VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
INFORMATIKOS INSTITUTAS
PROGRAMŲ SISTEMŲ STUDIJŲ PROGRAMA

Vaizdų klasifikavimas naudojant konvoliucinius neuroninius tinklus

4 užduotis

Atliko: 4 kurso 1 grupės studentė
Rosita Raišuotytė

2023

TURINYS

1. Įvadas

1.1. Tikslas

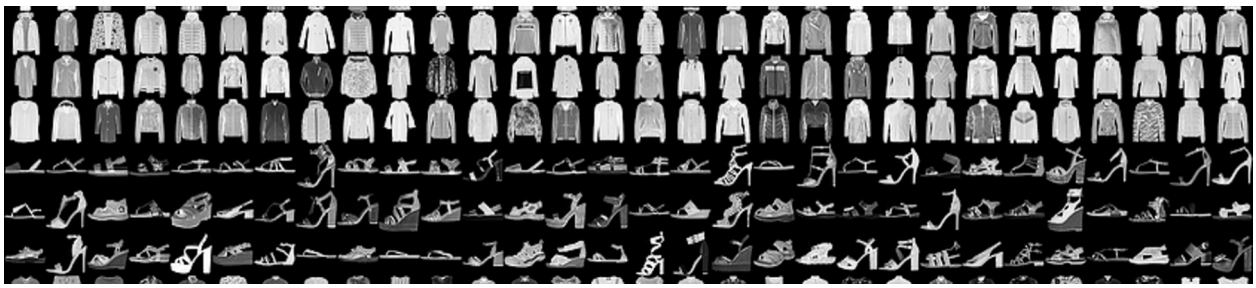
Užduoties tikslas - apmokyti konvoliucinį neuroninį tinklą vaizdams klasifikuoti ir ištirti modelio gebėjimą klasifikuoti vaizdus priklausomai nuo skirtingų hyperparametrų.

1.2. Uždavinai

- Parašyti programą, kurioje įgyvendintas konvoliucinis neuroninis tinklas vaizdams klasifikuoti.
- Atlikti tyrimą ir išsiaiškinti:
  - kaip rezultatai priklauso nuo tinklo architektūros lyginant tris skirtingas architektūras;
  - kaip rezultatai priklauso prie architektūros pridėjus išmetimo sluoksnius;
  - kaip rezultatai priklauso prie architektūros pridėjus normalizavimo sluoksnius;
  - kaip rezultatai skiriasi naudojant skirtingas aktyvacijos funkcijas;
  - kaip rezultatai skiriasi naudojant skirtingus optimizatorius.
- Įvertinti testavimo duomenų klasifikavimo tikslumą ir apskaičiuoti klasifikavimo matricą.

2. Duomenys

Tyrimui naudojamas Fashion-MNIST duomenų rinkinys. Jį sudaro 60 000 mokymo ir 10 000 testavimo duomenų įrašų, kurie susideda iš paveikslėlio ir klasės. Duomenys yra 28x28 dydžio juodai balti (vieno kanalo) rubų paveikslėliai, kurie skirstomi į 10 klasių, pavyzdžiui, suknelės, paltai, rankinės. 1 galima matyti įvairių klasių paveikslėlius. Kiekvienai klasei yra po 6 000 mokymo ir 1 000 testavimo įrašų.



1 pav. Fashion-MNIST duomenų rinkinys

2.1. Duomenų paruošimas

Duomenys į programinį kodą buvo įsikelti iš PyTorch sistemoje esančio torchvision paketo, kuriame gali rasti populiariausius duomenų rinkinius. Parsisiunčiant duomenis jiems buvo pritaikytos transformacijos: paveikslėlio vertimas į tenzorių. Įsikėlus mokymo ir testavimo duomenis, jie buvo apjungti į vieną duomenų rinkinį, sumaišyti ir išdalinti į tris grupes: mokymosi, testavimo, validavimo; santykiu 60:20:20. Padalinus gavosi, kad mokymosi duomenis sudarė 42 000, o

testavimo ir validavimo po 14 000. Dalinant duomenis kiekvienai duomenų aibei buvo sukurtas duomenų užkrovėjas (angl. dataloader), kuris duomenis pateikia paketais. Vieno paketo domenys yra sudaryti iš duomenų poros [paveikslėliai, klasės], kur paveikslėlių tenzoriaus dydis [paketo dydis, 1 - kanalas, 28 - pav. aukštis, 28 - pav. plotis], klasės - [paketo dydis].

3. Programos kodas

Programinis kodas rašytas Python kalba, naudojantis PyTorch mašininio mokymosi sistema. Programos kodą galite rasti paspaudę nuorodą čia.

3.1. Skaičiavimo resursai

Naudoti debesijos sprendimai „Google Colab“ aplinkoje, GPU versija.

4. Tyrimas

4.1. Hyperparametrai

Tyrimas buvo atliktas su šiais nustatytas hyperparametrais:

- epochų skaičius - 15,
- mokymosi greitis - 0,001,
- paketo dydis - 64,
- optimizavimo algoritmas - Adam,
- nuostolių funkcija - CrossEntropyLoss.

4.2. Konvoliucinio tinklo architektūra

Tyrimui, kuriame lyginome skirtingų architektūrų rezultatus, buvo pasirinktos šios trys architektūros:

- LeNet architektūra iš interneto:
  - 2 konvoliuciniai sluoksniai,
  - 3 pilnai sujungti sluoksniai.
- AlexNet architektūra iš interneto:
  - 5 konvoliuciniai sluoksniai,
  - 3 pilnai sujungti sluoksniai.
- Mano parašyto modelio architektūrą:
  - 4 konvoliuciniai sluoksniai,
  - 3 pilnai sujungti sluoksniai.

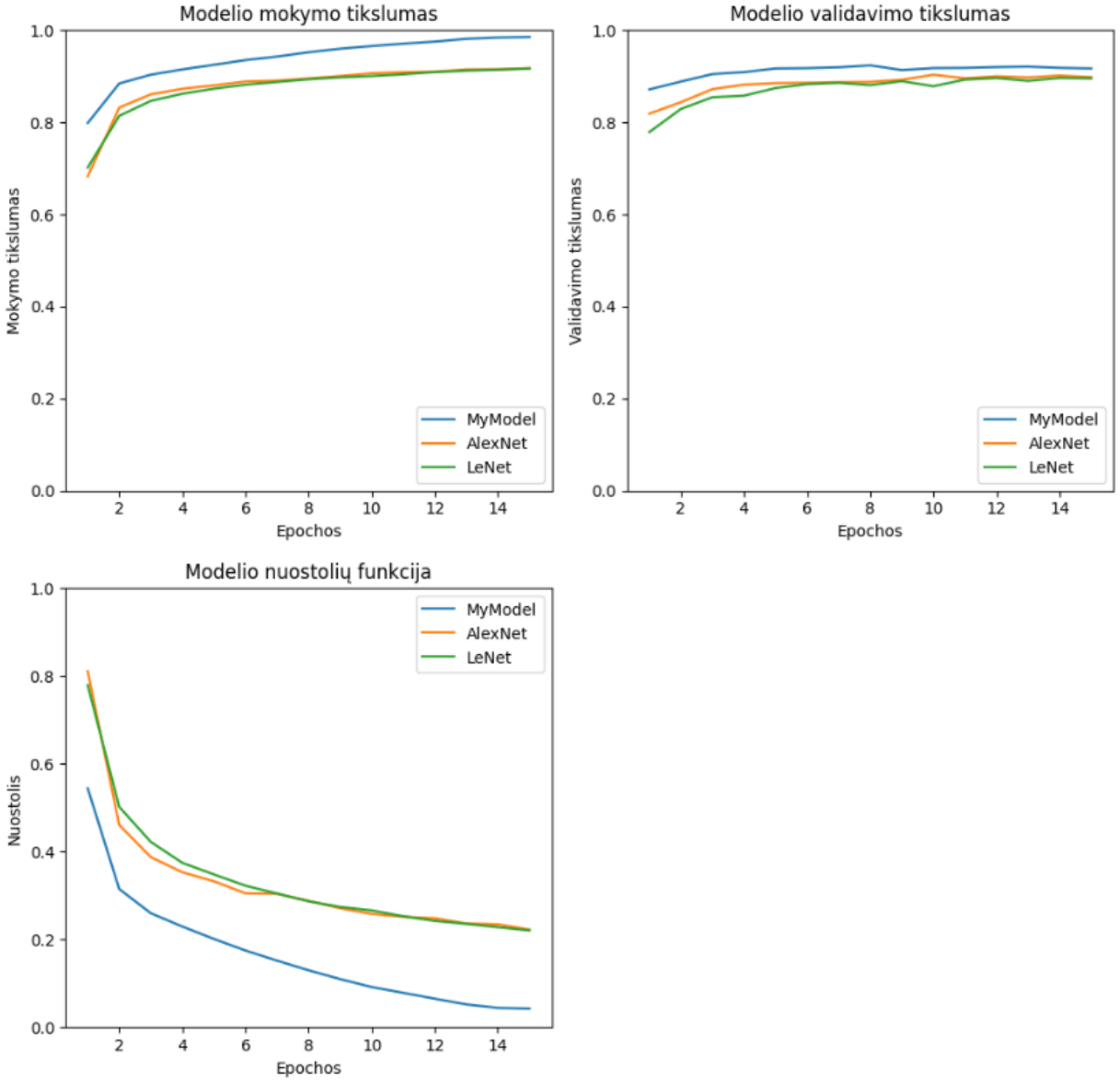
Nors visos architektūros yra panašios savo sluoksniais, tačiau skiriasi sluoksnių vidiniai parametrai: branduolių kiekis, branduolių dydis, filtro dydis ir kt, kas padaro modelius tokius skirtingus, nes skiriasi jų parametrų kiekiai: LeNet turi 44 426 parametrų, mano parašytas modelis - 797 770, o AlexNet - 57 022 154.

Mano parašytas modelis bus naudojamas toliamesniame tyrime, kai stebėsime rezultatų priklausomybę nuo hyperparametrų. Su juo lyginsime gautus naujus rezultatus papildžius ar pakeitus mano modelio architektūrą. Taip pat tą patį modelį, su ta pačia architektūra, apmokysime su skirtingu optimizatoriumi.

5. Rezultatai

5.1. Skirtingų architektūrų lyginimas

Tirant kaip skiriasi Fashion-MNIST duomenų klasifikavimas nuo architektūros, buvo lyginami šie trys modeliai: mano parašytas MyModel ir iš interneto paimti AlexNet ir LeNet. Kaip galima matyti 2 pav. esančiuose grafikuose AlexNet ir Lenet modelių rezultatai labai panašūs, nors AlexNet turi daugiau konvoliucinių sluoksnių ir parametrų kiekis skiriasi daugiau nei tūkstantį kartų. Galima manyti, kad AlexNet modeliui trūko mokymo, todėl buvo apmokytaas papildomai 15 epochų, tačiau rezultatas labai nesikeitė. Todėl šiuo atveju, mano parašytas modelis MyModel yra pranašesnis, nes apsimoko per trumpesnį laiką ir pasiekia geresnius rezultatus. MyModel pasiektas mokymosi klasifikavimo tikslumas (toliau tikslumas) per 15 epochų yra 98,56, validavimo tikslumas - 91,73, nuostolių funkcijos reikšmė - 0,042.



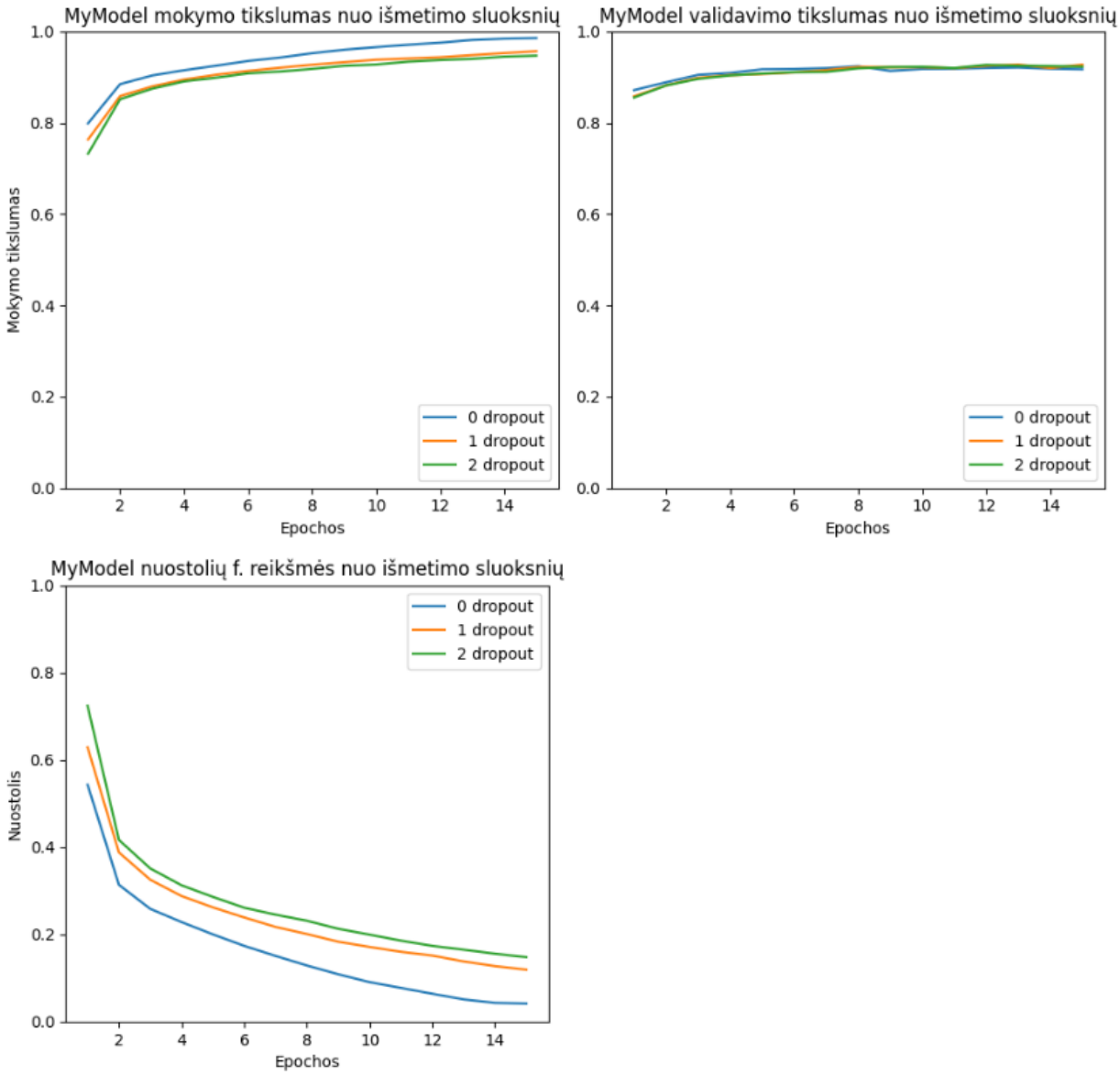
2 pav. Skirtingų architektūrų rezultatų grafikai

5.2. Skirtingų išmetimo sluoksnių skaičiaus ir skirtingų išmetimo tikimybių lyginimas

5.2.1. Skirtingas skaičius išmetimo sluoksnių

Tiriant kaip išmetimo sluoksniai keičia tikslumą bei nuostolių funkcijos reikšmę buvo atlikti bandymai pridėjus vieną ir du išmetimo sluoksnius prie MyModel architektūros. Išmetimo sluoksniai buvo pridėti tarp pilnai sujungtų sluoksniš prieš aktyvacijos funkciją su 0,5 išmetimo tikimybe.

3 pav. matomi rezultatai kaip MyModel modelio rezultatai skiriasi nuo skirtingo skaičiaus išmetimo sluoksnių. Antrame grafike matomas validavimo tikslumas beveik vienodas visiems atvejams, geriausią validavimo tikslumą gavo modelis su vienu išmetimo sluoksniu. Didžiausias skirtumas matomas trečiame grafike, kuriame vaizduojama nuostolių funkcijos reikšmė, greičiausiai apsimoko modelis be išmetimo sluoksnių. Vis dėlto bendrai geriausi rezultatai pasiekti be išmetimo sluoksnių.



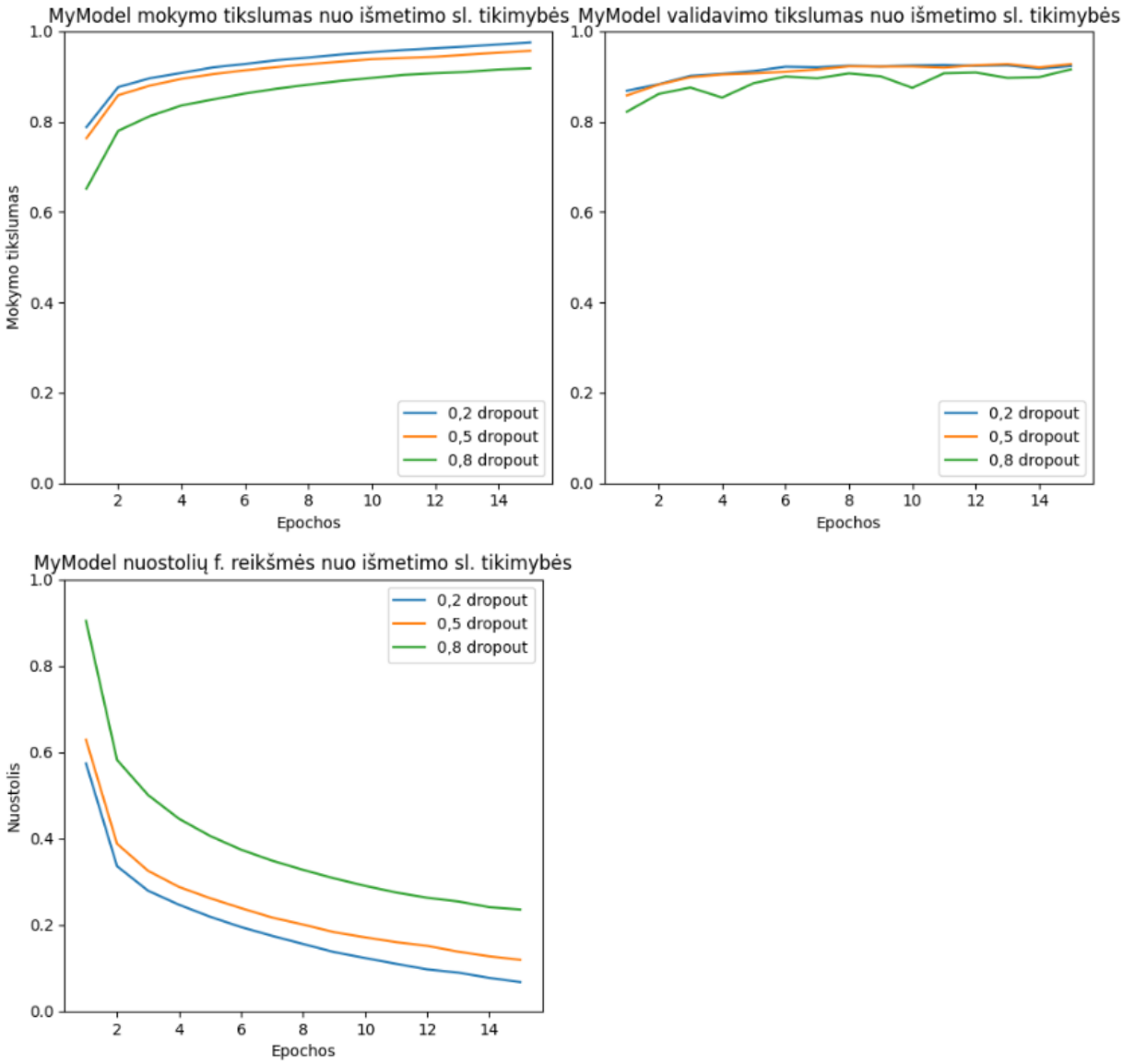
3 pav. MyModel su skirtingu skaičiumi išmetimo sluoksniu rezultatų grafikai

5.2.2. Skirtingos vieno sluoksnio išmetimo tikimybės

Kadangi su vienu išmetimo sluoksniu modelio rezultatai buvo šiek tiek geresni nei su dviem sluoksniais, buvo pasirinkta skirtingų išmetimo sluoksnio tikimybių gaunamus rezultatus tirti su vienu išmetimo sluoksniu. Pasirinktos tikimybės buvo: 0,2, 0,5, 0,8.

4 pav. pirmame grafike matyti, kad, kai išmetimo tikimybė yra 0,8, mokymosi tikslumo skirtumas lyginant su kitais modeliais yra didesnis, pats tikslumas mažesnis, taip pat antrame grafike matosi, kad validavimo tikslumas tampa ne toks stabilus, kreivė turi kelis įdubimus. Taip pat žiūrint į trečią grafiką matosi, kad nuostolių funkcijos reikšmė po 15 epochų yra dar pakankamai didelė, kas reiktų, kad modelis dar galėtų pasimokyti. Geriausias rezultatas pasiektas kai išmetimo tikimybė 0,2.

Lyginant MyModel su vienu išmetimo sluoksniu, kai išmetimo tikimybė 0,2, bei MyModel be išmetimo sluoksnių, po apmokymo geresnį validavimo tikslumą pasiekė modelis su išmetimo sluoksniu. Su išmetimo sluoksniu validavimo tiklsumas - 92,3, be - 91,73.

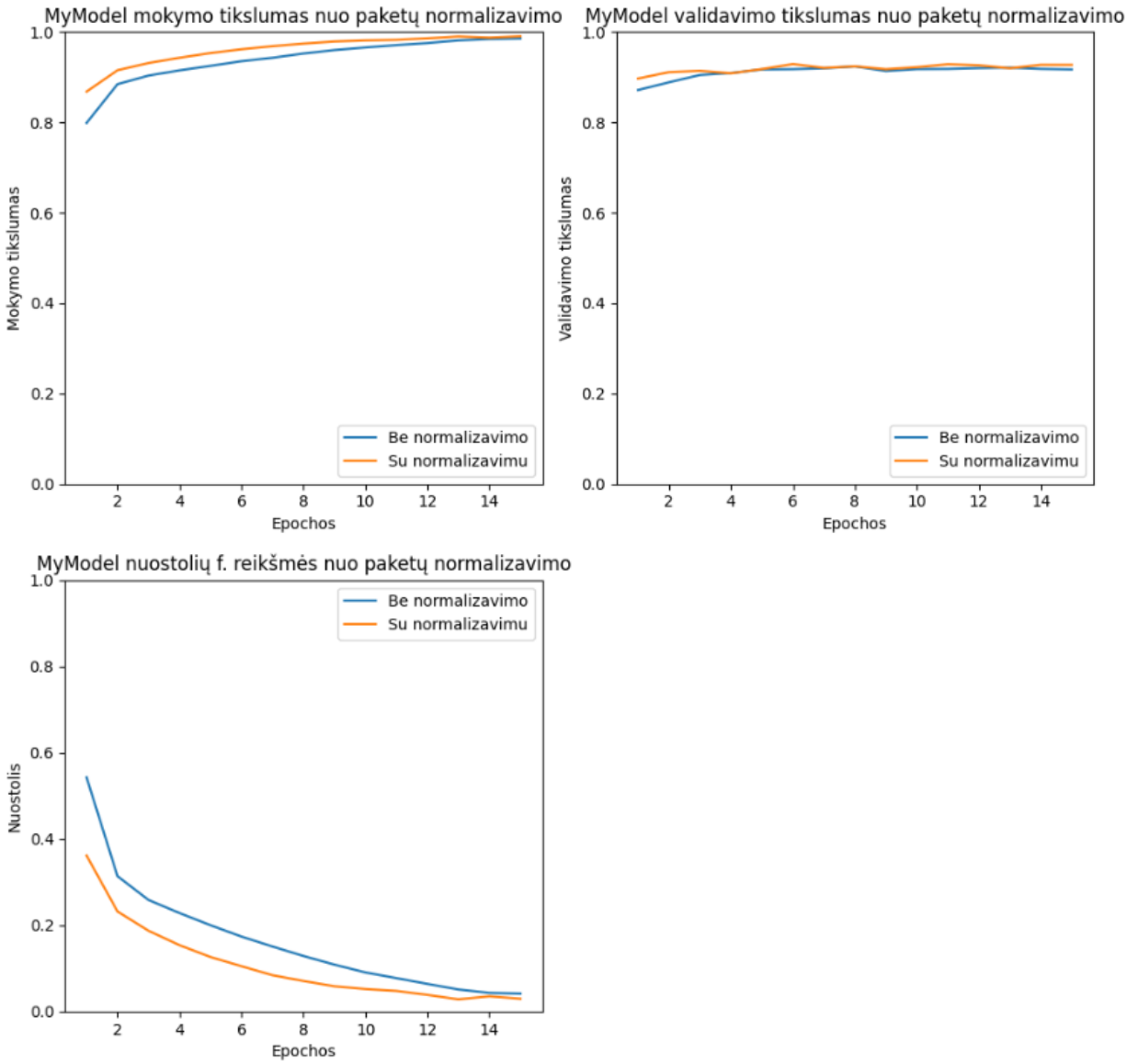


4 pav. MyModel su skirtingomis vieno išmetimo sluoksnio tikimybėmis rezultatų grafikai

5.3. Paketų normalizavimo daroma įtaka rezultatams

Tyrimas atliktas prie MyModel pridėjus du paketo normalizavimo sluoksnius. Pirmas sluoksnis pridėtas po dvijų konvoliucijos sluoksnių, ir kitas po dar dviejų konvoliucijos sluoksnių.

5 pav. matome, kad rezultatai po apmokymo panašūs abiem atvejais - su ir be paketų normalizavimo. Galima pastebėti, kad su paketų normalizavimu modelis iškarto pradėjus mokymą rodo geresnius rezultatus, tačiau mokymosi laikas išlieka toks pat norint pasiekti tokius pat ar šiek tiek geresnius rezultatus kaip modelis be paketų normalizavimo. Antrame grafike matomas validavimo kreivės nedidelis netolygumas, taip pat šeštoje epochoje pasiektas tikslumas 92,91 yra, nors ir nežymiai, didžiausias tarp visų nagrinėtų modelių.

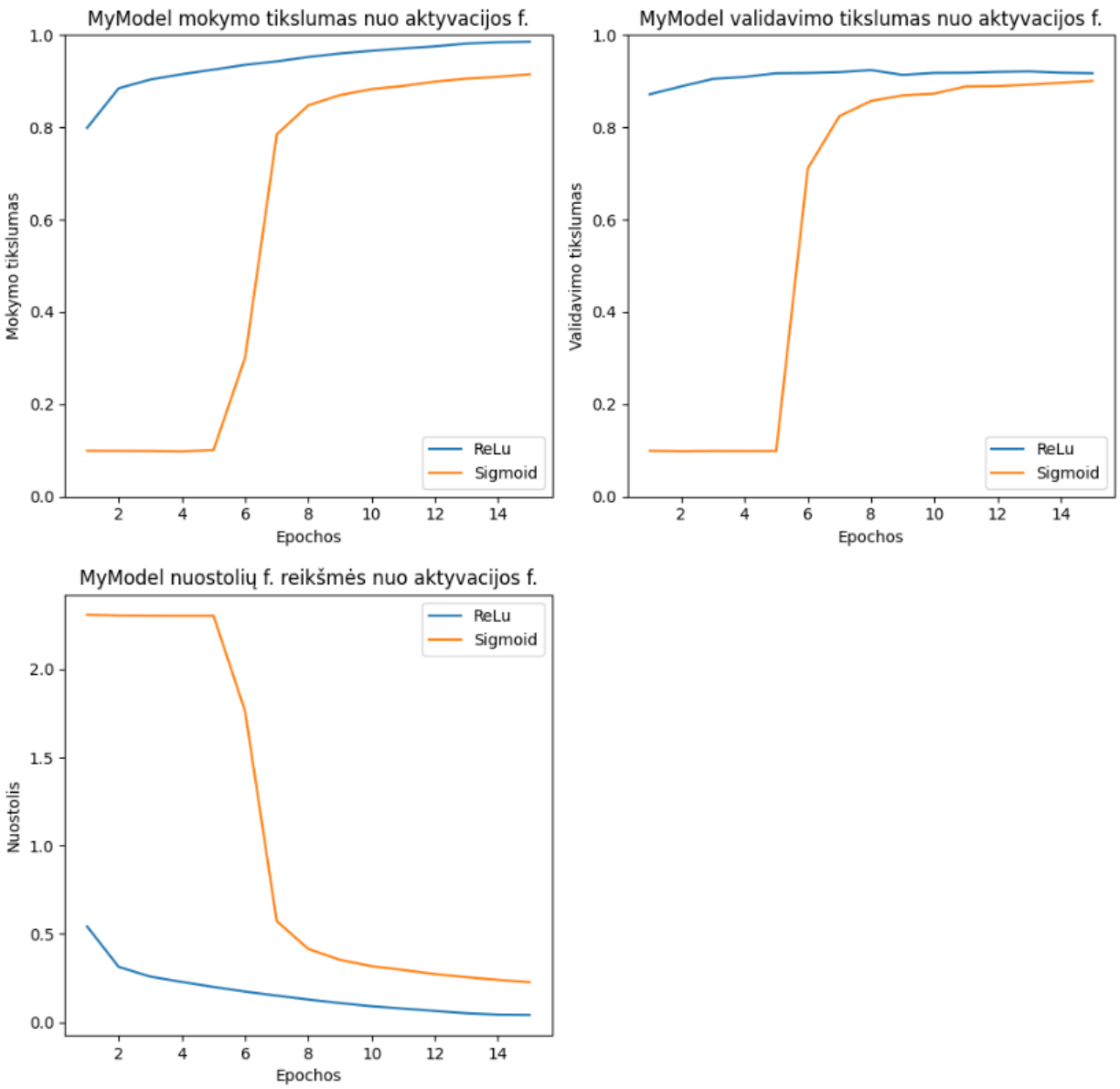


5 pav. MyModel su ir be paketų normalizavimo rezultatų grafikai

5.4. Kaip keičiasi rezultatai nuo aktyvacijos funkcijos

Tiriant aktyvacijos funkcijas lyginome ReLu ir Sigmoid aktyvacijos funkcijas modelyje MyModel.

6 pav. matosi didžiulis skirtumas tarp kreivių. Visuose grafikuose matyti, kad modelis su sigmoidine aktyvacijos funkcija mokymosi pradžiose nesimoko, rezultatai labai prasti. Trečiame grafike matyti, kad pradėjus mokyti modelį su aktyvacijos funkcija, nuostolių funkcijos reikšmė išskirtinai didelė palyginus su visais nagrinėtais modeliais. Taip pat matome, kad į mokymosi pabaigą visi rezultatai artėja prie panašių rezultatų kaip modelio su ReLu aktyvacijos funkcija. Vis dėlto po 15 mokymosi epochų modelio su ReLu aktyvacijos funkcija rezultatai yra geresni nei su sigmoidine.

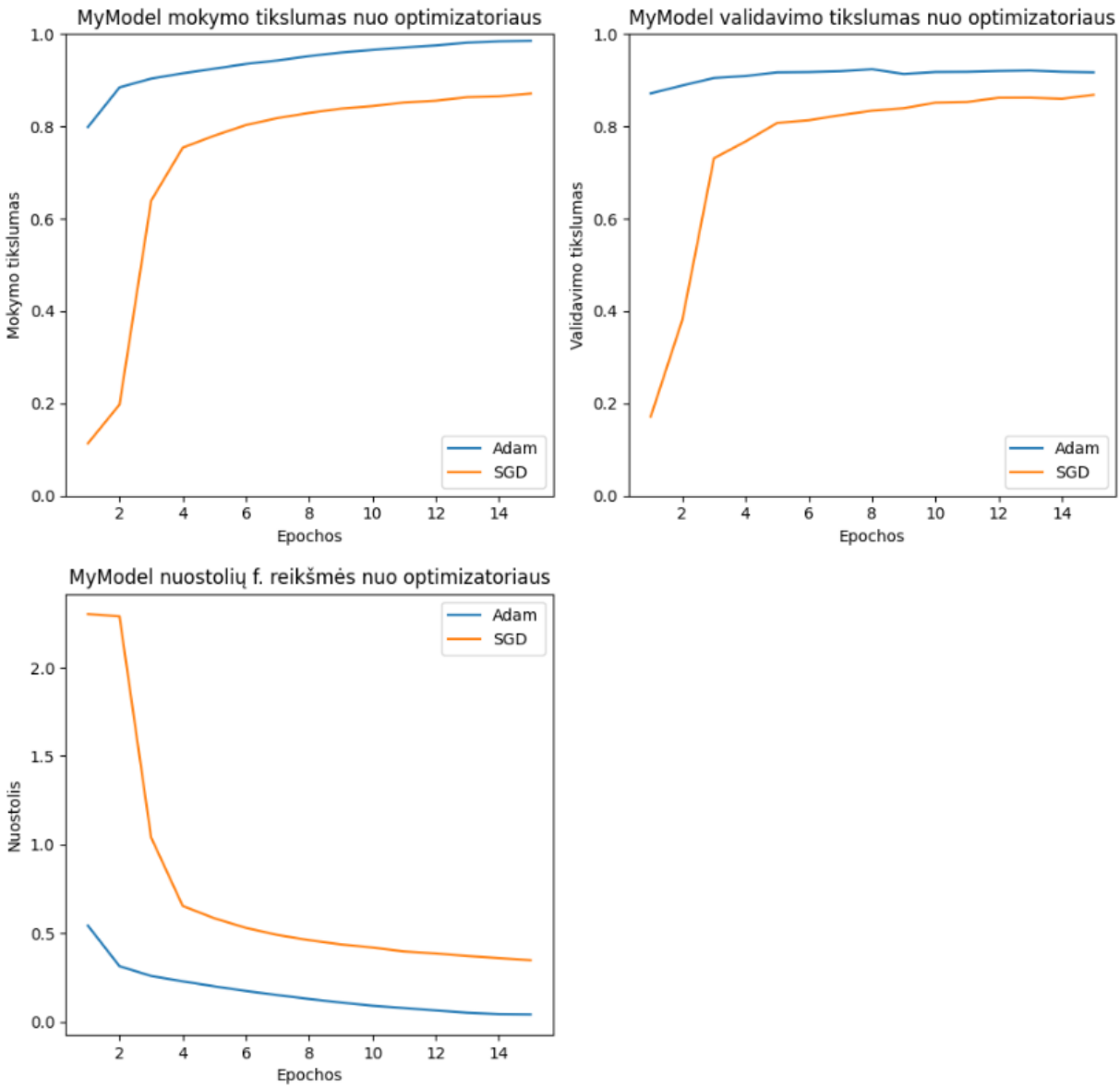


6 pav. MyModel su skirtingomis aktyvacijos funkcijomis rezultatų grafikai

5.5. Optimizavimo algoritmo daroma įtaka modelio rezultatams

Tyrimo metu buvo lyginami Adam ir SGD optimizatoriai.

7 pav. visuose grafikuose matyti didelis skirtumas tarp abiejų optimizatorių, Adam optimizatorius ryškiai pranašesnis už SGD optimizatorių. Trečiame grafike matosi, kad po apmokymo modelio naudojančio SGD optimizatorių nuostolių funkcijos reikšmė dar pakankamai didelė - 0,348, kas galimai reiškia, kad modeliui trūksta apmokymo.



7 pav. MyModel su skirtingais optimizatoriais rezultatų grafikai

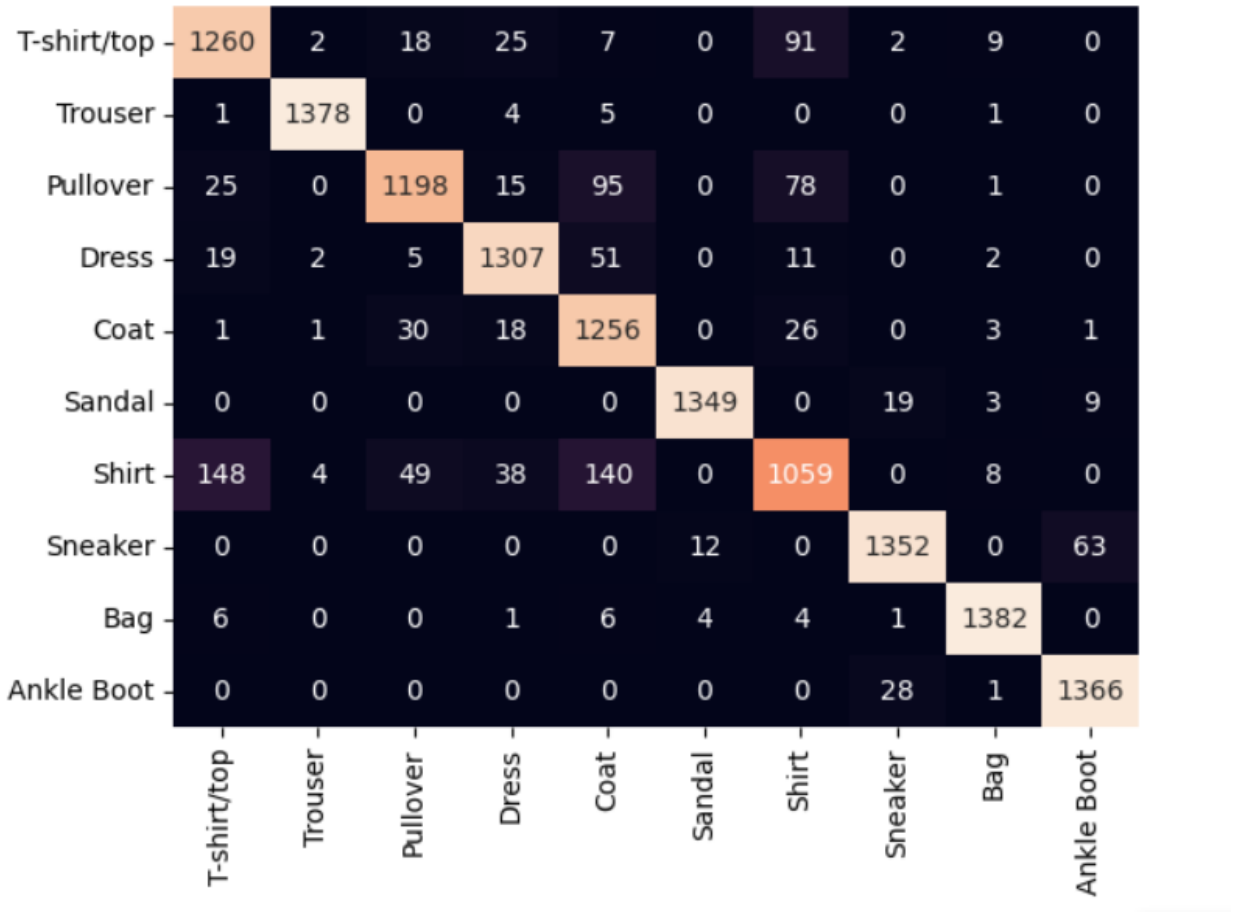
5.6. Geriausias modelis ir jo testavimo rezultatai

Geriausi rezultatai pasiekti su MyModel bei pridėtais paketų normalizavimo sluoksniais, kai epochų skaičius - 11, naudojama aktyvacijos funkcija - ReLu, optimizatorius - Adam. Gauti rezultatai: nuostolių funkcijos reikšmė 0,041, mokymosi tikslumas - 98,55, validavimo tikslumas - 92,61.

Šio modelio testavimo tikslumas - 92,19. Iš testavimo rezultatų buvo sudaryta klasifikavimo matrica matoma 8 pav. Iš jos matyti, kad blogiausiai suklasifikuoti yra marškiniai (angl. shirt). Jie maišomi su marškinėliais/topais (angl. t-shirt/top) ir paltais (angl. coat). Taip pat prastai klasifikuojami ir megztiniai (angl. pullover), jie maišomi su paltais ir marškiniais. Geriausiai klasifikuojamos rankinės (angl. bag).

1 lentelėje pateikiami 30 pasirinktų testavimo įrašų, kur būtų bent po vieną iš visų klasių. Matyti, kad iš 30 įrašų net 8 įrašai suklasifikuoti neteisingai, vadinasi šių duomenų klasifikavimo

tikslumas tik 0,73. Blogai suklasifikuoti duomenų įrašai lentelėje paryškinti.



8 pav. MyModel su paketų normalizavimu testavimo duomenų klasifikavimo matrica

1 lentelė. Testavimo duomenų tikrosios ir MyModel su paketų normalizavimu spėjamos klasės

Nr.	Tikroji klasė	Spėjama klasė
1	Dress	Dress
2	Coat	Coat
3	Shirt	Shirt
4	**Pullover**	**Shirt**
5	Pullover	Pullover
6	Coat	Coat
7	Bag	Bag
8	Shirt	Shirt
9	**Shirt**	**Coat**
10	Sandal	Sandal
11	**Shirt**	**T-shirt/top**
12	Pullover	Pullover
13	Bag	Bag
14	**Shirt**	**Coat**
15	Sandal	Sandal
16	Dress	Dress
17	**T-shirt/top**	**Shirt**
18	Ankle Boot	Ankle Boot
19	**Shirt**	**T-shirt/top**
20	Trouser	Trouser
21	Trouser	Trouser
22	Dress	Dress
23	Sandal	Sandal
24	**Sneaker**	**Ankle Boot**
25	Shirt	Shirt
26	**Bag**	**Coat**
27	Ankle Boot	Ankle Boot
28	Coat	Coat
29	Sandal	Sandal
30	Trouser	Trouser

6. Išvados

- Daugiau modelio parametrų negarantuoja geresnių rezultatų.
- Architektūros nepaisant panašių sluoksnių labai skiriasi savo vidiniais hyperparametrais.
- Modelio gaunami rezultatai priklauso ne tik nuo architektūros, bet ir nuo pačių duomenų.
- Išmetimo sluoksniai padeda išvengti modelio persimokymo, modelio mokymas užtrunka ilgiau. Modelio su vienu išmetimo sluoksniu validavimo tiklsumas buvo geresnis nei modelio be išmetimo sluoksnių.
- Didėjant išmetimo sluoksnių išmetimo tikimybei, modelio tikslumas mažėja.
- Paketų normalizavimas šiek tiek gerina rezultatus, mokymosi pradžiose akivaizdžiai matomi geresni rezultatai lyginant su modeliu be paketų normalizavimo.
- Aktyvacijos funkcija turi didelę įtaką rezultatams, šiuo atveju ReLu aktyvacijos funkcija yra geresnė ir tolydesnė palyginmus su sigmoidine.
- Optimizatorius taip pat turi didelį skirtumą rezultatams, šiame tyrime Adam optimizatorius ryškiu skirtumu lenkia SGD optimizatorių.
- Geriausiu modeliu išrinktas MyModel su paketu normalizavimu.
- Blogiausiai klasifikuojami buvo marškiniai, jie maišomi su marškinėliais ir paltais.