VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
PROGRAMŲ SISTEMŲ STUDIJŲ PROGRAMA

Priešiškų trukdžių nustatymas naudojant neuroninius tinklus praplėstus adaptyvaus mokymosi žingsnio algoritmais

Adversarial Attack Detection using Neural Networks with Adaptive Learning Rate Algorithms

Magistro baigiamasis darbas

Atliko:	Miglė Vaitulevičiūtė	(parašas)
Darbo vadovas:	asist. dr. Vytautas Valaitis	(parašas)
Darbo recenzentas:	asist. dr. Valdas Dičiūnas	(parašas)

Vilnius – 2021

Santrauka

Teorinėje darbo dalyje buvo aprašomi dirbtinis ir konvoliucinis neuroniniai tinklai, jų sudėtis ir veikimas. Taip pat aprašomi skirtingų tipų duomenys bei jų problemos - dimensijų prakeiksmas ir persimokymas. Aprašomos Liapunovo funkcijos ir Liapunovo eksponentės bei jų panaudojimas neuroninių tinklų kontekste. Toliau apibrėžiamas pasirinktas duomenų rinkinys, vertinimo metrikos ir technologijos. Eksperimentinėje dalyje apibrėžtos naudotos vertinimo metrikos, programų veikimas ir patys bandymai. Tinklas su Liapunovo funkcijos algoritmais - LF I ir patobulintu LF I - bei Adam funkcija naudotas spręsti XOR ir paveiksliukų paveiktų priešiškų trukdžių klasifikavimo uždavinius. Patobulintas LF I pasiekia geriausias metrikas, bet atlikus konvergavimo bandymus buvo nustatyta, kad tinklui su šiuo algoritmu reikia žymiai daugiau epochų negu kitiems, kad pasiekti labai mažą nuostolio funkcijos reikšmę. LF I algoritmas pasiekia konvergavimo sąlygą greičiausiai sprendžiant XOR problemą, o Adam - paveiksliukų klasifikavimo. Šie rezultatai parodo, kad nors ir Liapunovo funkcijos algoritmai gali spręsti šias užduotis, bet jie yra neefektyvūs su sudėtingomis užduotimis kaip paveiksliukų klasifikacija. Kita eksperimento dalis - Liapunovo eksponenčių gavimas, kurio metu buvo naudojami priešiškų trukdžių paveikti MNIST ir CIFAR bei sudarytas duomenų rinkiniai. Buvo pastebėta, kad Liapunovo eksponentės iš nesudėtingų duomenų buvo susigrupavusios į atitinkamas klases (originalius ir priešiškų trukdžių paveiktus), o sudėtingesnių duomenų - klasės buvo stipriai persidengusios. Panaudojus Liapunovo eksponentes tinklo apmokymui buvo nustatyta - kuo smarkiau jos yra susigrupavusios, tuo didesnį tikslumą tinklas gali pasiekti. Taip pat buvo nustatyta, kad vidutiniškai tinklas su patobulintu LF I algoritmu pasiekia geriausią tikslumą, kai apmokymui naudojamas Liapunovo eksponenčių rinkinys.

Raktiniai žodžiai: konvoliuciniai neuroniniai tinklai, prisitaikantis mokymosi žingsnis, Liapunovo funkcija, priešiški trukdžiai, Liapunovo eksponentės

Summary

The theoretical part of the work described artificial and convolutional neural networks, their structure, and function. It also describes the different types of data and their problems - the curse of dimensionality and overfitting. Lyapunov functions and Lyapunov exponents and their application in the context of neural networks are described. The selected dataset, assessment metrics, and technologies are also defined. The experimental part defines the evaluation metrics used, the overview of the created programs, and the experiments themselves. The network with Lyapunov function algorithms - LF I and improved LF I - and Adam function were used to solve the XOR problem and adversarial attacked image classification. The improved LF I achieves the best metrics, but convergence experiments have shown that a network with this algorithm needs significantly more epochs than others to achieve very low loss function value. The LF I algorithm achieves the convergence condition fastest when solving the XOR problem, and Adam when classifying images. These results show that although the Lyapunov function algorithms can solve these problems, they are less efficient with complex tasks such as image classification. The next part of the experiment was the calculation of Lyapunov exponents using MNIST and CIFAR datasets affected by adversarial attacks and an additional dataset compiled in this work. It was observed that Lyapunov exponents were more clustered from simple data into appropriate classes (original and adversarial attacked), than from more complex data in which case the classes were strongly overlapped. The use of Lyapunov exponents for network training has shown that the more clustered the exponents are, the higher the accuracy the network can achieve. It was also found that, on average, when a set of Lyapunov exponents is used for training a network with an improved LF I algorithm, it achieves the highest accuracy.

Keywords: convolutional neural network, adaptive learning rate, Lyapunov function, adversarial attack, Lyapunov exponents

TURINYS

Įvadas

Vienas iš dirbtinių neuroninių tinklų tipų yra konvoliuciniai neuroniniai tinklai, kurių viena iš užduočių yra spęsti klasifikacijos uždavinį [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 įvairiose srityse - nuo medicinos iki savivaldžių automobilių. Šiam uždaviniui spręsti naudojant konvoliucinius neuroninius tinklus geriausia yra turėti didelį („ImageNet“ duomenų bazė turi apie 15 milijonų paveiksliukų) ir įvairų duomenų rinkinį (angl. dataset) [HKY19; KSH12]. Tačiau ne visose mokslo ar komercijos srityse tai yra įmanoma. Pavyzdžiui, medicinos [SK17], psichologijos [EBF18] ir kitose srityse mažas duomenų kiekis yra norma, kadangi tyrimai vyksta su realiais žmonėmis. Todėl iš mažo kiekio duomenų gali kilti persimokymo (angl. overfitting) problema. Ji atsiranda, kai neuroninis tinklas išmoksta specifinius duomenų požymius ir jis neberanda bendrų pasikartojimų (angl. patterns) [Web10]. Taigi, permokytas tinklas pasirodo puikiai mokymosi metu, tačiau labai prastai su naujais duomenimis.

Bendrų pasikartojimų radimo problema gali kilti ir dėl didelio dimensijų (angl. dimensions) kiekio įeities duomenyse (angl. input data) [KSK19]. Ši problema vadinama dimensijų prakeiksmu (angl. curse of dimensionality), o dimensijos yra įvairūs požymiai, kuriuos turi duomenys, bei jų kiekis yra nustatomas įeities sluoksnyje. Šis efektas pasireiškia, kai duomenų dimensijų (skirtingų požymių) kiekis didėja ir taip pat tų pačių duomenų retumas didėja, todėl norint gauti patikimus rezultatus reikalingų duomenų kiekis irgi auga [SM14]. Priešingu atveju mokymo duomenys suteikia tik vietinių užuominų, tačiau neužpildo aukštos dimensijos erdvės. Taigi, dirbant su mažu kiekiu duomenų ir norint išanalizuoti didelį kiekį tų duomenų požymių tai gali būti neįmanoma dėl dimensijų prakeiksmo ir persimokymo. Pavyzdžiui, apmokant neuroninį tinklą atlikti kalbos atpažinimą gali prireikti kelių šimtų dimensijų [ACS+18]. Todėl mokant neuroninį tinklą su didele įeities dimensija privaloma struktūrizuoti duomenis. Šiuo atveju konvoliuciniai neuroniniai tinklai turi pranašumą, nes apibrėžta paveiksliukų struktūra stipriai sumažina išmokstamų parametrų kiekį [Wój19].

Neuroniniai tinklai turi dar vieną pranašumą prieš dimensijų prakeiksmo efektą - mokymo metu tinklas pradinį didelio dimensijų kiekio duomenų rinkinį projektuoja (angl. to project) į mažesnio dydžio dimensijų reprezentacijos erdves [GSM+18]. Tai yra dimensijų mažinimo (angl. dimensionality reduction) procesas. Jis vyksta kiekviename tinklo sluoksnyje, kadangi kiekvienas neuronas tinkle bus jautrus tik specifiniam aktualiam požymiui. Tad, neuroniniai tinklai geba

brėžti netiesinių ir sudėtingų formų sprendimo ribas (angl. decision boundaries) aplink taškus dimensijų erdvėje [REH+19]. Tačiau iš prigimties konvoliuciniai neuroniniai tinklai negeba spręsti kitos su dimensijų prakeiksmu susijusios problemos - persimokymo. Šios problemos efektas gali būti sumažintas atliekant tokius veiksmus: pridedant daugiau įeities duomenų, keičiant (angl. augment) turimus įeities duomenis [PW17], pridedant reguliarizavimą (angl. regularization) kad būtų pasiektas geresnis generalizavimas [SHK+14], sumažinti naudojamos architektūros sudėtingumą [XZG+18], pasirinkti architektūrą, kuri gerai generalizuoja.

Dimensijų mažinimo procesas gali būti suskirstytas į du tipus - požymių išrinkimas (angl. feature selection) ir požymių išgavimas (angl. feature extraction). Abu tipai gali būti dar labiau suskirstyti atitinkamai į filtrų, įvyniojimo (angl. wrapper) ir įterptus metodus bei tiesinius ir netiesinius metodus. Požymių išrinkimui ir išgavimui gali būti naudojama Liapunovo eksponentė arba jų rinkinys, vadinamas Liapunovo spektras. Liapunovo eksponentė gali identifikuoti chaotišką, periodišką ir stabilų judėjimą. Toks būdas buvo naudojamas analizuojant elektroencefalografijos signalus [ÜG07], hiperspektrinius paveiksliukus [YGJ12], priešiškus trukdžius (angl. adversarial perturbation) paveiksliukams [PDW18]. Visuose šiuose darbuose iš pradinių duomenų buvo gautas požymis, kuris atitiko Liapunovo eksponentę taip sutraukiant dimensijas, kas padėjo įveikti dimensijų prakeiksmą (plačiau išdėstyta „Liapunovo eksponentės panaudojimas" skyriuje).

Nors ir tinklai pasiekia vis geresnį tikslumą ir yra vis labiau patikimi, tačiau jie turi vieną saugumo spragą, kurią gali eksploatuoti piktus ketinimus turintys asmenys - duomenų paveikimas priešiškais trukdžiais (angl. adversarial attack). Tai yra maži, beveik nepastebimi duomenų pakeitimai, dėl kurių modelis padaro neteisingą spėjimą su aukštu užtikrintumu, nors tuos pačius duomenis pamatęs žmogus juos atpažįsta teisingai. Pavyzdžiui, kelio ženklai gali būti apipaišyti arba apklijuoti lipdukais taip, kad savarankiškai vairuojantys automobiliai su labai aukštu užtikrintumu neteisingai atpažintų tokius ženklus ir sukelti pavojų žmonėms [PMG+16]. Šiai neuroninių tinklų spragai yra sukurta įvairių apsisaugojimo būdų (plačiau išdėstyta „Duomenų rinkinys" skyriuje), tačiau dauguma metodų gali apginti tik nuo specifinių trukdžių ir dažniausiai besiginantysis neturi informacijos apie galimus priešiškus trukdžius iš anksto [WLW+18].

Vienas iš galimų apsisaugojimo būdų yra papildomo modelio naudojimas, kuris nustatytų, ar pateikti duomenys yra paveikti priešiškų trukdžių ar ne. Taigi, šio darbo metu buvo naudotas duomenų rinkinys, kuris turėjo dvi klases - tai originalūs ir paveikti priešiško trukdžio paveiksliukai. Persimokymo problemos išvengimui buvo naudoti algoritmai, kurių nuostolio funkcija buvo įrodyta kaip griežta Liapunovo funkcija siekiant užtikrinti modelio konvergavimą. Norint, kad funkcija būtų laikoma Liapunovo funkcija, ji turi atitikti tam tikras prielaidas (daugiau skyriuje „Liapu-

novo funkcija“). Tokiu būdu, jei funkcija yra Liapunovo funkcija, tai yra pakankamas įrodymas, kad ji yra stabili - pradinės sąlygos, kurios prasideda prie pusiausvyros taško (angl. equilibrium point), išlieka prie jo. Tačiau sugriežtinus prielaidas, galima teigti, kad arti esančios trajektorijos konverguoja į pusiausvyros tašką ir tai yra vadinama asimptotiškai stabili. Šie algoritmai buvo sukonstruoti [BKP06] ir [Rav15] straipsniuose, tačiau juose aprašyti algoritmai buvo išbandyti sprendžiant paprastesnes problemas ir su daug paprastesniu neuroniniu tinklu. Taip pat duomenų rinkinys remiantis [PDW18] straipsniu buvo konvertuotas į Liapunovo eksponentes tikintis, jog jos išgaus požymius, kurie apibūdins abi klases, ir taip sumažins dimensijų kiekį bei gautas Liapunovo eksponentes bus galima panaudoti neuroninio tinklo apmokymui.

Magistro darbo tikslas

Šio darbo tikslas yra išanalizuoti Liapunovo funkcijos praplėstą neuroninį tinklą ir Liapunovo eksponenčių panaudojamumą tokio tinklo apmokymui sprendžiant klasifikacijos uždavinį su priešiškų trukdžių paveiktais duomenimis.

Magistro darbo uždaviniai

1. Atlikti teorinę analizę apie dimensijų prakeiksmą, persimokymą, Liapunovo funkciją bei Liapunovo eksponentes, ištirti, kaip jos gali būti pritaikytos neuroniniams tinklams.
2. Paruošti duomenų rinkinį, kuris būtų paveiktas priešiškų trukdžių, bei naudojant jį suderinti pasirinktą neuroninį tinklą.
3. Pritaikyti Liapunovo eksponentes kaip dimensijų mažinimo metodą ir su apskaičiuotomis Liapunovo eksponentėmis apmokyti sukurtą neuroninį tinklą.

Eksperimento žingsniai:

1. Sukonstruoti neuroninį tinklą ir jį praplėsti su algoritmais, paremtais Liapunovo funkcija.
2. Suderinti sukurtą tinklą su pasirinktu duomenų rinkiniu.
3. Sukonstruoti duomenų konvertavimo sistemą iš paveiksliukų į Liapunovo eksponentes ir panaudoti ją pasirinkto duomenų rinkinio konvertavimui.
4. Naudojant konvertuotus duomenis apmokyti paprastesnį neuroninį tinklą.

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 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 - tai paveiksliukai, vaizdo medžiaga, garsas ir t.t. Ši informacija yra reikalinga tam, kad tinklas galėtų ją išanalizuoti ir išmokti, 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.

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 funkcijos“). Jei šie rezultatai skiriasi, slenksčio reikšmės ir svoriai yra keičiami atitinkamai [RS17], tai vadina-

ma sklidimo 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 reikėtų 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 ir jį galima įvardinti kaip funkciją (1):

$$\frac{\partial C_0}{\partial w^L} = \frac{\partial z^L}{\partial w^L}\frac{\partial a^L}{\partial z^L}\frac{\partial C_0}{\partial a^L}. \tag{1}$$

Šioje funkcijoje vienas sluoksnis turi vieną neuroną, priklausomai nuo neuronų ir sluoksnių skaičiaus prie funkcijos parametrų prisidėtų atitinkami indeksai. Funkcija parodo nuostolio funkcijos dalinės išvestinės ir svorio (arba slenksčio reikšmės) santykio išvestinę, 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. Taip pat, jei dvi viena po kitos apskaičiuotos dalinės išvestinės artimai nesutampa, tai parodo, kad įvyko klaida.

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):

$$Y = A(\Sigma(w * d) + b). \tag{2}$$

Formulėje (2) pateikta raidė A reiškia bet kokią pasirinktą aktyvavimo funkciją, 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 sudaugi-

nus su įvesties duomenimis. Taip pat slenksčio reikšmė parodo, ar reikia sustiprinti ar susilpninti 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; 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 nuokrypio (angl. mean squared error). Ši funkcija apskaičiuoja vidutinį 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 sprendžiamos problemos 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, siekiant sumažinti gaunamų rezultatų netikslumą [Nik16]. Visos optimizavimo funkcijos gali būti suskirstytos į du tipus - nuolatinio mokymosi žingsnio ir prisitaikančio mokymosi. 1 lentelė buvo parengta remiantis [Rud16] straipsniu. Joje išvardintos visos populiariausios optimizavimo funkcijos.

1 lentelė. Optimizavimo funkcijos

Pavadinimas	Tipas	Privalumai	Trūkumai	Veikimas
SGD	Nuolatinio mokymosi žingsnio	Parametrų atnaujinimai turi aukštą dispersiją, tai leidžia lengviau rasti lokalų minimumą.	Didelis svyravimas trukdo konverguoti.	Parametrų atnaujinimas vykdomas kiekvienai mokymo iteracijai.
Adam	Prisitaikančio mokymosi	Greitai konverguoja, ir modelio mokymosi žingsnio yra didelis bei efektyvus.	Praleidžia mažą lokalų minimumą.	Suskaičiuoja mokymosi žingsnį kiekvienam parametrui bei saugo eksponentiškai nykstantį prieš tai buvusį kvadratinio gradiento vidurkį ir eksponentiškai mažėjantį prieš tai buvusį gradiento vidurkį, panašų į inercija (angl. momentum).
Adagrad	Prisitaikančio mokymosi	Nereikia rankiniu būdu derinti mokymosi žingsnio.	Mokymosi žingsnis visada yra mažėjantis ir nykstantis, o tai lėtina konvergavimą.	Leidžia mokymosi žingsniui priklausyti nuo parametrų. Dideli atnaujinimai nedažniems parametrams, maži atnaujinimai dažniems parametrams.
RMSprop	Prisitaikančio mokymosi	Greitai konverguoja.	Inercija nedidina funkcijos efektyvumo.	Dalija mokymosi žingsnį 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 parametrų atnaujinimui yra naudojama optimizavimo funkcija, kuri apskaičiuoja gradientą. Svoriai yra keičiami pagal priešingą apskaičiuoto gradiento kryptį - bandoma leistis į gradiento minimumą.

Optimizavimo funkcijos turi parametrą - mokymosi žingsnį (angl. learning rate). Jis privalo būti nustatytas, tačiau pasirinkti tinkamą mokymosi žingsnį 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 tą pasiekti galima gradientu judant į žemiausią jo vietą, tačiau pasirinkus per didelį mokymosi žingsnį yra galimybė, kad žemiausia vieta bus peršokta ir bus tolstama nuo jos.

1.6. Hiperparametrai

Mašininiame 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. Dalis hiperparametrų yra: mokymosi žingsnis, epochų ir partijų (angl. batch) dydžiai, 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 tai, kad konvoliucinio neuroninio tinklo įvesties sluoksnis 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 sudėtas viena ant kitos tris 2D matricas. 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].

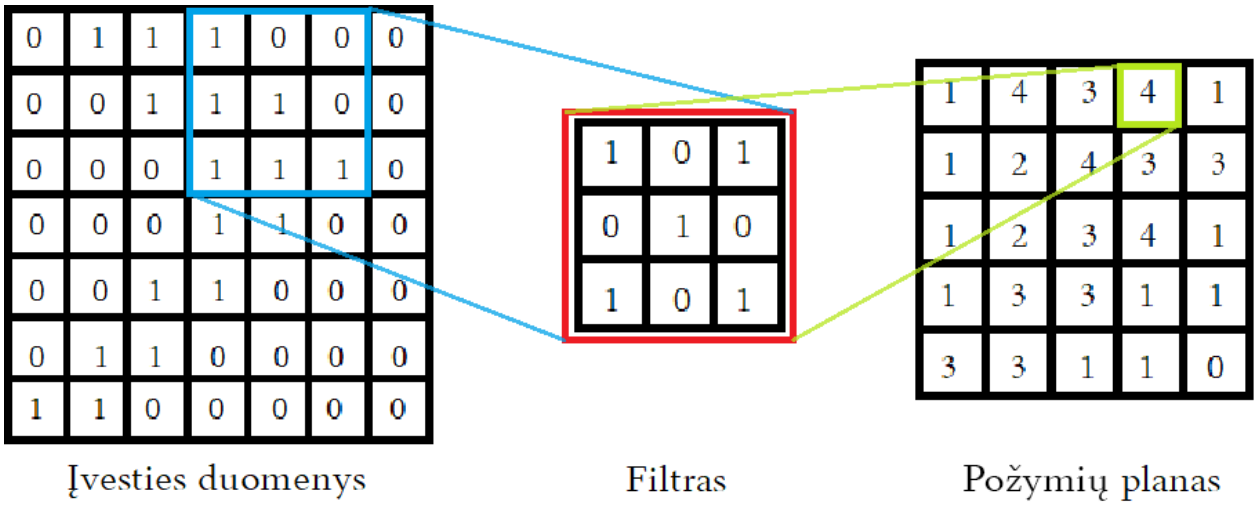
Objektų atpažinimas paveiksliukuose yra sudėtingas dėl šių iššūkių:

- Segmentavimas - paveiksliukai gali atvaizduoti įvairias scenas, kuriose gali būti pavaizduota daug objektų, kurie vienas kitą gali dalinai uždengti.
- Šviesa - pikselių intensyvumas gali būti paveiktas šviesos šaltinio ar paties objekto.
- Deformacija - objektai gali būti deformuoti įvairiais būdais, pavyzdžiui, žmogaus ranka parašyti skaičiai.
- Galimybės - objektų klasės dažnai nustatomos pagal tai, kaip patys objektai yra naudojami, pavyzdžiui, kėdės yra sukurti sėdėti objektai, tačiau jos gali būti skirtingo dizaino.
- 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, parodančią kaip reikia sujungti du informacijos rinkinius [PG17].

Pagal paveiksliuką (1 pav.) matyti, kad įvesties duomenys ir filtras, sudarytas iš svorių, yra pateikti 2D matricomis. Filtras juda nuo įvesties duomenų matricos kairės viršutinės dalies į dešinę, tada yra nuleidžiamas žemiau per vieną eilutę. 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 požymį [Epp17]. Pirmuose sluoksniuose šiuos filtrus galima apibūdinti kaip horizontalių, vertikalių ar įstrižų linijų filtrus, kurie sukuria paveiksliuko kraštų planą.



1 pav. Konvoliucijos veikimas

2.2. Konvoliucinio neuroninio tinklo sluoksniai

Konvoliuciniai neuroniniai tinklai yra sluoksnių rinkinys, kuris turi įvesties, vidinius ir išvesties sluoksnius. Tačiau vidiniai sluoksniai gali skirtis priklausomai nuo konvoliucinio neuroninio tinklo tipo. 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 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 tikimybę p ir $1 - p$. Dažniausiai yra nustatyta 50 procentų.

2.2.1. Konvoliucinis sluoksnis

Konvoliucinis sluoksnis (angl. convolutional layer) yra pagrindinis konvoliucinio neuroninio tinklo sluoksnis, kuris nustato visus paveikslėlio požymius. Kadangi įvesties informacija (paveikslėlis) yra didelės dimensijos, neefektyvu visus neuronus sujungti vienus su kitais, todėl neuronai yra sujungiami su lokaliu informacijos kiekiu, kuris yra lygus filtro dydžiui ir vadinamas erdviniu mastu (angl. receptive field) [Li15].

Neuronų kiekis po konvoliucijos (požymių 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 paveiksliuko mastą. Šis veiksmas yra atliekamas dėl to, kad būtų sumažintas parametrų ir skaičiavimų kiekis. 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 kurių 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 požymius, kurie buvo gauti iš prieš tai buvusių sluoksnių, nustatyti, kokioms klasėms priklauso įvesties paveiksliukas pagal mokymo informacijos imtį, kai neuroninio tinklo problema yra klasifikacija [Sin18]. Šiam sluoksniui yra priskiriama aktyvacijos funkcija, kuri neprivalo būti tokia pati kaip vidiniuose sluoksniuose naudota aktyvacijos funkcija.

2.3. Architektūros

Konvoliuciniai neuroniniai tinklai turi keletą skirtingų architektūrų, kurios naudojamos pagal sprendžiamą problemą. 2 lentelėje pateikta informacija 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
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

3. Duomenys ir jų problemos

Duomenys yra suskirstyti į tris pagrindinius tipus - struktūrizuoti, pusiau struktūrizuoti (angl. semi-structured) bei nestruktūrizuoti. Visų tipų duomenys yra apibūdinami dimensijomis, kurios parodo požymių kiekį duomenyse [KR18].

Struktūrizuoti duomenys yra tokie duomenys, kurie atitinka nustatytą schemą bei gali būti patalpinti reliacinėse duomenų bazėse ar lentelėse [SSS+09]. Jie turi aiškiai nurodytas dimensijas, pavyzdžiui, viena duomenų įeitis lentelėje yra eilutė, o lentelės stulpeliai suskirsto duomenų įeitį į skirtingas dimensijas, kurios gali būti kokybinės arba kiekybinės.

Pusiau struktūrizuoti duomenys yra hibridas tarp struktūrizuotų ir nestruktūrizuotų duomenų tipų - jie turi organizacinį karkasą, bet neturi pilnos struktūros, kuri yra reikalinga duomenis laikyti reliacinėje duomenų bazėje [CZY+16]. Tokie duomenys turi save apibūdinančią struktūrą, kuri turi žymas (angl. tag), kurios atskiria skirtingas duomenų esybes. Pusiau struktūrizuotų duomenų pavyzdžiai yra XML, JSON, el. laiškai ir t.t.

Duomenys, kurie neturi identifikuojamos struktūros, yra vadinami nestruktūrizuoti (paveiksliukai, garso įrašai, ir t.t.). Galima teigti, kad paveiksliukas yra masyvas, kurį sudaro skirtingos pikselių reikšmės, ir tai yra struktūra, tačiau iš tokio masyvo žmogus negali suprasti, kas yra pavaizduota paveiksliuke, dėl to tokie duomenys yra laikomi nestruktūrizuoti [Rao19]. Nustatyti požymių arba dimensijų kiekį nestruktūrizuotuose duomenyse yra sunku, kadangi paveiksliukai, tekstas ir kiti nestruktūrizuoti duomenys turi skirtingus vienetus - pikselis, žodis ir t.t. - todėl toliau bus nagrinėjami būtent paveiksliukai.

Dimensijų skaičius paveiksliukuose yra apibrėžiamas pikselių kiekiu [ZHW10], kitaip tariant, kiekvieno pikselio skaliaras yra atskira dimensija. Pavyzdžiui, juodai balta nuotrauka turi tiek pat dimensijų kiek pikselių, o spalvota nuotrauka (RŽM) turi tris kartus daugiau [Cel16] dėl to, kad kiekviena spalva atitinka vieną matricą.

Duomenys yra vadinami didelės dimensijos, jei jie turi mažą kiekį duomenų ir didesnį kiekį dimensijų [Sku14]. Tokie duomenys dažniausiai pasitaiko kaip medicininiai paveiksliukai, teksto atpažinimas, finansai ir chemometrija.

3.1. Dimensijų prakeiksmas

Terminas dimensijų prakeiksmas (angl. curse of dimensionality) buvo sugalvotas E.Bellman 1961 metais, kai jis nagrinėjo dinaminį programavimą (angl. dynamic programming)[Bel61]. Jis teigė, kad didinant dimensijų kiekį reikalingų duomenų skaičius didėja eksponentiškai, dėl to kyla

problemos su duomenų vizualizavimu ir tabuliavimu bei mašinų atminties reikalavimais. Tačiau šiais laikais duomenų analizės ir mašininio mokymosi sferoje dimensijų prakeiksmas sukelia kitą aktualią problemą - analizuojant didelės dimensijos duomenis didėja duomenų retumas. Jis yra gerai perteikiamas tuščios erdvės fenomenu (angl. empty space phenomenon). Jis parodo šiuos požymius [LV07]:

1. Sferos tūris su vieneto (angl. unit) spinduliu artėja link nulio, kai dimensijų skaičius didėja.
2. Tokio paties spindulio sferos ir kubo tūrių santykis artėja link nulio, kai didėja dimensijų kiekis, todėl galima teigti, kad kubo tūris koncentruojasi kraštuose, t.y. jis pasidaro spygliuotas.
3. Sferų su vienetiniu ir 1-ϵ spinduliais tūrių santykis artėja link vieneto, kai didėja dimensijų kiekis. Tai reiškia, kad beveik visas tūris kaupiasi išorinėje apvalkalo dalyje.

Šie požymiai leidžia teigti, kad didelės dimensijos erdvės yra daugiausia tuščios. Tai parodo, kad lokalios taškų grupės yra daugiausia tuščios, ir net tolygaus pasiskirstymo (angl. uniform distribution) atveju duomenys koncentruojasi tūrio kraštuose [VFS+03].

Duomenų analizės ir mašininio mokymosi atveju dirbant su dideliu dimensijų kiekiu reikia vis daugiau duomenų, norint tiriamą erdvę užpildyti. Tai sukelia problemų, pavyzdžiui, atliekant klasifikacijos uždavinį - žemesnėse dimensijose du duomenų taškai gali atrodyti arti vienas kito, tačiau didinant dimensijų kiekį jie tolsta vienas nuo kito.

3.2. Persimokymas

Modelis, kuris yra apmokomas su dideliu kiekiu duomenų gali nukentėti nuo persimokymo (angl. overfitting). Šiuo atveju modelis išmoksta duomenyse esantį triukšmą (angl. noise) ir duomenų netikslumus. Tai reiškia, kad apmokytas modelis neberanda bendrų pasikartojimų (angl. patterns). Kitaip tariant, permokytas modelis pasirodo puikiai mokymosi metu, tačiau labai prastai su naujais duomenimis. Taip yra todėl, kad modelis per daug įsimena mokymo metu naudotus duomenis ir nesugeba generalizuoti su naujais duomenimis.

Egzistuoja įvairių technikų kaip išvengti persimokymo [Web10]:

- Maža dispersija (angl. low variance) - tai reiškia, kad mokymosi algoritmas bus mažiau jautrus naudojamų duomenų specifikai (triukšmui, netikslumams, specifiniams pastebėjimams).
- Apkarpymas (angl. pruning) - kuris yra suskaidytas į išankstinį apkarpymą (angl. pre-pruning), kai geriausio modelio ieškojimas yra sustabdomas, kai pakankamai sudėtingas modelis yra rastas; kiti modeliai, kurie yra sudėtingesni, yra išmetami (nukirpti), ir po apkarpymo (angl. post-pruning), kai pirmiausia yra sukuriamas sudėtingas modelis, kuris atitinka

mokymo duomenis, ir tada yra apkarpomas tikslu jį supaprastinti [DB13].

- Normalizavimas (angl. regularization) - papildomų parametrų pridėjimas prie nuostolio funkcijos su tikslu bausti sudėtingus modelius [Liu17].
- Stabdymo kriterijai (angl. stopping criteria) - modelio mokymo užbaigimas, kai nuostolio funkcija su validavimo duomenimis pradeda didėti (generalizavimas pradeda blogėti) [NE19].
- Mažiausias aprašymo ilgis (angl. minimum description length) - pagal duomenų reguliarumą galima juos suspausti (duomenis apibūdinti su mažiau simbolių negu kad tai padaryti reikėtų pažodžiui), kas reiškia, kad kuo daugiau duomenys yra suspaudžiami, tuo daugiau apie juos yra išmokstama [Grü07].
- Mažiausias pranešimo ilgis (angl. minimum message length) - karkasas, kuris stengiasi paaiškinti duomenis pagal hipotezę arba geriausių hipotezių rinkinį, koduojant ir perduodant dviejų dalių žinutę, kurios pirmoje dalyje yra optimaliai užkoduota hipotezė, o kitoje - duomenys užkoduoti pagal hipotezę [Gun06].

4. Liapunovo teorijos

Aleksandr Mikhailovich Lyapunov prisidėjo prie keleto mokslo sričių - tai diferencialinės lygtys, potencialo teorija, dinaminės sistemos ir tikimybių teorija. Iš visų sričių jo didžiausias įdirbis buvo srityje apie pusiausvyros taško stabilumą ir mechaninių sistemų judėjimą. Tolimesniuose skyriuose bus apžvelgtos kelios jo sukurtos ar įkvėptos teorijos, ir kaip jos gali būti panaudojamos neuroninių tinklų kontekste.

4.1. Liapunovo funkcija

Skaliarinė funkcija, apibrėžta fazinėje erdvėje, kurioje yra atvaizduotos visos galimos sistemos būsenos, yra vadinama Liapunovo funkcija [Kho12]. Ji yra skirta pusiausvyros taško (angl. equilibrium point) stabilumo įrodymui.

Dinaminės sistemos pusiausvyros taškas yra būsenos kintamųjų reikšmė, kur būsenos kintamieji nesikeičia. Kitaip tariant, pusiausvyros taškas yra sprendinys, kuris nesikeičia su laiku [RAA67]. Taigi, jei sistema prasideda pusiausvyros taške, tai jos būsena amžinai išliks tame taške. Taip pat, jeigu dinaminė sistema yra apibūdinta diferencialine lygtimi arba jų sistema, tada pusiausvyros taškas gali būti apskaičiuotas nustatant išvestinę kaip nulį [BDH12].

Pusiausvyros taško stabilumas nustato, ar arti jo esantys sprendiniai liks prie pusiausvyros taško, tols nuo jo ar artės [VM14]. Pusiausvyros taškas gali turėti skirtingus stabilumo tipus [Far06]:

- Stabilus (angl. stable) - jei pradinės sąlygos, kurios prasideda arti pusiausvyros taško, išlieka šalia jo. Kitaip tariant, pusiausvyros taškas x_e yra stabilus, jeigu visi $\epsilon > 0$, tuomet egzistuoja $\delta > 0$ visiems $t > 0$:

$$\|x(0) - x_e\| < \delta \Rightarrow \|x(t) - x_e\| < \epsilon \tag{3}$$

  Toks apibrėžimas nereiškia, kad $x(t)$ artėja link pusiausvyros taško su laiku, tačiau kad išlieka netoliese jo.
- Nestabilus (angl. unstable) - jeigu pusiausvyros taškas nėra stabilus.
- Asimptotiškai stabilus (angl. asymptotically stable) - jeigu yra stabilus, bet taip pat $x(t) \rightarrow 0$ ir $t \rightarrow \infty$. Tai reiškia, kad arti esančios trajektorijos konverguoja į pusiausvyros tašką, kai laikas tampa didelis.

Nėra apibrėžtų tikslių procedūrų, kaip sukonstruoti Liapunovo funkciją [PPB09]. Tačiau galima įrodyti, jog funkcija yra Liapunovo funkcija, jei funkcija V turi minimumą x_e, tai yra $V(x) \geq 0$ su visais $x \in U$ ir $V(x) = 0 \Leftrightarrow x = x_e$, kur $V\colon U \rightarrow \mathbb{R}$. Bei $\dot{V}(x) \leq 0$ visiems $x \in U$, tai reiškia,

kad funkcija nedidėja palei sprendimo trajektoriją. Tačiau jeigu $\dot{V}(x) < 0$ visiems $x \in U$, tuomet funkcija griežtai mažėja ir tai reiškia, kad funkcija yra griežta Liapunovo funkcija [HG15].

4.2. Liapunovo eksponentė

Pagrindinis chaoso bruožas yra jautri priklausomybė nuo pradinės sąlygos. Dvi šalia viena kitos pradinės sąlygos esančios atraktoriuje (angl. attractor) yra atskirtos atstumo, kuris auga eksponentiškai su laiku. Skaičiuojant vidurkį palei trajektoriją yra gaunamas ilgalaikis nenuspėjamumas. Liapunovo eksponentė (angl. exponent) yra vidutinis šio atstumo augimo greitis. Šio dydžio reikšmės gali būti:

- Teigiamos - nestabilus ir chaotiškas, kitaip tariant, šalia esantys taškai, nesvarbu kaip arti vienas kito bebūtų, vis viena diverguos vienas nuo kito.
- Nulinė - periodiškas, tai reiškia, kad sistema yra kažkokioje pastovioje būsenoje.
- Neigiamos - stabilus, o tai lemia, kad tokios sistemos yra asimptotiškai stabilios ir kuo neigatyvesnė yra reikšmė tuo didesnis jos stabilumas.

Liapunovo eksponentės apskaičiavimas yra konceptualiai paprasta užduotis, nes reikia sekti dviejų iš pradžių arti esančių taškų trajektorijas ir pritaikyti jų atstumo logaritmą tiesinei laiko funkcijai. Šio pritaikymo nuolydis yra Liapunovo eksponentė. Tačiau vis viena egzistuoja keli sunkumai [Spr10]:

- Reikia pakankamai ilgai sekti trajektoriją, kad būtų įsitikinta, jog esama ant atraktoriaus prieš apskaičiuojant eksponentę.
- Pasirinktos pradinės sąlygos dviems trajektorijoms ne visada yra orientuotos į greičiausią plėtimąsi, t.y. pirmiausia atstumas gali mažėti, o tik po kažkiek laiko pradėti didėti.
- Atsiskyrimo greitis dažniausiai labai skiriasi pagal poziciją atraktoriuje, todėl yra svarbu gauti vidurį per labai ilgą laiką.
- Pradinės sąlygos turi pakankamai didelį atstumą, kad būtų išreikštos tiksliai pagal kažkokį tikslumą (angl. precision), taip kad nemažėtų į nulį ar nedidėtų iki reikšmingos atraktoriaus dalies dydžio.

4.3. Liapunovo spektras

Liapunovo spektras (angl. spectrum) apibūdina tiesinių diferencinių lygčių stabilumą, įskaitant stabilią, centrinę ir nestabilią suberdves [CKK+00]. Kitaip tariant, Liapunovo eksponenčių rinkinys, išdėstytas mažėjimo tvarka, yra Liapunovo spektras. Taip pat šis spektras charakterizuoja

chaotišką atraktorių, nes leidžia nustatyti jo dimensijas. Liapunovo spektrą yra sunku charakterizuoti, ir jis neturi logiškų tęstinumo savybių [HLN+98]. Jo apskaičiavimui yra reikalingi matricų manipuliacijos metodai, kurie greitai pasidaro per sudėtingi (skaičiavimo laiko ir atminties galimybėmis), nes orginalios sistemos parametrų skaičius tampa didelis [COH+99].

4.4. Liapunovo eksponentės apskaičiavimas

Bendrai Liapunovo eksponentę galima apskaičiuoti naudojant dinaminės sistemos judėjimo lygtį arba rekonstruojant atraktorių iš laiko eilučių. Tačiau šiame darbe nebus nagrinėjama, kaip apskaičiuoti Liapunovo eksponentę naudojant dinaminės sistemos judėjimo lygtį, kadangi ji yra labai retai žinoma bei analizuoti straipsniai šio būdo nenaudoja.

Taigi, šiame skyriuje bus pateikti algoritmai, kurie nustato Liapunovo eksponentes rekonstruojant atraktorių iš laiko eilučių - Wolf ir kiti [WSS+85], Eckmann ir kiti [EKC87]. Buvo pasirinkti būtent šie algoritmai, nes jie buvo naudoti nagrinėtuose straipsniuose.

4.4.1. Wolf algoritmas

Pirmasis algoritmas, kuris buvo sukurtas tikslu surasti didžiausią Liapunovo eksponentę iš laiko eilučių, yra Wolf ir kitų (toliau - Wolf) algoritmas. Šis algoritmas yra paremtas netoliese esančių trajektorijų nuo vienos atskaitos trajektorijos vidutinės divergacijos stebėjimu norint nustatyti didžiausią Liapunovo eksponentę [Ste18].

Wolf algoritmo žingsniai:

1. Tarkim *x(t)* yra skaliarinės laiko eilutės su ilgiu *N*, kur *t = 1, 2,...,N*. Pasirenkama įterpimo dimensija (angl. embedding dimension) *m* ir uždelsimo laikas (angl. delay time) τ būsenos erdvės rekonstrukcijai.
2. Pasirenkamas įterpimo taškas kaip pradinė sąlyga (toliau $x(t_0)$). Šis taškas yra uždelsimo vektorius ir atraktoriuje yra pateiktas $\{x(t), x(t+\tau),..., x(t+[m-1]\tau)\}$. Pagal šį vektorių yra sugeneruojama atskaitos trajektorija.
3. Randamas taškas (toliau $z_0(t_0)$), esantis artimiausiai įterpimo taško, kuris yra pateiktas kaip $\{x(t_0), x(t_0+\tau),..., x(t_0+[m-1]\tau)\}$ kitoje trajektorijoje. Jis yra randamas apskaičiuojant Euklidinį atstumą tarp visų taškų, esančių atraktoriuje, ir pasirenkant tašką, kuris yra mažiausiai nutolęs.
4. Apskaičiuojamas atstumas tarp abiejų taškų $L_0 = \|z_0(t_0) - x(t_0)\|$
5. „Einant toliau" trajektorijomis yra vėl apskaičiuojamas $L_0(i)$ iki tol, kol $L_0(i) > \epsilon$, kai ši

sąlyga yra tenkinama, tas $L_0(i)$ tampa L'_0 tam t_1 laikui.

6. Surandamas $z_1(t_1)$ taškas, kur L_{i} (atstumas tarp jų) ir kampinis atsiskyrimas tarp išsivysčiusio ir $z_1(t_1)$ taškų yra maži. Jeigu tinkamas $z_1(t_1)$ taškas nėra randamas, turimi taškai nėra keičiami.
7. Jeigu tinkamas $z_1(t_1)$ taškas yra rastas, tuomet algoritmo žingsniai (nuo 4 žingsnio) yra kartojami iki tol, kol pasibaigia duomenų dokumentas.

Pasiekus duomenų dokumento pabaigą galima apskaičiuoti didžiausią Liapunovo eksponentę (λ_1) su šia formule:

$$\lambda_1 \approx \frac{1}{N\Delta t} \sum_{1}^{M-1} \log_2 \frac{L'(i)}{L(i)} \tag{4}$$

kur N yra atskaitos trajektorijos x vystymosi žingsnių kiekis, o M - z taško keitimų kiekis bei Δt yra skirtumas tarp t_{i} ir t_0.

4.4.2. Eckmann algoritmas

Eckmann ir kitų (toliau - Eckmann) algoritmas leidžia surasti Liapunovo spektrą ir susideda iš trijų pagrindinių dalių - a) dinamika rekonstruojama baigtinių dimensijų erdvėje, b) rekonstruotos dinamikos liestinės atvaizdų gavimas naudojant mažiausių kvadratų metodą, c) surandamos Liapunovo eksponentės iš liestinių atvaizdų.

Turimas laiko eilutes galima apibrėžti kaip skaičių seką $x_1, x_2,...,x_N$. Galima teigti, kad laiko intervalas τ tarp matavimų yra fiksuotas taip, kad $x_i = x(i\tau)$.

Eckmann algoritmo a) dalies žingsniai:

1. Pasirenkama įterpimo dimensija d_{E} ir sukonstruojama orbita turinti d_{E} dimensijų, kuri pateikia sistemos laiko vystymąsi naudojant laiko uždelsimo metodą. Taip yra apibrėžiamas:

$$\overrightarrow{x_{\text{i}}} = (x_{\text{i}}, x_{\text{i+1}}, ..., x_{\text{id}_{\text{e}}\text{-1}}), \tag{5}$$

   kur $i = 1,2,..., N - d_{\text{e}} + 1$.
2. Nustatomi $\overrightarrow{x_{\text{i}}}$ kaimyniniai taškai $\overrightarrow{x_{\text{j}}}$, kurie yra išsidėstę orbitoje su tam tikru spinduliu r ir $\overrightarrow{x_{\text{i}}}$ esančiu centre, $\|\overrightarrow{x_{\text{j}}} - \overrightarrow{x_{\text{i}}}\| \leq r$. Kaimyninių taškų suradimui galima naudoti Euklidinę normą, tačiau algoritmo autoriai siūlo naudoti maksimumo normą (angl. max-norm), nes su ja galima greičiau surasti $\overrightarrow{x_{\text{j}}}$, kurie tenkintų ankstesniame sakinyje minėtą sąlygą.
3. x_i yra išrūšiuojami taip kad $x_{\prod(1)} \leq x_{\prod(2)} \leq \cdots \leq x_{\prod(N)}$ ir išsaugojami perstatymai $\prod$ ir jų inversija. Tuomet siekiant surasti x_i kaimyninius taškus pirmoje dimensijoje, yra žiūrima

į $k = \prod^{-1}(i)$ ir skanuojami $x_{\prod(s)}$, kur $s = k+1, k+2$,... iki tol, kol $x_{\prod(s)} - x_i > r$ ir taip pat su $s = k-1, k-2, ...$

4. Įterpimo dimensijai būnant $d_E > 1$, pirmiausia yra pasirenkamos s reikšmės, kurios $|x_{\prod(s)-x_i} \leq r|$ bei $|x_{\prod(s)+\alpha} - x_{i+\alpha}| \leq r$, kai $\alpha = 1, 2, ..., d_{E-1}$.

Atlikus šiuos žingsnius, dinaminė sistema buvo įterpta į d_E dimensijas. Tolesni žingsniai nusako Eckmann algoritmo b) dalį:

1. Matrica T_i ($d_\mathrm{E} \times d_\mathrm{E}$) yra gaunama žiūrinti į $\overrightarrow{x_\mathrm{i}}$ kaimynus $\overrightarrow{x_\mathrm{j}}$ ir:

$$T_i(\overrightarrow{x_\mathrm{j}} - \overrightarrow{x_\mathrm{i}}) \approx \overrightarrow{x_{\mathrm{j+1}}} - \overrightarrow{x_{\mathrm{i+1}}}. \tag{6}$$

Tačiau vektoriai $\overrightarrow{x_\mathrm{j}} - \overrightarrow{x_\mathrm{i}}$ gali neapimti R^{d_E}, tai reiškia, jog matrica T_i gali būti dalinai nustatyta. Šią problemą galima išspręsti leidžiant T_i būti $d_M \times d_M$, kur $d_M \leq d_E$, o atitinkamai yra laiko vystymasis iš $\overrightarrow{x_\mathrm{i}}$ į $\overrightarrow{x}_{\mathrm{i+m}}$. Daroma prielaida, kad egzistuoja toks skaičius $m \geq 1$, kad $d_E = (d_M - 1)m + 1$ ir asocijuojamas su $\overrightarrow{x_\mathrm{i}}$ vektoriumi, turinčiu d_M dimensijų: $\overrightarrow{x_\mathrm{i}} = (x_i, x_{i+m}, ..., x_{i+(d_M-1)m})$. Taigi, kai $m > 1$, (6) formulė yra pakeičiama:

$$T_i(\overrightarrow{x_\mathrm{j}} - \overrightarrow{x_\mathrm{i}}) \approx \overrightarrow{x_{\mathrm{j+m}}} - \overrightarrow{x_{\mathrm{i+m}}}. \tag{7}$$

2. Nustatant $S_i^E(r)$ kaip rinkinį $\overrightarrow{x_\mathrm{j}}$ kaimyninių taškų su indeksu j nuo $\overrightarrow{x_\mathrm{i}}$ centrinio taško su atstumu r, tuomet gauname a_k naudojant mažiausių kvadratų metodą.

Pabaigus b) dalies žingsnius yra gaunama matricų seka T_i, T_{i+m}, T_{i+2m},.... Paskutinėje Eckmann algoritmo dalyje yra nustatomos stačiakampės matricos $Q_{(j)}$ ir viršutinių trikampių matricos $R_{(j)}$ su teigiamais įstrižainės elementais taip, kad $Q_{(0)}$ yra vieneto matrica ir

$$\begin{aligned} T_1 Q_{(0)} &= Q_{(1)} R_{(1)}, \\ T_{1+m} Q_{(1)} &= Q_{(2)} R_{(2)}, \\ &..., \\ T_{1+jm} Q_{(j)} &= Q_{(j+1)} R_{(j+1)}, \\ &... \end{aligned} \tag{8}$$

Tuomet Liapunovo eksponentes galima apskaičiuoti su šia formule:

$$\lambda_k m = \frac{1}{\tau K} \sum_{j=0}^{K-1} \ln R_{(j)kk}, \tag{9}$$

kur $K \leq (N - d_M m - 1)/m$ yra turimų matricų kiekis, o τ yra atrankos laiko žingsnis.

4.5. Liapunovo eksponentės panaudojimas

Buvo analizuoti keturi straipsniai, kurių duomenys buvo elektroencefalografijos (toliau EEG) signalai [ÜG07], hiperspektriniai paveiksliukai [YGJ12], priešiški trukdžiai (angl. adversarial perturbation) paveiksliukams [PDW18] ir [PDB20].

Pirmasis straipsnis neįvardino tikslaus naudoto algoritmo, tik kad Liapunovo eksponentės buvo apskaičiuotos iš laiko eilučių ir buvo naudotas programos „MatLab" paketas. Tad, šiame straipsnyje EEG signalai (sudaryti iš 256 atskirų duomenų) buvo panaudoti kaip laiko eilutės. Vienam EEG segmentui buvo gauta 128 Liapunovo eksponentės, siekiant sumažinti dimensionalumą modelio apmokymui buvo naudoti statistiniai požymiai kiekvienam EEG segmentui - visų Liapunovo eksponenčių vidurkis, maksimali Liapunovo eksponentės reikšmė, Liapunovo eksponenčių vidutinė galia, Liapunovo eksponenčių standartinė deviacija ir Liapunovo eksponenčių pasiskirstymo iškraipymas. Šie požymiai buvo naudoti EEG signalų klasifikacijai. Liapunovo eksponentės puikiai apibūdino EEG signalų chaotiškumą turint tris skirtingas klases, pasiekiant 96.33 procentų tikslumą.

Antrame analizuotame straipsnyje, kur duomenų tipas buvo hiperspektriniai paveiksliukai, buvo naudotas Wolf algoritmas (daugiau skyriuje „Wolf algoritmas") siekiant nustatyti didžiausią Liapunovo eksponentę. Kiekvieno pikselio hiperspektrinės atspindžio kreivės (arba spektras) buvo laikytos kaip laiko eilutės, kadangi hiperspektrinių duomenų atspindžio kreivių pasikartojimai turėtų būti panašūs tarp tokių pačių medžiagų. Taigi, duomenų dimensionalumas buvo sutrauktas nuo kelių šimtų iki dviejų - didžiausios Liapunovo eksponentės ir Hurst eksponentės (pastaroji šiame darbe nebus analizuojama). Naudojant šiuos du požymius yra gaunamas didžiausias tikslumas palyginus su kitais požymių išgavimo metodais.

Trečias analizuotas straipsnis, kuriame buvo tiriami priešiški trukdžiai paveiksliukams, buvo naudotas Eckmann algoritmas (daugiau skyriuje „Eckmann algoritmas") tikslu nustatyti Liapunovo eksponentes. Straipsnio autorių eksperimentuose buvo pastebėta, jog didinant priešiškų trukdžių triukšmą, auga teigiamų Liapunovo eksponenčių vyravimas. Taigi, prieš pateikiant duomenis konvoliuciniam neuroninam tinklui, galima patikrinti tų duomenų originalumą jiems apskaičiuojant Liapunovo eksponentes. Naudojami paveiksliukai buvo vektorizuoti (suplokštinti), kad jie atitiktų laiko eilutes, ir buvo indeksuoti pagal pikselio vietą. Toks Liapunovo eksponentės panaudojimas buvo patikrintas su MNIST (juodai balti ranka parašytų skaičių paveiksliukai) ir mados MNIST

(juodai balti dešimties drabužių tipų paveiksliukai) duomenų rinkiniais bei įvairiomis priešiškų trukdžių atakomis - Carlini Wagner, FGSM, JSMA, Madry ir kiti bei DeepFool - ir jų visų, išskyrus JSMA atakas, pavyko nustatyti su aukštu tikslumu. Šio autoriaus bandymuose baltai juodi paveiksliukai buvo suplokštinti ir jiems apskaičiuojamos keturios Liapunovo eksponentės (Liapunovo spektras). Bandymuose su MNIST duomenų rinkiniu originalūs paveiksliukai (nepaveikti priešiškų trukdžių) neturėjo teigiamų Liapunovo eksponenčių, tačiau priešiškų trukdžių paveikti paveiksliukai turėjo bent vieną teigiamą Liapunovo eksponentę, o tai yra stiprus chaoso įrodymas. Autorius taip pat pateikė Liapunovo eksponentes, gautas iš mados MNIST duomenų rinkinio, tačiau nepadarė jokių išvadų apie juos, nors gauti rezultatai parodė, jog net originalus paveiksliukas turi teigiamas Liapunovo eksponentes, ir kai kurie paveiksliukai, paveikti priešiškų trukdžių, turi mažesnį kiekį teigiamų Liapunovo eksponenčių nei originalūs.

Paskutinis analizuotas darbas, kaip ir priešpaskutinis straipsnis, analizavo priešiškų trukdžių paveiktų paveiksliukų klasifikavimą ir Liapunovo eksponenčių apskaičiavimui naudojo Eckmann algoritmą. Šiame darbe buvo naudoti duomenų rinkiniai - MNIST, mados MNIST ir CIFAR-10 (sumažinto dydžio paveiksliukai iš ImageNet duomenų rinkinio, tačiau straipsnyje spalvoti paveiksliukai yra konvertuoti į juodai baltus). Išvardinti rinkiniai buvo atakuoti su Carlini Wagner, FGSM, JSMA, DeepFool, BIM, EAD, Virtual, Madry ir kiti, Spatial, HopSkip bei Sparse metodais, kai atakuojamas tinklas yra LeNet. Taip pat šio straipsnio autoriai analizavo entropijos (angl. entropy) panaudojamumą priešiškų trukdžių paveiktų paveiksliukų nustatymui. Gautose išvadose buvo nustatyta, jog entropijos ir Liapunovo eksponenčių kombinacija gauna geresnius rezultatus negu kiekvienas metodas atskirai.

4.6. Liapunovo funkcijos panaudojimas

Neuroninių tinklų kontekste Liapunovo funkcijos sudarymas gali būti paremtas neuroninio tinklo mokymosi klaidų būsenų mažinimu [Tay06]. Iš esmės svorių keitimas sklidimo atgal algoritmu turi užtikrinti tinklo stabilumą, ir tai galima įrodyti su Liapunovo funkcija [Gra13]. Kitaip tariant, dinaminė sistema, pavyzdžiui neuroninis tinklas, turinti Liapunovo funkciją, įrodo, kad funkcija konverguoja į pusiausvyros tašką. Taip pat Liapunovo funkcija padeda nustatyti, kur yra pusiausvyros taškas - funkcijos lokaliame minimume [SRM96].

Šiame darbe buvo analizuoti du straipsniai [BKP06] ir [Rav15], kurie panaudojo Liapunovo funkciją neuroninių tinklų kontekste.

Pirmasis straipsnis ([BKP06]) apibrėžė du skirtingus algoritmus, tačiau šiame darbe bus ana-

lizuojamas ir naudojamas tik vienas iš jų, kadangi jo nuostolio funkcija buvo tokia pati, kaip ir antrojo straipsnio. Šio algoritmo pavadinimas yra „Liapunovo function (LF I) based learning algorithm“ (toliau LF I). Šio algoritmo esmė yra sklidimo atgal algoritmo fiksuoto mokymosi žingsnio pakeitimas su prisitaikančiu mokymosi žingsniu (angl. adaptive learning rate), kuris yra išvedamas naudojant Liapunovo funkciją. Taigi, bendrai naudota nuostolio funkcija yra:

$$C = \frac{1}{2} e^{\mathrm{T}} e, \quad e = \widehat{y} - y. \tag{10}$$

Formulės (10) parametras $\widehat{y}$ yra teisingas atsakymas, o y yra tinklo spėjimas. Ši nuostolio funkcija yra vidutinio kvadratinio nuokrypio funkcija. Pirmojo straipsnio prisitaikantis mokymosi žingsnis, kuris naudojamas LF I algoritme, yra apibrėžtas kaip (originaliame straipsnyje formulė (13)):

$$\eta = \mu \frac{\|e\|^2}{\|J_{\mathrm{p}}{}^{\mathrm{T}} e\|^2}. \tag{11}$$

Formulės (11) parametras μ yra konstanta, kuri pasirenkama euristiškai. Šio straipsnio autorių daromuose bandymuose buvo nustatyta, jog ši konstanta turi būti tarp 0.2 ir 0.8 reikšmių. Kitas formulės (11) parametras e yra taip pat apibrėžiamas kaip ir formulėje (10), o J_{p} yra apskaičiuojamas pagal $\partial y / \partial W$, kur y yra tinklo spėjimas, o W yra svorių vektorius.

Svorių atnaujinimo formulė šiame straipsnyje buvo gauta pagal Liapunovo funkcijos (formulė (10)) laiko išvestinę. Tuomet pagal svorių atnaujinimo algoritmą buvo sudaryta svorių skirtumo lygtis, kuri buvo palyginta su sklidimo atgal algoritmu paremtu gradiento nusileidimo metodu, ir buvo pastebėta, jog sklidimo atgal algoritmo fiksuotas mokymosi žingsnis yra pakeistas formule (11).

Šis algoritmas straipsnyje buvo išbandytas su dviejų sluoksnių tiesioginio sklidimo tinklu ir sprendžiami paprasti uždaviniai kaip kad XOR uždavinys.

Antrasis straipsnis [Rav15], kuris buvo parašytas 2015 metais, bandė patobulinti prieš tai aprašytą LF I algoritmą, kadangi jis priklausė nuo nuostolio funkcijos reikšmės ir jos išvestinės. Straipsnio autorius norėjo apibrėžti algoritmą (toliau patobulintas LF I) su prisitaikančiu mokymosi žingsnis, kuris nepriklausytų nuo nuostolio funkcijos, siekiant užtikrinti, kad algoritmo veikimas neblogėtų artėjant prie sprendimo.

Šio straipsnio naudojama nuostolio funkcija yra tokia pati, kaip ir buvo apibrėžta (10) formulėje. O sukurta mokymosi žingsnio apskaičiavimo formulė buvo (originaliame straipsnyje formulė aprašyta 3.2. pastaboje):

$$\eta = \frac{\alpha_0}{2E[\nabla^2{}_{\mathrm{W}}]_{\mathrm{M}} E[\nabla^2{}_{\mathrm{w}}]_{\mathrm{P}}}. \tag{12}$$

Formulėje (12) pateiktas α_0 parametras priklauso [0, 1], kai norima pasiekti stabilumą nuostolio funkcijos mažinime, kur ∇_{W} yra funkcijos gradientas pagal W (paskutinio sluoksnio svorius) ir atitinkamai ∇_{w} yra funkcijos gradientas pagal w (likę svoriai atmetus paskutinio sluoksnio svorius). Bei $E[]_{\mathrm{M}}$ ir $E[]_{\mathrm{P}}$ yra tikėtinos reikšmės su P (mokymui pateiktų duomenų kiekis) ir M (paskutinio sluoksnio svorių kiekis) rinkinių elementų skaičiumi.

Formulė (10) buvo įvardinta kaip Liapunovo funkcija ir buvo apibrėžtas pasikeitimas nuostolio funkcijoje. Tuomet svorių atnaujinimo formulė buvo parinkta pagal sklidimo atgal algoritmą, ir pagal ją atnaujintas nuostolio funkcijos pasikeitimo apskaičiavimas. Tai leido padaryti išvadą, jog funkcija yra Liapunovo funkcija.

Šiame straipsnyje [Rav15] buvo pateikti du bandymai - neuroninių tinklų modeliavimo ir klasifikacijos uždaviniai. Naudotas neuroninis tinklas turėjo du vidinius sluoksnius su 100-200 svorių konfigūracija. Kadangi mokymosi žingsnio formulė (formulė (12)) buvo maža to darbo dalis, dėl to bandymų rezultatuose tiksliai neapibrėžta, jog jų metu formulė buvo naudota.

Taigi, abiejuose straipsniuose buvo apibrėžtos adaptyvaus mokymosi žingsnio formulės, kurios turi būti naudojamos su specifine nuostolio funkcija bei naudojant gradiento nusileidimo metodą.

5. Duomenų rinkinys

Pasirinktas duomenų rinkinys buvo sudarytas iš „DAmangeNet“ [CHH+19] ir „ImageNet“ rinkinių (daugiau skaityti „ImageNet“). „DAmangeNet“ duomenų rinkinys yra sudarytas iš paveiksliukų, kurie buvo sugeneruoti iš „ImageNet“ duomenų rinkinio paveiksliukų, paveiktų priešiškų trukdžių (angl. adversarial attack).

Duomenys, paveikti priešiškų trukdžių, yra skirti suklaidinti neuroninį tinklą. Priešiški trukdžiai yra maži, dažnai nepastebimi pakeitimai normaliuose duomenyse [BA18], pavyzdžiui, paveiksliukuose. Pateikus tokius duomenis apmokytam modeliui, jis padaro neteisingą spėjimą su labai dideliu užtikrintumu. Tačiau pateikus tuos pačius duomenis žmogui, jis atpažįsta juos teisingai. Ši problema yra labai aktuali, kadangi naudojamo modelio saugumas negali būti užtikrinamas.

Egzistuojantys priešiški trukdžiai gali būti kategorizuoti į baltos dėžės (angl. white-box), pilkos dėžės (angl. gray-box) ir juodos dėžės (angl. black-box) [RZQ+20]. Skirtumas tarp šių kategorijų yra žinios, kurias turi asmenys, norintys panaudoti priešiškus trukdžius prieš modelį. Baltos dėžės trukdžiuose yra manoma, kad tie asmenys turi visą informaciją apie modelį - jo architektūrą ir parametrus. Manoma, jog asmenys, naudojantys pilkos dėžės kategorijos trukdžius, turi informaciją apie modelio struktūrą. O juodos dėžės kategorijos - turi tik informaciją iš užklausų (angl. query), kai trukdžius kuriantis asmuo pateikia modeliui duomenis ir gauna rezultatą ir pagal tai sukuria priešiškus trukdžius. Šiame darbe naudojamas duomenų rinkinys „DAmangeNet“ jo autorių laikomas kaip nulio užklausų (angl. zero-query), kadangi priešiškų trukdžių rinkiniui sukurti nebuvo naudojamas specifinis modelis.

Sukurti apsisaugojimo būdai nuo priešiškų trukdžių gali būti išskirti į dvi grupes - apsisaugojimas nuo priešiškų trukdžių (angl. adversarial defense) ir priešiškų trukdžių nustatymas (angl. adversarial detection) [AD19]. Vienas iš pirmosios grupės būdų yra priešiškas mokymas (angl. adversarial training), jo tikslas yra mokymo metu inkorporuoti duomenis, paveiktus priešiškų trukdžių. Tačiau šis būdas apsaugo tik nuo apmokymo metu naudotų priešiškų trukdžių [SKC18]. Antros grupės būdai bando nustatyti duomenis, paveiktus priešiškų trukdžių, prieš jiems patenkant į modelį. Tai yra padaroma su modeliu, kuris yra apmokytas su originaliais ir paveiktais duomenimis, ir jis yra panaudojamas nustatymui, ar įeinantys duomenys yra paveikti priešiško trukdžio, ar ne.

Taigi, sudarytas duomenų rinkinys savyje turėjo dvi klases - priešiškų trukdžių paveiktus paveiksliukus ir originalius „ImageNet“ paveiksliukus. Abiejų klasių paveiksliukų pavyzdžiai yra pateikti 2 paveiksliuke. Duomenų rinkinio paveiksliukai yra spalvoti (raudona, žalia ir mėlyna) bei

200 x 200 dydžio. Mokymo metu kiekviena klasė turėjo po 400 paveiksliukų, o testavimo metu turėjo 100.

2 pav. Duomenų rinkinio pavyzdys. Kairėje originalus paveiksliukas, o dešinėje - paveiktas priešiškų trukdžių (paveiksliukų pav. „ILSVRC2012_val_00000003.png“).

Šis duomenų rinkinys buvo pasirinktas, nes yra sudėtingesnis - daugiau dimensijų turintis (spalvotas ir didesnės rezoliucijos paveiksliukas) - negu kad [PDW18] straipsnyje naudotas MNIST ir mados MNIST duomenų rinkiniai (daugiau „Liapunovo eksponenčių panaudojimas“).

6. Vertinimo metrikos

Šiame skyriuje apibrėžtos bendros neuroninių tinklų vertinimo metrikos, kurios buvo naudotos eksperimentuose.

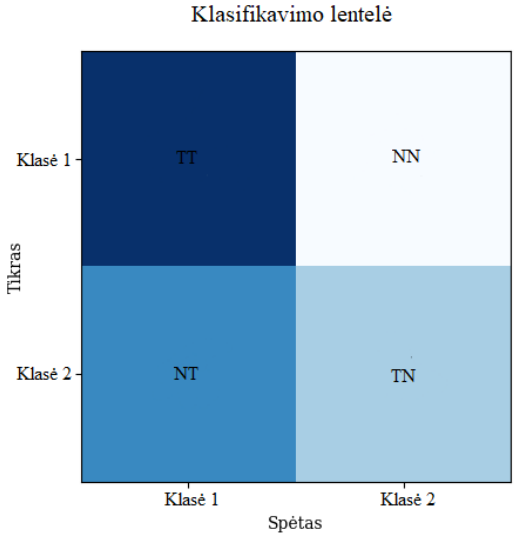
6.1. Klasifikavimo lentelė

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

Klasifikavimo lentelėje 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.

Šių sąvokų sutrumpinimai yra pavaizduoti 3 paveiksliuke.



3 pav. Klasifikavimo lentelės pavyzdys

Jeigu modelis įvardija dalį klasės 1 paveiksliukų kaip klasę 1, tai tų paveiksliukų kiekis yra įrašomas į TT lentelės laukelį. Tačiau jei modelis klasės 1 paveiksliukus įvardina kaip klasę 2, tai tas paveiksliukų skaičius yra įrašomas į NT laukelį. Tas pat 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.

6.2. Tikslumas

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

$$tikslumas = \frac{TT + TN}{TT + TN + NT + NN}. \tag{13}$$

Formulės (13) 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 nevisiškai atskleidžia modelio kokybę. Pavyzdžiui, kačių nuotraukų yra 9, o šunų nuotraukų yra 91, modelis teisingai atpažįsta 1 katę ir 90 šunų, tikslumas bus 0.91, nors tik viena katė buvo atpažinta.

6.3. Atšaukimas

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

$$atšaukimas = \frac{TT}{TT + NN}. \tag{14}$$

Formulės (14) 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).

6.4. Precizija

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

$$precizija = \frac{TT}{TT + NT}. \tag{15}$$

Formulės (15) 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 tiesų yra teigiama (mažas NT).

6.5. F-balas

F-balas (angl. F-score) gali būti matomas kaip svertinis precizijos ir atšaukimo vidurkis, taigi, ši metrika atsižvelgia kartu į netikrus teigiamus ir netikrus neigiamus spėjimus:

$$F\text{-}balas = 2 * \frac{atšaukimas * precizija}{atšaukimas + precizija}. \tag{16}$$

Formulės (16) atšaukimo ir precizijos parametrai yra apibrėžti anksčiau buvusiuose skyriuose „Atšaukimas“ ir „Precizija“. Ši metrika yra itin naudinga, jeigu duomenų kiekis kiekvienoje klasėje nėra vienodas. Jeigu F-balas pasiekia vieneto reikšmę, tuomet yra gaunamas geriausias rezultatas, o kai pasiekia nulį - blogiausias.

7. Technologijos

Šiame skyriuje pateiktos populiariausios šių laikų technologijos bei glaustai apibrėžti jų pagrindiniai funkcionalumai.

7.1. ImageNet

Projektas „ImageNet“ buvo sudarytas profesorės Li Fei-Fei 2009 metais. Projekto tikslas buvo sukurti didelę sukategorizuotų paveiksliukų ir jų etikečių duomenų bazę, kuri būtų skirta vizualinio objektų atpažinimo programinės įrangos tyrimams. Ši duomenų bazė 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 rengia konkursą, vadinamą „ImageNet Large Scale Visual Recognition Challenge“ (trumpinys ILSVRC). Konkurso užduotis yra išmokyti 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[RDS+15].

7.2. Keras

„Keras“ yra aukšto lygio programų sąsaja, skirta neuroniniams tinklams. Sąsaja parašyta „Python“ programavimo kalba ir vidinėje pusėje gali 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.

7.3. 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, taigi, yra aktyviai palaikomas automatinis ir gilusis mokymasis, tačiau dėl bibliotekos ir skaičiavimų lankstumo yra naudojamas įvairiose mokslinėse srityse.

7.4. 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 specifinei 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 išmoksta požymių, panašių į Gaboro filtrą (tiesinis filtras, naudojamas tekstūroms analizuoti) ir spalvų dėmes. Šie pirmojo sluoksnio požymiai nepriklauso nuo duomenų rinkinio, bet yra bendri ir tinkami daugeliui duomenų rinkinių ir užduočių [YCB+14].

8. Eksperimentas

Šio darbo eksperimentas buvo suskaidytas į dvi dalis - programų sukūrimą ir tinklų suderinimą. Buvo sukurtos šios programos:

1. Konvoliucinis neuroninis tinklas praplėstas su algoritmu, kurio nuostolio funkcija yra griežta Liapunovo funkcija.
2. Paveiksliukų duomenų rinkinio konvertavimas į Liapunovo eksponentes.
3. Sukurtas pradinis tinklas pritaikytas būti apmokytu su Liapunovo eksponentėmis.

Prieš gaunant rezultatus su pasirinktu duomenų rinkiniu sukurtos programos buvo testuojamos su paprastesniu išskirtinio arba (angl. XOR, toliau XOR) uždaviniu ir su paprastesniais duomenų rinkiniais - MNIST ir CIFAR.

Taigi, XOR uždavinys yra klasifikacijos uždavinys, kai neuroniniui tinklui yra pateikiami du dvejetainiai įeities duomenys ir norima gauti spėjimą, kuris atitiktų XOR logiką. Šis uždavinys yra itin tinkamas neuroninio tinklo testavimui, kadangi duomenys yra labai paprasti ir funkcija yra netiesinė. Tad, jeigu neuroninis tinklas pasiekia aukštą tikslumą sprendžiant šį uždavinį, galima teigti, jog tinklas sugebėjo nubrėžti netiesines sprendimo ribas.

MNIST ir CIFAR duomenų rinkiniai buvo atakuoti naudojant Carlini Wagner, PGD, FGSM, Madry ir kiti metodais, norint užtikrinti kad parašyta programa gauna panašius rezultatus, kaip ir [PDW18] ir [PDB20] straipsniuose, tačiau šiame darbe paveiksliukų atakavimo metodai nebus nagrinėjami.

Šio eksperimento pirmos programos tikslas buvo pritaikyti „Liapunovo funkcijos panaudojimas“ skyriuje pateiktus algoritmus sukurtam neuroniniam tinklui ir jį suderinti su pasirinktu duomenų rinkiniu. Tokiu būtu įrodant, jog naudojant šiuos algoritmus galima užtikrinti modelio konvergavimą ir jo nepersimokymą. Sukurtų algoritmų palyginimui buvo naudota Adam optimizacijos funkcija, kuri yra viena iš labiausiai naudojamų giliuosiuose mokymuose [Soy20]. Adam algoritmas apskaičiuoja prisitaikančio mokymosi žingsnio reikšmes kiekvienam tinklo parametrui. Taigi, naujas parametras yra apskaičiuojamas pagal funkciją (atnaujinant vieną tinklo parametrą):

$$w_{t+1} = w_t - \eta * \frac{v_t}{\sqrt{s_t} + \epsilon} * g_t. \tag{17}$$

Kurioje η yra pradinis mokymosi žingsnis, v_t yra eksponentiškai nykstantis gradiento vidurkis palei w laiku t, s_t - eksponentiškai nykstantis kvadratu pakeltų gradientų vidurkis palei w laiku t, o g_t yra gradientas palei w laiku t. ϵ yra labai mažas skaičius užtikrinti, kad nebūtų dalybos iš nulio. Bandymuose buvo naudota „TensorFlow“ bibliotekoje apibrėžta optimizacijos funkcija

„tf.compat.v1.train.AdamOptimizer“, kurios mokymosi žingsnis buvo keičiamas. Tad, hiperparametrai β_1 ir β_2 yra nustatyti 0.9 ir 0.999 atitinkamai.

Remiantis skyriumi „Liapunovo eksponentės panaudojimas“ likusių dviejų programų tikslas buvo įrodyti, jog galima Liapunovo eksponentes panaudoti kaip dimensijų mažinimo žingsnį ir taip užtikrinti, kad modelis nebus paveiktas dimensijų prakeiksmo.

Straipsnyje [PDW18] priešiškų trukdžių atakuotų paveiksliukų gavimui yra naudojama „CleverHans“ biblioteka [PFC+18], o apskaičiuoti Liapunovo eksponentes yra naudojama „nolds“ biblioteka [Sch19].

8.1. Naudojamos vertinimo metrikos

Eksperimentuose buvo naudojama klasifikavimo lentelė, tikslumo metrikos diagrama, mokymo metu gautų nuostolio funkcijos ir mokymo žingsnio grafikas, atšaukimo, precizijos ir F-balo reikšmės, kurios buvo apibrėžtos skyriuje „Vertinimo metrikos“. Svarbu pabrėžti, kad mokymo metu pasiektos nuostolio funkcijos ir mokymo žingsnio reikšmės yra pateiktos po kiekvienos partijos, o ne po epochos, kad pamatytume tikslias reikšmes, o ne jų vidurkius.

Pagal [BKP06] straipsnį buvo nuspręsta lyginti jų sukurtą algoritmą (LF I) su tomis pačiomis metrikomis, tai - konvergavimo laikas, kuris yra lygus epochų kiekiui, reikalingam pasiekti specifinę nuostolio funkcijos reikšmę (10^{-4}), ir konvergavimo laiko vidurkis, apskaičiuojamas penkiasdešimčiai skirtingų tinklo apmokymų. Ši informacija bus pateikta kaip lentelė su algoritmo pavadinimu, epochų skaičiumi bei laiku. Taip pat bus pateiktos diagramos su mokymo epochų ir skirtingų mokymų kiekiu bei konvergavimo laiku ir skirtingų mokymų kiekiu skirtingiems algoritmams. Tuo tarpu straipsnyje [Rav15] klasifikacijos bandyme buvo pateikta tik testavimo nuostolio funkcijos reikšmių intervalas, tačiau šiame darbe bus pateiktos nuostolio funkcijos diagramos.

Straipsniai [YGJ12] ir [ÜG07] naudojo tikslumą, kad parodytų jų sprendimo tinkamumą.

8.2. Programų 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“. Kompiuteris, naudotas eksperimentui, turėjo Nvidia 1070 Ti ir Intel Core i7-5820K procesorių.

Pirmajai eksperimento daliai buvo sukurti du modeliai: kelių vidinių sluoksnių ir praplėstas VGG. Taigi, pirmasis modelis buvo skirtas XOR uždaviniui spręsti turintis du vidinius sluoksnius (nustatytus kaip tankumo (angl. dense) su ReLU aktyvacijos funkcija bei neuronų kiekis, nustatytas 16) ir paskutinis sluoksnis - tankumo su sigmoidine aktyvacijos funkcija. Vienos partijos (angl. batch) dydis buvo nustatytas keturiems.

Antrasis modelis buvo VGG, kuris buvo naudojamas iš „Keras“ pateiktų modelių, apmokytų su „ImageNet“. Importavimo metu padaryti nustatymai - pilnai sujungtas sluoksnis nepridedamas, kadangi galima parinkti, kokių dimensijų paveiksliukai naudojami ir kiek spalvų sluoksnių jie turi (spalvoti RGB paveiksliukai turi 3). Tuomet reikėjo nustatyti kiek importuoto modelio sluoksnių bus nenorima mokinti; šiame darbe buvo pasirinkta - 5. Po šių egzistuojančio modelio paruošimų reikėjo sukurti naują „Kero“ modelį, prie kurio buvo pridėtas paruoštas egzistuojantis modelis bei pridėti keli kiti sluoksniai tam tikru išsidėstymu - plokštinimo (angl. flatten), tankumo, plokštinimo, tankumo bei vėl tankumo. Pirmi du tankumo sluoksniai turėjo aktyvacijos funkciją ReLU, o paskutinis - sigmoidinę. Geriausias partijos dydis naudotam kompiuteriui buvo mokymosi partijai 20 paveiksliukai, o validacijos - 10 paveiksliukų. Klasių tipas buvo nustatytas į binarinį, nes duomenų rinkinys buvo sudarytas iš dviejų klasių - originalūs ir paveikti priešiško trukdžio paveiksliukai.

Po modelių paruošimo buvo sukurtas specialus mokymosi ciklas, siekiant kad būtų įmanoma naudoti specialią nuostolio funkciją bei pagal algoritmus keisti ir pritaikyti mokymosi žingsnį tinklo mokymo metu. Nuostolio funkcija buvo apibrėžta kaip „Python“ metodas, o optimizacijos funkcija buvo nustatyta kaip „TensorFlow“ „GradientDescentOptimizer“ klasė bei abi šios funkcijos priskirtos tinklui. Specialiam mokymosi ciklui sukurti buvo apibrėžti „TensorFlow“ metodai „train_step“ ir „test_step“. Gradientų gavimui buvo naudojamas „TensorFlow“ „GradientTape“ klasė. Metode „train_step“ buvo apskaičiuojamas mokymosi žingsnis ir priskiriamas optimizacijos funkcijai prieš pritaikant gradientus.

Kita programa, skirta Liapunovo eksponenčių apskaičiavimui, buvo atskirta nuo modelio, kadangi Liapunovo eksponenčių apskaičiavimas užtrunka ilgai, ir apskaičiuotos Liapunovo eksponentės yra deterministinės, todėl neverta jas vis iš naujo skaičiuoti. Ši programa atidaro paveiksliuko failą naudojant „Pillow“ biblioteką [vKwM+21], nuskaito visus pikselius kaip vienos dimensijos masyvą (kiekvienas spalvos kanalas yra atskirame masyve). Gauti masyvai yra pateikiami „lyap_e“ funkcijai, kuri yra aprašyta „nolds“ bibliotekoje [Sch19]. Gautos Liapunovo eksponentės ir atitinkami paveiksliukų pavadinimai yra išsaugojami kableliais atskirtos vertės (angl. csv) dokumente.

Paskutinė programa turi į pirmąją programą panašią struktūrą, tačiau ji yra pritaikyta priimti struktūrizuotus duomenis. Taigi, teko pakeisti duomenų skaitymą, kad būtų galima priimti duome-

nis iš kableliais atskirtos vertės dokumento. Taip pat buvo pakeistas pats modelis - buvo sukurtas negilus tinklas su penkiais tankumo sluoksniais. Visų tankumo sluoksnių aktyvacijos funkcija yra ReLU, išskyrus paskutinį, kurio aktyvacijos funkcija yra sigmoidinė.

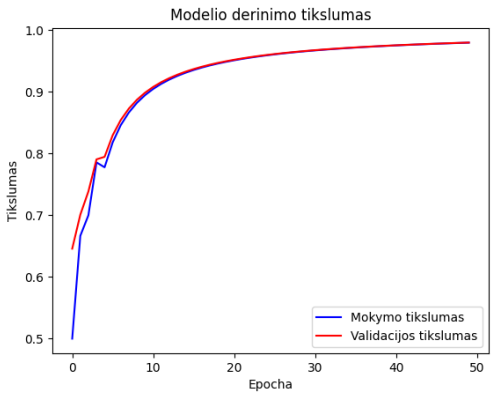
8.3. Eksperimento bandymai

Sukūrus reikalingas programas buvo pradėti bandymai. Visuose pateiktuose rezultatuose tikslumo reikšmės yra galimos nuo nulio iki vieneto, kur vienetas yra geriausias tikslumas, o nuostolio funkcijos reikšmė turėtų artėti link nulio. Neuroninio tinklo derinimo nuostolio funkcijos grafikai yra pateikti skyriuje „Priedas nr. 1“. Eksperimentas suskaidytas į kelias dalis - Liapunovo funkcijos algoritmų panaudojimas ir Liapunovo eksponenčių apskaičiavimas bei jų pritaikymas neuroninio tinklo suderinimui.

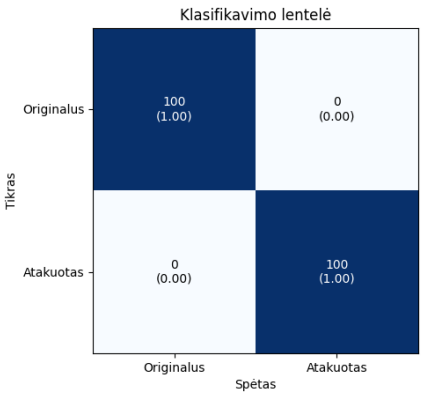
8.3.1. Liapunovo funkcijos algoritmai

Neuroninis tinkas su sukurtais LF I ir patobulintu LF I algoritmais bei Adam optimizacijos funkcija buvo išbandytas sprendžiant XOR uždavinį. Taigi, pirmieji bandymai buvo atlikti su 50 epochų.

Iš 4 ir 5 paveiksliukų matyti, jog tinklas su LF I algoritmu pasiekė aukščiausia tikslumą ir po apmokymo suskirstė visus duomenis į teisingas klases. Tai parodo, kad modelis su šiuo algoritmu sugeba išspręsti XOR uždavinį.



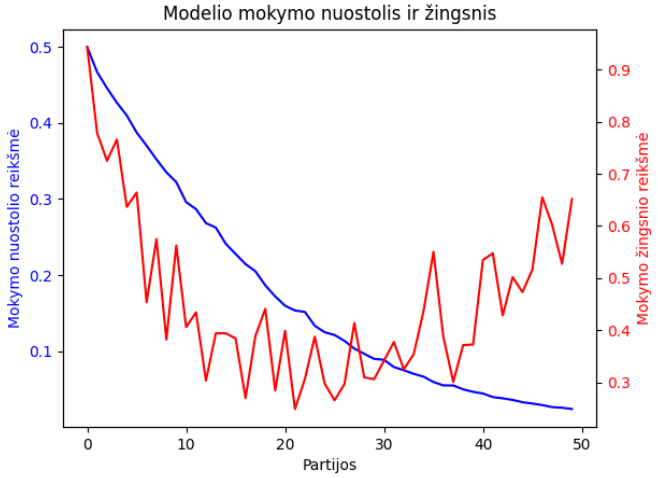
4 pav. Modelio su LF I algoritmu tikslumo grafikas per 50 epochų sprendžiant XOR uždavinį



5 pav. Modelio su LF I algoritmu klasifikavimo lentelė po 50 epochų sprendžiant XOR uždavinį

Pagal 6 paveiksliuką matyti, kad nuostolio funkcija viso mokymo metu mažėjo. Tame pačiame paveiksliuke mokymo žingsnis pačioje pradžioje buvo aukštas ir sparčiai mažėjo, bet įpusėjus

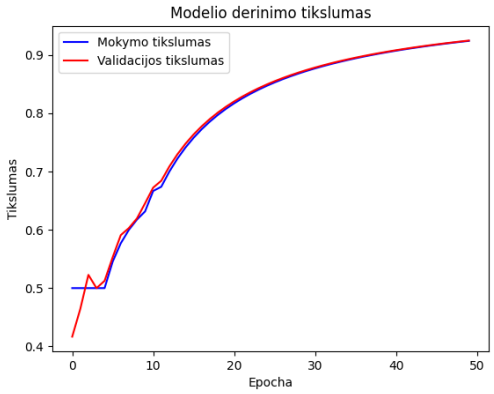
mokymui pradėjo didėti. Aukštos pradinės mokymo žingsnio reikšmės gali būti paaiškintos dideliu skirtumu tarp tikros reikšmės ir tinklo spėjimo bei dideliais pokyčiais svoriuose. Tačiau neu-



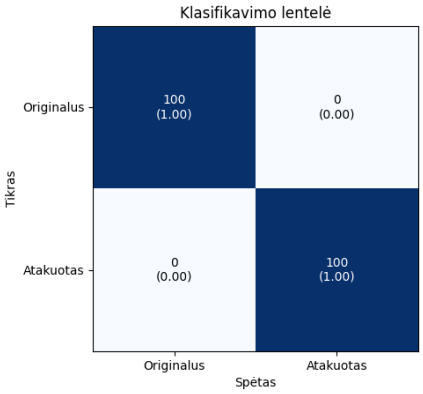
6 pav. Modelio su LF I algoritmu mokymo nuostolio funkcija ir mokymo žingsnis per 50 epochų sprendžiant XOR uždavinį

roniniam tinklui pasiekus aukštą tikslumą bei nuostolio funkcijai vis mažėjant mokymo žingsnis pradėjo vėl augti dėl tų pačių reikšmių, dėl kurių mokymo žingsnis mokymo pradžioje mažėjo.

Kitas bandymas buvo atliktas su patobulintu LF I algoritmu ir tokiu pačiu epochų kiekiu. Šio bandymo tikslumas ir klasifikavimo lentelė yra pateikti 7 ir 8 paveiksliukuose atitinkamai, iš kurių matyti, kad tinklas sugebėjo išspręsti XOR uždavinį.

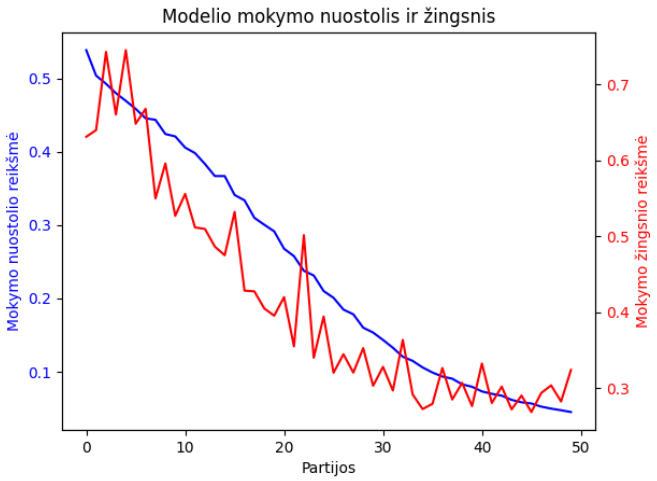


7 pav. Modelio su patobulintu LF I algoritmu tikslumo grafikas per 50 epochų sprendžiant XOR uždavinį



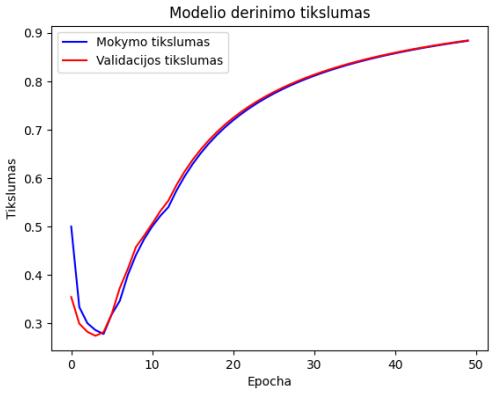
8 pav. Modelio su patobulintu LF I algoritmu klasifikavimo lentelė po 50 epochų sprendžiant XOR uždavinį

Patobulintas LF I algoritmas priklauso nuo paskutinio sluoksnio ir likusių svorių gradientų, tad dėl šios priežasties nuostolio funkcija ir mokymosi žingsnis (9 pav.) turi panašų mažėjimo polinkį.

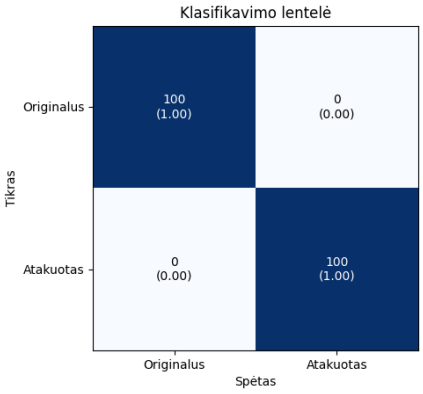


9 pav. Modelio su patobulintu LF I algoritmu mokymo nuostolio funkcija ir mokymo žingsnis per 50 epochų sprendžiant XOR uždavinį

Likęs algoritmas buvo Adam, kuris buvo išbandytas su tokiu pačiu neuroniniu tinklu ir tiek pat epochų. Iš 10 ir 11 paveiksliukų matyti, kad modelis su Adam algoritmu, kaip ir su visais kitais algoritmais, sugebėjo išspręsti XOR uždavinį bei pasiekti aukštą tikslumą ir po apmokymo nepadarė nei vienos klaidos priskiriant įeinančius duomenis skirtingoms klasėms.



10 pav. Modelio su Adam algoritmu tikslumo grafikas per 50 epochų sprendžiant XOR uždavinį

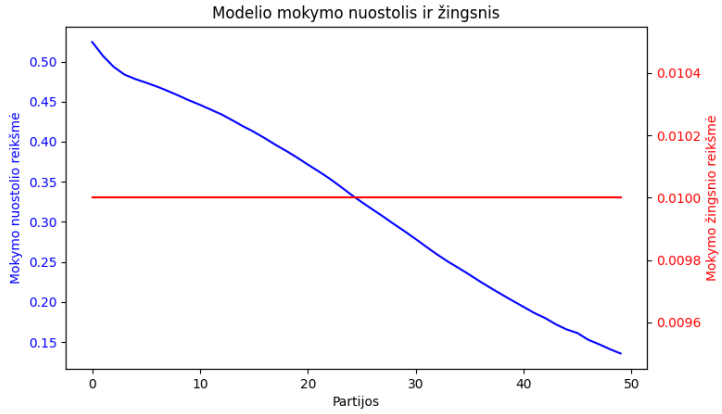


11 pav. Modelio su Adam algoritmu klasifikavimo lentelė po 50 epochų sprendžiant XOR uždavinį

Iš 12 paveiksliuko matyti, kad nuostolio funkcija viso mokymo metu mažėja. O mokymo žingsnis išlieka toks pats, kadangi yra pateikiamas formulės (17) pradinis mokymosi žingsnis, nes Adam algoritmas apskaičiuoja mokymosi žingsnį kiekvienam parametrui.

3 lentelėje pateiktos visos vertinimo metrikos po 50 epochų mokymo. Iš jos matyti, kad modelis su visais algoritmais sugebėjo išspręsti XOR uždavinį.

Atlikus šiuos bandymus buvo nuspręsta padidinti epochų kiekį iki 250 ir įsitikinti, kaip kinta nuostolio funkcijos ir mokymo žingsnio reikšmės neuroniniam tinklui sprendžiant XOR uždavinį.

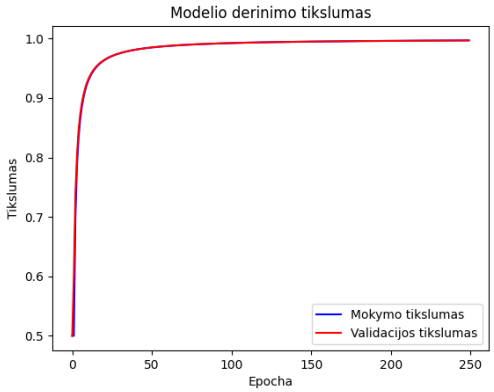


12 pav. Modelio su Adam algoritmu mokymo nuostolio funkcija ir mokymo žingsnis per 50 epochų sprendžiant XOR uždavinį

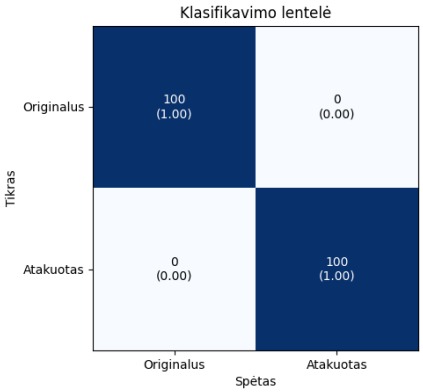
3 lentelė. Atliktų bandymų metrikos su 50 epochų sprendžiant XOR uždavinį

Algoritmas	Precizija	Atšaukimas	F-balas	Tikslumas
LF I	1.0	1.0	1.0	1.0
Patobulintas LF I	1.0	1.0	1.0	1.0
Adam	1.0	1.0	1.0	1.0

Taigi, pirmasis panaudotas algoritmas buvo LF I. Iš pateiktų 13 ir 14 paveiksliukų matyti, jog neuroninis tinklas pasiekia aukščiausią tikslumą ir po suderinimo klasifikavimo lentelėje nepadaro nei vienos klaidos. Tai, kaip ir prieš tai, parodo, kad tinklas pilnai išmoko spręsti XOR uždavinį.

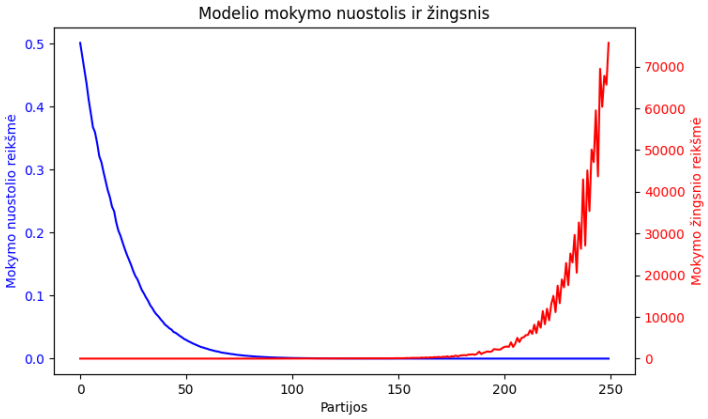


13 pav. Modelio su LF I algoritmu tikslumo grafikas per 250 epochų sprendžiant XOR uždavinį



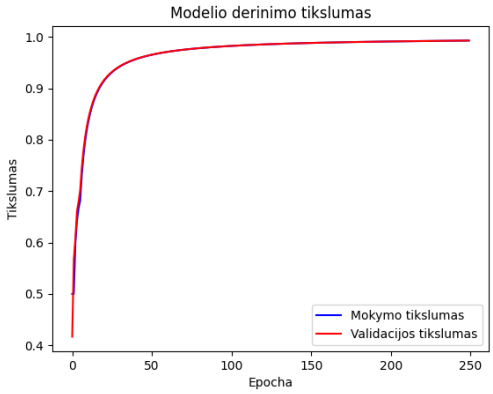
14 pav. Modelio su LF I algoritmu klasifikavimo lentelė po 250 epochų sprendžiant XOR uždavinį

15 paveiksliuke pateiktos mokymo metu buvusios nuostolio funkcijos ir mokymo žingsnio reikšmės, iš kurio matyti, jog nuostolio funkcijos reikšmė mažėjo iki nulio, o po 200 partijos mokymo žingsnis pradėjo sparčiai augti. Šis augimas prasidėjo dėl to, kad mokymo žingsniui apskaičiuoti yra naudojamas skirtumas tarp tikrosios reikšmės ir tinklo spėjimo, kurie tapo labai maži, bei Jakobijano matrica, kuri irgi pasiekė labai mažas reikšmes.

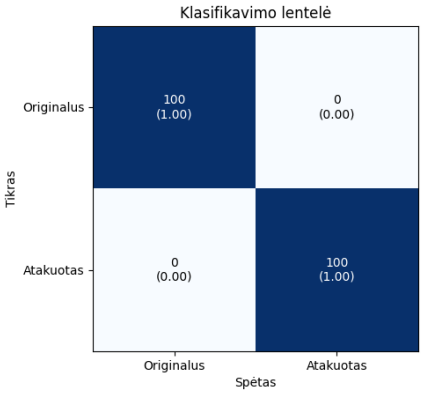


15 pav. Modelio su LF I algoritmu mokymo nuostolio funkcija ir mokymo žingsnis per 250 epochų sprendžiant XOR uždavinį

Suderintas neuroninis tinklas su patobulintu LF I algoritmu per 250 epochų pasiekė aukštą tikslumą (16 pav.) ir po suderinimo gautoje klasifikavimo lentelėje (17 pav.) matyti, jog tinklas nepadarė nei vienos klaidos. Taip kaip ir su LF I algoritmu, parodo, kad tinklas išmoko pilnai išspręsti XOR uždavinį.



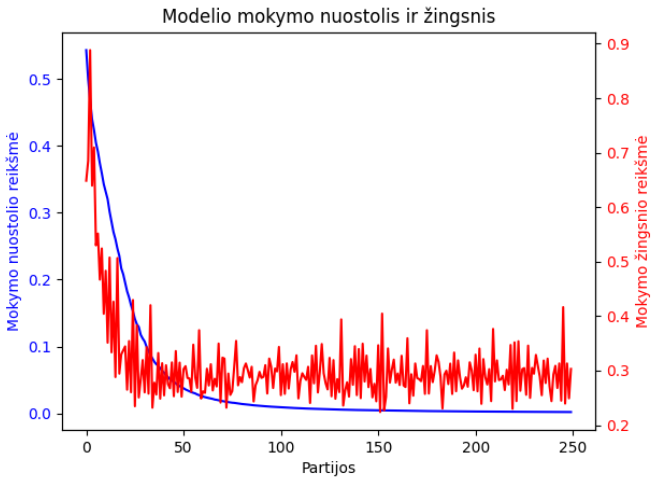
16 pav. Modelio su patobulintu LF I algoritmu tikslumo grafikas per 250 epochų sprendžiant XOR uždavinį



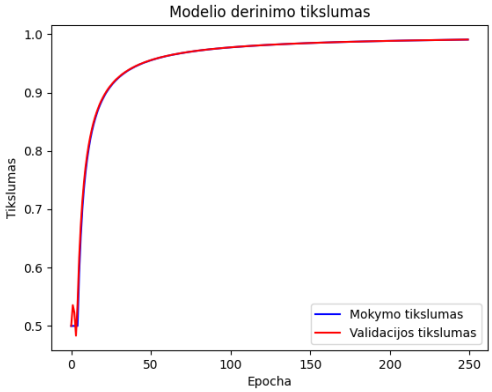
17 pav. Modelio su patobulintu LF I algoritmu klasifikavimo lentelė po 250 epochų sprendžiant XOR uždavinį

Iš 18 pav. matyti, jog nuostolio funkcijos reikšmė greitai mažėjo ir tuomet pradėjo plokštėti bei lėtai artėti link nulio. Šioje plokštėjimo dalyje mokymosi žingsnis nors ir svyravo, tačiau tam tikruose rėžiuose dėl mažai besikeičiančių svorių gradientų.

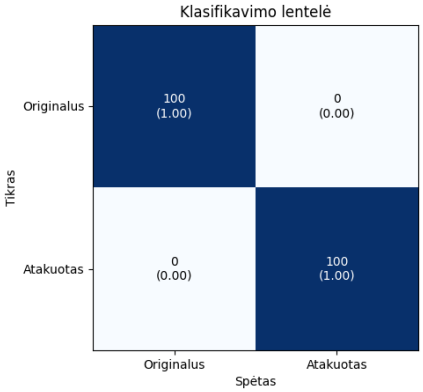
Kaip ir iš visų prieš tai buvusių bandymų su 250 epochų, neuroninis tinklas su Adam optimizacijos funkcija pasiekė aukštą tikslumą ir išmoko spręsti XOR uždavinį, o tai irgi matyti iš 19 ir 20 paveiksliukų.



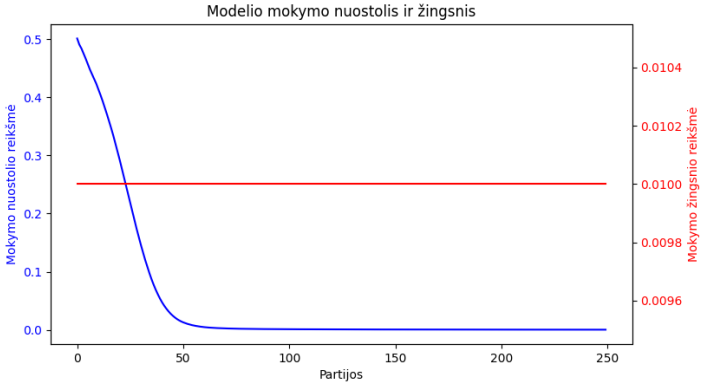
18 pav. Modelio su patobulintu LF I algoritmu mokymo nuostolio funkcija ir mokymo žingsnis per 250 epochų sprendžiant XOR uždavinį



19 pav. Modelio su Adam algoritmu tikslumo grafikas per 250 epochų sprendžiant XOR uždavinį



20 pav. Modelio su Adam algoritmu klasifikavimo lentelė po 250 epochų sprendžiant XOR uždavinį



21 pav. Modelio su Adam algoritmu mokymo nuostolio funkcija ir mokymo žingsnis per 250 epochų sprendžiant XOR uždavinį

Iš 21 paveiksliuko matyti, kad nuostolio funkcija pasiekė nulio reikšmę. Mokymosi žingsnis

nekinta dėl tokios pačios priežasties kaip ir bandyme su 50 epochų.

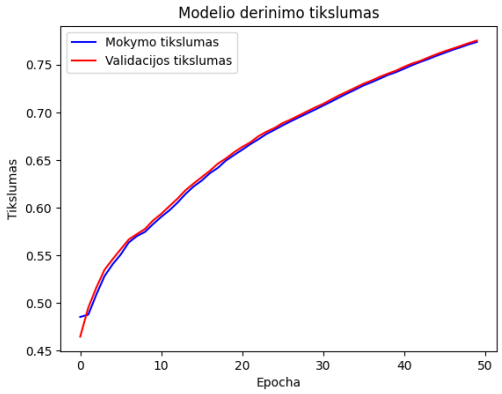
4 lentelėje pateiktos gautos precizijos, atšaukimo, f-balo ir tikslumo reikšmės po 250 epochų suderinimo naudojant modelį su skirtingais algoritmais. Iš lentelės matyti, kad visais atvejais modeliai pilnai išmoko spręsti XOR uždavinį.

4 lentelė. Atliktų bandymų metrikos su 250 epochų sprendžiant XOR uždavinį

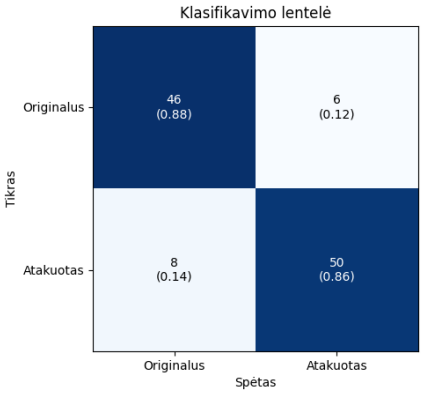
Algoritmas	Precizija	Atšaukimas	F-balas	Tikslumas
LF I	1.0	1.0	1.0	1.0
Patobulintas LF I	1.0	1.0	1.0	1.0
Adam	1.0	1.0	1.0	1.0

Atlikus visus bandymus su neuroniniu tinklu, pritaikytu spręsti XOR uždavinį, ir įsitikinus, jog įgyvendinti LF I ir patobulintas LF I algoritmai veikia kaip tikėtasi, buvo pradėti bandymai su pasirinktu duomenų rinkiniu, kuris aprašytas „Duomenų rinkinys“ skyriuje, ir sprendžiant paveiksliukų klasifikacijos uždavinį. Pirmiausia buvo pasirinktas 50 epochų kiekis.

Bandymai su paveiksliukų duomenų rinkiniu buvo pradėti naudojant LF I algoritmą. Iš 22 pav. matyti, jog modelio tikslumas auga ir nepradeda plokštėti, o po šio suderinimo gauta klasifikavimo lentelė (23 pav.) parodo, jog dauguma paveiksliukų buvo priskirti tinkamai klasei.



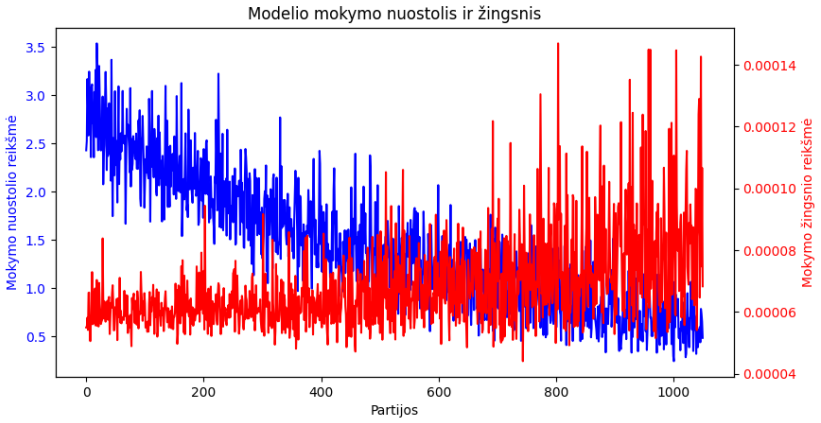
22 pav. Modelio su LF I algoritmu tikslumo grafikas per 50 epochų klasifikuojant paveiksliukus



23 pav. Modelio su LF I algoritmu klasifikavimo lentelė po 50 epochų klasifikuojant paveiksliukus

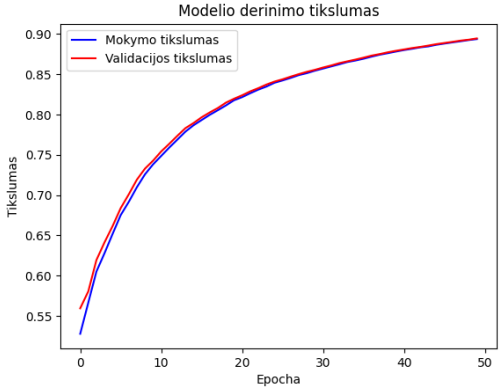
24 paveiksliuke yra pateikta, kad nuostolio funkcijos reikšmė nuolatos mažėja, o mokymo žingsnis turi didėjimo polinkį, ką galima buvo matyti su didesnėmis reikšmėmis ir XOR uždavinio sprendimo metu (6 ir 15 pav.). Taigi, galima daryti išvadą, kad mažėjant nuostolio funkcijai algoritmo LF I gaunama mokymo žingsnio reikšmė didėja dėl anksčiau minėtų priežasčių (skirtumo tarp tikrųjų reikšmių ir spėjimo bei gradientų).

Po bandymo su LF I algoritmu buvo pradėti bandymai su patobulintu LF I algoritmu. Naudojant jį tinklas pasiekė gan aukštą tikslumą (25 pav.) ir po 50 suderinimo epochų padarė mažiau

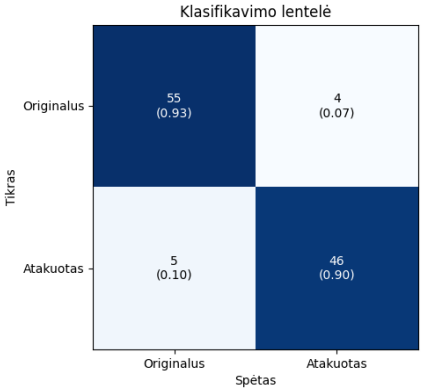


24 pav. Modelio su LF I algoritmu mokymo nuostolio funkcija ir mokymo žingsnis per 50 epochų klasifikuojant paveiksliukus

klaidų klasifikavimo lentelėje (26 pav.) negu kad LF I algoritmas. Tai leidžia spėti, jog patobulintas LF I algoritmas pasiekia aukštesnį tikslumą negu LF I algoritmas per mažesnį laiką.



25 pav. Modelio su patobulintu LF I algoritmu tikslumo grafikas per 50 epochų klasifikuojant paveiksliukus

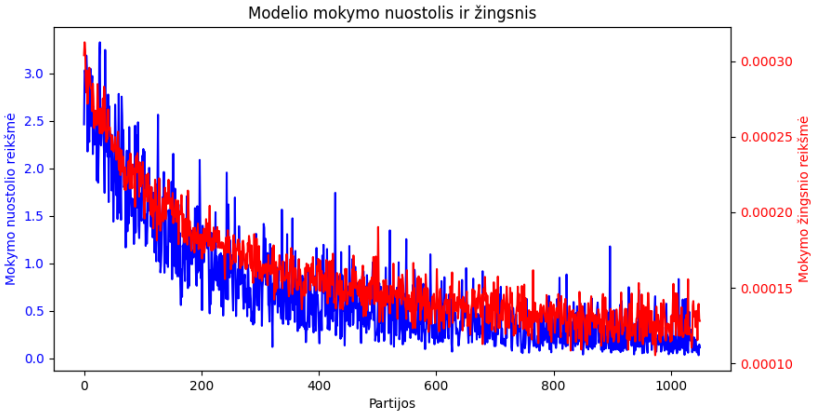


26 pav. Modelio su patobulintu LF I algoritmu klasifikavimo lentelė po 50 epochų klasifikuojant paveiksliukus

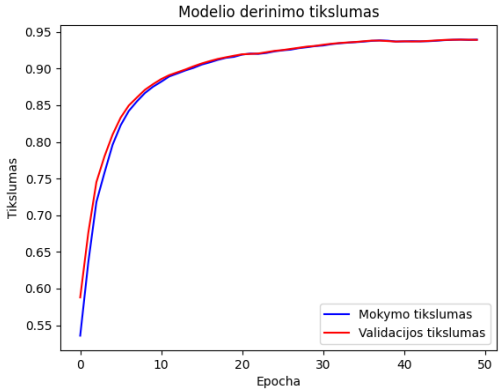
Taip pat 27 paveiksliukas parodo, jog modelio mokymo nuostolio funkcijos reikšmė ir mokymo žingsnis mažėja beveik kartu, o tai leidžia modelio suderinimo pradžioje daryti labai didelius pakeitimus ir taip sparčiau artėti link nuostolio funkcijos minimumo.

Su tokia pačia epochos reikšme buvo suderintas neuroninis tinkas su Adam optimizacijos funkcija. Šis modelis pasiekė aukštą tikslumą (28 pav.) bei po suderinimo gautoje klasifikavimo lentelė (29 pav.) turėjo daugiausia klaidų iš visų algoritmų - 15.

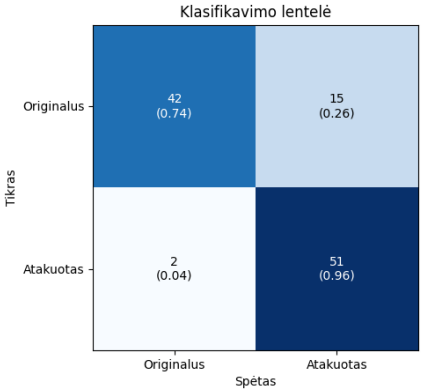
30 paveiksliuke matoma nuostolio funkcija pradžioje mažėjo link nulio, tačiau neišliko prie šio polinkio ir turėjo periodų, kai buvo vis didėjanti. Pagal gautą klasifikavimo lentelę (29 pav.) ir aukščiau pateiktus pastebėjimus galima daryti išvadą, jog modelis su Adam optimizacijos funkcija nors ir pasiekia aukštą tikslumą, tačiau neišlaiko nuostolio funkcijos arti nulio per 50 epochų.



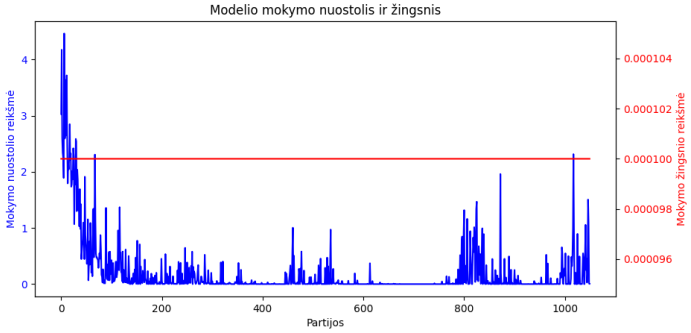
27 pav. Modelio su patobulintu LF I algoritmu mokymo nuostolio funkcija ir mokymo žingsnis per 50 epochų klasifikuojant paveiksliukus



28 pav. Modelio su Adam algoritmu tikslumo grafikas per 50 epochų klasifikuojant paveiksliukus



29 pav. Modelio su Adam algoritmu klasifikavimo lentelė po 50 epochų klasifikuojant paveiksliukus



30 pav. Modelio su Adam algoritmu mokymo nuostolio funkcija ir mokymo žingsnis per 50 epochų klasifikuojant paveiksliukus

O mokymosi žingsnis nekinta, nes paveiksliuke pateiktas pradinis mokymosi žingsnis, kuris yra nustatomas hiperparametras.

Gautos precizijos, atšaukimo, f-balo ir tikslumo reikšmės pateiktos 5 lentelėje po 50 suderinimo epochų naudojant modelį su skirtingais algoritmais. Iš šios lentelės matyti, jog modelis su

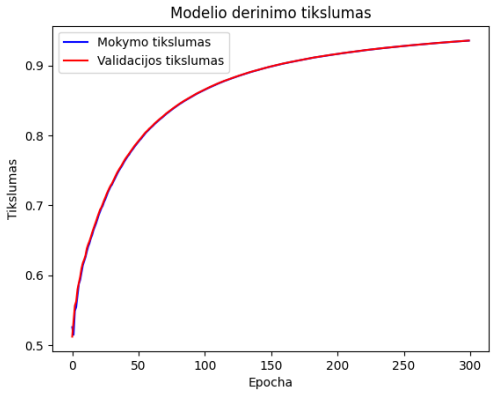
5 lentelė. Atliktų bandymų metrikos su 50 epochų atliekant paveiksliukų klasifikaciją

Algoritmas	Precizija	Atšaukimas	F-balas	Tikslumas
LF I	0.89	0.86	0.88	0.87
Patobulintas LF I	0.92	0.9	0.91	0.92
Adam	0.77	0.96	0.86	0.85

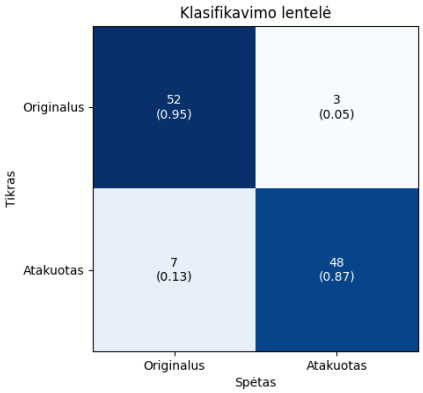
patobulintu LF I algoritmu pasiekia geriausius rezultatus, išskyrus atšaukimo reikšmę, kurią Adam funkcija pasiekia aukštesnę, ir tai parodo, kad Adam padarė daugiau teisingų teigiamų spėjimų (teisingai priskyrė duomenis į originalių paveiksliukų klasę).

Atlikus bandymus su 50 epochų buvo nuspręsta bandymą pakartoti su didesniu kiekiu epochų, kad būtų įsitikintą, kokį maksimalų tikslumą modelis gali pasiekti su kiekvienu algoritmu bei kaip kinta nuostolio funkcijos ir mokymo žingsnio reikšmės. Buvo nuspręsta naudoti 300 epochų.

Pirmasis bandymas buvo atliktas su tinklu, kuris buvo praplėstas su LF I algoritmu. Suderinus tinklą buvo gauti 31 ir 32 paveiksliukai, kuriuose parodytas tinklo pasiektas tikslumas, kuris nuolatos augo, ir po suderinimo gauta klasifikavimo lentelė, iš kurios matyti, kad tinklas neteisingai priskyrė 10 paveiksliukų į neteisingas klases. Tai parodo, kad tinklas sugebėjo pasiekti aukštus rezultatus sprendžiant klasifikacijos užduotį.



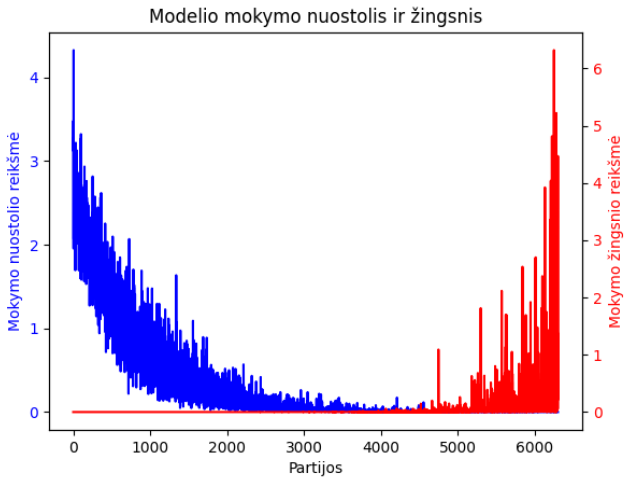
31 pav. Modelio su LF I algoritmu tikslumo grafikas per 300 epochų klasifikuojant paveiksliukus



32 pav. Modelio su LF I algoritmu klasifikavimo lentelė po 300 epochų klasifikuojant paveiksliukus

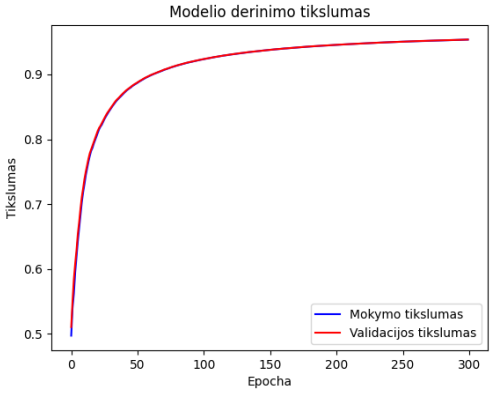
Gautas 33 paveiksliukas atitinka 15 paveiksliuką, kuris buvo gautas sprendžiant XOR uždavinį. Kadangi iš jų abiejų matyti, kad kai nuostolio funkcijos reikšmė tampa labai maža ir toliau artėja link nulio, mokymo žingsnis pradeda sparčiai augti. Tai vyksta dėl tokios pačios priežasties, kaip buvo išnagrinėta ties 15 paveiksliuku - skirtumas tarp tikros reikšmės ir spėjimo tampa labai mažas ir svorių pasikeitimai tapo labai maži, dėl ko Jakobijano matricos reikšmės taip pat tapo mažos.

Atliekant bandymą su patobulintu LF I algoritmu buvo gautas 34 paveiksliukas, kuris parodo

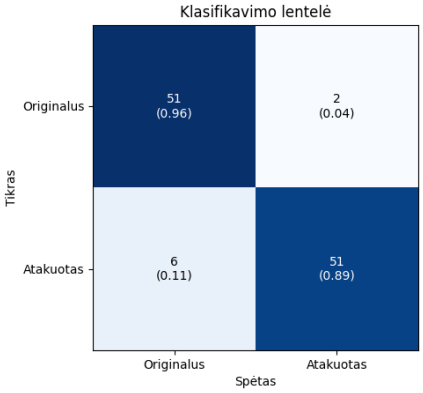


33 pav. Modelio su LF I algoritmu mokymo nuostolio funkcija ir mokymo žingsnis per 300 epochų klasifikuojant paveiksliukus

tikslumo augimą per epochas - jame matyti, kad nors tikslumas derinimo pradžioje ir sparčiai augo, tolimesnėse epochose tikslumo grafikas išsilygino. Taip pat iš 35 paveiksliuko matyti, kad tinklas po suderinimo padarė 8 klaidas.



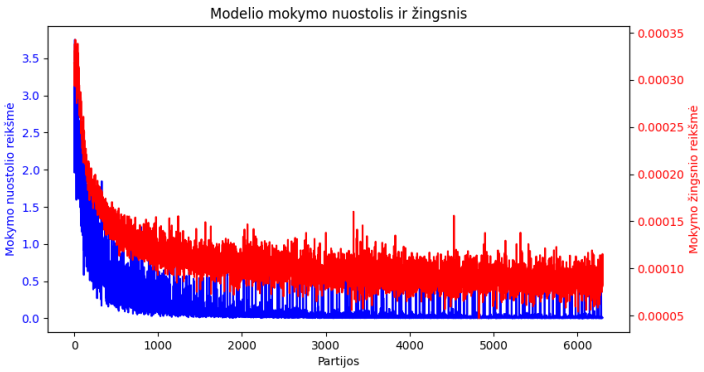
34 pav. Modelio su patobulintu LF I algoritmu tikslumo grafikas per 300 epochų klasifikuojant paveiksliukus



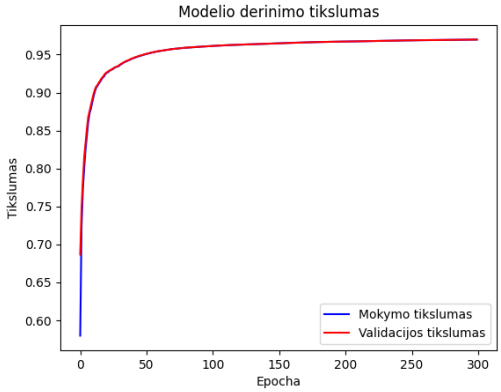
35 pav. Modelio su patobulintu LF I algoritmu klasifikavimo lentelė po 300 epochų klasifikuojant paveiksliukus

36 paveiksliukas parodo tokį patį polinkį, kaip ir 27 paveiksliukas, kur toks pat bandymas buvo atliktas su mažiau epochų. Tačiau panašius polinkius galima pamatyti ir iš 9 bei 18 paveiksliukų, kurie buvo gauti sprendžiant XOR uždavinį, nors juose mokymo žingsnis buvo žymiai didesnis.

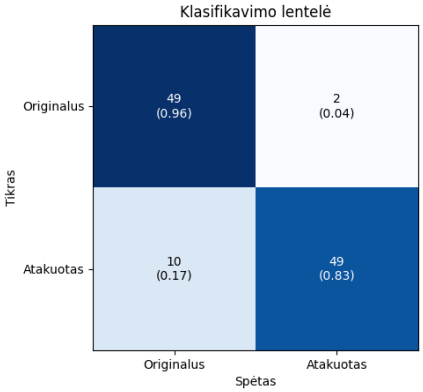
Gautas 37 paveiksliukas parodo tikslumo grafiko greitą augimą, po kurio seka išsilyginimas, kurio metu tikslumas smarkiai nebesikeičia. Tuo tarpu 38 paveiksliukas parodo, kad tinklas padarė daugiausia klaidų - 12. Tokia pati išvada buvo gauta suderinus modelį su mažiau epochų.



36 pav. Modelio su patobulintu LF I algoritmu mokymo nuostolio funkcija ir mokymo žingsnis per 300 epochų klasifikuojant paveiksliukus

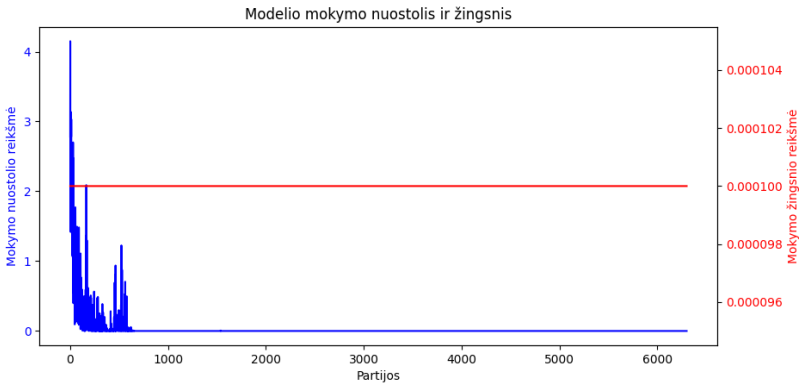


37 pav. Modelio su Adam algoritmu tikslumo grafikas per 300 epochų klasifikuojant paveiksliukus



38 pav. Modelio su Adam algoritmu klasifikavimo lentelė po 300 epochų klasifikuojant paveiksliukus

Nuostolio funkcijos reikšmės 39 paveiksliuke tik pačioje suderinimo pradžioje kito, tolimesnėje eigoje jos artėjo link nulio. Tai gali įvykti todėl, kad optimizacijos funkcija sėkmingai surado minimumą. Šiame paveiksliuke mokymosi žingsnis nekinta, kadangi yra pateiktas pradinis mokymosi žingsnis.



39 pav. Modelio su Adam algoritmu mokymo nuostolio funkcija ir mokymo žingsnis per 300 epochų klasifikuojant paveiksliukus

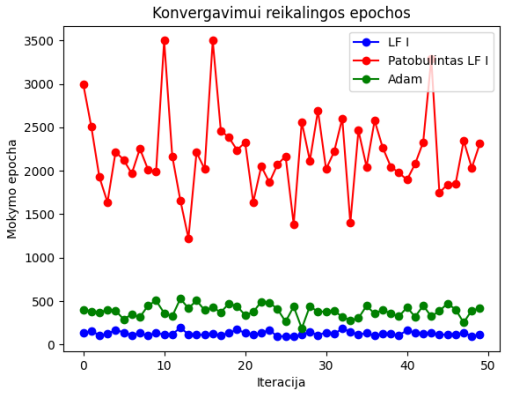
Visų atliktų bandymų vertinimo metrikos, kai epochų skaičius nustatytas kaip 300, yra pateiktos 6 lentelėje. Iš jos matyti, kad LF I ir patobulintas LF I pasiekė aukštesnį tikslumą negu kad Adam per 0.02 ir 0.04 atitinkamai. Taip pat visos metrikos turėjo aukščiausias reikšmes naudojant patobulintą LF I algoritmą, tačiau jų skirtumai tarp algoritmų, kaip matyti iš tikslumo palyginimo, yra labai maži.

6 lentelė. Atliktų bandymų metrikos su 300 epochų atliekant paveiksliukų klasifikaciją

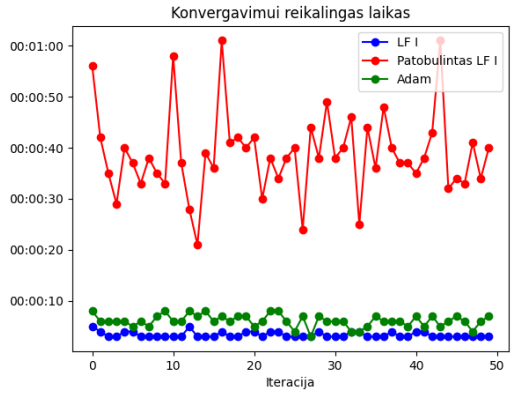
Algoritmas	Precizija	Atšaukimas	F-balas	Tikslumas
LF I	0.94	0.87	0.91	0.91
Patobulintas LF I	0.96	0.89	0.93	0.93
Adam	0.96	0.83	0.89	0.89

Po visų algoritmų rezultatų gavimo ir abiejų uždavinių sprendimo, buvo iš naujo gauti rezultatai, kurie palygintų jų konvergavimą taip, kaip tai buvo padarę [BKP06] straipsnio autoriai. Šio straipsnio autoriai laikė, kad konvergavimui reikia, kad nuostolio funkcijos reikšmė per epochą pasiektų 0.0001 reikšmę ir norint gauti šių metrikų vidurkį, modelius apmokė 50 atskirų kartų (iteracijų). Siekiant atkartoti bandymą su XOR uždaviniu buvo panaudota tokia pati sąlyga bei maksimalus epochų kiekis pasirinktas kaip 3500, o sprendžiant paveiksliukų klasifikavimo uždavinį nuostolio funkcijos reikšmė per epochą buvo pasirinkta kaip 0.001 bei nustatytas maksimalus epochų kiekis kaip 1000, o iteracijų skaičius buvo pasirinktas kaip 10, kadangi šis uždavinys reikalavo daugiau resursų.

Taigi, 40 ir 41 paveiksliukai pateikia gautus bandymo rezultatus sprendžiant XOR uždavinį. Matyti, jog visi modeliai su skirtingais algoritmais konverguoja pagal apsibrėžtą sąlygą. Tačiau algoritmai LF I ir Adam konverguoja žymiai greičiau nei patobulintas LF I.



40 pav. Epochų kiekis, reikalingas modeliui konverguoti sprendžiant XOR uždavinį



41 pav. Laikas reikalingas, modeliui konverguoti sprendžiant XOR uždavinį

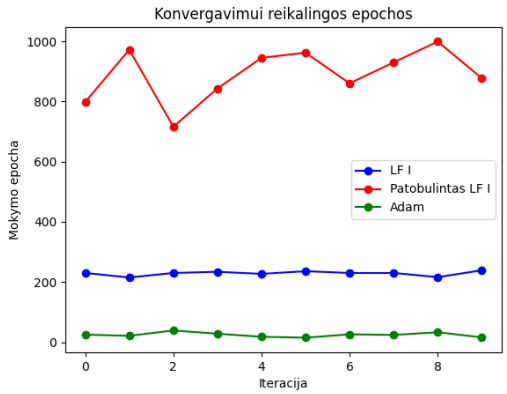
Visų iteracijų skirtingų reikšmių vidurkiai yra pateikti 7 lentelėje, iš kurios matyti, kad pagal

epochų skaičiaus vidurkį LF I algoritmas pasiekia konvergavimo sąlygą greičiausiai, tačiau nors ir patobulintam LF I algoritmui reikėjo 17 kartų daugiau epochų pasiekti tą pačią konvergavimo sąlygą kaip kad LF I, bet šis algoritmas pasiekė 0.9 tikslumą per mažiausią epochų skaičių iš visų algoritmų. Taigi, iš šios lentelės ir prieš tai darytų bandymų, iš kurių gauti nuostolio funkcijos grafikai, galima daryti išvadą, kad patobulintas LF I nors ir artėja link nulio, tačiau siekiant pasiekti labai mažą nuostolio funkcijos reikšmę, jam reikia daugiau epochų. Taip pat vėl galima matyti, kad visi algoritmai gali spręsti XOR uždavinį.

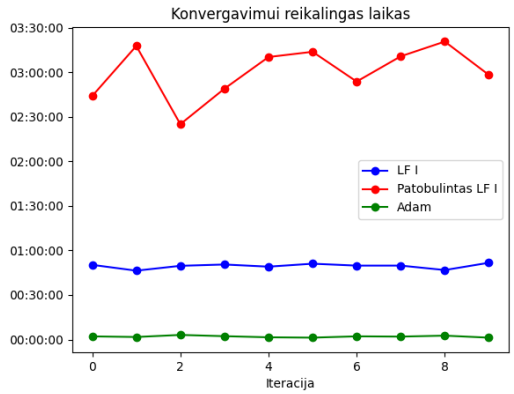
7 lentelė. Algoritmų konvergavimo duomenų vidurkiai iš 50 iteracijų sprendžiant XOR uždavinį

Algoritmas	LF I	Patobulintas LF I	Adam
Vid. epochų kiekis	127.44	2184.8	386.26
Vid. laikas (val:min:sek)	00:00:03	00:00:39	00:00:06
Vid. nuostolio funkcijos reikšmė	0.0000955	0.0001073	0.0001037
Vid. nuostolio funkcijos reikšmė per visą laiką	0.0938	0.0059	0.0309
Vid. tikslumas	0.9833	0.9991	0.9929
Vid. epochų kiekis tikslumui pasiekti 0.9	21.7	19.92	26.82

Panašus konvergavimo polinkis matomas ir iš 42 ir 43 paveiksliukų, tačiau juose apibrėžtą konvergavimo sąlygą žymiai greičiau pasiekia modelis su Adam funkcija, o ne su LF I ar patobulintu LF I algoritmais. Tai gali būti paaiškinta, kad Adam algoritmas yra pritaikytas naudojimui su dideliais duomenų rinkiniais, kurie gali būti sudaryti iš aukštos dimensijos duomenų [KB17].



42 pav. Epochų kiekis, reikalingas modeliui konverguoti sprendžiant paveiksliukų klasifikacijos uždavinį



43 pav. Laikas reikalingas, modeliui konverguoti sprendžiant paveiksliukų klasifikacijos uždavinį

Prieš tai pateiktą išvadą dar labiau pabrėžia 8 lentelės duomenys, iš kurių matyti, kad vidutiniškai modeliui su Adam algoritmu reikėjo tik 24.5 epochų, kai kitiems reikėjo bent kelių šimtų. Modelis su Adam algoritmu daug greičiau pasiekė 0.8 ir 0.9 tikslumo reikšmes - vidutiniškai per 3.2 ir 10.7 epochas atitinkamai. Tuo tarpu modeliui su LF I ir patobulintu LF I algoritmais reikėjo

bent 56.8 ir 17.2 epochų atitinkamai, kad pasiektų 0.8 tikslumą. Tai parodo, kad LF I ir patobulintas LF I algoritmai nors ir sugeba spręsti paveiksliukų klasifikavimo uždavinį, bet yra žymiai neefektyvesni palyginus su Adam algoritmu. Tačiau svarbu pabrėžti, kad Adam algoritmas yra jautrus pradinio mokymosi žingsnio reikšmei - modelis su Adam algoritmu ir pradiniu mokymosi žingsniu nustatytu didesniu kaip 0.0002 nesugebėjo mokytis (plačiau „Priedas nr. 2“).

8 lentelė. Algoritmų konvergavimo duomenų vidurkiai iš 10 iteracijų sprendžiant paveiksliukų klasifikacijos uždavinį

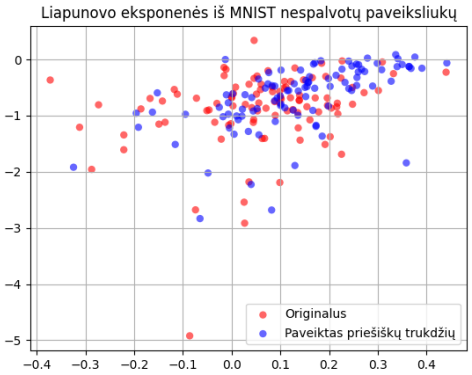
Algoritmas	LF I	Patobulintas LF I	Adam
Vid. epochų kiekis	228.7	890.2	24.5
Vid. laikas (val:min:sek)	0:49:30	3:00:16	0:01:59
Vid. nuostolio funkcijos reikšmė	0.0013	0.0010	0.0005
Vid. nuostolio funkcijos reikšmė per visą laiką	0.4025	0.0522	0.3056
Vid. tikslumas	0.9213	0.9640	0.9310
Vid. validacijos tikslumas	0.9213	0.9640	0.9308
Vid. epochų kiekis tikslumui pasiekti 0.8	56.8	17.2	3.2
Vid. epochų kiekis tikslumui pasiekti 0.9	159.4	58	10.7

8.3.2. Liapunovo eksponenčių gavimas

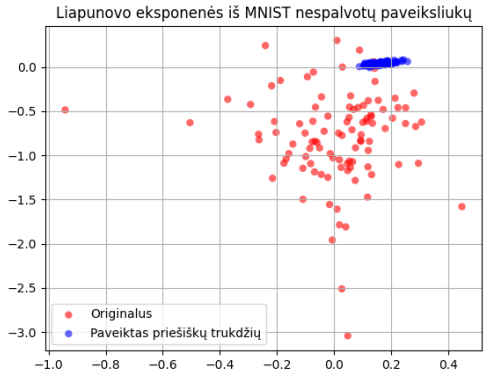
Atlikus bandymus su Liapunovo funkcijos algoritmais buvo pradėta kita eksperimento dalis- Liapunovo eksponenčių panaudojimas. Pirmiausia buvo bandyta atkartoti straipsnių [PDW18] ir [PDB20] gautus rezultatus. Abiejuose straipsniuose buvo naudoti MNIST ir mados MNIST duomenų rinkiniai, o [PDB20] straipsnis dar panaudojo ir CIFAR duomenų rinkinį. Tačiau šiame darbe buvo nuspręsta naudoti tik MNIST ir CIFAR duomenų rinkinius, kadangi vienas yra sudarytas iš juodai baltų paveiksliukų, o kitas iš spalvotų, kurie yra panašūs į ImageNet duomenų rinkinį, bei skyriuje „Duomenų rinkinys“ aprašytą duomenų rinkinį. Taip pat remiantis abiem prieš tai minėtais straipsniais buvo nuspręsta MNIST ir CIFAR duomenų rinkinius paveikti su Carlini Wagner, FGSM, PGD bei Madry ir kiti atakomis. Šių atakų parametrai buvo naudoti tokie patys, kaip ir buvo apibrėžti [PDB20] straipsnyje, kadangi jie atitiko tų atakų autorių numatytuosius parametrus.

Liapunovo eksponenčių apskaičiavimui buvo naudoti tokie patys parametrai, kurie buvo nustatyti [PDW18] straipsnyje, kadangi [PDB20] straipsnyje jie nebuvo tiksliai apibrėžti, bet buvo minima, kad vienam paveiksliukui buvo apskaičiuotos keturios eksponentės.

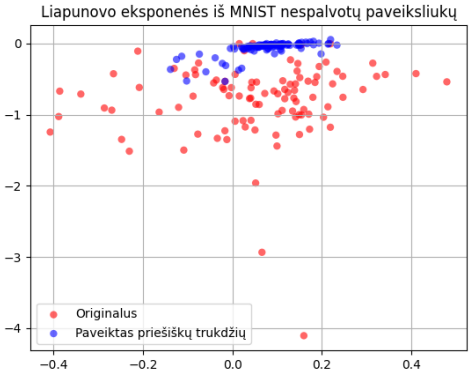
Taigi, pirmasis bandymas buvo atliktas su MNIST duomenų rinkiniu ir jį atakuojant su prieš tai apibrėžtomis atakomis. Originaliems ir atakuotiems paveiksliukams buvo apskaičiuotos keturios Liapunovo eksponentės, tačiau kaip ir [PDB20] straipsnyje, grafikuose pateiktos pirmos dvi Liapunovo eksponentės. Iš 45, 46 ir 47 grafikų matyti, kad paveikti ir originalūs paveiksliukai



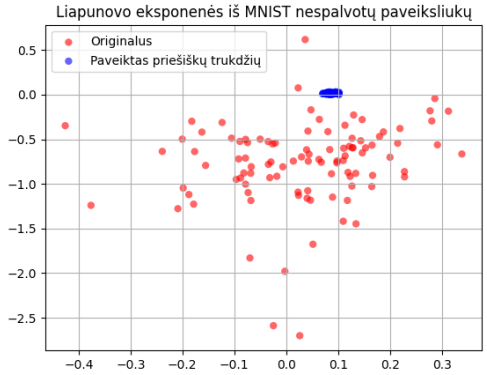
44 pav. Liapunovo eksponentės, gautos iš MNIST duomenų rinkinio, atakuoto su Carlini Wagner ataka



45 pav. Liapunovo eksponentės, gautos iš MNIST duomenų rinkinio, atakuoto su PGD ataka



46 pav. Liapunovo eksponentės, gautos iš MNIST duomenų rinkinio, atakuoto su FGSM ataka

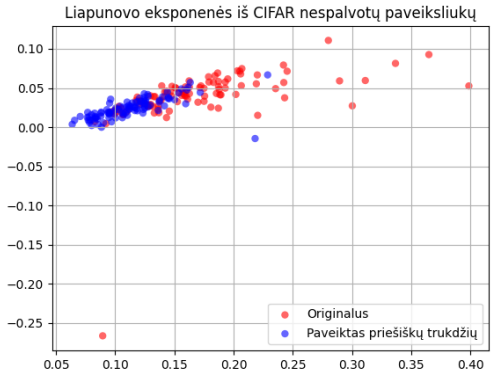


47 pav. Liapunovo eksponentės, gautos iš MNIST duomenų rinkinio, atakuoto su Madry ir kiti ataka

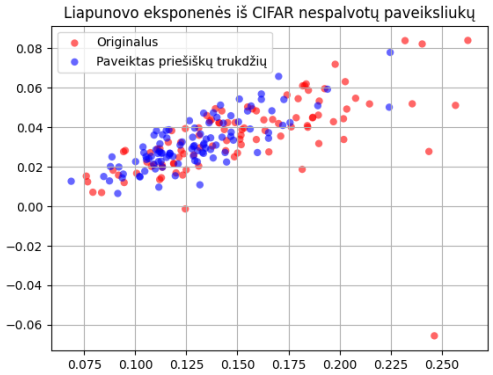
yra susiskirstę į labai aiškias grupes, tačiau 44 paveiksliuke matyti, kad originalių paveiksliukių ir Carlini Wagner atakos paveiktų paveiksliukų gautos Liapunovo eksponentės nesusiskirsto į aiškias grupes.

Antras bandymas buvo atliktas su CIFAR paveiksliukais, kurie prieš juos paveikiant priešiškais trukdžiais buvo paversti į juodai baltus paveiksliukus kaip tai buvo padarę [PDB20] autoriai. Gautuose 48, 49, 50 ir 51 paveiksliukuose pateiktos pirmosios dvi Liapunovo eksponentės kaip ir prieš tai. Iš 48 ir 51 paveiksliukų matyti, kad originalūs ir atakuoti paveiksliukai yra susigrupavę, tačiau persidengia. Toks persidengimas dar stipresnis 50 paveiksliuke. O 49 paveiksliuke originalių ir atakuotų paveiksliukų pirmosios dvi Liapunovo eksponentės jų nesugrupuoja.

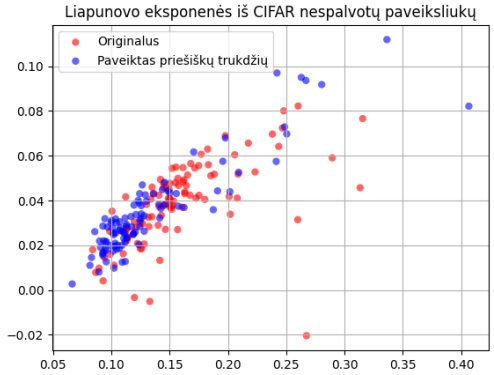
Pagal [PDW18] straipsnį duomenys, nepaveikti priešiškų trukdžių, neturėtų turėti arba turėti mažesnį kiekį teigiamų Liapunovo eksponenčių, kadangi tai yra stiprus chaoso indikatorius. Tačiau iš atliktų bandymų su MNIST ir CIFAR duomenų rinkiniais matyti, kad gautų Liapunovo eksponenčių ženklas nėra pakankamas įrodymas, kad paveiksliukas yra paveiktas priešiškų truk-



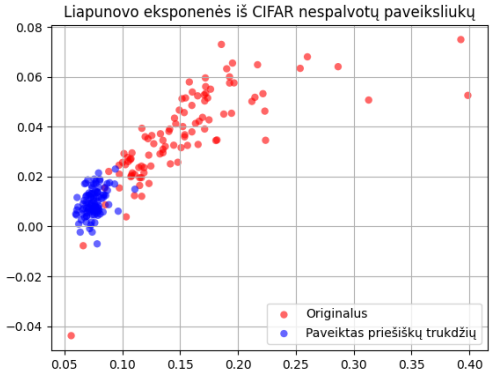
48 pav. Liapunovo eksponentės, gautos iš nespalvotų CIFAR duomenų rinkinio, atakuoto su Carlini Wagner ataka



49 pav. Liapunovo eksponentės, gautos iš nespalvotų CIFAR duomenų rinkinio, atakuoto su PGD ataka

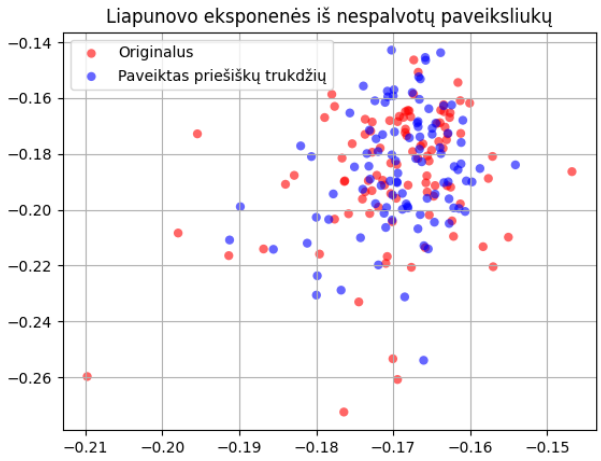


50 pav. Liapunovo eksponentės, gautos iš nespalvotų CIFAR duomenų rinkinio, atakuoto su FGSM ataka



51 pav. Liapunovo eksponentės, gautos iš nespalvotų CIFAR duomenų rinkinio, atakuoto su Madry ir kiti ataka

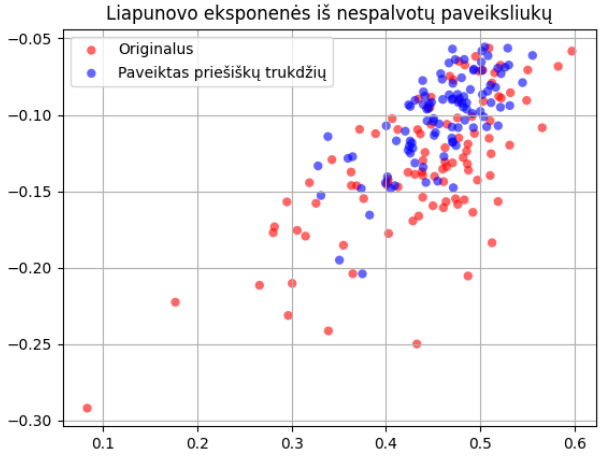
džių. Straipsnio [PDB20] autoriai gavo panašius į šio bandymo rezultatus bei padarė išvadą, kad Liapunovo eksponentės yra labai jautrios atakos paveiktų pikselių kiekiui - kuo daugiau pikselių yra paveiktų tuo lengviau atskirti originalius ir paveiktus priešiškų trukdžių paveiksliukus.



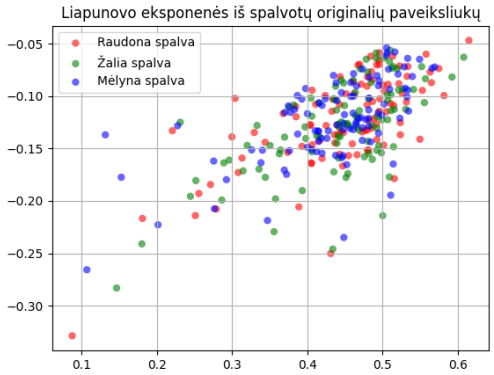
52 pav. Pagal nespalvotus originalius ir priešiškų trukdžių paveiktus paveiksliukus apskaičiuotos Liapunovo eksponentės

Atlikus aukščiau pateiktus bandymus buvo pradėtas bandymas su pasirinktu duomenų rinkiniu (daugiau pateikta „Duomenų rinkinys“ skyriuje). Kaip CIFAR duomenų rinkinys buvo paverstas į juodai baltus paveiksliukus, buvo nuspręsta tą patį padaryti su pasirinktu duomenų rinkiniu bei jam apskaičiuoti tokį patį kiekį Liapunovo eksponenčių. Gautos pirmos dvi Liapunovo eksponentės pateiktos 52 paveiksliuke, iš kurio matyti, kad originalūs ir priešiškų trukdžių paveikti paveiksliukai nėra susigrupavę.

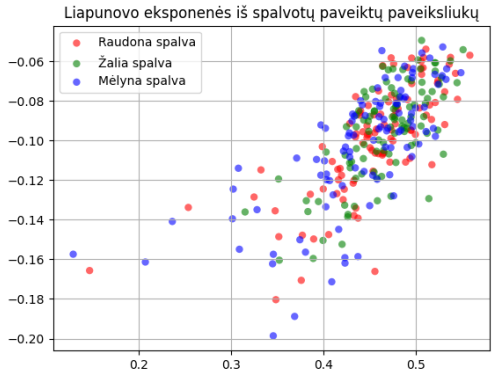
Buvo nuspręsta sumažinti apskaičiuojamų Liapunovo eksponenčių skaičių iki dviejų, kai spalvoti paveiksliukai yra konvertuoti į juodai baltus. Buvo gautas 53 pav., iš kurio matyti, kad Liapunovo eksponentės, gautos iš originalių ir priešiškų trukdžių paveiktų paveiksliukų, yra stipriai persidengusios, tačiau ne taip smarkiai nesusigrupavusios kaip 52 paveiksliuke.



53 pav. Pagal nespalvotus originalius ir priešiškų trukdžių paveiktus paveiksliukus apskaičiuotos dvi Liapunovo eksponentės



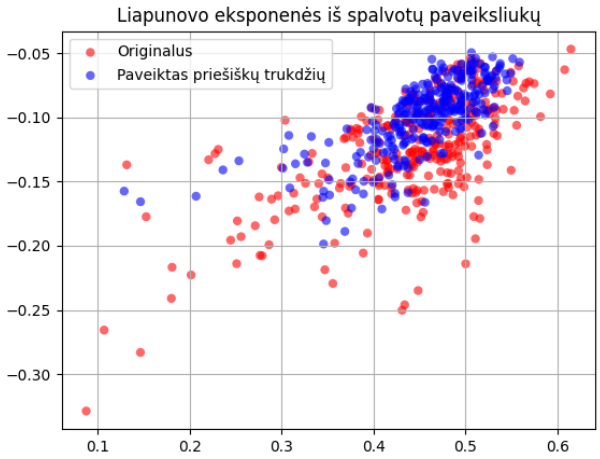
54 pav. Pagal spalvotus originalius duomenų rinkinio paveiksliukus apskaičiuotos Liapunovo eksponentės



55 pav. Pagal spalvotus duomenų rinkinio paveiksliukus paveiktus priešiškais trukdžiais apskaičiuotos Liapunovo eksponentės

Tikintis, kad dėl spalvotų paveiksliukų konvertavimo į juodai baltus yra prarandama dalis informacijos buvo nuspręsta apskaičiuoti kiekvienam spalvų kanalui po dvi Liapunovo eksponentes (iš viso apskaičiuotos 6 Liapunovo eksponentės). Gautos eksponentės pateiktos 54 ir 55 pav., kur trys grafiko taškai atitinka vieną konvertuojamų duomenų paveiksliuką.

Vizualiai perdengus Liapunovo eksponentes, gautas iš spalvotų originalių (54 pav.) ir paveiktų paveiksliukų (55 pav.), buvo gautas 56 paveiksliukas. Iš jo matyti, kad originalūs paveiksliukai turi didesnį pasiskirstymą negu paveikti priešiškų trukdžių. Taip pat matyti, jog abiejų klasių paveiksliukų Liapunovo eksponentės išlieka panašioje vietoje.



56 pav. Pagal spalvotus originalius ir priešiškų trukdžių paveiktus paveiksliukus apskaičiuotos Liapunovo eksponentės

Pagal anksčiau minėtą [PDB20] straipsnio išvadą buvo nuspręsta apskaičiuoti pikselių kiekį, kuris buvo pakeistas sudarytame duomenų rinkinyje. Taigi, kai paveiksliukai yra spalvoti vidutiniškas skirtingų pikselių kiekis yra 49 441.12 iš visų 50 176 pikselių, jeigu skaičiuojant visų skirtingų

spalvų kanalų reikšmes pikslelių kiekis yra trigubai didesnis - 148 323.35 pakeistų pikselių iš visų 150 528. Tuo tarpu paveiksliukus konvertavus į juodai baltus - pakeistų pikselių kiekis yra 22 271.74 iš visų 50 176 galimų. Taigi, spalvotų paveiksliukų pikselių 98.54 procentai yra pakeisti, o juos konvertavus į pilkus - 44.39 procentai. Pagal šią informaciją matoma, kad iš tiesų straipsnio [PDB20] padaryta išvada dėl Liapunovo eksponenčių jautrumo pakeistų pikselių kiekiui yra teisinga.

MNIST, CIFAR ir pasirinktam duomenų rinkiniams apskaičiavus Liapunovo eksponentes buvo pradėtas tinklo apmokymas su jomis. Sukurtas tinklas buvo apmokomas 100 epochų su kiekvienu gautu Liapunovo eksponenčių rinkiniu. Visų apmokymų tikslumo ir nuostolio funkcijos grafikai yra pateikti skyriuje „Priedas nr. 3“.

Taigi, pirmiausia buvo apmokytas tinklas su Liapunovo eksponenčių rinkiniu gautu iš MNIST duomenų rinkinio atakuoto su Carlini Wagner, PGD, FGSM bei Madry ir kiti metodais. Visos gautos vertinimo metrikos pateiktos 9 lentelėje. Iš jos matyti, kad su tais Liapunovo eksponenčių rinkiniais, kurie buvo aiškiai susigrupavę 44-47 pav. tinklas su visais algoritmais pasiekia aukštą tikslumą. Svarbu pastebėti kad su FGSM bei Madry ir kiti atakomis tinklas su LF I algoritmu po 100 epochų nepasiekia aukšto tikslumo. Žvelgiant į tinklo su LF I algoritmu tikslumo grafiką, kuris yra pateiktas skyriuje „Priedas nr. 3“, galima matyti kad tikslumas staigiai auga ir tuomet krenta žemyn. Tai įvyko dėl per didelio mokymosi žingsnio - skirtumas tarp tikrųjų ir spėtų reikšmių tapo labai mažas ir dėl to mokymosi žingsnis staiga pradėjo augti pagal LF I algoritmo mokymosi žingsnio formulę.

9 lentelė. Vertinimo metrikos gautos iš tinklo apmokyto su Liapunovo eksponentėmis gautomis iš MNIST duomenų rinkinio

Ataka	Algoritmas	Precizija	Atšaukimas	F-balas	Tikslumas
Carlini Wagner	LF I	0.39	0.64	0.49	0.53
	Patobulintas LF I	0.74	0.67	0.7	0.7
	Adam	0.52	0.5	0.51	0.48
PGD	LF I	1.0	1.0	1.0	1.0
	Patobulintas LF I	0.92	1.0	0.96	0.95
	Adam	1.0	1.0	1.0	1.0
FGSM	LF I	0.48	1.0	0.64	0.48
	Patobulintas LF I	0.91	1.0	0.95	0.95
	Adam	0.95	0.95	0.95	0.95
Madry ir kiti	LF I	0.45	1.0	0.62	0.45
	Patobulintas LF I	0.94	1.0	0.97	0.98
	Adam	1.0	1.0	1.0	1.0

Antrasis bandymas buvo atliktas apmokant tinklą su Liapunovo eksponenčių rinkiniu gautu

iš nespalvotų CIFAR duomenų rinkinio atakuoto su Carlini Wagner, PGD, FGSM bei Madry ir kiti metodais. Bandymo metu gautos vertinimo metrikos pateiktos 10 lentelėje, iš kurios matyti panašūs polinkiai kaip ir apmokant tinklą su MNIST duomenų rinkiniu - kai klasės yra aiškiai susigrupavusios tinklas su bet kuriuo algoritmu pasiekia aukštą tikslumą. Išskyrus LF I algoritmą, kadangi jis kaip ir pastebėta anksčiau priartėjęs prie aukšto tikslumo pradeda sparčiai mažėti.

10 lentelė. Vertinimo metrikos gautos iš tinklo apmokyto su Liapunovo eksponentėmis gautomis iš pilkų CIFAR duomenų rinkinio

Ataka	Algoritmas	Precizija	Atšaukimas	F-balas	Tikslumas
Carlini Wagner	LF I	0.6	1.0	0.75	0.6
	Patobulintas LF I	0.64	0.82	0.72	0.73
	Adam	0.66	0.88	0.76	0.75
PGD	LF I	0.48	0.5	0.49	0.48
	Patobulintas LF I	0.62	0.68	0.65	0.65
	Adam	0.48	0.5	0.49	0.48
FGSM	LF I	0.62	0.87	0.72	0.75
	Patobulintas LF I	0.59	0.63	0.61	0.68
	Adam	0.75	0.63	0.68	0.65
Madry ir kiti	LF I	0.0	0.0	0.0	0.63
	Patobulintas LF I	0.95	0.95	0.95	0.95
	Adam	0.89	0.94	0.92	0.93

Paskutinis bandymas buvo atliktas su sudarytu duomenų rinkiniu. Tinklo apmokymui buvo naudoti Liapunovo eksponenčių rinkiniai - keturios eksponentės gautos iš nespalvotų paveiksliukų, dvi eksponentės gautos taip pat iš nespalvotų bei šešios eksponentės gautos iš spalvotų paveiksliukų. Po apmokymo gautos vertinimo metrikos pateiktos 11 lentelėje. Iš jos galima daryti tokias pačias išvadas kaip su MNIST ir CIFAR duomenų rinkiniais paveiktais skirtingomis atakomis - mažiau susigrupavusios klasės pasiekia mažesnį tikslumą. Taigi, iš 52 paveiksliuko matyti, kad originalūs ir paveikti paveiksliukai yra panašiai pasiskirstę ir dėl šios priežasties visos vertinimo metrikos nepasiekia aukštų reikšmių, nes yra sunku ar neįmanoma apibrėžti ribų tarp skirtingų grupių. Taip pat svarbu pabrėžti, kad dėl stipraus persidengimo matomo 53 ir 56 paveiksliukuose tinklo pasiekiamas aukščiausias tikslumas yra apribotas.

Iš visų atliktų bandymų su Liapunovo eksponentėmis matyti, kad jeigu Liapunovo eksponentės, gautos iš duomenų rinkinio, yra susigrupavusios, neuroninį tinklą galima apmokyti. Tačiau svarbu pabrėžti, kad Liapunovo eksponenčių apskaičiavimas gali užtrukti daug laiko priklausomai nuo pasirinkto eksponenčių skaičiaus ir pradinių duomenų dydžio (paveiksliuko pikselių kiekio). Bei svarbu pabrėžti, kad gautos Liapunovo eksponentės gali nesugrupuoti pasirinktų klasių. Taip pat iš visų apmokymų naudojant skirtingus algoritmus su Liapunovo eksponentėmis matyti, kad

11 lentelė. Vertinimo metrikos gautos iš tinklo apmokyto su Liapunovo eksponentėmis gautomis iš sudaryto duomenų rinkinio

Duomenys	Algoritmas	Precizija	Atšaukimas	F-balas	Tikslumas
Nespalvoti (4 eksponentės)	LF I	0.44	0.42	0.43	0.48
	Patobulintas LF I	0.53	0.53	0.53	0.55
	Adam	0.44	0.35	0.39	0.45
Nespalvoti (2 eksponentės)	LF I	0.68	0.94	0.79	0.8
	Patobulintas LF I	0.71	0.85	0.77	0.75
	Adam	0.7	0.83	0.76	0.7
Spalvoti	LF I	0.69	0.9	0.78	0.75
	Patobulintas LF I	0.83	0.63	0.71	0.7
	Adam	0.79	0.95	0.86	0.85

LF I algoritmas pasiekęs aukštą tikslumą kartais pradeda blogėti. Tuo tarpu patobulintas LF I algoritmas net esant aiškiai nesugrupuotoms klasėms pasiekia aukščiausią tikslumą iš visų algoritmų.

Rezultatai ir išvados

Atlikus šį darbą buvo gauti tokie rezultatai:

1. Padaryta teorijos analizė - dimensijų prakeiksmo, persimokymo, Liapunovo teorijos (funkcija, eksponentė, spektras) bei Liapunovo teorijos pritaikymas neuroninių tinklų kontekste.
2. Sukurtas neuroninis tinklas VGG ir praplėstas su LF I, patobulintu LF I algoritmais (plačiau skyriuje „Liapunovo funkcijos panaudojimas“) ir Adam funkcija bei buvo suderintas naudojant pasirinktą duomenų rinkinį (plačiau skyriuje „Duomenų rinkinys“). Taip pat bandymai buvo pakartoti su paprastesniu neuroniniu tinklu sprendžiant XOR uždavinį su LF I ir patobulintu LF I bei Adam algoritmais.
3. Sukurta paveiksliukų konvertavimo sistema į Liapunovo eksponentes bei originalūs ir priešiškų trukdžių paveikti MNIST ir CIFART bei pasirinktas duomenų rinkiniai konvertuoti. Naudojant konvertuotus duomenis buvo apmokytas neuroninis tinklas su LF I, patobulintu LF I bei Adam algoritmais.

Šiame darbe buvo teoriškai išnagrinėta, jog neuroniniams tinklams yra aktualios dimensijų prakeiksmo ir persimokymo problemos. Taip pat buvo surinkta informacija apie skirtingas Liapunovo teorijas - funkciją ir eksponentes - ir paaiškintas jų veikimas bei kaip jos gali būti panaudoti neuroniniuose tinkluose.

Taigi, dimensijų prakeiksmas parodo, kad didinant dimensijų kiekį reikalingų duomenų skaičius auga eksponentiškai. Kitaip tariant, duomenų retumas didėja, kai dimensijų skaičius auga. Kita aktuali problema yra persimokymas, kai modelis išmoksta duomenyse esantį triukšmą ir pačių duomenų netikslumus. Kas lemia, kad mokymosi metu modelis pasirodo labai gerai, tačiau su naujais duomenimis-labai prastai.

Egzistuoja skirtingų būdų kaip įrodyti stabilumą - tai Liapunovo funkcija, Liapunovo eksponentė arba spektras. Siekiant įrodyti pusiausvyros taško stabilumą naudojant Liapunovo funkciją, analizuojamos sistemos funkcija arba potenciali Liapunovo funkcija turi atitikti bent dvi prielaidas:

- Potenciali Liapunovo funkcija turi turėti minimumą.
- Jeigu pradinė sąlyga nėra pusiausvyros taške, tai potenciali Liapunovo funkcija mažėja per specifinę reikšmę. Kitaip tariant, funkcija nedidėja palei sprendimo trajektoriją.

Norint, kad potenciali Liapunovo funkcija būtų griežta Liapunovo funkcija, ji turi griežtai mažėti į pusiausvyros tašką.

Kitas būdas kaip parodyti stabilumą yra naudojant Liapunovo eksponentę, kuri parodo, ar dvi arti viena kitos esančios pradinės sąlygos su laiku artėja, tolsta ar išlieka šalia viena kitos.

Kitaip tariant, Liapunovo eksponentė yra vidutinis atstumo augimo greitis tarp dviejų trajektorijų. Liapunovo spektras yra visų Liapunovo eksponenčių rinkinys.

Taigi, iš analizės, atliktos skyriuje „Liapunovo eksponentės panaudojimas" galima daryti išvadą, jog Liapunovo eksponentė gali būti sėkmingai naudojama požymių išrinkimui ir išskyrimui, jeigu yra dirbama su duomenimis, kurie apibūdina dinaminę sistemą (smegenų biosrovės matavimai, atspindžių spektrai, paveiksliukų paveikimas priešiškais trukdžiais) bei gali būti paversti į laiko eilutes. Suradus duomenų rinkinio Liapunovo eksponentes, jas galima toliau naudoti modelio apmokymui kaip požymius apibūdinančius duomenis ir taip sumažinant dimensijų kiekį bei išvengiant dimensijų prakeiksmo.

Šio darbo eksperimentinėje dalyje buvo atlikti skirtingi bandymai, paremti [BKP06] ir [Rav15] straipsniuose pateiktais Liapunovo funkciją naudojančiais algoritmais ir [PDW18] ir [PDB20] straipsniuose pateiktu priešiškų trukdžių nustatymo metodu bei [YGJ12] ir [ÜG07] straipsniuose aprašytu Liapunovo eksponenčių panaudojimu neuroninių tinklų apmokymui. Atlikus šiuos eksperimentus buvo gauti tokie rezultatai:

1. Tinklas su LF I, patobulintu LF I ir Adam algoritmais sugeba pilnai išspręsti XOR uždavinį (4-21 pav. bei 3 ir 4 lentelės), kas parodo, kad pavyko atkartoti [BKP06] ir [Rav15] straipsnių bandymus. Taip pat modelis su LF I, patobulintu LF I ir Adam algoritmais sugeba klasifikuoti didesnio dydžio paveiksliukus paveiktus priešiškų trukdžių (plačiau „Duomenų rinkinys" skyriuje). Tai galima matyti iš 22-37 paveiksliukų bei 5 ir 6 lentelių. Iš jų matyti, kad po tinklo suderinimų (50 ir 300 epochų) geriausias vertinimo metrikas pasiekia patobulintas LF I algoritmas (išskyrus po 50 epochų Adam pasiekia geresnę atšaukimo metrikos reikšmę bei po 300 epochų Adam pasiekia tokia pačią preciziją).
2. Po 50 iteracijų sprendžiant XOR uždavinį (40 ir 41 pav.) matyti, kad konvergavimo sąlygą (nuostolio funkcija per epochą sumažėja iki 0.0001) pasiekia visi algoritmai. Tačiau pagal gautus vidurkius pateiktus 7 lentelėje, LF I ir Adam pasiekia greičiau - vidutiniškai per 127.44 ir 386.26 epochų atitinkamai, o patobulintas LF I - per 2184.8 epochas. Bet vidutiniškai patobulintas LF I algoritmas pasiekia 0.9 tikslumą greičiausiai - per 19.92 epochas, kai LF I algoritmui reikia 21.7, o Adam - 26.82 epochų. Tačiau po 10 iteracijų klasifikuojant priešiškų trukdžių paveiktus paveiksliukus (42 ir 43 pav.) buvo nustatyta, jog Adam algoritmas pasiekia konvergavimo sąlygą (nuostolio funkcija per epochą sumažėja iki 0.001) žymiai greičiau negu LF I ar patobulintas LF I algoritmai. Pagal vidurkius pateiktus 8 lentelėje matyti, kad Adam pasiekia konvergavimo sąlygą vidutiniškai per 24.5 epochas, o LF I ir patobulintas LF I per 228.7 ir 890.2 epochas atitinkamai. Taip pat Adam greičiau pasiekia 0.8 ir 0.9 tikslumą - per

3.2 ir 10.7 epochas atitinkamai. Tuo tarpu tokius pačius tikslumus LF I pasiekia per 56.8 ir 159.4 epochas atitinkamai, o patobulintas LF I per 17.2 ir 58 epochas.

3. Bandant atkartoti bandymus atliktus [PDW18] ir [PDB20] straipsniuose buvo atakuoti MNIST ir CIFART duomenų rinkiniai bei jiems apskaičiuotos Liapunovo eksponentės. Iš gautų rezultatų (44-51 pav.) matyti, kad pirmosios dvi Liapunovo eksponentes daugumoje atveju aiškiai sugrupuoja originalius ir priešiškų trukdžių paveiktus duomenis. Tačiau grupių persidengimas didėja sudėtingėjant duomenų rinkiniams ir naudojant sudarytą duomenų rinkinį abi grupės buvo pilnai persidengusios (52, 53 ir 56 pav.). Buvo pastebėta, kad rezultatai neatitinka [PDW18] straipsnyje padarytos išvados - duomenys paveikti priešiškų trukdžių turėtų turėti labiau vyraujančias teigiamas Liapunovo eksponentes. Tačiau [PDB20] straipsnyje, kuriame buvo analizuotas ne vien MNIST duomenų rinkinys buvo padaryta išvada, kad Liapunovo eksponentės yra labai jautrios atakos paveiktų pikselių kiekiui. Ši išvada buvo išbandyta apskaičiavus pakeistų pikselių kiekius sudarytam duomenų rinkiniui - spalvotų paveiksliukų pikselių buvo pakeisti 98.54 procentai, o juos konvertavus į pilkus - 44.39 procentai. Pagal 52 ir 56 paveiksliukus matyti, kad pilkų paveiksliukų Liapunovo eksponentės yra mažiau susigrupavusios. Svarbu pabrėžti, kad apskaičiavus dvi Liapunovo eksponentes pilkiems paveiksliukams, jų susigrupavimas tampa panašus į spalvotų paveiksliukų.
4. Naudojant Liapunovo eksponenčių rinkinius gautus iš originalių ir priešiškų trukdžių paveiktų MNIST ir CIFAR duomenų rinkinių bei sudaryto duomenų rinkinio pavyko apmokyti tinklą su LF I, patobulintu LF I ir Adam algoritmais. Po apmokymo gautos metrikos pateiktos 9, 10 ir 11 lentelėse, iš kurių matyti, kad gautas modelio tikslumas priklausė nuo to kaip stipriai buvo sugrupuotos Liapunovo eksponentės gautos iš originalių ir paveiktų paveiksliukų. Tinklas su LF I per visus bandymus su Liapunovo eksponentėmis vidutiniškai pasiekė 0.63 tikslumą, su patobulintu LF I vidutiniškai pasiekė 0.78 tikslumą, o su Adam pasiekė - 0.75. Taigi, patobulintas LF I algoritmas vidutiniškai yra labiau tinkamas modelio apmokymui su Liapunovo eksponentėmis - jei Liapunovo eksponentės yra persidengusios ir neturi aiškių grupių, šis algoritmas pasiekia aukštesnį tikslumą negu kiti algoritmai.

Išvados gautos atlikus šį darbą:

1. Tinklas su LF I, patobulintu LF I ir Adam algoritmais gali spręsti XOR ir priešiškų trukdžių paveiktų paveiksliukų klasifikacijos uždavinius. Tačiau Adam algoritmas pastarąją užduotį sprendžia efektyviau nei LF I ir patobulintas LF I algoritmai, bet yra jautresnis pasirinkto pradinio mokymosi žingsnio dydžiui.

2. Liapunovo eksponenčių rinkinius galima naudoti neuroninio tinklo apmokymui ir vidutiniškai geriausias metrikas pasiekia tinklas su patobulintu LF I algoritmu palyginus su LF I ir Adam algoritmais. Tačiau tinklo pasiekiamas tikslumas priklauso nuo Liapunovo eksponenčių gautų iš originalių ir priešiškų trukdžių paveiktų paveiksliukų susigrupavimo aiškumo - kuo aiškiau Liapunovo eksponentės yra susigrupavusios, tuo aukštesnį tikslumą tinklas gali pasiekti.

Literatūra

[ACS+18] Triantafyllos Afouras, Joon Son Chung, Andrew W. Senior, Oriol Vinyals ir Andrew Zisserman. Deep audio-visual speech recognition. *CoRR*, abs/1809.02108, 2018. arXiv: 1809.02108. URL: http://arxiv.org/abs/1809.02108.

[AD19] Zahid Akhtar ir Dipankar Dasgupta. A brief survey of adversarial machine learning and defense strategies, 2019.

[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.

[BA18] Avishek Joey Bose ir Parham Aarabi. Adversarial attacks on face detectors using neural net based constrained optimization. *CoRR*, abs/1805.12302, 2018. arXiv: 1805.12302. URL: http://arxiv.org/abs/1805.12302.

[BDH12] P. Blanchard, R.L. Devaney ir G.R. Hall. *Differential Equations*. Cengage Learning, 2012. Skyr. A Population Model for Two Competing Species. ISBN: 9781133388081.

[Bel61] Richard E. Bellman. *Adaptive Control Processes*. Princeton University Press, Princeton, 1961. ISBN: 978-1-4008-7466-8. URL: https://www.degruyter.com/view/title/522011.

[BKP06] L. Behera, S. Kumar ir A. Patnaik. On adaptive learning rate that guarantees convergence in feedforward networks. *IEEE Transactions on Neural Networks*, 17(5):1116–1125, 2006. DOI: 10.1109/TNN.2006.878121.

[Cel16] N. Celebi. *Image Mining*. Advances in data mining and database management (ADMDM) book series. IGI Global, 2016. Skyr. 4, p. 71–72. ISBN: 9781522500759. URL: https://books.google.lt/books?id=a-NAjwEACAAJ.

[CHH+19] Sizhe Chen, Xiaolin Huang, Zhengbao He ir Chengjin Sun. Damagenet: A universal adversarial dataset. *CoRR*, abs/1912.07160, 2019. arXiv: 1912.07160. URL: http://arxiv.org/abs/1912.07160.

[CKK+00] F. Colonius, W.H. Kliemann, W. Kliemann ir L. Grüne. *The Dynamics of Control*. Systems & Control: Foundations & Applications. Birkhäuser Boston, 2000. Skyr. Dynamics, Perturbations, and Control. ISBN: 9780817636838. URL: https://books.google.lt/books?id=MSPNFJd1%5C_PcC.

[COH+99] Ricardo Carretero-Gonzalez, S. Orstavik, J. Huke, David Broomhead ir J. Stark. Scaling and interleaving of subsystem lapunov exponents for spatio-temporal systems. *Chaos (Woodbury, N.Y.)*, 9:466–482, 1999-07. DOI: 10.1063/1.166420.

[CZY+16] Z. Chen, F. Zhong, X. Yuan ir Y. Hu. Framework of integrated big data: a review. *2016 IEEE International Conference on Big Data Analysis (ICBDA)*, p. 1–5, 2016.

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

[DB13] Guozhu Dong ir James Bailey. *Contrast Data Mining: Concepts, Algorithms, and Applications*. 2013. Skyr. Overfitting Avoidance by CP-Based Approaches. ISBN: 9781439854327.

[EBF18] Sacha Epskamp, Denny Borsboom ir Eiko I. Fried. Estimating psychological networks and their accuracy: a tutorial paper. *Behavior Research Methods*, 50(1):195–212, 2018-02. ISSN: 1554-3528. DOI: 10.3758/s13428-017-0862-1. URL: https://doi.org/10.3758/s13428-017-0862-1.

[EKC87] Jean-Pierre Eckmann, Sylvie Kamphorst ir Sergio Ciliberto. Liapunov exponents from time series. *Physical review. A*, 34:4971–4979, 1987-01. DOI: 10.1103/PhysRevA.34.4971.

[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.

[Far06] S.J. Farlow. *An Introduction to Differential Equations and Their Applications*. Dover Books on Mathematics. Dover Publications, 2006. Skyr. Nonlinear Differential Equations and Chaos. ISBN: 9780486445953.

[Fuk80] Kunihiko Fukushima. Neocognitron: a self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. 36:193–202, 1980-02.

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

[Gra13] D. Graupe. *Principles of Artificial Neural Networks*. Advanced series in circuits and systems. World Scientific, 2013. Skyr. Hopfield Networks. ISBN: 9789814522748. URL: https://books.google.lt/books?id=W6W6CgAAQBAJ.

[Grü07] Peter Grünwald. *The Minimum Description Length Principle*. 2007-01. DOI: 10.7551/mitpress/4643.001.0001.

[GSM+18] Yuan Gao, Qi She, Jiayi Ma, Mingbo Zhao, Wei Liu ir Alan L. Yuille. NDDR-CNN: layer-wise feature fusing in multi-task CNN by neural discriminative dimensionality reduction. *CoRR*, abs/1801.08297, 2018. arXiv: 1801.08297. URL: http://arxiv.org/abs/1801.08297.

[Gun06] Samuel Gundry. *Minimum Message Length Model Selection for Autoregressive Conditional Heteroskedastic Time Series*. Disertacija, Monash University, 2006.

[HG15] Sigurður Hafstein ir Peter Giesl. Computational methods for lyapunov functions. *Discrete and Continuous Dynamical Systems - Series B*, 20, 2015-08. DOI: 10.3934/dcdsb.2015.20.8i.

[HKY19] A. Habib, C. Karmakar ir J. Yearwood. Impact of ecg dataset diversity on generalization of cnn model for detecting qrs complex. *IEEE Access*, 7:93275–93285, 2019. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2019.2927726.

[HLN+98] J. Hilgert, J.D. Lawson, K.H. Neeb ir E.B. Vinberg. *Positivity in Lie Theory: Open Problems*. De Gruyter expositions in mathematics. Walter de Gruyter, 1998. ISBN: 9783110161120. URL: https://books.google.lt/books?id=C89Kg-d7sgsC.

[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.

[YGJ12] J. Yin, C. Gao ir X. Jia. Using hurst and lyapunov exponent for hyperspectral image feature extraction. *IEEE Geoscience and Remote Sensing Letters*, 9(4):705–709, 2012.

[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.

[KB17] Diederik P. Kingma ir Jimmy Ba. Adam: a method for stochastic optimization, 2017. arXiv: 1412.6980 [cs.LG].

[Kho12] M. Khosrowpour. *Dictionary of Information Science and Technology*. Information Science Reference, 2012. ISBN: 9781466626744.

[KR18] Sandeep Kumar ir Santosh Singh Rathore. *Software Fault Prediction - A Road Map*. Springer Briefs in Computer Science. Springer, 2018. ISBN: 978-981-10-8714-1. DOI: 10.1007/978-981-10-8715-8. URL: https://doi.org/10.1007/978-981-10-8715-8.

[KSH12] Alex Krizhevsky, Ilya Sutskever ir Geoffrey Hinton. Imagenet classification with deep convolutional neural networks. *Neural Information Processing Systems*, 25, 2012-01. DOI: 10.1145/3065386.

[KSK19] M. A. Kutlugün, Y. Sirin ir M. Karakaya. The effects of augmented training dataset on performance of convolutional neural networks in face recognition system. *2019 Federated Conference on Computer Science and Information Systems (FedCSIS)*, p. 929–932, 2019-09. DOI: 10.15439/2019F181.

[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.

[Liu17] Y. Liu. *Python Machine Learning By Example*. Packt Publishing, 2017. ISBN: 9781783553129. URL: https://books.google.lt/books?id=0nc5DwAAQBAJ.

[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.

[LV07] John A. Lee ir Michel Verleysen, red. *Nonlinear Dimensionality Reduction*. Springer New York, 2007. DOI: 10.1007/978-0-387-39351-3. URL: https://doi.org/10.1007%2F978-0-387-39351-3.

[NE19] Yağız Nalçakan ir Tolga Ensari. Decision of neural networks hyperparameters with a population-based algorithm. Giuseppe Nicosia, Panos Pardalos, Giovanni Giuffrida, Renato Umeton ir Vincenzo Sciacca, redaktoriai, *Machine Learning, Optimization, and Data Science*, p. 276–281. Springer International Publishing, 2019. ISBN: 978-3-030-13709-0.

[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.

[PDB20] Anibal Pedraza, Oscar Deniz ir Gloria Bueno. Approaching adversarial example classification with chaos theory. *Entropy*, 22(11), 2020. ISSN: 1099-4300. DOI: 10.3390/e22111201. URL: https://www.mdpi.com/1099-4300/22/11/1201.

[PDW18] Vinay Uday Prabhu, Nishant Desai ir John Whaley. On lyapunov exponents and adversarial perturbation. *CoRR*, abs/1802.06927, 2018. arXiv: 1802.06927. URL: http://arxiv.org/abs/1802.06927.

[PFC+18] Nicolas Papernot, Fartash Faghri, Nicholas Carlini, Ian Goodfellow ir k.t. Technical report on the cleverhans v2.1.0 adversarial examples library. *arXiv preprint arXiv:1610.00768*, 2018.

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

[PMG+16] Nicolas Papernot, Patrick D. McDaniel, Ian J. Goodfellow, Somesh Jha, Z. Berkay Celik ir Ananthram Swami. Practical black-box attacks against deep learning systems using adversarial examples. *CoRR*, abs/1602.02697, 2016. arXiv: 1602.02697. URL: http://arxiv.org/abs/1602.02697.

[PPB09] Ana Porto-Pazos, Alejandro Pazos ir Washington Buño. *Advancing Artificial Intelligence Through Biological Process Applications*. 2009-01. ISBN: 978-1-59904-996-0. DOI: 10.4018/978-1-59904-996-0.

[PW17] Luis Perez ir Jason Wang. The effectiveness of data augmentation in image classification using deep learning, 2017-12.

[RAA67] W.N. Redisch, United States. National Aeronautics ir Space Administration. *Equilibrium Analysis and Its Application to Attitude Control Systems*. NASA technical report. National Aeronautics ir Space Administration, 1967. Skyr. Mathematical Development for General Systems.

[Rao19] Dattaraj Jagdish Rao. *Handling unstructured data. Keras to Kubernetes®*. John Wiley & Sons, Ltd, 2019. Skyr. 3, p. 71–109. ISBN: 9781119564843. DOI: 10.1002/9781119564843.ch3.

[Rav15] Udhaya Ravishankar. A lyapunov based adaptive and stable neural network weight regularization algorithm. *Neural, Parallel and Scientific Computations*, 23:343–356, 2015-01.

[RDS+15] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause ir k.t. Imagenet large scale visual recognition challenge, 2015. arXiv: 1409.0575 [cs.CV].

[REH+19] Francisco Rodrigues, Mateus Espadoto, Roberto Hirata ir Alexandru Telea. Constructing and visualizing high-quality classifier decision boundary maps. *Information*, 10:280, 2019-09. DOI: 10.3390/info10090280.

[RPA+18] Reza Reiazi, Reza Paydar, Ali Abbasian Ardakani ir Maryam Etedadialiabadi. Mammography lesion detection using faster r-cnn detector:111–115, 2018-01.

[RS17] Thaqif Rajab ir Roselina Salleh. Classıfıcatıon of dıabetes dısease usıng backpropagatıon and radıal basıs functıon 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.

[RZQ+20] Kui Ren, Tianhang Zheng, Zhan Qin ir Xue Liu. Adversarial attacks and defenses in deep learning. *Engineering*, 6(3):346–360, 2020. ISSN: 2095-8099. DOI: https://doi.org/10.1016/j.eng.2019.12.012. URL: http://www.sciencedirect.com/science/article/pii/S209580991930503X.

[Sch19] Christopher Schölzel. Nonlinear measures for dynamical systems, versija 0.5.2, 2019-06. DOI: 10.5281/zenodo.3814723. URL: https://doi.org/10.5281/zenodo.3814723.

[Shi12] D. Shiffman. *The Nature of Code*. D. Shiffman, 2012. Skyr. 10. ISBN: 9780985930806. URL: https://books.google.lt/books?id=hoK6lgEACAAJ.

[SHK+14] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever ir Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014. URL: http://jmlr.org/papers/v15/srivastava14a.html.

[Sin18] Arvind Kumar Sinha. Interweaving convolutions : an application to audio classification harsh. 2018.

[SK17] Torgyn Shaikhina ir Natasha A. Khovanova. Handling limited datasets with neural networks in medical applications: a small-data approach. *Artificial intelligence in medicine*, 75:51–63, 2017.

[SKC18] Pouya Samangouei, Maya Kabkab ir Rama Chellappa. Defense-gan: protecting classifiers against adversarial attacks using generative models. *CoRR*, abs/1805.06605, 2018. arXiv: 1805.06605. URL: http://arxiv.org/abs/1805.06605.

[Sku14] Ewa Skubalska-Rafajłowicz. Small sample size in high dimensional space - minimum distance based classification. Leszek Rutkowski, Marcin Korytkowski, Rafał Scherer, Ryszard Tadeusiewicz, Lotfi A. Zadeh ir Jacek M. Zurada, redaktoriai, *Artificial Intelligence and Soft Computing*, p. 610–621, Cham. Springer International Publishing, 2014. ISBN: 978-3-319-07173-2.

[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.

[SM14] Hari Seetha ir M. Narasimha Murty. Efficient high dimensional data classification. 2014.

[Soy20] Derya Soydaner. A comparison of optimization algorithms for deep learning. *International Journal of Pattern Recognition and Artificial Intelligence*, 34, 2020-02. DOI: 10.1142/S0218001420520138.

[Spr10] J.C. Sprott. *Elegant Chaos: Algebraically Simple Chaotic Flows*. World Scientific, 2010. Skyr. Fundamentals. ISBN: 9789812838827. URL: https://books.google.lt/books?id=buILBDre9S4C.

[SRM96] Paul Smolensky, David E. Rumelhart ir Michael C. Mozer. *Mathematical Perspectives on Neural Networks*. L. Erlbaum Associates Inc., USA, 1st leid., 1996. Skyr. Overview: Dynamical Perspectives on Neural Networks. ISBN: 0805812016.

[SSS+09] Rolf Sint, Stephanie Stroka, Sebastian Schaffert ir Roland Ferstl. Combining unstructured, fully structured and semi-structured information in semantic wikis. 2009-01.

[Ste18] N. Stergiou. *Nonlinear Analysis for Human Movement Variability*. CRC Press, 2018. Skyr. Lyapunov Exponent. ISBN: 9781315360089. URL: https://books.google.lt/books?id=nat-DwAAQBAJ.

[SZ15] Karen Simonyan ir Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2015.

[Tay06] B.J. Taylor. *Methods and Procedures for the Verification and Validation of Artificial Neural Networks*. Springer US, 2006. ISBN: 9780387294858. URL: https://books.google.lt/books?id=ax3Q%5C_YBuXFEC.

[TAL16] Sasha Targ, Diogo Almeida ir Kevin Lyman. Resnet in resnet: generalizing residual architectures. *CoRR*, abs/1603.08029, 2016. arXiv: 1603.08029. URL: http://arxiv.org/abs/1603.08029.

[ÜG07] Elif Derya Übeyli ir Inan Güler. Statistics over lyapunov exponents for feature extraction: electroencephalographic changes detection case. *World Academy of Science, Engineering and Technology, International Journal of Medical, Health, Biomedical, Bioengineering and Pharmaceutical Engineering*, 1:134–137, 2007.

[VFS+03] M. Verleysen, D. Francois, G. Simon ir V. Wertz. On the effects of dimensionality on data analysis with neural networks. *Artificial Neural Nets Problem Solving Methods*, p. 105–112. Springer Berlin Heidelberg, 2003. DOI: 10.1007/3-540-44869-1_14. URL: https://doi.org/10.1007%2F3-540-44869-1_14.

[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.

[vKwM+21] Hugo van Kemenade, wiredfool, Andrew Murray, Alex Clark ir k.t. Python-pillow/pillow: 8.1.0, versija 8.1.0, 2021-01. DOI: 10.5281/zenodo.4411934. URL: https://doi.org/10.5281/zenodo.4411934.

[VM14] D.D. Vecchio ir R.M. Murray. *Biomolecular Feedback Systems*. Princeton University Press, 2014. Skyr. Analysis of Dynamic Behavior. ISBN: 9781400850501.

[Web10] Geoffrey I. Webb. *Overfitting*. *Encyclopedia of Machine Learning*. Claude Sammut ir Geoffrey I. Webb, redaktoriai. Springer US, Boston, MA, 2010, p. 744–744. ISBN: 978-0-387-30164-8. DOI: 10.1007/978-0-387-30164-8_623. URL: https://doi.org/10.1007/978-0-387-30164-8_623.

[WLW+18] Derek Wang, Chaoran Li, Sheng Wen, Yang Xiang, Wanlei Zhou ir Surya Nepal. Defensive collaborative multi-task training - defending against adversarial attack towards deep neural networks. *CoRR*, abs/1803.05123, 2018. arXiv: 1803.05123. URL: http://arxiv.org/abs/1803.05123.

[Wój19] Marcin Wójcik Piotr Iwoand Kurdziel. Training neural networks on high-dimensional data using random projection. *Pattern Analysis and Applications*, 22(3):1221–1231, 2019-08. ISSN: 1433-755X. DOI: 10.1007/s10044-018-0697-0. URL: https://doi.org/10.1007/s10044-018-0697-0.

[WSS+85] Alan Wolf, Jack Swift, Harry Swinney ir John Vastano. Determining lyapunov exponents from a time series. *Physica D: Nonlinear Phenomena*, 16:285–317, 1985-07. DOI: 10.1016/0167-2789(85)90011-9.

[XZG+18] Qi Xu, Ming Zhang, Zonghua Gu ir Gang Pan. Overfitting remedy by sparsifying regularization on fully-connected layers of cnns. *Neurocomputing*, 2018-08. DOI: 10.1016/j.neucom.2018.03.080.

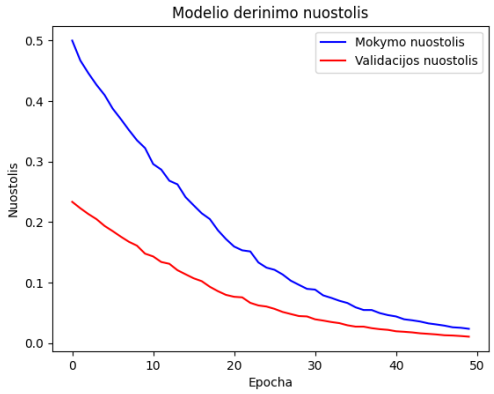
[ZGD03] Qi-Jun Zhang, K. C. Gupta ir V. K. Devabhaktuni. Artificial neural networks for rf and microwave design - from theory to practice. *IEEE Transactions on Microwave Theory and Techniques*, 51(4):1339–1350, 2003-04. ISSN: 0018-9480. DOI: 10.1109/TMTT.2003.809179.

[ZHW10] J. Zhang, H. Huang ir J. Wang. Manifold learning for visualizing and analyzing high-dimensional data. *IEEE Intelligent Systems*, 25(4):54–61, 2010.

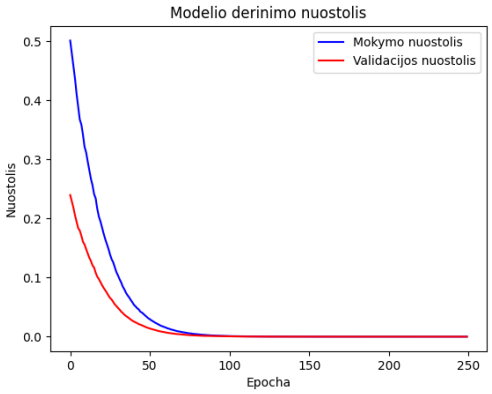
Priedas nr. 1

Papildomi Liapunovo funkcijos bandymų grafikai

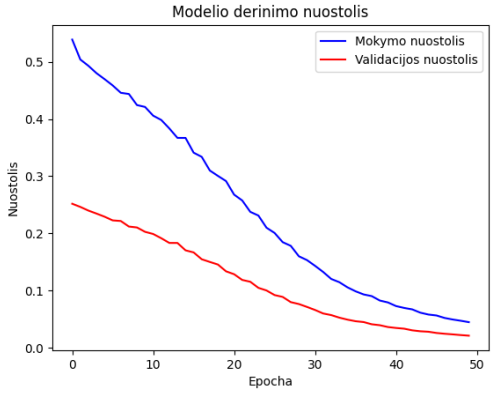
Šiame skyriuje pateiktos neuroninių tinklų praplėstų su LF I ir patobulintu LF I bei Adam algoritmais nuostolio funkcijos grafikai sprendžiant XOR ir paveiksliukų klasifikacimo uždavinius. Gauti duomenys atitinka skyriuje „Liapunovo funkcijos algoritmai“ pateiktus tikslumo grafikus, klasifikavimo lenteles bei nuostolio funkcijos ir mokymosi greičio grafikus.



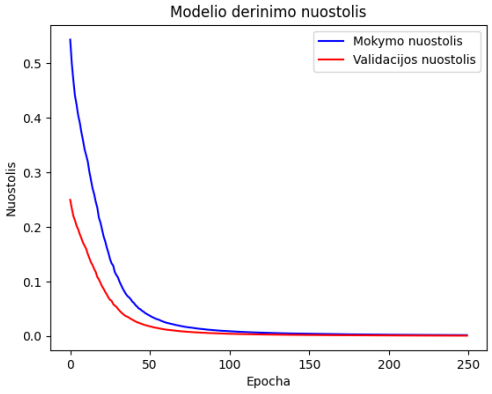
57 pav. Modelio su LF I algoritmu nuostolio funkcijos grafikas per 50 epochų sprendžiant XOR uždavinį



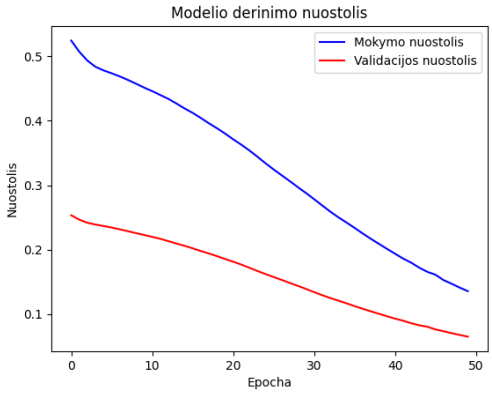
58 pav. Modelio su LF I algoritmu nuostolio funkcijos grafikas per 250 epochų sprendžiant XOR uždavinį



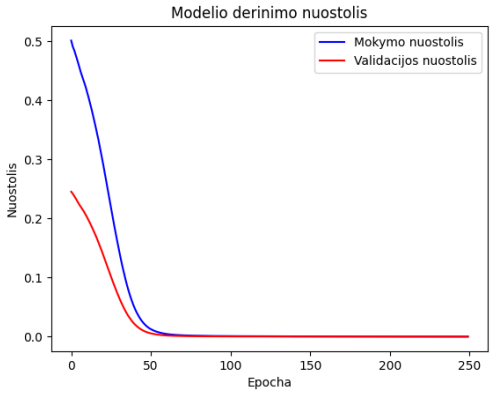
59 pav. Modelio su patobulintu LF I algoritmu nuostolio funkcijos grafikas per 50 epochų sprendžiant XOR uždavinį



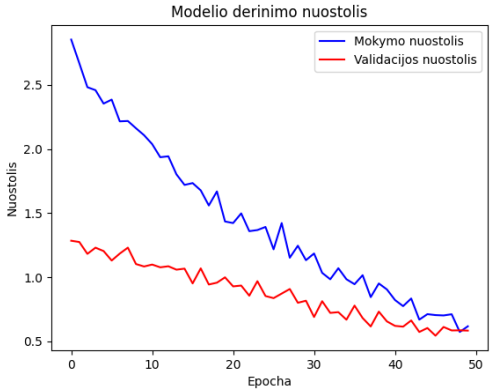
60 pav. Modelio su patobulintu LF I algoritmu nuostolio funkcijos grafikas per 250 epochų sprendžiant XOR uždavinį



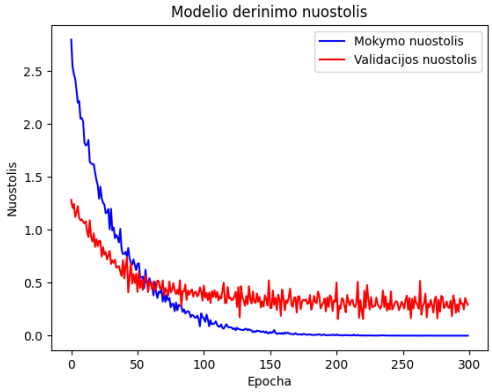
61 pav. Modelio su Adam algoritmu nuostolio funkcijos grafikas per 50 epochų sprendžiant XOR uždavinį



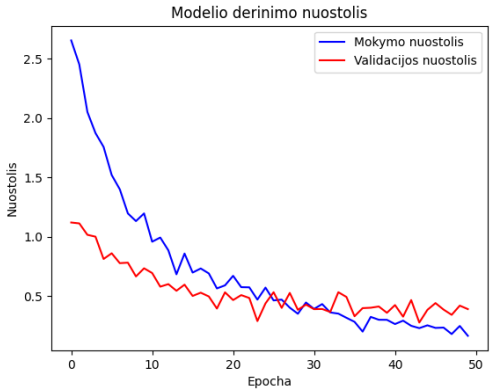
62 pav. Modelio su Adam algoritmu nuostolio funkcijos grafikas per 250 epochų sprendžiant XOR uždavinį



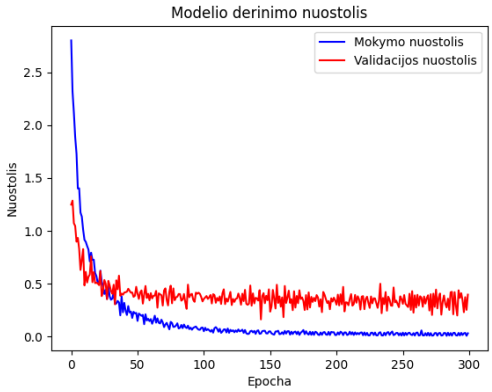
63 pav. Modelio su LF I algoritmu nuostolio funkcijos grafikas per 50 epochų klasifikuojant paveiksliukus



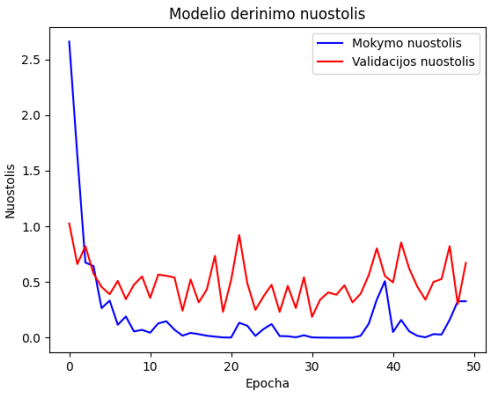
64 pav. Modelio su LF I algoritmu nuostolio funkcijos grafikas per 300 epochų klasifikuojant paveiksliukus



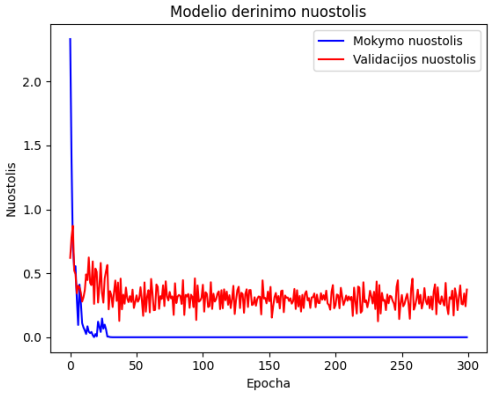
65 pav. Modelio su patobulintu LF I algoritmu nuostolio funkcijos grafikas per 50 epochų klasifikuojant paveiksliukus



66 pav. Modelio su patobulintu LF I algoritmu nuostolio funkcijos grafikas per 300 epochų klasifikuojant paveiksliukus



67 pav. Modelio su Adam algoritmu nuostolio funkcijos grafikas per 50 epochų klasifikuojant paveiksliukus

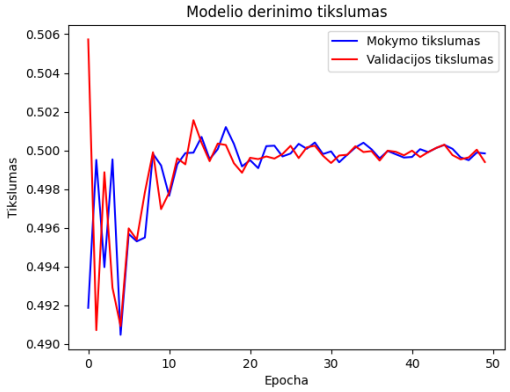


68 pav. Modelio su Adam algoritmu nuostolio funkcijos grafikas per 300 epochų klasifikuojant paveiksliukus

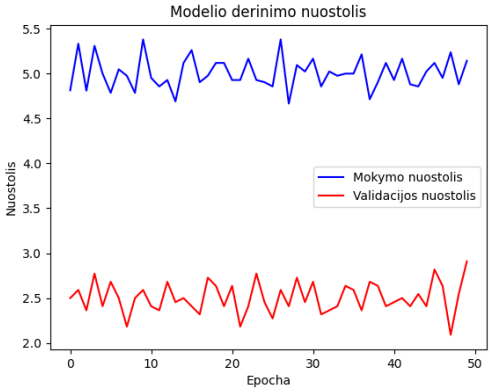
Priedas nr. 2

Tinklo su Adam algoritmu derinimas su skirtingais mokymosi žingsniais

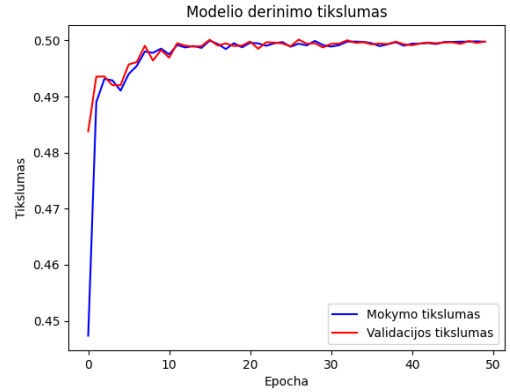
Šiame skyriuje pateikti tikslumo ir nuostolio funkcijų grafikai, kai yra derinamas tinklas su Adam optimizacijos funkcija atliekant paveiksliukų klasifikaciją per 50 epochų ir keičiant mokymosi žingsnį - 0.001, 0.0005, 0.0002. Buvo nustatyta, kad nustačius pradinį mokymosi žingsnį kaip 0.0002 ar mažiau buvo gaunami tinkami rezultatai, tad darbe buvo naudojamas 0.0001 mokymosi žingsnis.



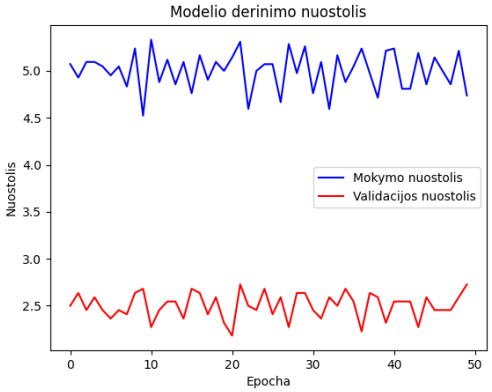
69 pav. Tikslumo grafikas, kai mokymosi žingsnis yra 0.001



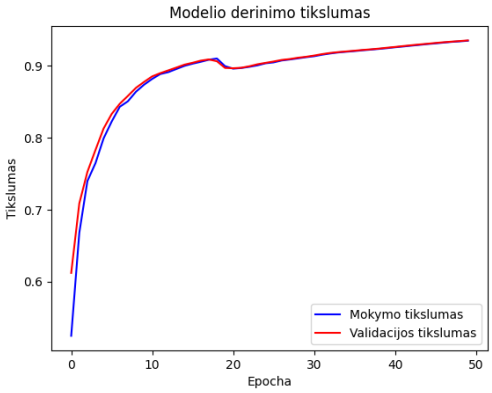
70 pav. Nuostolio funkcijos grafikas, kai mokymosi žingsnis yra 0.001



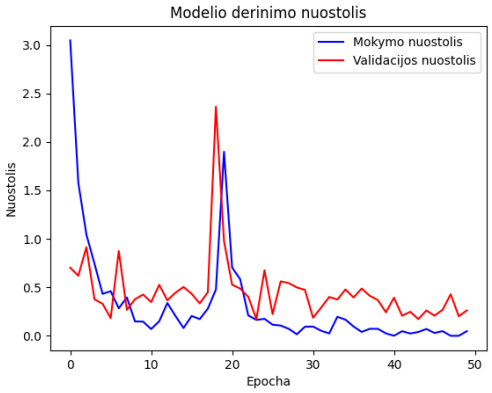
71 pav. Tikslumo grafikas, kai mokymosi žingsnis yra 0.0005



72 pav. Nuostolio funkcijos grafikas, kai mokymosi žingsnis yra 0.0005



73 pav. Tikslumo grafikas, kai mokymosi žingsnis yra 0.0002

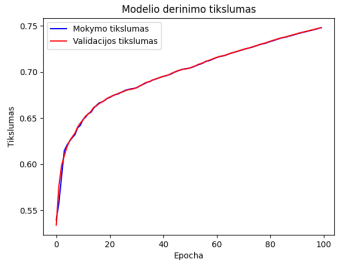


74 pav. Nuostolio funkcijos grafikas, kai mokymosi žingsnis yra 0.0002

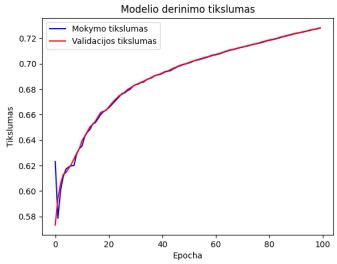
Priedas nr. 3

Tinklo apmokymo grafikai su Liapunovo eksponentėmis

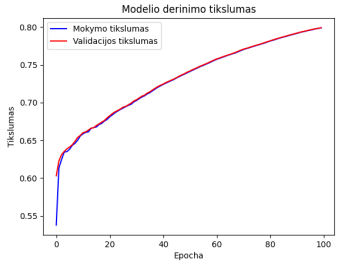
Gautos Liapunovo eksponentės buvo panaudotos paprasto tinklo apmokymui. Šiame skyriuje yra pateikti tikslumo ir nuostolio funkcijų grafikai po 100 epochų. Liapunovo eksponentės buvo gautos iš MNIST ir CIFAR duomenų rinkinių, kai jie buvo atakuoti su Carlini Wagner, PGD, FGSM bei Madry ir kiti atakomis. Taip pat šiame skyriuje pateiktos tikslumo ir nuostolio funkcijų grafikai, kai tinklas buvo apmokytas su Liapunovo eksponentėmis gautomis iš sudaryto duomenų rinkinio, kai jis buvo nepakeistas ir kaip jis buvo konvertuotas į juodai baltus paveiksliukus.



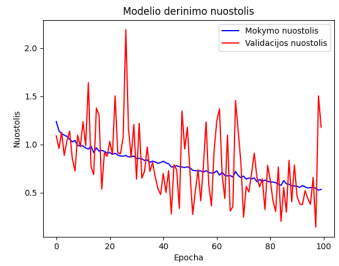
75 pav. Tinklo su LF I algoritmu tikslumo grafikas kai MNIST duomenų rinkinys paveiktas su Carlini Wagner ataka



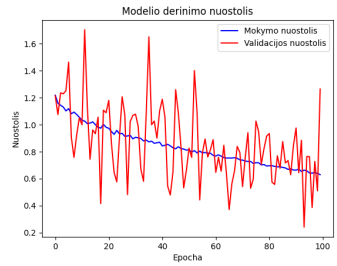
76 pav. Tinklo su patobulintu LF I algoritmu tikslumo grafikas kai MNIST duomenų rinkinys paveiktas su Carlini Wagner ataka



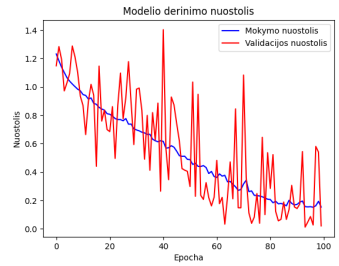
77 pav. Tinklo su Adam algoritmu tikslumo grafikas kai MNIST duomenų rinkinys paveiktas su Carlini Wagner ataka



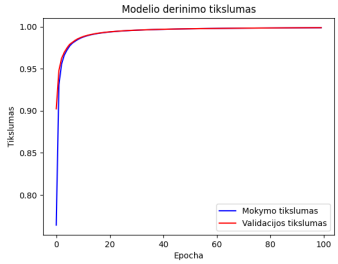
78 pav. Tinklo su LF I algoritmu nuostolio funkcijos grafikas kai MNIST duomenų rinkinys paveiktas su Carlini Wagner ataka



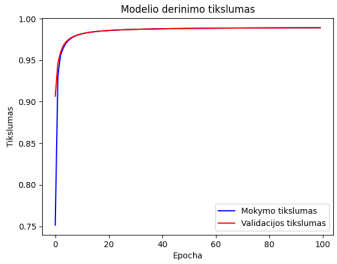
79 pav. Tinklo su patobulintu LF I algoritmu nuostolio funkcijos grafikas kai MNIST duomenų rinkinys paveiktas su Carlini Wagner ataka



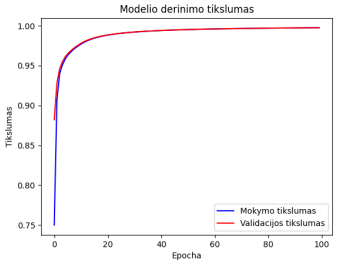
80 pav. Tinklo su Adam algoritmu nuostolio funkcijos grafikas kai MNIST duomenų rinkinys paveiktas su Carlini Wagner ataka



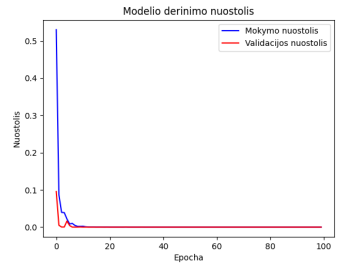
81 pav. Tinklo su LF I algoritmu tikslumo grafikas kai MNIST duomenų rinkinys paveiktas su PGD ataka



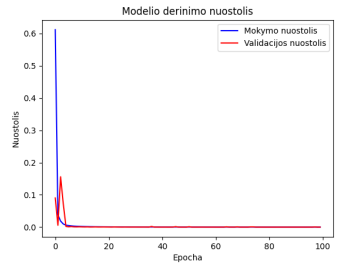
82 pav. Tinklo su patobulintu LF I algoritmu tikslumo grafikas kai MNIST duomenų rinkinys paveiktas su PGD



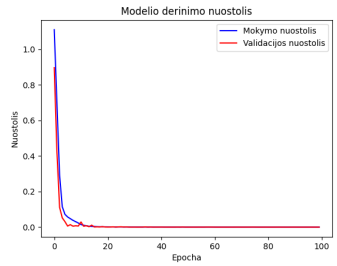
83 pav. Tinklo su Adam algoritmu tikslumo grafikas kai MNIST duomenų rinkinys paveiktas su PGD ataka



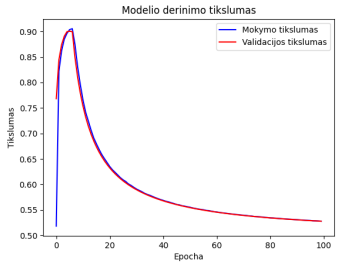
84 pav. Tinklo su LF I algoritmu nuostolio funkcijos grafikas kai MNIST duomenų rinkinys paveiktas su PGD ataka



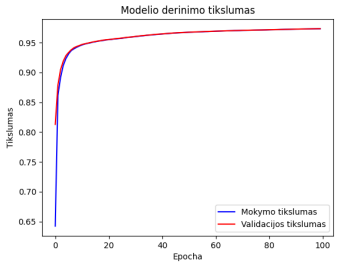
85 pav. Tinklo su patobulintu LF I algoritmu nuostolio funkcijos grafikas kai MNIST duomenų rinkinys paveiktas su PGD ataka



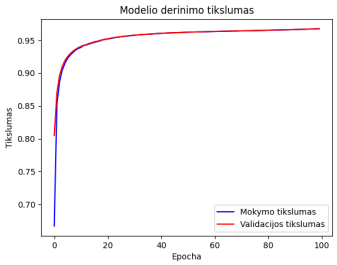
86 pav. Tinklo su Adam algoritmu nuostolio funkcijos grafikas kai MNIST duomenų rinkinys paveiktas su PGD ataka



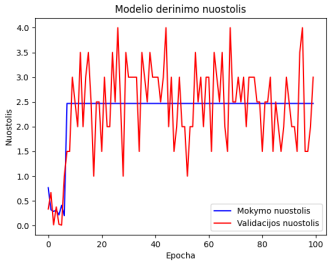
87 pav. Tinklo su LF I algoritmu tikslumo grafikas kai MNIST duomenų rinkinys paveiktas su FGSM ataka



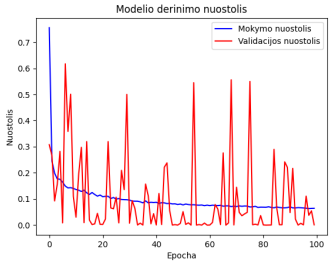
88 pav. Tinklo su patobulintu LF I algoritmu tikslumo grafikas kai MNIST duomenų rinkinys paveiktas su FGSM



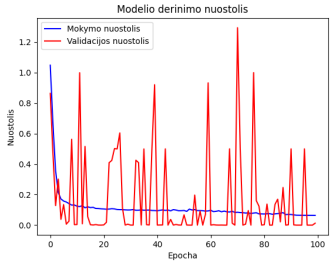
89 pav. Tinklo su Adam algoritmu tikslumo grafikas kai MNIST duomenų rinkinys paveiktas su FGSM ataka



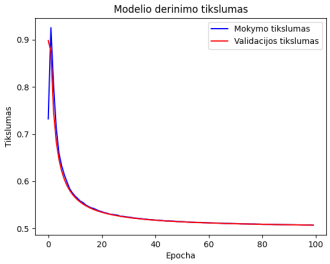
90 pav. Tinklo su LF I algoritmu nuostolio funkcijos grafikas kai MNIST duomenų rinkinys paveiktas su FGSM ataka



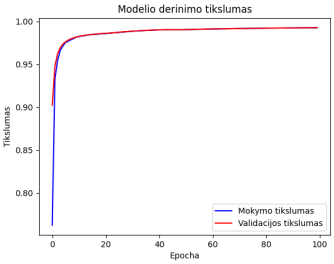
91 pav. Tinklo su patobulintu LF I algoritmu nuostolio funkcijos grafikas kai MNIST duomenų rinkinys paveiktas su FGSM ataka



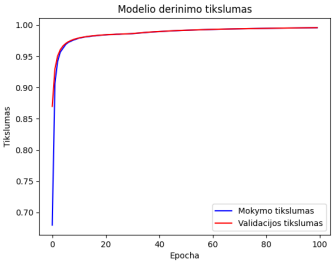
92 pav. Tinklo su Adam algoritmu nuostolio funkcijos grafikas kai MNIST duomenų rinkinys paveiktas su FGSM ataka



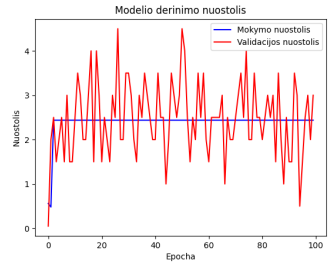
93 pav. Tinklo su LF I algoritmu tikslumo grafikas kai MNIST duomenų rinkinys paveiktas su Madry ir kiti ataka



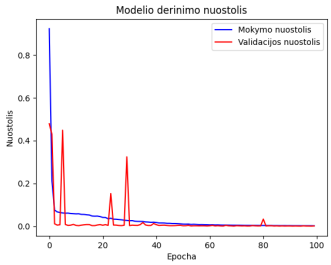
94 pav. Tinklo su patobulintu LF I algoritmu tikslumo grafikas kai MNIST duomenų rinkinys paveiktas su Madry ir kiti ataka



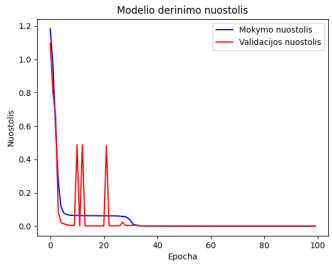
95 pav. Tinklo su Adam algoritmu tikslumo grafikas kai MNIST duomenų rinkinys paveiktas su Madry ir kiti ataka



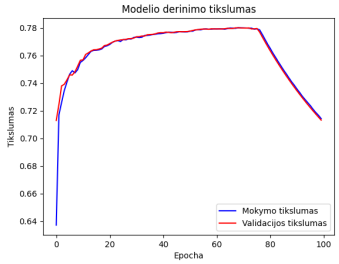
96 pav. Tinklo su LF I algoritmu nuostolio funkcijos grafikas kai MNIST duomenų rinkinys paveiktas su Madry ir kiti ataka



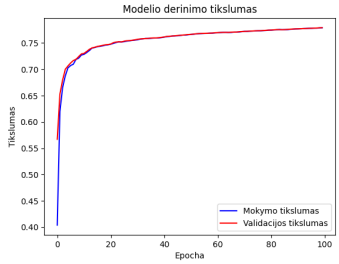
97 pav. Tinklo su patobulintu LF I algoritmu nuostolio funkcijos grafikas kai MNIST duomenų rinkinys paveiktas su Madry ir kiti ataka



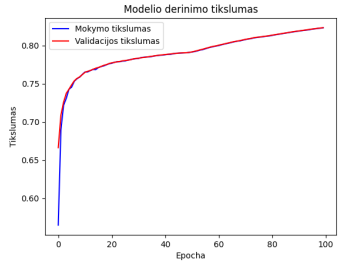
98 pav. Tinklo su Adam algoritmu nuostolio funkcijos grafikas kai MNIST duomenų rinkinys paveiktas su Madry ir kiti ataka



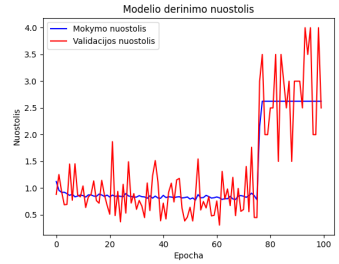
99 pav. Tinklo su LF I algoritmu tikslumo grafikas kai CIFAR duomenų rinkinys paveiktas su Carlini Wagner ataka



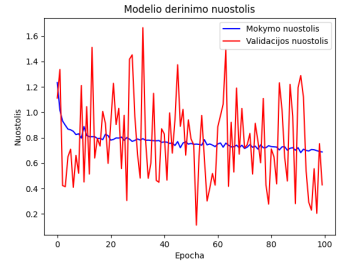
100 pav. Tinklo su patobulintu LF I algoritmu tikslumo grafikas kai CIFAR duomenų rinkinys paveiktas su Carlini Wagner ataka



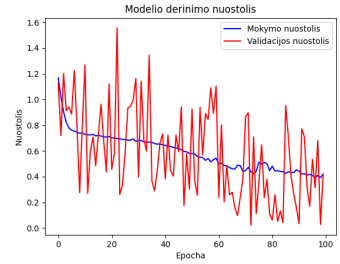
101 pav. Tinklo su Adam algoritmu tikslumo grafikas kai CIFAR duomenų rinkinys paveiktas su Carlini Wagner ataka



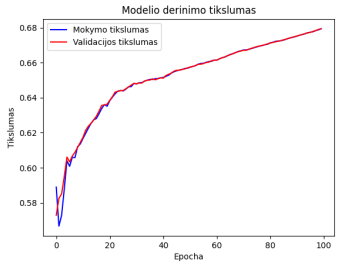
102 pav. Tinklo su LF I algoritmu nuostolio funkcijos grafikas kai CIFAR duomenų rinkinys paveiktas su Carlini Wagner ataka



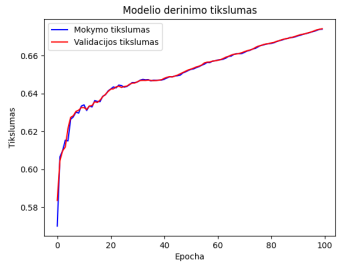
103 pav. Tinklo su patobulintu LF I algoritmu nuostolio funkcijos grafikas kai CIFAR duomenų rinkinys paveiktas su Carlini Wagner ataka



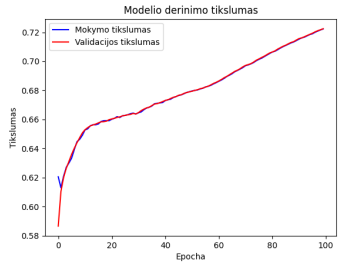
104 pav. Tinklo su Adam algoritmu nuostolio funkcijos grafikas kai CIFAR duomenų rinkinys paveiktas su Carlini Wagner ataka



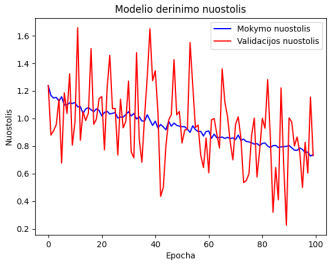
105 pav. Tinklo su LF I algoritmu tikslumo grafikas kai CIFAR duomenų rinkinys paveiktas su PGD ataka



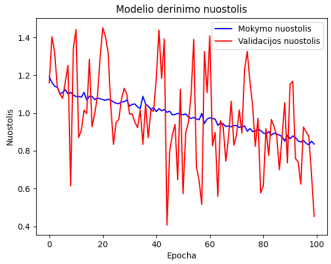
106 pav. Tinklo su patobulintu LF I algoritmu tikslumo grafikas kai CIFAR duomenų rinkinys paveiktas su PGD



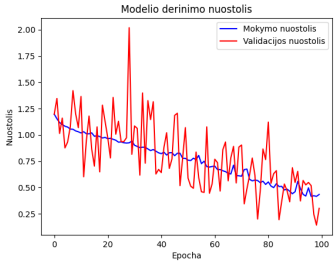
107 pav. Tinklo su Adam algoritmu tikslumo grafikas kai CIFAR duomenų rinkinys paveiktas su PGD ataka



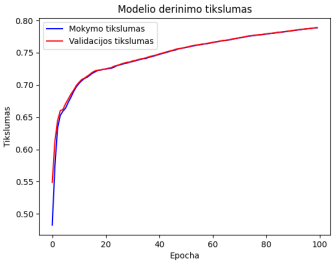
108 pav. Tinklo su LF I algoritmu nuostolio funkcijos grafikas kai CIFAR duomenų rinkinys paveiktas su PGD ataka



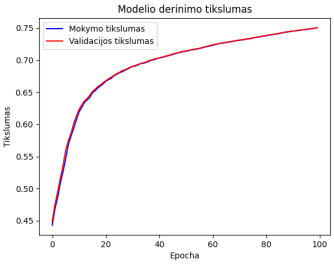
109 pav. Tinklo su patobulintu LF I algoritmu nuostolio funkcijos grafikas kai CIFAR duomenų rinkinys paveiktas su PGD ataka



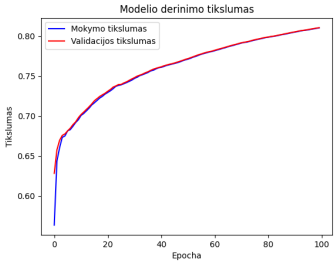
110 pav. Tinklo su Adam algoritmu nuostolio funkcijos grafikas kai CIFAR duomenų rinkinys paveiktas su PGD ataka



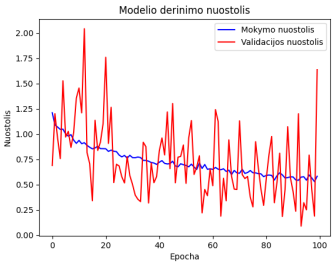
111 pav. Tinklo su LF I algoritmu tikslumo grafikas kai CIFAR duomenų rinkinys paveiktas su FGSM ataka



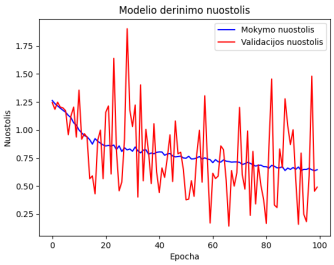
112 pav. Tinklo su patobulintu LF I algoritmu tikslumo grafikas kai CIFAR duomenų rinkinys paveiktas su FGSM



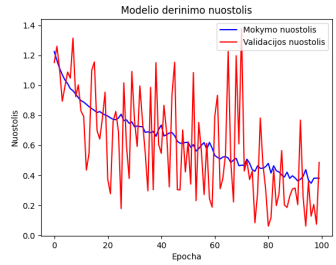
113 pav. Tinklo su Adam algoritmu tikslumo grafikas kai CIFAR duomenų rinkinys paveiktas su FGSM ataka



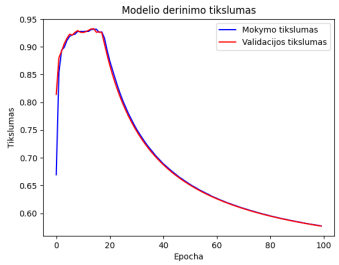
114 pav. Tinklo su LF I algoritmu nuostolio funkcijos grafikas kai CIFAR duomenų rinkinys paveiktas su FGSM ataka



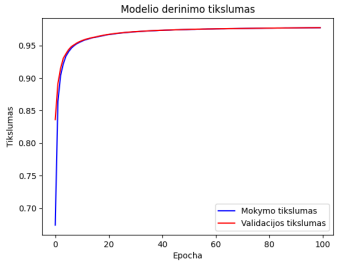
115 pav. Tinklo su patobulintu LF I algoritmu nuostolio funkcijos grafikas kai CIFAR duomenų rinkinys paveiktas su FGSM ataka



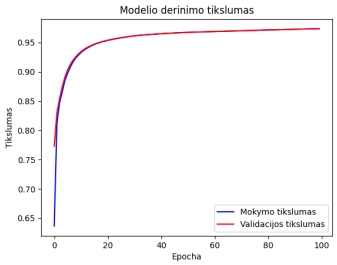
116 pav. Tinklo su Adam algoritmu nuostolio funkcijos grafikas kai CIFAR duomenų rinkinys paveiktas su FGSM ataka



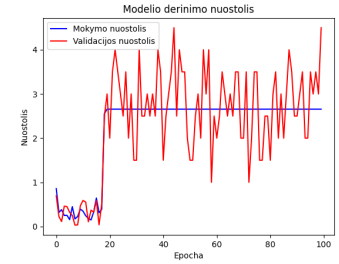
117 pav. Tinklo su LF I algoritmu tikslumo grafikas kai CIFAR duomenų rinkinys paveiktas su Madry ir kiti ataka



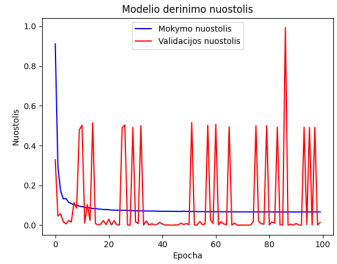
118 pav. Tinklo su patobulintu LF I algoritmu tikslumo grafikas kai CIFAR duomenų rinkinys paveiktas su Madry ir kiti ataka



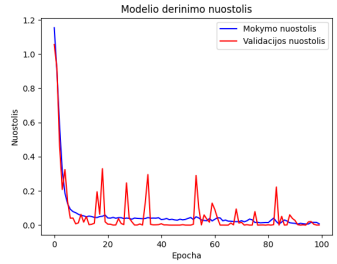
119 pav. Tinklo su Adam algoritmu tikslumo grafikas kai CIFAR duomenų rinkinys paveiktas su Madry ir kiti ataka



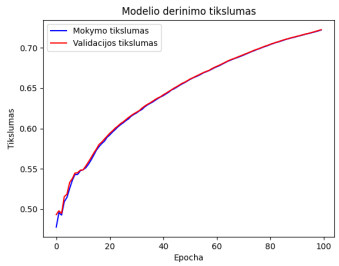
120 pav. Tinklo su LF I algoritmu nuostolio funkcijos grafikas kai CIFAR duomenų rinkinys paveiktas su Madry ir kiti ataka



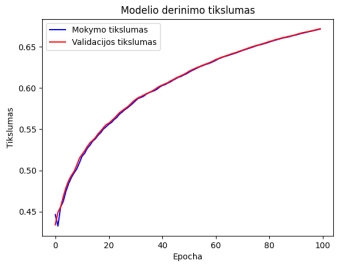
121 pav. Tinklo su patobulintu LF I algoritmu nuostolio funkcijos grafikas kai CIFAR duomenų rinkinys paveiktas su Madry ir kiti ataka



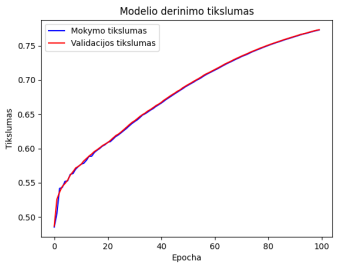
122 pav. Tinklo su Adam algoritmu nuostolio funkcijos grafikas kai CIFAR duomenų rinkinys paveiktas su Madry ir kiti ataka



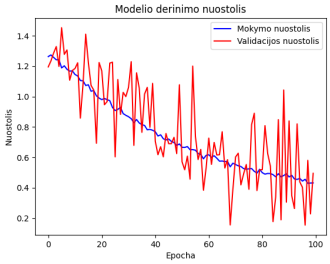
123 pav. Tinklo su LF I algoritmu bei apmokytu su sudarytu nespalvotu duomenų rinkiniu, kuriam apskaičiuotos keturios Liapunovo eksponentės, tikslumo grafikas



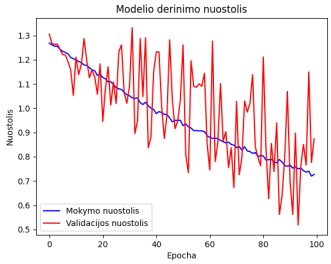
124 pav. Tinklo su patobulintu LF I algoritmu bei apmokytu su sudarytu nespalvotu duomenų rinkiniu, kuriam apskaičiuotos keturios Liapunovo eksponentės, tikslumo grafikas



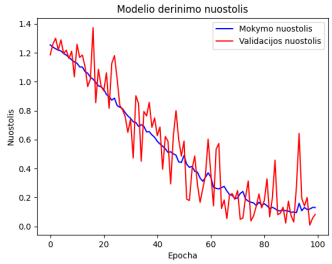
125 pav. Tinklo su Adam algoritmu bei apmokytu su sudarytu nespalvotu duomenų rinkiniu, kuriam apskaičiuotos keturios Liapunovo eksponentės, tikslumo grafikas



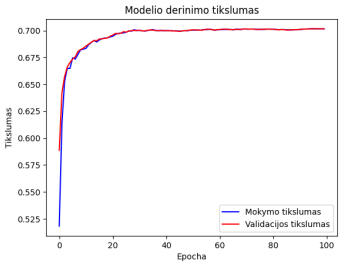
126 pav. Tinklo su LF I algoritmu bei apmokytu su sudarytu nespalvotu duomenų rinkiniu, kuriam apskaičiuotos keturios Liapunovo eksponentės, nuostolio funkcijos grafikas



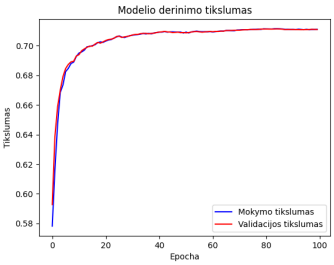
127 pav. Tinklo su patobulintu LF I algoritmu bei apmokytu su sudarytu nespalvotu duomenų rinkiniu, kuriam apskaičiuotos keturios Liapunovo eksponentės, nuostolio funkcijos grafikas



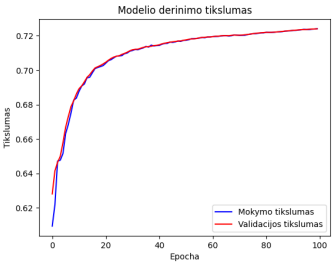
128 pav. Tinklo su Adam algoritmu bei apmokytu su sudarytu nespalvotu duomenų rinkiniu, kuriam apskaičiuotos keturios Liapunovo eksponentės, nuostolio funkcijos grafikas



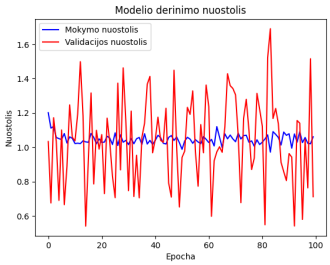
129 pav. Tinklo su LF I algoritmu bei apmokytu su sudarytu nespalvotu duomenų rinkiniu, kuriam apskaičiuotos dvi Liapunovo eksponentės, tikslumo grafikas



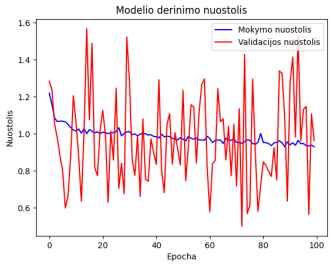
130 pav. Tinklo su patobulintu LF I algoritmu bei apmokytu su sudarytu nespalvotu duomenų rinkiniu, kuriam apskaičiuotos dvi Liapunovo eksponentės, tikslumo grafikas



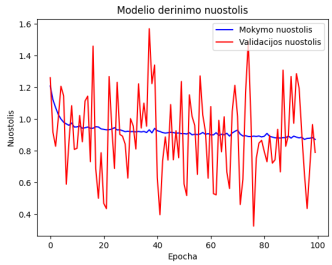
131 pav. Tinklo su Adam algoritmu bei apmokytu su sudarytu nespalvotu duomenų rinkiniu, kuriam apskaičiuotos dvi Liapunovo eksponentės, tikslumo grafikas



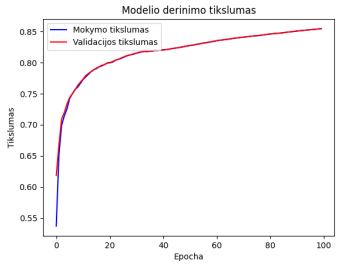
132 pav. Tinklo su LF I algoritmu bei apmokytu su sudarytu nespalvotu duomenų rinkiniu, kuriam apskaičiuotos dvi Liapunovo eksponentės, nuostolio funkcijos grafikas



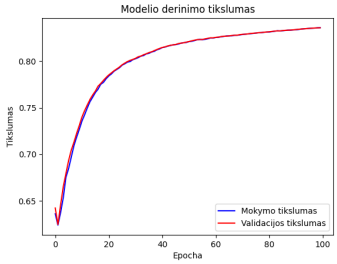
133 pav. Tinklo su patobulintu LF I algoritmu bei apmokytu su sudarytu nespalvotu duomenų rinkiniu, kuriam apskaičiuotos dvi Liapunovo eksponentės, nuostolio funkcijos grafikas



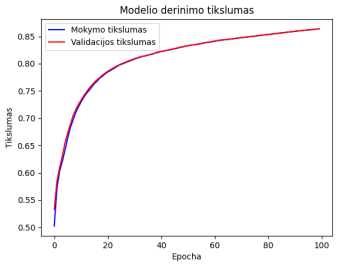
134 pav. Tinklo su Adam algoritmu bei apmokytu su sudarytu nespalvotu duomenų rinkiniu, kuriam apskaičiuotos dvi Liapunovo eksponentės, nuostolio funkcijos grafikas



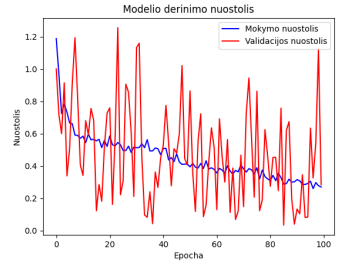
135 pav. Tinklo su LF I algoritmu bei apmokytu su sudarytu spalvotu duomenų rinkiniu tikslumo grafikas



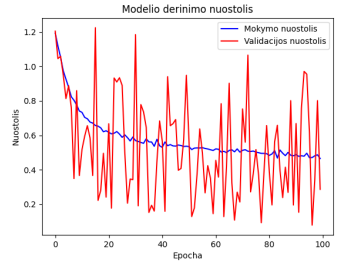
136 pav. Tinklo su patobulintu LF I algoritmu bei apmokytu su sudarytu spalvotu duomenų rinkiniu tikslumo grafikas



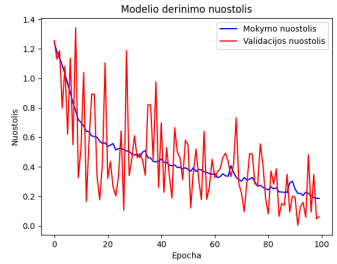
137 pav. Tinklo su Adam algoritmu bei apmokytu su sudarytu spalvotu duomenų rinkiniu tikslumo grafikas



138 pav. Tinklo su LF I algoritmu bei apmokytu su sudarytu spalvotu duomenų rinkiniu nuostolio funkcijos grafikas



139 pav. Tinklo su patobulintu LF I algoritmu bei apmokytu su sudarytu spalvotu duomenų rinkiniu nuostolio funkcijos grafikas



140 pav. Tinklo su Adam algoritmu bei apmokytu su sudarytu spalvotu duomenų rinkiniu nuostolio funkcijos grafikas