilgsnio sudėtingumo metodą.



9 pav. MNIST LeNet1 testavimo imties neteisingai klasifikuotų vaizdų dalis priklausomai nuo treniravimo imties dydžio filtruojamo a) elementų užmiršimo metodu b) pirmo žvilgsnio sudėtingumo metodu sklaidos diagrama. Abejoms ašims pritaikyta logaritminė transformacija.



10 pav. MNIST LeNet2 testavimo imties neteisingai klasifikuotų vaizdų dalis priklausomai nuo treniravimo imties dydžio filtruojamo a) elementų užmiršimo metodu b) pirmo žvilgsnio sudėtingumo metodu sklaidos diagrama. Abejoms ašims pritaikyta logaritminė transformacija.



11 pav. CIFAR-10 LeNet3 testavimo imties neteisingai klasifikuotų vaizdų dalis priklausomai nuo treniravimo imties dydžio filtruojamo a) elementų užmiršimo metodu b) pirmo žvilgsnio sudėtingumo metodu sklaidos diagrama.

3.3. Geriausių modelių kiekio parametro įtaka filtravimo metodų efektyvumui

Abu filtravimo metodai naudoja $m_{geriausi}$ parametrą, kuris nustato koks geriausią tikslumą įvertį turinčių modelių skaičius yra naudojamas filtruojant treniravimo duomenų imtį. Kadangi naudojama min agregavimo funkcija, tad ar tai pamiršimų skaičius, ar pirmos epochos nuostolių funkcijos reikšmė, kuo šis parametras didesnis, tuo daugiau duomenų yra išfiltruojama. Tad yra būtina atlikti

šio parametro analizę ir atrasti parametrą, kuriuo yra filtruojami tik neprasmingi duomenų elementai. Šiai analizei yra pasitelkiami tie patys **9 pav.**, **10 pav.**, **11 pav.** grafikai.

Pirmo žvilgsnio sudėtingumo metodas duomenų imčiai MNIST ir hiperparametrų paieškos erdvei LeNet1 **9 pav.** didžiausią tikslumą ilgiausiai išlaiko kai $m_{geriausi} = 3$ (žalia punktyrinė linija). Kuomet $m_{geriausi} = 10$ (geltona punktyrinė linija) tikslumas yra panašus prie panašių duomenų kiekių, tačiau parametrui priskyrus šią vertę, yra išfiltruojama ir 53,000 duomenų elementų, taip padidinant neteisingai klasifikuotų vaizdų skaičių 10 kartų.

MNIST LeNet1 ir LeNet2 eksperimentams taikant elementų pamiršimo filtravimo metodą **9 pav.**, **10 pav.**, geriausią tikslumą prie mažiausio duomenų kiekio gaunanti $m_{geriausi}$ reikšmė yra taip pat 3 (žalia punktyrinė linija). Panašūs rezultatai pasiekiami ir naudojant 2 (mėlyna punktyrinė linija), tačiau išfiltravus tik kiek daugiau nei 90% imties, modelių tikslumas ženkliai sumažėja.

CIFAR-10 LeNet3 eksperimentų rezultatų grafikuose **11 pav.** matomos tos pačios tendencijos. Nepriklausomai nuo filtravimo metodo, parametrui $m_{geriausi}$ įgavus reikšmę 3 yra gaunami didžiausio efektyvumo rezultatai, o su reikšme 2 – rezultatai tik vos prastesni.

Taigi, efektyviausia filtravimo agregavimui naudojamų modelių kiekio parametro reikšmė yra 3. Naudojant šią vertę yra gaunamas mažiausia branduolio duomenų aibė. Kuomet $m_{geriausi} = 2$, rezultatai yra nedaug prastesni. Agregavimui naudojant 5 geriausius modelius, išfiltravus mažiau duomenų yra gaunami prastesni rezultatai, o naudojant 10 – išfiltruojamas per didelis elementų skaičius.

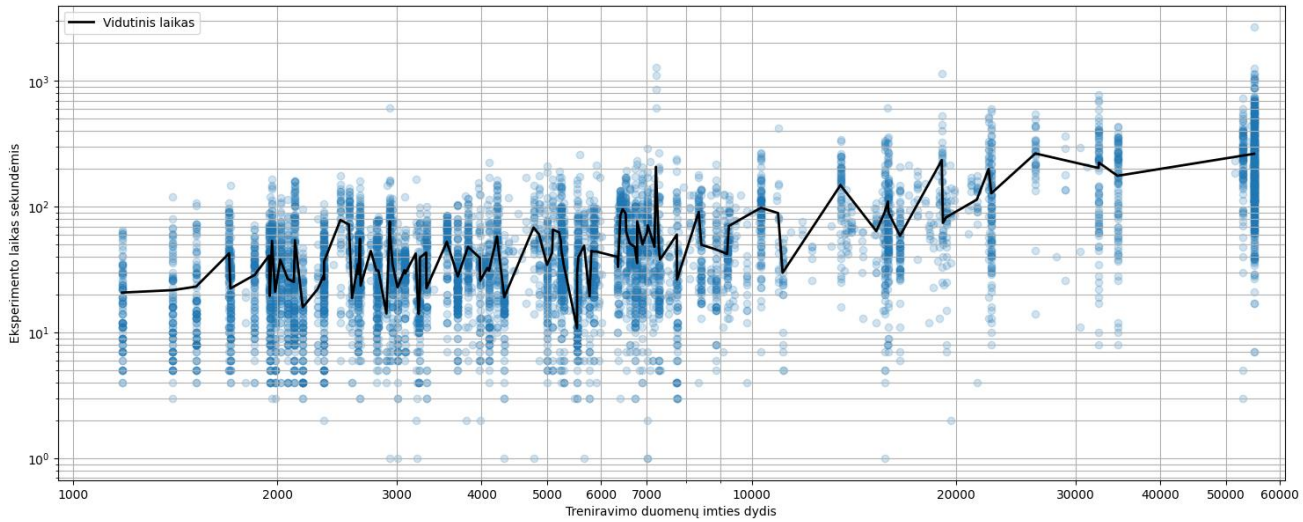
3.4. Filtravimo metodų greitaveikos analizė

Mažinant treniravimo duomenų imtį mažėja ir iteracijų skaičius epochai, tad ir individualaus modelio treniravimo epochos trunkamas laikas. Tačiau naudojant išankstinį modelio sustabdymą dažnu atveju modelis pasiekia savo aukščiausią įmanomą tikslumą prieš pasiekdamas paskutinę įmanomą epochą, o drastiškai sumažinus treniravimo duomenų aibės dydį, šių epochų prireikia kur kas daugiau. To pasėkoje yra būtina analizė parodanti viso modelio mokymo trukmės priklausomybę nuo treniravimo duomenų aibės dydžio. Analizei naudojamos sklaidos diagramos rodančios sąryšį tarp duomenų kiekio ir modelio treniravimo laiko **12 pav.**, **13 pav.**

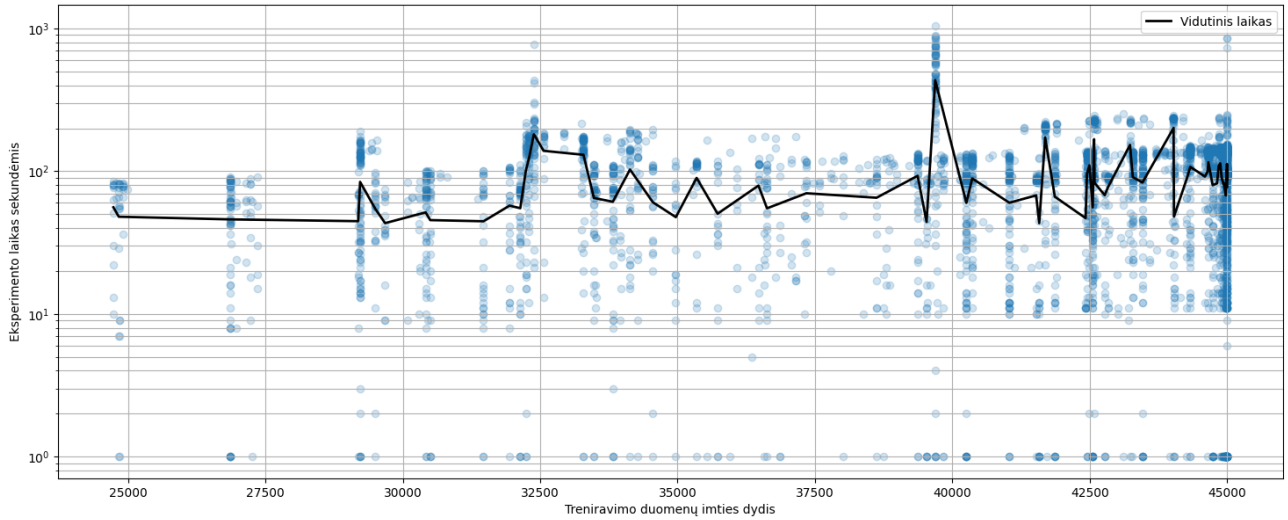
MNIST duomenų aibės grafikas **12 pav.** rodo, jog, mažinant duomenų kiekį, mažėja pilno modelio apmokymo laikas nuo vidutiniškai 264 sekundžių iki 21. Taip pat, galima pastebėti, jog naudojant vis šis mažėjimas nėra tiesiškas, naudojant vis mažiau duomenų greitaveikos padidėjimas tampa vis mažiau žymus, o pasiekus 4000 elementų ribą, greitis lieka panašus – vidutiniškai 35 sekundės. Taip yra dėl vis didėjančio reikalingų epochų skaičiaus kiekvieno modelio apmokymui. Pirsono koreliacijos koeficientas tarp treniravimosi laiko ir duomenų imties dydžio lygus 0.625, kas indikuoja stiprią koreliaciją tarp šių kintamųjų.

CIFAR-10 treniravimo duomenų imties dydžio ir eksperimento trukmės sklaidos diagrama **13 pav.**, nors ir triukšmingesnė, rodo, jog vidutinis eksperimento laikas taip pat mažėja kartu su duomenų imtimi. Eksperimentai naudoję virš 40000 vaizdų treniravimui vidutiniškai truko 90 sekundes, o naudojant mažiau nei 35000 – 79 sekundes. Ekvivalentus Pirsono koreliacijos koeficientas čia taip pat reikšmingai mažesnis – 0.076. Mažiau reikšmingas greitaveikos padidėjimas čia atrandamas dėl sudėtingesnės treniravimo aibės bei paieškos erdvės susidedančios iš gilesnių architektūrų.

Atliktus šią analizę galima teigti, jog DNT treniravimo greitis priklauso nuo aibės kintamųjų, kurių vienas yra treniravimo aibės dydis.



12 pav. MNIST LeNet1 ir LeNet2 apmokymo laiko priklausomybės nuo treniravimo imties dydžio sklaidos diagrama. Abejoms ašims pritaikyta logaritminė transformacija.



13 pav. CIFAR-10 LeNet3 apmokymo laiko priklausomybės nuo treniravimo imties dydžio sklaidos diagrama. Y ašiai pritaikyta logaritminė transformacija.

Išvados

1. Buvo sukurtas, realizuotas ir išanalizuotas hiperparametrų paieškai Bajeso optimizaciją naudojantis ir dvejais metodais optimizavimo metu treniravimo duomenų imtį filtruojantis metodas.
2. Atlikus atradimo funkcijos diversifikaciją ir intensifikaciją balansuojančio parametro k analizę buvo pastebėta, jog optimali reikšmė kinta priklausomai nuo užduoties. Siekiant geriausių rezultatų LeNet1 ir LeNet2 eksperimentams ši reikšmė turi būti lygi 4, o LeNet3 eksperimentams 2.5. Tačiau yra aišku, jog ši reikšmė turi būti didesnė nei 1.
3. Elementų pamiršimo metodas iš MNIST treniravimo duomenų imties išfiltruoja 90% elementų nepakenkdamas treniruojamų modelių tikslumui, kuomet naudojant pirmo žvilgsnio sudėtingumo metodą galima išfiltruoti tik 70% treniravimo imties. Atitinkamai iš CIFAR-10 imties pirmasis metodas išgauna 35% mažesnę branduolio duomenų aibę lyginant su pilna duomenų imtimi, o antrasis tik 9%. Todėl atlikus analizę yra aišku, jog elementų pamiršimo kiekio filtravimo metodas yra pranašesnis.
4. Ištyrus geriausių modelių kiekio parametro įtaką filtravimo metodams buvo atrasta, jog abejiems metodams geriausia šio parametro vertė yra 3. Ši vertė efektyviai balansuoja tarp išfiltruoto duomenų kiekio bei pasiekiamo tikslumo, kas indikuoja efektyvią branduolio duomenų aibės paiešką.
5. Atlikus duomenų kiekio įtakos greitaveikai analizę buvo pastebėta, jog greitaveika priklauso nuo daugiau kintamųjų nei tik mokymo duomenų kiekis, tačiau šis kiekis yra viena greitaveiką veikiančių dedamųjų. Šis efektas matomas ypatingai gerai MNIST duomenų imties analizėje, kur tarp mokymo duomenų kiekio ir vidutinio mokymo trukmės Pirsono koreliacijos koeficientas lygus 0. 625.

Rekomendacijos tolimesniems tyrimams

1. Atlikti šiame darbe atliktą analizę naudojant diversifikaciją skatinančius metodus Bajeso optimizacijai. Pažangesnių metodų tyrimas atskleistų jų efektyvumą taikant duomenų filtravimo metodus.
2. Naudoti kelių tikslų Bajeso optimizacijos algoritmą, į tikslus įtraukiant tokius įverčius kaip modelio dydis ar greitis. Tokiu atveju bus ieškomos ne tik didžiausią tikslumo įvertį gaunančios hiperparametrų aibės, bet ir mažiausiai resursų reikalaujančios.
3. Palyginti šiame darbe naudotus branduolio aibės konstravimo metodus su kitais aprašytais literatūroje.
4. Išsaugotus duomenis apie treniravimą panaudoti kitiems tikslams, tokiems kaip sunkių elementų paieška ar mokymo plano sudarymas.

Literatūros sąrašas

1. [1] H.-J. Yoo, “Deep Convolution Neural Networks in Computer Vision: a Review,” *IEIE Transactions on Smart Processing and Computing*, vol. 4, no. 1, pp. 35–43, Feb. 2015, doi: 10.5573/IEIESPC.2015.4.1.035.
2. [2] E. Real *et al.*, “Large-Scale Evolution of Image Classifiers,” 2017, doi: 10.48550/ARXIV.1703.01041.
3. [3] C. White, A. Zela, B. Ru, Y. Liu, and F. Hutter, “How Powerful are Performance Predictors in Neural Architecture Search?,” 2021, doi: 10.48550/ARXIV.2104.01177.
4. [4] U. Gupta *et al.*, “The Architectural Implications of Facebook’s DNN-based Personalized Recommendation.” arXiv, 2019. doi: 10.48550/ARXIV.1906.03109.
5. [5] M. Wistuba, A. Rawat, and T. Pedapati, “A Survey on Neural Architecture Search.” arXiv, 2019. doi: 10.48550/ARXIV.1905.01392.
6. [6] F. Rosenblatt, *The Perceptron, a Perceiving and Recognizing Automaton Project Para*. in Report: Cornell Aeronautical Laboratory. Cornell Aeronautical Laboratory, 1957. [Online]. Available: https://books.google.lt/books?id=P_XGPgAACAAJ
7. [7] A. Griewank, “Who invented the reverse mode of differentiation?,” in *Documenta Mathematica Series*, 1st ed., M. Grötschel, Ed., EMS Press, 2012, pp. 389–400. doi: 10.4171/dms/6/38.
8. [8] A. Toosi, A. G. Bottino, B. Saboury, E. Siegel, and A. Rahmim, “A Brief History of AI: How to Prevent Another Winter (A Critical Review),” *PET Clinics*, vol. 16, no. 4, pp. 449–469, Oct. 2021, doi: 10.1016/j.cpet.2021.07.001.
9. [9] E. H. Shortliffe and B. G. Buchanan, “A model of inexact reasoning in medicine,” *Mathematical Biosciences*, vol. 23, no. 3–4, pp. 351–379, Apr. 1975, doi: 10.1016/0025-5564(75)90047-4.
10. [10] L. E. Baum and T. Petrie, “Statistical Inference for Probabilistic Functions of Finite State Markov Chains,” *Ann. Math. Statist.*, vol. 37, no. 6, pp. 1554–1563, Dec. 1966, doi: 10.1214/aoms/1177699147.
11. [11] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998, doi: 10.1109/5.726791.
12. [12] V. Zue, S. Seneff, and J. Glass, “Speech database development at MIT: Timit and beyond,” *Speech Communication*, vol. 9, no. 4, pp. 351–356, Aug. 1990, doi: 10.1016/0167-6393(90)90010-7.
13. [13] O. Russakovsky *et al.*, “ImageNet Large Scale Visual Recognition Challenge,” 2014, doi: 10.48550/ARXIV.1409.0575.
14. [14] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks,” in *Advances in Neural Information Processing Systems*, F. Pereira, C. J. Burges, L. Bottou, and K. Q. Weinberger, Eds., Curran Associates, Inc., 2012. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf
15. [15] D. Silver *et al.*, “Mastering the game of Go with deep neural networks and tree search,” *Nature*, vol. 529, no. 7587, pp. 484–489, Jan. 2016, doi: 10.1038/nature16961.

16. [16] K. Simonyan and A. Zisserman, “Very Deep Convolutional Networks for Large-Scale Image Recognition.” arXiv, 2014. doi: 10.48550/ARXIV.1409.1556.
17. [17] C. Szegedy *et al.*, “Going Deeper with Convolutions.” arXiv, 2014. doi: 10.48550/ARXIV.1409.4842.
18. [18] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition.” arXiv, 2015. doi: 10.48550/ARXIV.1512.03385.
19. [19] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, “Densely Connected Convolutional Networks.” arXiv, 2016. doi: 10.48550/ARXIV.1608.06993.
20. [20] A. G. Howard *et al.*, “MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications.” arXiv, 2017. doi: 10.48550/ARXIV.1704.04861.
21. [21] M. Tan and Q. V. Le, “EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks,” 2019, doi: 10.48550/ARXIV.1905.11946.
22. [22] C. Ying, A. Klein, E. Real, E. Christiansen, K. Murphy, and F. Hutter, “NAS-Bench-101: Towards Reproducible Neural Architecture Search,” 2019, doi: 10.48550/ARXIV.1902.09635.
23. [23] P. Ren *et al.*, “A Comprehensive Survey of Neural Architecture Search: Challenges and Solutions.” arXiv, 2020. doi: 10.48550/ARXIV.2006.02903.
24. [24] X. He, K. Zhao, and X. Chu, “AutoML: A survey of the state-of-the-art,” *Knowledge-Based Systems*, vol. 212, p. 106622, Jan. 2021, doi: 10.1016/j.knosys.2020.106622.
25. [25] Y. Sun, B. Xue, M. Zhang, and G. G. Yen, “Completely Automated CNN Architecture Design Based on Blocks,” *IEEE Trans. Neural Netw. Learning Syst.*, vol. 31, no. 4, pp. 1242–1254, Apr. 2020, doi: 10.1109/TNNLS.2019.2919608.
26. [26] B. Zoph and Q. V. Le, “Neural Architecture Search with Reinforcement Learning,” 2016, doi: 10.48550/ARXIV.1611.01578.
27. [27] H. Pham, M. Y. Guan, B. Zoph, Q. V. Le, and J. Dean, “Efficient Neural Architecture Search via Parameter Sharing,” 2018, doi: 10.48550/ARXIV.1802.03268.
28. [28] H. Cai, L. Zhu, and S. Han, “ProxylessNAS: Direct Neural Architecture Search on Target Task and Hardware.” arXiv, 2018. doi: 10.48550/ARXIV.1812.00332.
29. [29] W. Chen *et al.*, “Understanding and Accelerating Neural Architecture Search With Training-Free and Theory-Grounded Metrics,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 46, no. 2, pp. 749–763, Feb. 2024, doi: 10.1109/TPAMI.2023.3328347.
30. [30] W. Wen, H. Liu, H. Li, Y. Chen, G. Bender, and P.-J. Kindermans, “Neural Predictor for Neural Architecture Search,” 2019, doi: 10.48550/ARXIV.1912.00848.
31. [31] L. Xie and A. Yuille, “Genetic CNN,” 2017, doi: 10.48550/ARXIV.1703.01513.
32. [32] A. E. Eiben and J. E. Smith, “Evolutionary Computing: The Origins,” in *Introduction to Evolutionary Computing*, in Natural Computing Series. , Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, pp. 13–24. doi: 10.1007/978-3-662-44874-8_2.
33. [33] H. Liu, K. Simonyan, O. Vinyals, C. Fernando, and K. Kavukcuoglu, “Hierarchical Representations for Efficient Architecture Search.” arXiv, 2017. doi: 10.48550/ARXIV.1711.00436.
34. [34] G. Bender, P.-J. Kindermans, B. Zoph, V. Vasudevan, and Q. Le, “Understanding and Simplifying One-Shot Architecture Search,” in *Proceedings of the 35th International Conference on Machine Learning*, J. Dy and A. Krause, Eds., in Proceedings of Machine Learning Research,

vol. 80. PMLR, Jul. 2018, pp. 550–559. [Online]. Available: https://proceedings.mlr.press/v80/bender18a.html

35. [35] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, “Learning Transferable Architectures for Scalable Image Recognition,” 2017, doi: 10.48550/ARXIV.1707.07012.
36. [36] A. Krizhevsky, “Learning Multiple Layers of Features from Tiny Images,” 2009.
37. [37] T. Domhan, J. T. Springenberg, and F. Hutter, “Speeding Up Automatic Hyperparameter Optimization of Deep Neural Networks by Extrapolation of Learning Curves,” in *International Joint Conference on Artificial Intelligence*, 2015. [Online]. Available: https://api.semanticscholar.org/CorpusID:369457
38. [38] A. Klein, S. Falkner, J. T. Springenberg, and F. Hutter, “Learning Curve Prediction with Bayesian Neural Networks,” in *International Conference on Learning Representations*, 2017. [Online]. Available: https://openreview.net/forum?id=S11KBYclx
39. [39] J. Wu, X.-Y. Chen, H. Zhang, L.-D. Xiong, H. Lei, and S.-H. Deng, “Hyperparameter Optimization for Machine Learning Models Based on Bayesian Optimizationb,” *Journal of Electronic Science and Technology*, vol. 17, no. 1, pp. 26–40, 2019, doi: https://doi.org/10.11989/JEST.1674-862X.80904120.
40. [40] H. J. Kushner, “A New Method of Locating the Maximum Point of an Arbitrary Multipeak Curve in the Presence of Noise,” *Journal of Basic Engineering*, vol. 86, no. 1, pp. 97–106, Mar. 1964, doi: 10.1115/1.3653121.
41. [41] J. Močkus, “On bayesian methods for seeking the extremum,” in *Optimization Techniques IFIP Technical Conference Novosibirsk, July 1–7, 1974*, vol. 27, G. I. Marchuk, Ed., in Lecture Notes in Computer Science, vol. 27. , Berlin, Heidelberg: Springer Berlin Heidelberg, 1975, pp. 400–404. doi: 10.1007/3-540-07165-2_55.
42. [42] J. B. Mockus and L. J. Mockus, “Bayesian approach to global optimization and application to multiobjective and constrained problems,” *J Optim Theory Appl*, vol. 70, no. 1, pp. 157–172, Jul. 1991, doi: 10.1007/BF00940509.
43. [43] J. Mockus, V. Tiesis, and A. Zilinskas, “The application of Bayesian methods for seeking the extremum,” in *Towards Global Optimization*, vol. 2, 2014, pp. 117–129.
44. [44] D. Lizotte, T. Wang, M. Bowling, and D. Schuurmans, “Automatic Gait Optimization with Gaussian Process Regression.,” Jan. 2007, pp. 944–949.
45. [45] Z. Wang, B. Shakibi, L. Jin, and N. de Freitas, “Bayesian Multi-Scale Optimistic Optimization.” arXiv, 2014. doi: 10.48550/ARXIV.1402.7005.
46. [46] R. Garnett, M. A. Osborne, and S. J. Roberts, “Bayesian optimization for sensor set selection,” in *Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks*, Stockholm Sweden: ACM, Apr. 2010, pp. 209–219. doi: 10.1145/1791212.1791238.
47. [47] K. Kandasamy *et al.*, “Tuning Hyperparameters without Grad Students: Scalable and Robust Bayesian Optimisation with Dragonfly.” arXiv, 2019. doi: 10.48550/ARXIV.1903.06694.
48. [48] H. Shi, R. Pi, H. Xu, Z. Li, J. T. Kwok, and T. Zhang, “Bridging the Gap between Sample-based and One-shot Neural Architecture Search with BONAS.” arXiv, 2019. doi: 10.48550/ARXIV.1911.09336.
49. [49] M. Malu, G. Dasarathy, and A. Spanias, “Bayesian Optimization in High-Dimensional Spaces: A Brief Survey,” in *2021 12th International Conference on Information, Intelligence,*

Systems & Applications (IISA), Chania Crete, Greece: IEEE, Jul. 2021, pp. 1–8. doi: 10.1109/IISA52424.2021.9555522.

50. [50] H. Cho, Y. Kim, E. Lee, D. Choi, Y. Lee, and W. Rhee, "Basic Enhancement Strategies When Using Bayesian Optimization for Hyperparameter Tuning of Deep Neural Networks," *IEEE Access*, vol. 8, pp. 52588–52608, 2020, doi: 10.1109/ACCESS.2020.2981072.
51. [51] M. Lozano and C. García-Martínez, "Hybrid metaheuristics with evolutionary algorithms specializing in intensification and diversification: Overview and progress report," *Computers & Operations Research*, vol. 37, no. 3, pp. 481–497, Mar. 2010, doi: 10.1016/j.cor.2009.02.010.
52. [52] S. Zapotecas Martínez and C. A. Coello Coello, "A novel diversification strategy for multi-objective evolutionary algorithms," in *Proceedings of the 12th annual conference companion on Genetic and evolutionary computation*, Portland Oregon USA: ACM, Jul. 2010, pp. 2031–2034. doi: 10.1145/1830761.1830852.
53. [53] E. Nava, M. Mutny, and A. Krause, "Diversified Sampling for Batched Bayesian Optimization with Determinantal Point Processes," in *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*, G. Camps-Valls, F. J. R. Ruiz, and I. Valera, Eds., in Proceedings of Machine Learning Research, vol. 151. PMLR, Mar. 2022, pp. 7031–7054. [Online]. Available: https://proceedings.mlr.press/v151/nava22a.html
54. [54] D. Golovin, B. Solnik, S. Moitra, G. Kochanski, J. Karro, and D. Sculley, "Google Vizier: A Service for Black-Box Optimization," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Halifax NS Canada: ACM, Aug. 2017, pp. 1487–1495. doi: 10.1145/3097983.3098043.
55. [55] L. Prechelt, "Automatic early stopping using cross validation: quantifying the criteria," *Neural Networks*, vol. 11, no. 4, pp. 761–767, Jun. 1998, doi: 10.1016/S0893-6080(98)00010-0.
56. [56] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.
57. [57] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift." arXiv, 2015. doi: 10.48550/ARXIV.1502.03167.
58. [58] Y. Bai *et al.*, "Understanding and Improving Early Stopping for Learning with Noisy Labels." arXiv, 2021. doi: 10.48550/ARXIV.2106.15853.
59. [59] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization," *Advances in neural information processing systems*, vol. 24, 2011.
60. [60] C.-J. Wu *et al.*, "Sustainable ai: Environmental implications, challenges and opportunities," *Proceedings of Machine Learning and Systems*, vol. 4, pp. 795–813, 2022.
61. [61] R. Ratcliff, "Connectionist models of recognition memory: Constraints imposed by learning and forgetting functions.," *Psychological Review*, vol. 97, no. 2, pp. 285–308, 1990, doi: 10.1037/0033-295X.97.2.285.
62. [62] J. Kirkpatrick *et al.*, "Overcoming catastrophic forgetting in neural networks," *Proc. Natl. Acad. Sci. U.S.A.*, vol. 114, no. 13, pp. 3521–3526, Mar. 2017, doi: 10.1073/pnas.1611835114.
63. [63] C. Guo, B. Zhao, and Y. Bai, "DeepCore: A Comprehensive Library for Coreset Selection in Deep Learning," in *Database and Expert Systems Applications*, vol. 13426, C. Strauss, A. Cuzzocrea, G. Kotsis, A. M. Tjoa, and I. Khalil, Eds., in Lecture Notes in Computer Science, vol. 13426. , Cham: Springer International Publishing, 2022, pp. 181–195. doi: 10.1007/978-3-031-12423-5_14.

64. [64] G. Frahling and C. Sohler, “Coresets in dynamic geometric data streams,” in *Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, Baltimore MD USA: ACM, May 2005, pp. 209–217. doi: 10.1145/1060590.1060622.
65. [65] S. Har-Peled and S. Mazumdar, “On coresets for k-means and k-median clustering,” in *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, Chicago IL USA: ACM, Jun. 2004, pp. 291–300. doi: 10.1145/1007352.1007400.
66. [66] I. Jubran, A. Maalouf, and D. Feldman, “Introduction to Coresets: Accurate Coresets.” arXiv, 2019. doi: 10.48550/ARXIV.1910.08707.
67. [67] Y. Chen, M. Welling, and A. Smola, “Super-Samples from Kernel Herding.” arXiv, 2012. doi: 10.48550/ARXIV.1203.3472.
68. [68] O. Sener and S. Savarese, “Active Learning for Convolutional Neural Networks: A Core-Set Approach.” arXiv, 2017. doi: 10.48550/ARXIV.1708.00489.
69. [69] C. Coleman *et al.*, “Selection via Proxy: Efficient Data Selection for Deep Learning.” arXiv, 2019. doi: 10.48550/ARXIV.1906.11829.
70. [70] P.-T. De Boer, D. P. Kroese, S. Mannor, and R. Y. Rubinstein, “A Tutorial on the Cross-Entropy Method,” *Ann Oper Res*, vol. 134, no. 1, pp. 19–67, Feb. 2005, doi: 10.1007/s10479-005-5724-z.
71. [71] M. Toneva, A. Sordoni, R. T. des Combes, A. Trischler, Y. Bengio, and G. J. Gordon, “An Empirical Study of Example Forgetting during Deep Neural Network Learning.” arXiv, 2018. doi: 10.48550/ARXIV.1812.05159.
72. [72] M. Paul, S. Ganguli, and G. K. Dziugaite, “Deep Learning on a Data Diet: Finding Important Examples Early in Training,” 2021, doi: 10.48550/ARXIV.2107.07075.
73. [73] Y. Liu, Y. Sun, B. Xue, M. Zhang, G. G. Yen, and K. C. Tan, “A Survey on Evolutionary Neural Architecture Search,” 2020, doi: 10.48550/ARXIV.2008.10937.
74. [74] A. Zela, J. Siems, L. Zimmer, J. Lukasik, M. Keuper, and F. Hutter, “Surrogate NAS Benchmarks: Going Beyond the Limited Search Spaces of Tabular NAS Benchmarks,” 2020, doi: 10.48550/ARXIV.2008.09777.
75. [75] A. Zela, J. Siems, and F. Hutter, “NAS-Bench-1Shot1: Benchmarking and Dissecting One-shot Neural Architecture Search,” 2020, doi: 10.48550/ARXIV.2001.10422.
76. [76] G. Cohen, S. Afshar, J. Tapson, and A. van Schaik, “EMNIST: an extension of MNIST to handwritten letters,” 2017, doi: 10.48550/ARXIV.1702.05373.
77. [77] B. Recht, R. Roelofs, L. Schmidt, and V. Shankar, “Do CIFAR-10 Classifiers Generalize to CIFAR-10?,” 2018, doi: 10.48550/ARXIV.1806.00451.
78. [78] E. Brochu, V. M. Cora, and N. de Freitas, “A Tutorial on Bayesian Optimization of Expensive Cost Functions, with Application to Active User Modeling and Hierarchical Reinforcement Learning.” arXiv, 2010. doi: 10.48550/ARXIV.1012.2599.
79. [79] E. Schulz, M. Speekenbrink, and A. Krause, “A tutorial on Gaussian process regression: Modelling, exploring, and exploiting functions,” *Journal of Mathematical Psychology*, vol. 85, pp. 1–16, Aug. 2018, doi: 10.1016/j.jmp.2018.03.001.
80. [80] F. Nogueira, “Bayesian Optimization: Open source constrained global optimization tool for Python.” 2014. [Online]. Available: https://github.com/bayesian-optimization/BayesianOptimization

81. [81] TensorFlow Developers, “TensorFlow.” [object Object], Mar. 08, 2024. doi: 10.5281/ZENODO.4724125.
82. [82] J. S. Bridle, “Training stochastic model recognition algorithms as networks can lead to maximum mutual information estimation of parameters,” in *Proceedings of the 2nd International Conference on Neural Information Processing Systems*, in NIPS’89. Cambridge, MA, USA: MIT Press, 1989, pp. 211–217.
83. [83] D. P. Kingma and J. Ba, “Adam: A Method for Stochastic Optimization.” arXiv, 2014. doi: 10.48550/ARXIV.1412.6980.
84. [84] G. van Rossum and F. L. Drake, *The Python language reference*, Release 3.0.1 [Repr.]. in Python documentation manual / Guido van Rossum; Fred L. Drake [ed.], no. Pt. 2. Hampton, NH: Python Software Foundation, 2010.
85. [85] The pandas development team, “pandas-dev/pandas: Pandas.” Zenodo, Apr. 10, 2024. doi: 10.5281/ZENODO.3509134.
86. [86] E. Bott and C. Stinson, *Windows 10 inside out*. Microsoft Press, 2019.
87. [87] J. Berk, S. Gupta, S. Rana, and S. Venkatesh, “Randomised Gaussian Process Upper Confidence Bound for Bayesian Optimisation,” 2020, doi: 10.48550/ARXIV.2006.04296.