VILNIAUS UNIVERSITETSS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
INFORMATIKOS INSTITUTAS
PROGRAMŲ SISTEMŲ BAKALAURO STUDIJŲ PROGRAMA

Vaizdų klasifikavimas naudojant konvoliucinius neuroninius tinklus

Image Classification Using CNN

Laboratorinio darbo ataskaita

Atliko: Armintas Pakenis

Darbo vadovas: prof. dr. Olga Kurasova

Vilnius – 2023

TURINYS

1. Užduoties tikslas

Užduoties tikslas — išbandyti sukurti skirtingas konvoliucinio neuroninio tinklo architektūras ir rasti iš išbandytų geriausią. Rašytas programos kodas naudoja PyTorch biblioteką. Naudotos programos kodą galima rasti GitHub repositorijoje: https://github.com/ArmintasP/Computational-intelligence/tree/main/Lab4.

1.1. Duomenys

Naudotas CIFAR-10 duomenų rinkinys https://www.cs.toronto.edu/ kriz/cifar.html.

Rašyta programa buvo išskaidyta į dvi dalis: viena apdoruoja duomenis, kita treniruoja modelį. Programos dalys skirtingose Jupyter užrašų knygutėse.

CIFAR-10 duomenų rinkinys buvo užkoduotas failuose. Failai buvo nuskaitomi panaudojant metodą "unpickle" iš to paties duomenų rinkinio autorių puslapo. Likusi pirmos programos dalies kodo dalis originali, kur atrenkamos RGB reikšmės ir atkuriamos nuotraukos .jpg formatu. Kiekvienos klasės nuotraukos išsaugomos atskirame kataloge su tos klasės pavadinimu.

Vaizdų klasių 10: lėktuvai, automobiliai, paukščiai, katės, elniai, šunys, varlės, arkliai, laivai ir sunkvežimiai.

Originalus CIFAR-10 rinkinys pateikia 6000 nuotraukų kiekvienai klasei. Iš viso 10 klasių, 50000 nuotraukų skirtų mokymui ir 10000 testavimui. Po failų apdorojimo, 50000 ir 10000 nuotraukų buvo sumaišytos ir padalintos atsitiktinai pagal santykiu 7:3. Nuotraukų dydis: 32×32.

1.2. Įranga

Modelio treniravimas vyko ant asmeninio kompiuterio su AMD Ryzen 5 5600G APU (CPU) ir 32 GB RAM.

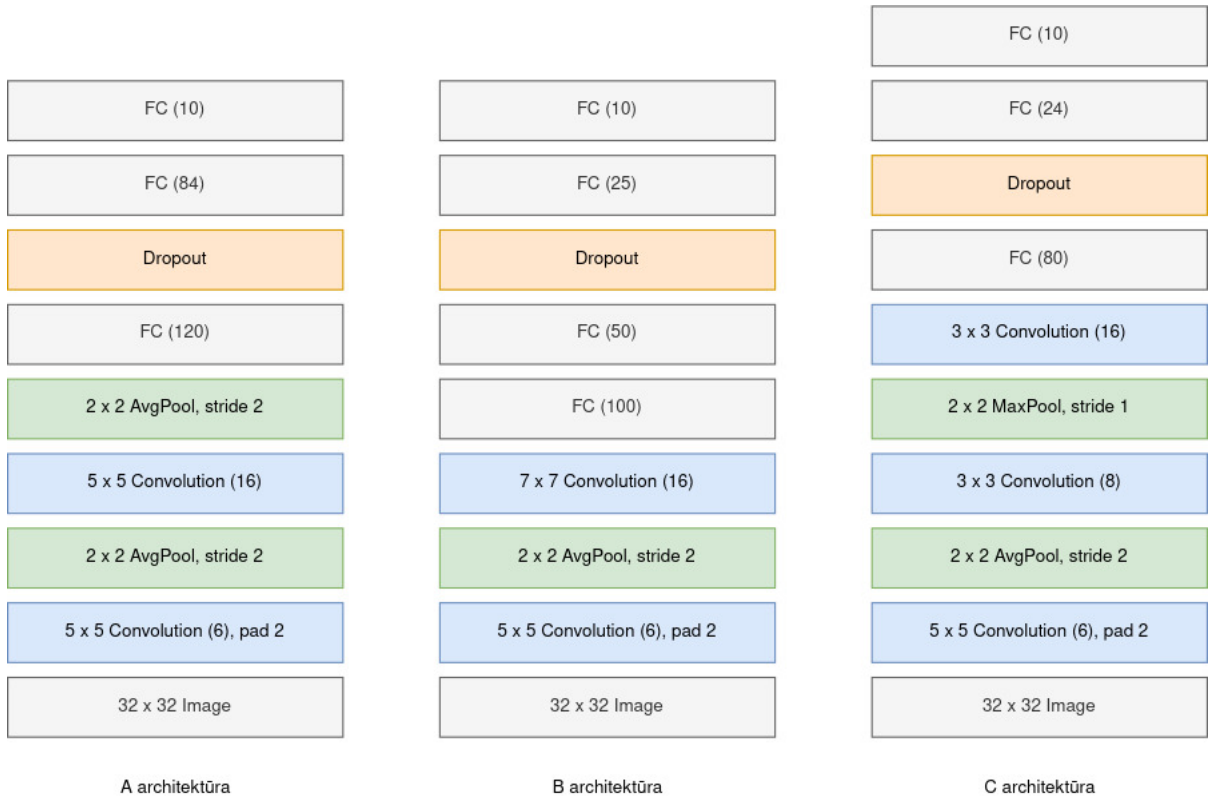
2. Modelių architektūros

Nagrinėtos 3 architektūros: A, B ir C (žr. 1 pav.). A architektūra paremta LeNet modelio architektūra, atlikti tokie pakeitimai: įvesties nuotraukos ilgis ir plotis padidinti iki 32 pikselių, prieš priešpaskutinį sluoksnį atsiranda išmetimo (Dropout) sluoksnis. Lyginant architektūras, užfiksuoti šie hiperparametrai:

- Aktyvacijos funkcija — ELU.
- Paketo dydis — 32.
- Išmetimo sluoksnio tikimybė — 0.4.
- Optimizavimo algoritmas — Adam.
- Nuostolių funkcija — kryžminė entropija.
- Mokymosi greitis — 0,001.
- Epochų skaičius - 30.

Duomenų rinkinio mokymo duomenims buvo naudotos šios augmentacijos:

- Spalvų variacija (color jitter).
- Atsitiktinis horizontalus apvertimas.
- Atsitiktinis RGB nuotraukos pavertimas į monochromatinę.



1 pav. Trys architektūros. Sluoksniai ar jų operacijos taikomi iš apačios į viršų

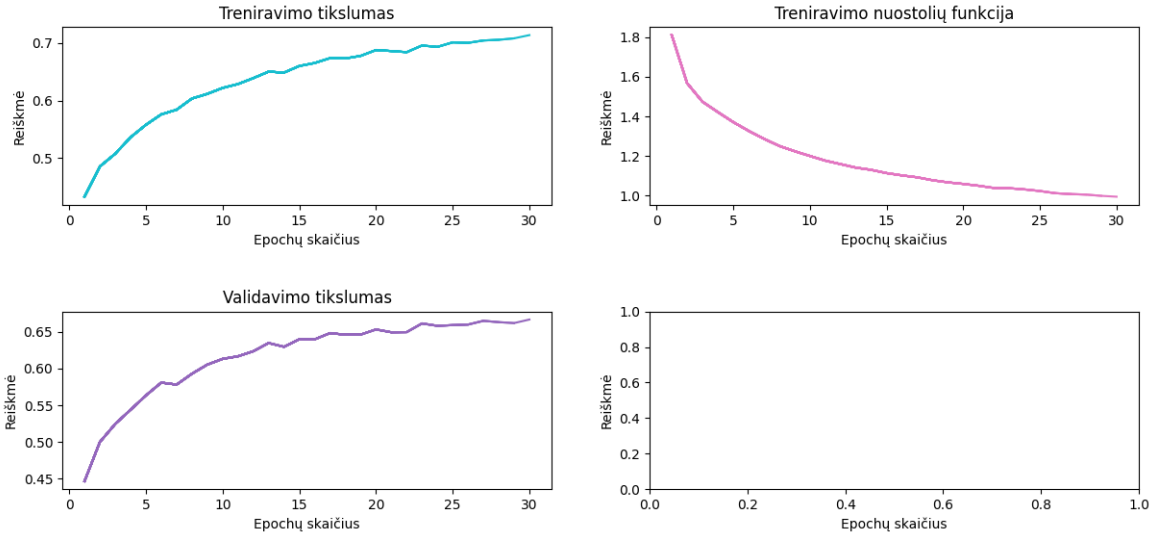
2.1. Architektūrų palyginimas

A modelis turi 83 126 parametrų, B — 171 861, C — 159 218.

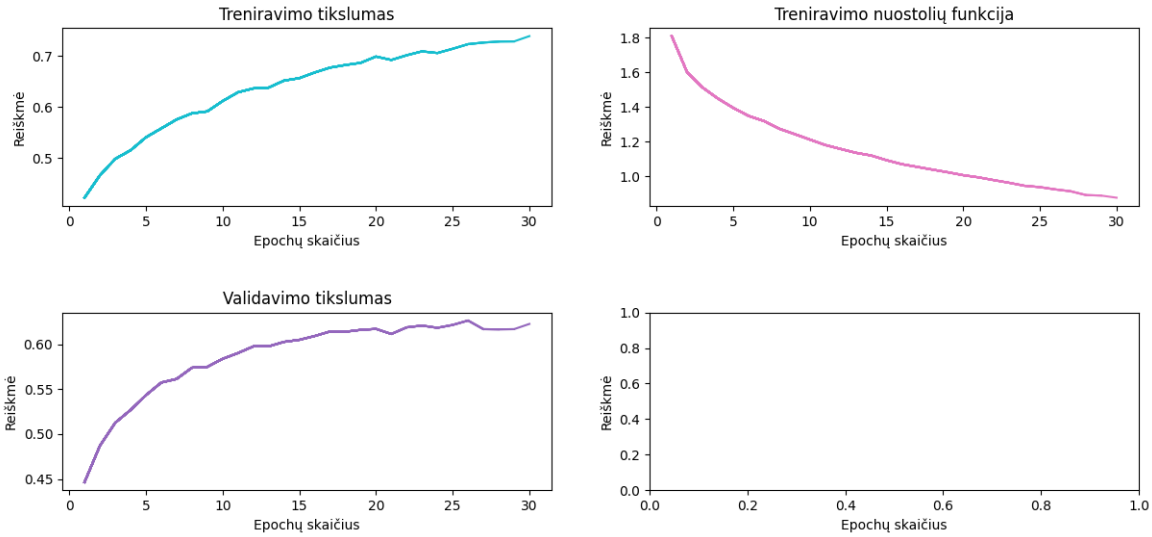
Iš mokymo rezultatų matyti, kad tikslumas ir skirtumas tarp mokymo bei validavimo

tikslumo nesiskiaria architektūrose A ir C (žr. 2 ir 4 pav.). Nors C turi bene dvigubai daugiau parametrų nei A, C modelis nepronoko A. Blogiausiai pasirodė B modelis (žr. refimg:b-results pav.) su daugiausiai parametrų. Pašalinus sujungimo sluoksnį (AvgPool) ir panaudojus pilnai sujungtą, rezultatai net pablogėjo: nuo 20 epochos validavimo tikslumas nedidėjo, kai didėjo treniravimo — ženklas, kad B modelis persimokė.

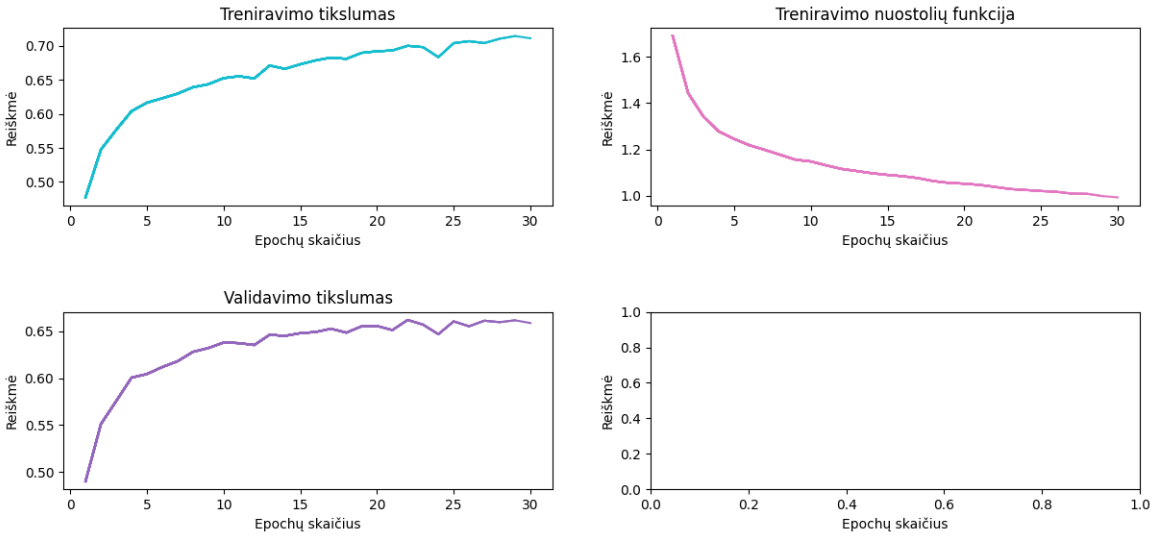
Galima daryti išvadą, kad šitame klasifikavimo uždaviny su esamu rinkiniu daug svarbiau turėti konvoliucinius sluoksnius. Tačiau kaip matoma iš C pavyzdžio, sluoksniai papildomo tikslumo ženkliai nepradėjo. Nors C pirmosios epochos turėjo geresnius tikslumus, griežtai to patvirtinti negalima, kadangi tai galėjo būti sutapimas dėl atsitiktinai inicializuotų svorių mokymo pradžioje.



2 pav. A architektūros tikslumo ir nuostolių funkcijos grafikai



3 pav. A architektūros tikslumo ir nuostolių funkcijos grafikai



4 pav. A architektūros tikslumo ir nuostolių funkcijos grafikai

1 lentelė. Tikslumo ir nuostolių funkcijos priklausomybė nuo hiperparametrų

Optimizavimo algoritmas	Išmetimo sluoksnio tik.	Aktyvacijos f-ija	Nuostolių f-ijos reikšmė	Mokymosi tiksl.	Validavimo tiksl.
Adam	0,4	ELU	1,22	62,7 %	61,85 %
SGD	0,4	ELU	2,08	28,63 %	30,10 %
Adam	0	ELU	1,03	66,17%	61,94 %
Adam	0,8	ELU	1,47	53,32 %	54,18 %
Adam	0,4	ReLU	1,15	64,00 %	62,41 %
Adam	0,4	Sigmoid	1,66	47,37 %	47,97 %

3. Hiperparametrų įtaka konvoliuciniam neuroniniam tinklui

Toliau nagrinėjama tik A modelio architektūra su skirtingais specifiniais hiperparametrais.

Epochų skaičius, mokymosi greitis, paketo dydis ir skirtingų augmentacijų poveikis mokymosi ir validavimo tikslumui nebus nagrinėjami. Šie hiperparametrai nekis ir bus atitinkamai 10, 0,001, 32 ir anksčiau minėtos augmentacijos. Nuostolių funkcija irgi išliks kryžminė entropija.

Rezultatuose (žr. 1 lentelę) matoma, kad stohastinio gradiento nusileidimo optimizavimo algoritmas veikia blogiau nei Adam algoritmas. Tačiau taip yra todėl, kad SGD yra taikomas paketui, kurio dydis yra lygus vienetui. Lentelėje pateikti hiperparametrai buvo keisti paliekant tą patį paketo dydį. Tačiau atlikus papildomą modelio apmokymą su 2 eilutės duomenimis (naudojant SGD optimizatorių) ir pakeitę paketo dydį į 1, gauta nuostolių funkcijos reikšmė yra arti 1,23, o mokymosi tikslumas su validavimo — 59,05 ir 60,03 %. Tad parinkus tinkamą paketo dydį, SGD algoritmas veikia beveik tiek pat gerai, kiek Adam. Tačiau atlikus šį papildomą apmokymą su vieneto dydžio paketais modelio apmokymas užtruko dvigubai ilgiau (18 minučių).

Trečioje lentelės eilutėje matome geriausias pasiektas statistikas, kur išmetimo sluoksnio tikimybė yra 0; kitaip tariant, kai jo nėra. Tačiau verta atkreipti dėmesį, kad išmetimo sluoksnis yra taikomas modelio permokymo problemai spręsti. Šiuo atveju tarp mokymosi tikslumo ir validavimo tikslumo yra $66{,}17\% - 61{,}94\% = 4{,}23\%$. Lyginant su kitomis eilutėmis panašu, kad su tokiu išmetimo sluoksnio hiperparametru mūsų modelis persimokys. Ketvirtoje eilutėje matome aukštą išmetimo sluoksnio tikimybę, tačiau ji per daug aukšta ir geriausiai, panašu, kad robastiškumui pasiekti geresnė tikimybė yra $0{,}4$, ypač jei norima, kad modelis nepersimokytų.

Paskutiniose trijose eilutėse išbandomos skirtingos funkcijos. Blogiausiai tam tinka sigmoidinė. Nors modelsi nėra architektūriškai didelis, panašu, kad artėjama prie nykstančių gradientų ir nuo to kenčia mokymosi tikslumas. Panašu, kad geriausi rezultatai gali būti pasiekiami naudojant ReLU funkciją, kuri, priešingai nei ELU, negali įgyti neigiamų reikšmių. Būtent išmokytą modelį, kuris naudoja ReLU funkciją ir kitus tos lentelės eilutės hiperparemtrus, laikysime geriausiu, kadangi jis pasiekė aukščiausią validavimo tikslumą, skirtumas tarp mokymosi tikslumo ir validavimo tikslumo mažesnis nei 3 eilutės, o nuostolių funkcijos reikšmė irgi nėra reikšmingai nutolusi nuo 3 eilutės reikšmės (1,03).

2 lentelėje, t. y. klasifikavimo matricoje, matome, kad ne visos klasės vienodai yra preciziškai

2 lentelė. Klasifikavimo matrica testavimo duomenims

		Prognozuotos klasės									
		airplane	**automobile**	**bird**	**cat**	**deer**	**dog**	**frog**	**horse**	**ship**	**truck**
Trokštamos klasės	**airplane**	**1051**	100	71	30	77	31	16	27	251	147
	automobile	12	**1346**	0	9	14	7	15	12	59	305
	bird	118	19	**708**	120	334	214	141	67	39	49
	cat	28	27	83	**667**	186	468	183	85	35	64
	deer	55	19	50	106	**1176**	63	117	139	27	29
	dog	15	15	85	271	151	**1018**	106	96	17	45
	frog	12	13	42	118	179	71	**1245**	25	18	60
	horse	20	6	34	68	202	176	23	**1146**	17	83
	ship	114	96	13	28	32	15	12	4	**1399**	104
	truck	27	157	6	31	13	12	20	19	47	**1478**

3 lentelė. Kiekvienos klasės 3 nuotraukų prognozės

Trokštama	**Prognozuota**
airplane	ship
airplane	truck
airplane	airplane
automobile	automobile
automobile	truck
automobile	automobile
bird	dog
bird	bird
bird	deer
truck	truck

Trokštama	**Prognozuota**
cat	cat
cat	dog
cat	dog
deer	deer
deer	bird
deer	deer
dog	dog
dog	dog
dog	dog
truck	truck

Trokštama	**Prognozuota**
frog	frog
frog	deer
frog	frog
horse	horse
horse	horse
horse	deer
ship	ship
ship	deer
ship	airplane
truck	truck

identifikuojamos, nors bendras validavimo tikslumas yra aukštas. Galėtume suskaičiuoti įvairias metrikas pasinaudojant lentelės duomenimis. Tikslūs suskaičiavimai pateikiami kodo repozitorijoje. Šiuo atveju klasės „truck“, „ship“ ir „automobile“ turi aukščiausią preciziškumą ir atkūrimą; o „cat“ ar „bird“ vieną iš mažiausių. Tuo pačiu 3 lentelėje matyti, kad net aukštesnį preciziškumą turinčios klasės gali nebūti identifikuojamos, tad nepaisant nežemo modelio tikslumo, modelis visgi geriau linkęs atpažinti tam tikras klases. Šis modelis daug mažiau tiktų katėms, paukščiams, šunims identifikuoti nei laivams ar sunkvežimiams.

4. Išvados

Architektūra be sujungimo sluoksnio ir su daugiau pilnai sujungtų sluoksnių blogiau mokėsi ir persimokė, lyginant su kiek daugiau turinčio konvoliucijom. Tačiau nėra taip, kad pridedant daugiau konvoliucijų modelio mokymasis stipriai pagerėja – tam ištirti reikėtų ilgesnių epochų ir daugiau nei dviejų architektūrų palyginimų. Pasirinktas optimizatorius įtakoja, kaip greitai mokysis modelis, dar labiau svarbu, koks bus rinkinio (paketo) dydis, jei renkamas optimizatorius. Išmetimo sluoksniai sumažina persimokymą, tačiau jei išmetimo sluoksnio tikimybė per didelė, nuo to gali kentėti mokymasis. Aktyvacijos funkcija ne ką mažiau svarbi, net nedidelės architektūros neuroninis tinklas blogiau apsimoko, kai naudojama sigmoidinė aktyvacijos funkcija, geriau naudoti ReLU ar ELU, kurios sprendžia gradiento nykimo problemą. Bendras modelio tikslumas nėra pakankamas matas spręsti apie modelio tikslumą, kiekviena klasė gali turėt savo tikslumą ir kitas metrikas, į kurias priklausomai nuo uždavinio, tektų atsižvelgti.