Univerza v Mariboru

Fakulteta za elektrotehniko,
računalništvo in informatiko

Doktorska disertacija

Metoda prilagodljivega uglaševanja slojev konvolucijskih nevronskih mrež pri strojnem učenju s prenosom znanja

Junij 2021

Grega Vrbančič

Univerza v Mariboru

Fakulteta za elektrotehniko,
računalništvo in informatiko

Doktorska disertacija

Metoda prilagodljivega uglaševanja slojev konvolucijskih nevronskih mrež pri strojnem učenju s prenosom znanja

Junij 2021

Grega Vrbančič

Mentor: red. prof. dr. Vili Podgorelec

UDK: 004.032.26:004.651(043.3)

Avtor:	Grega Vrbančič
Mentor:	red. prof. dr. Vili Podgorelec
Naslov:	Metoda prilagodljivega uglaševanja slojev konvolucijskih nevronskih mrež pri strojnem učenju s prenosom znanja
Naslov v angleščini:	Method for adaptive fine-tuning of convolutional neural network layers using transfer learning
UDK:	004.032.26:004.651(043.3)
Ključne besede:	strojno učenje, globoko učenje, učenje s prenosom znanja, klasifikacija, uglaševanje, optimizacija
Število izvodov:	8
Lektoriranje:	Alenka Mirkac

Doktorsko disertacijo posvečam očetu,
Robertu.

Zahvala

Iskreno se zahvaljujem rednemu profesorju dr. Viliju Podgorelcu za ponujeno priložnost ter vpeljavo v raziskovalno delo. S številnimi nasveti, diskusijami in idejami izven okvirov me je spodbujal in usmerjal na moji raziskovalni poti, obenem pa mi omogočal raziskovalno svobodo, za kar sem mu še posebej hvaležen.

Javni agenciji za raziskovalno dejavnost Republike Slovenije se zahvaljujem za financiranje raziskovalnega dela v sklopu programa mladi raziskovalec.

Zahvala gre tudi sodelavcem Inštituta za informatiko za vso pomoč, nasvete in komentarje, ki so pripomogli h kakovosti doktorske disertacije.

Najlepša hvala staršem, ki so mi omogočili študij. Mama, hvala za vse, brez tebe te doktorske disertacije gotovo ne bi bilo.

Posebna zahvala gre dekletu Katarini za vso ljubezen, podporo in razumevanje.

Povzetek

V doktorski disertaciji predstavimo problematiko izbire uglaševanih slojev konvolucijskih nevronskih mrež pri strojnem učenju s prenosom znanja. Z izvedeno analizo vpliva izbire uglaševanih slojev konvolucijske nevronske mreže na uspešnost učenja potrdimo domnevo, da je primerna izbira uglaševanih slojev s ciljem doseganja visoke klasifikacijske uspešnosti odvisna od izbrane arhitekture konvolucijske nevronske mreže ter ciljnega problema oz. izbrane podatkovne zbirke. Z namenom naslovitve problema izbire uglaševanih slojev razvijemo in predlagamo prilagodljivo metodo *DEFT*, ki temelji na algoritmu diferencialne evolucije in deluje popolnoma samodejno, ne glede na uporabljeno arhitekturo konvolucijske nevronske mreže ali ciljni problem. Zaradi velike časovne kompleksnosti predlagane metode v nadaljevanju razvijemo in predlagamo na funkciji izgube temelječo metriko *LDM*, ki v zgodnji fazi učenja uspešno zaznava manj primerne izbire uglaševanih slojev, kar nam omogoča, da za zaznane manj primerne izbire uglaševanih slojev predčasno zaključimo učenje in na tak način zmanjšamo časovno zahtevnost predlagane metode. Uspešnost predlagane metode ovrednotimo z uporabo treh različnih arhitektur globokih konvolucijskih mrež nad tremi raznolikimi slikovnimi podatkovnimi zbirkami. Klasifikacijsko uspešnost predlagane metode z in brez uporabe metrike *LDM* smo primerjali s klasičnimi pristopi učenja globokih konvolucijskih nevronskih mrež. Primerjavo izvedemo z uporabo najpogostejših klasifikacijskih metrik, časom, potrebnim za učenje, ter porabljenim številom epoh. Rezultate smo preverili z uporabo klasičnih metod statistične analize kot tudi z naprednim pristopom Bayesove analize. Izsledki slednje so potrdili tezo, da je mogoče z uporabo metode prilagodljivega uglaševanja slojev konvolucijske nevronske mreže uspešno nasloviti problem izbire slojev ter da lahko z uporabo metrike *LDM* za zaznavo manj primernih izbir uglaševanih slojev učinkovito zmanjšamo število epoh, potrebnih za učenje, ob doseganju primerljivih rezultatov.

Ključne besede: strojno učenje, globoko učenje, učenje s prenosom znanja, klasifikacija, uglaševanje, optimizacija

Abstract

In this Doctoral Dissertation, we present the problem of selecting fine-tunable layers when utilizing transfer learning with the fine-tuning approach for training deep convolutional neural networks. With the conducted empirical analysis of layer selection impact on the training performance, we confirmed the assumption that the most suitable selection of fine-tuned layers depends on the chosen convolutional neural network architecture, as well as on the target problem. In order to address the problem of selecting the most suitable combination of fine-tunable layers, we developed and proposed an adaptive method, *DEFT*, based on a differential evolution algorithm, which works in a straightforward automatic manner using different convolutional neural network architectures. Due to the high time complexity of the proposed method, we developed and proposed a metric derived from the loss value, which is capable of detecting less suitable selections of fine-tunable layers at an early stage of training, which allows us to terminate training early, and, thus, reduce the time complexity of the proposed method. The performance of the proposed method was evaluated by utilizing three different convolutional neural network architectures against three different image datasets. Classification performance of the proposed *DEFT* method, with or without the proposed metric *LDM*, was compared against conventional approaches for training convolutional neural networks. The performance comparison was conducted using the most common classification metrics, consumed time for training, and consumed number of epochs. The statistical analysis of the obtained results was conducted using conventional statistical methods, as well as modern Bayesian analysis based approaches. The results confirmed the initial thesis that the problem of layer selection when utilizing transfer learning with fine-tuning, can be addressed successfully using the proposed adaptive *DEFT* method, and that utilization of the proposed *LDM* metric reduces the number of epochs needed for training effectively, while achieving comparable results.

Keywords: machine learning, deep learning, transfer learning, classification, fine-tuning, optimization

Kazalo

Seznam slik

Seznam tabel

Seznam algoritmov

Seznam uporabljenih kratic

AUC Površina pod krivuljo ROC (angl. area under the curve)

CCE Kategorična prečna entropija (angl. categorical cross-entropy)

CE Prečna entropija (angl. cross-entropy)

CNN Konvolucijska nevronska mreža (angl. convolutional neural network)

DE Diferencialna evolucija (angl. differential evolution)

GD Gradientni spust (angl. gradient descent)

K-CV K-kratno prečno preverjanje (angl. k-fold cross validation)

LDM Na funkciji izgube temelječa metrika (angl. loss derived metric)

MLP Večslojni perceptron (angl. multlayer perceoptron)

NHST Preverjanje statistične značilnosti ničelne domneve (angl. null hypothesis significance testing)

NP Nedeterministični polinomski čas (angl. nondeterministical polyomial time)

ReLU Popravljena linearna enota (angl. rectified linear units)

ROPE Regija praktične enakosti (angl. region of practical equivalence)

RS Naključno iskanje (angl. random search)

SGD Stohastični gradientni spust (angl. stochastic gradient descent)

TNR Delež pravilno klasificiranih primerkov negativnega razreda (angl. true negative rate)

TPR Delež pravilno klasificiranih pozitivnih primerkov (angl. true positive rate)

WSI Zajem celotnega diapozitiva (angl. whole slide imaging)

Poglavje 1

Uvod

If we knew what it was we were doing, it would not be called research, would it?

- Albert Einstein

1.1 Opredelitev raziskovalnega problema

V zadnjih letih je mogoče zaznati velik razvoj raziskovalnega področja strojnega učenja, ki na različne načine vpliva na druga raziskovalna področja. Z doseganjem zavidljivih klasifikacijskih točnosti modernih globokih nevronskih mrež področje globokega učenja privablja mnoge raziskovalce iz številnih raziskovalnih področij, vse od programskega inženirstva [1–4], modeliranja poslovnih procesov [5–8] do biologije [9–12], zdravstva [13, 14], biomedicine [15–17] in medicine [18–24]. Z uporabo modernih tehnik in pristopov globokega učenja jim je omogočeno naslavljanje različnih domensko specifičnih problemov [25], s katerimi se soočajo. Vsi ti moderni pristopi globokega učenja poleg prednosti prinašajo tudi številne izzive [13, 22, 24, 26]. Med pogostejše izzive zagotovo sodijo časovna zahtevnost učenja, optimizacija hiperparametrov [27–29] uporabljenih metod in algoritmov ter ne nazadnje potreba po velikih podatkovnih zbirkah [22, 26, 30], nujnih za doseganje pričakovane oz. želene napovedne točnosti tako naučenih modelov strojnega učenja. S slednjim se še posebej ukvarjajo raziskovalci s področja medicine in njej podobnih področij [13, 22, 24], kjer je pogosto zbiranje zadostne količine podatkov zaradi same narave oz. občutljivosti podatkov oteženo ali celo nemogoče, kot tudi raziskovalci iz področij, kjer so z vpeljavo modernih pristopov globokega učenja začeli šele pred kratkim in posledično ni na voljo dovolj kakovostnih in velikih podatkovnih zbirk.

Kot eden izmed učinkovitejših pristopov naslavljanja problema pomanjkanja kakovostnih in dovolj obsežnih podatkovnih zbirk se je v zadnjem času uveljavilo strojno učenje s prenosom znanja (angl. transfer learning) [31, 32]. V splošnem je pristop učenja s prenosom znanja mogoče definirati kot učenje nove naloge s pomočjo predhodno pridobljenega oz. naučenega znanja iz podobne naloge [33]. Pri reševanju klasifikacijskih problemov z uporabo učenja s prenosom znanja se, kot predhodno naučeno znanje najpogosteje uporablja prednaučene napovedne modele. Tipično so ti modeli predhodno naučeni nad podatkovno zbirko *ImageNet* [34], ki vsebuje več kot 17.000 različnih kategorij slik in skupno več kot 14 milijonov slik. *ImageNet* je v splošnem odlična podatkovna zbirka za učenje napovednih modelov, ki so kasneje uporabljeni kot »vir znanja« pri uporabi učenja s prenosom znanja, saj ni specializirana za specifično domensko področje, kar posledično daje modelom, prednaučenim na tej podatkovni zbirki, bolj splošno »znanje« [35]. Z uporabo takšnih prednaučenih modelov se pri učenju s prenosom znanja posledično zmanjša potreba po ogromni količini podatkov, saj globoke nevronske mreže ne učimo od začetka, ampak ima model globoke konvolucijske nevronske mreže (angl. convolution neural network, CNN) že določeno predznanje. To prednaučeno znanje je shranjeno v obliki uteži posameznih slojev globoke nevronske mreže, ki jih prenesemo iz prednaučenega modela v model, ki naslavlja podobno, vendar ne enako nalogo [36]. S takšnim prenosom znanja se posledično zmanjša tudi časovna zahtevnost učenja modela na novem domenskem problemu, saj v splošnem učenje s takim pristopom zahteva manj ponovitev kot pri klasičnem učenju globoke CNN, obenem pa se lahko poveča tudi uspešnost napovedovanja [37–39].

Učenje s prenosom znanja se v večini primerov uporablja na dva načina. S prvim načinom prednaučene globoke CNN uporabimo kot mehanizem za ekstrakcijo značilnic (angl. feature extraction) [40, 41], slednje pa nato uporabimo kot vhod za konvencionalne metode strojnega učenja. Drugi način pa je z uporabo uglaševanja (angl. fine-tuning) [38, 39, 42, 28] posameznih slojev prednaučene globoke CNN na nov domensko specifičen problem. Uglaševanje slojev pri učenju s prenosom znanja poteka na način, da uteži konvolucijskih slojev prednaučenih modelov globokih CNN prenesemo v novo ustvarjeno ciljno globoko CNN, končne klasifikacijske polno povezane sloje pa pustimo naključno inicializirane [36]. Pri klasifikaciji slik so namreč konvolucijski sloji ključni gradniki, ki vsebujejo znanje oz. sposobnost luščenja pomembnih značilnic iz podanih vhodnih slik. Uglaševanje slojev pri učenju s prenosom znanja poteka na način, da izbrane sloje uglašujemo oz. jih omogočimo za učenje, medtem ko preostale onemogočimo (zamrznemo) za učenje. Tako omogočimo, da določeno znanje iz prednaučenega modela zadržimo (zamrznjeni sloji), znanje uglaševanih slojev pa prilagodimo glede na ciljni problem. Ključna za uspešno uporabo uglaševanja slojev pri učenju s prenosom znanja je torej izbira, katere

sloje uglaševati in katere pustiti zamrznjene [43, 44]. Izbira uglaševanih slojev namreč neposredno vpliva na učinkovitost prenosa oz. na prilagoditev znanja iz prednaučenega modela na izbran ciljni problem. Neprimerna izbira uglaševanih slojev se lahko tako posledično odraža kot nezmožnost učenja modela ali pa kot nizka napovedna točnost naučenega modela [37, 38].

Čeprav pristop učenja s prenosom znanja rešuje problem manka velikih podatkovnih množic, pa se raziskovalci, ki uporabljajo pristop z uglaševanjem slojev, soočajo s specifičnim problemom izbire, katere sloje je smiselno uglaševati in katere pustiti zamrznjene [38, 43]. V splošnem je izbira uglaševanja proces, ki se izvaja ročno in temelji na pristopu s preizkušanjem, dokler ne najdemo ustrezne kombinacije, ki nam vrača zadovoljive rezultate. Posamezne izbire uglaševanih slojev in uspešnost samega uglaševanja tipično ocenjujemo z opazovanjem napredka procesa učenja, ki je merjen z izbrano funkcijo izgube (angl. loss function), pri čemer stremimo h kontinuiranemu padanju vrednosti le-te skozi celoten proces učenja. Zaznavanje primernosti izbir uglaševanih slojev v čim zgodnejših fazah učenja bi tako pripomoglo k pohitritvi samega procesa iskanja najprimernejših slojev za uglaševanje. Manj primerne kombinacije izbir slojev, katerih učenje ne bi napredovalo po pričakovanjih, pa bi lahko na tak način predčasno zaustavili in s tem prihranili čas, ki bi ga sicer porabili za učenje takšne kombinacije.

Najpogosteje motivacija za strategijo izbire uglaševanih slojev nevronske mreže temelji na empiričnih dokazih [43, 45], da spodnji sloji globoke CNN (sloji bližje vhodnemu sloju) ohranjajo bolj abstraktne značilnice, ki so primerne za širši nabor domensko specifičnih problemov, medtem ko zgornji sloji (sloji bližje izhodnemu sloju) večinoma ohranjajo značilnice, specifične za posamezni problem. Najnovejše študije [37, 44] sicer zavračajo to tezo in nakazujejo, da je uspešnost prenosa znanja odvisna od podobnosti primarnega ciljnega problema, na katerem je bila nevronska mreža v osnovi naučena, z novim ciljnim problemom. Ker je večina mrež pri uporabi učenja s prenosom znanja prednaučena na podatkovni množici *ImageNet* ali njej podobni množici, je razlika med slikami omenjene množice in ciljnim problemom pogosto nezanemarljiva. Še posebej je to izrazito pri naslavljanju problemov z domenskih področij, kjer se metode globokega učenja šele uvajajo ter posledično ni na voljo velikih, kakovostnih, označenih podatkovnih množic. Problematika izbire uglaševanih slojev in vpliva izbranih uglaševanih slojev na uspešnost učenja globokih konvolucijskih nevronskih mrež pa ostaja pri tem nenaslovljena.

1.2 Cilji doktorske disertacije

Glede na predstavljen raziskovalni problem izbire uglaševanih slojev globokih konvolucijskih nevronskih mrež z uporabo učenja s prenosom znanja smo zastavili naslednje cilje doktorske disertacije:

Temeljni cilj doktorske disertacije je razviti prilagodljivo metodo, temelječo na pristopu učenja s prenosom znanja z uglaševanjem, ki z uporabo optimizacijskega algoritma izbere najprimernejše sloje konvolucijske nevronske mreže, ki jih je treba uglasiti za namen dosega čim višje točnosti tako naučenega napovednega modela. Ker pa z vpeljavo optimizacijskih algoritmov povečamo časovno zahtevnost že sicer časovno zahtevnega procesa učenja globokih nevronskih mrež, je cilj doktorske disertacije tudi razvoj lastne, na funkciji izgube temelječe metrike (angl. loss derived metric, LDM), ki v zgodnjih fazah učenja omogoča zaznavo manj primernih izbir slojev konvolucijske nevronske mreže. Razvito prilagodljivo metodo in metriko bomo formalno predstavili ter uporabili z različnimi tipi arhitektur globokih CNN. Uspešnost razvite metode ter kombinacije te metode z uporabo metrike *LDM* bomo ovrednotili na množici domensko različnih slikovnih podatkovnih zbirk.

V okviru razvoja metode prilagodljivega uglaševanja slojev konvolucijske nevronske mreže ter metrike *LDM* so cilji doktorske disertacije še:

- preučiti obstoječe pristope in metode za reševanje problema izbire slojev pri uporabi pristopa strojnega učenja s prenosom znanja z uglaševanjem
- analizirati vpliv izbire uglaševanih slojev na uspešnost učenja pri uporabi pristopa učenja s prenosom znanja
- identificirati in analizirati pristope ter metode za predčasno oz. zgodnjo prekinitev učenja
- pregledati metrike klasifikacije za nadzorovano učenje
- implementirati predlagane metode in metrike
- ovrednotiti razvite metode in metrike

1.3 Teza doktorske disertacije

Glede na zastavljene cilje doktorske disertacije smo oblikovali naslednjo tezo:

> *Z uporabo metode prilagodljivega uglaševanja slojev konvolucijske nevronske mreže pri učenju s prenosom znanja je mogoče uspešno nasloviti problem izbire*

slojev. Z vpeljavo zaznavanja manj primernih izbir slojev v tovrstno metodo lahko učinkovito zmanjšamo časovno zahtevnost ob doseganju primerljivih ali boljših rezultatov od primerjanih klasičnih metod učenja globokih konvolucijskih nevronskih mrež.

Izhajajoč iz predstavljene teze doktorske disertacije smo oblikovali naslednje hipoteze:

(H1) Izbira uglaševanih slojev konvolucijske nevronske mreže je odvisna od ciljnega problema ter uporabljene arhitekture globoke konvolucijske nevronske mreže.

(H2) Razvita namenska metrika *LDM* omogoča zaznavo manj primernih izbir slojev konvolucijske nevronske mreže.

(H3) Razvita metrika *LDM* bolje zaznava manj primerne izbire slojev konvolucijske nevronske mreže kot klasični pristop predčasnega ustavljanja učenja.

(H4) Razvita metoda prilagodljivega uglaševanja slojev konvolucijske nevronske mreže pri učenju s prenosom znanja dosega boljše rezultate v primerjavi s klasičnimi pristopi.

(H5) Razvita metoda z uporabo metrike *LDM* v krajšem času dosega primerljive ali boljše rezultate od razvite metode brez uporabe metrike *LDM*.

(H6) Razvita metoda z uporabo metrike *LDM* v enakem času dosega boljše rezultate v primerjavi z metodo brez uporabe *LDM*.

1.4 Izvirni znanstveni prispevki

V sklopu doktorske disertacije bomo ustvarili naslednje izvirne znanstvene prispevke:

(IZP1) Empirična analiza vpliva izbire uglaševanih slojev globoke konvolucijske nevronske mreže pri učenju s prenosom znanja.

(IZP2) Razvita namenska metrika *LDM*, ki v zgodnjih fazah učenja omogoča zaznavo manj oz. bolj primernih izbir uglaševanih slojev konvolucijske nevronske mreže.

(IZP3) Primerjava uspešnosti razvite metrike *LDM* s klasičnim pristopom predčasnega ustavljanja učenja.

(IZP4) Razvita metoda prilagodljivega uglaševanja slojev konvolucijske nevronske mreže pri učenju s prenosom znanja.

(IZP5) Podrobna primerjava in ovrednotenje rezultatov razvite metode z in brez uporabe metrike *LDM* s klasičnimi metodami in pristopi.

1.5 Predpostavke in omejitve

V sklopu doktorske disertacije nismo postavili predpostavk, bomo pa upoštevali naslednje omejitve:

(O1) Pri uporabi konvolucijskih nevronskih mrež se bomo omejili zgolj na nadzorovano učenje ter učenje s prenosom znanja.

(O2) Razvito metodo bomo aplicirali na omejeno množico arhitektur konvolucijskih nevronskih mrež (*VGG16* [46], *ResNet50* [47], *MobileNet* [48]).

(O3) Pri primerjavi razvite metode s klasičnimi metodami se bomo omejili na metodo nadzorovanega učenja konvolucijskih nevronskih mrež, metodo učenja s prenosom znanja ter metodo učenja s prenosom znanja z uporabo uglaševanja slojev konvolucijske nevronske mreže.

(O4) Pri primerjavi razvite metode s klasičnimi metodami se bomo omejili na primerjavo omejene množice podatkovnih zbirk (rentgenske slike, histološke slike, športne slike).

(O5) Pri primerjavi razvite metode z drugimi metodami se bomo omejili na splošno uveljavljene metrike napovedne uspešnosti klasifikacijskih modelov.

1.6 Struktura doktorske disertacije

Doktorska disertacija obsega devet poglavij. Po uvodnem poglavju sledi pregled področja strojnega učenja, v katerem podrobneje predstavimo globoko učenje, arhitekture globokih konvolucijskih nevronskih mrež ter učenje s prenosom znanja. Tretje poglavje zajema predstavitev in izvedbo analize vpliva izbire slojev konvolucijske nevronske mreže na uspešnost učenja, kar nas privede do prilagodljivega uglaševanja slojev konvolucijske nevronske mreže, predstavljenega v četrtem poglavju. V petem poglavju predstavimo pristop zaznavanja primernosti izbire slojev konvolucijske nevronske mreže pri uglaševanju, s katerim nadgradimo pristop prilagodljivega uglaševanja slojev. V šestem poglavju predstavimo eksperimentalno ogrodje, ki zajema opis uporabljenih podatkovnih zbirk, arhitektur globokih konvolucijskih nevronskih mrež, nastavitev parametrov metod in metrik, način vrednotenja rezultatov ter eksperimentalno okolje. V sedmem poglavju predstavimo in z različnih

vidikov analiziramo rezultate opravljenih eksperimentov. Osmo poglavje je namenjeno diskusiji, interpretaciji rezultatov, predstavitvi omejitev in možnosti uporabe ter nadaljnjega razvoja. Doktorsko disertacijo zaključimo z devetim poglavjem, v katerem povzamemo rezultate, zaključke in sklepe raziskovanja.

Poglavje 2

Strojno učenje

Take any old classification problem where you have a lot of data, and it's going to be solved by deep learning. There's going to be thousands of applications of deep learning.

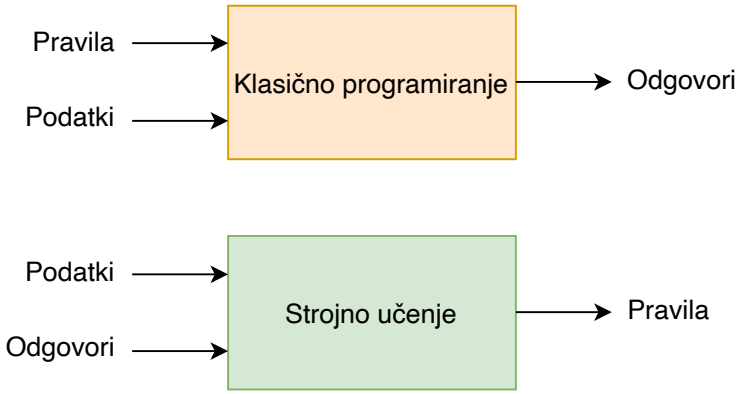
- Geoffrey Hinton

Učenje s prenosom znanja je ena izmed tehnik, ki je pogosto uporabljena pri globokem učenju, globoko učenje pa je specifična vrsta strojnega učenja. Z namenom boljšega razumevanja v doktorski disertaciji uporabljenih tehnik v tem poglavju najprej predstavimo učne algoritme v splošnem, nato pa se podrobneje posvetimo klasifikacijskim metodam in vrednotenju klasifikacijskih modelov. Zatem se osredotočimo na globoko učenje ter arhitekture globokih konvolucijskih nevronskih mrež in na koncu predstavimo še pristop učenja s prenosom znanja.

2.1 Algoritmi strojnega učenja

Algoritem strojnega učenja je algoritem, ki se je sposoben učiti iz podanih podatkov. Na tem mestu si velja najprej zastaviti vprašanje, kaj sploh je strojno učenje. Mitchell v svoji knjigi [49] strojno učenje definira tako: »Računalniški program, ki se uči iz izkušenj E glede na posamezen razred naloge T in merila uspešnosti P, če se njegova uspešnost pri reševanju nalog T, ki je merjena s P, izboljša z izkušnjo E.« Na primer, računalniški program, ki se uči igranja šaha, lahko izboljša svojo uspešnost, *merjeno z njegovo zmožnostjo zmage*, pri skupini nalog, ki so vključene pri *igranju šaha*, skozi izkušnje, *pridobljene z igranjem iger sam proti sebi* [49].

V splošnem nam strojno učenje oz. algoritmi strojnega učenja omogočajo reševanje nalog oz. problemov, ki so prezahtevni, da bi jih lahko rešili s klasičnimi, rigidnimi



Slika 2.1 Primerjava klasičnega programiranja s strojnim učenjem

računalniškimi programi, ki imajo vnaprej določeno obnašanje, to obnašanje pa je definirano z uporabo pravil, sprejetih s strani ljudi. V prej podani, precej formalni Mitchellovi definiciji strojnega učenja naloga ni sam proces učenja, ampak je učenje zgolj sredstvo za doseganje sposobnosti opravljanja neke naloge. Če na primer želimo robota naučiti metati na koš, potem je naša ciljna naloga metanje na koš. Takšnega robota lahko programiramo na način, da se bo sam učil metati na koš, lahko pa poskusimo neposredno spisati program, ki definira, kako mora robot metati na koš ročno.

Takšen pristop k reševanju nalog oz. problemov nas tako pripelje do drugačne paradigme programiranja – strojnega učenja (glej Slika 2.1). Do sedaj smo ljudje s programiranjem vnašali v programe pravila in podatke, pri čemer smo kot rezultat programa dobili odgovore, pri strojnem učenju pa s programiranjem v programe vnašamo podatke in odgovore za določeno specifično nalogo oz. problem ter kot rezultat programa dobimo pravila.

Naloge strojnega učenja so v splošnem predstavljene na način, ki določa, kako mora sistem strojnega učenja procesirati posamezen primer opravljene naloge. S strojnim učenjem je tako mogoče reševati različne naloge, med najpogostejše pa sodijo naslednje [50]:

- **Klasifikacija:** V to skupino nalog spadajo takšne, kjer je računalniški program postavljen pred nalogo, katere cilj je vhodni podatek razvrstiti v eno izmed k kategorij oz. razredov.
- **Regresija:** Med regresijske naloge spadajo takšne, kjer je računalniški program postavljen pred nalogo, katere cilj je številčna napoved za podan vhodni podatek. Ta skupina nalog je podobna klasifikaciji s to razliko, da je oblika pričakovanega izhoda drugačna.

- **Zaznavanje anomalij:** Pri nalogah zaznavanja anomalij računalniški program preseje niz dogodkov ali objektov, pri čemer nekatere izmed njih označi kot nenavadne ali netipične.

- **Sintetizacija in vzorčenje:** Pri tovrstnih nalogah od algoritma strojnega učenja zahtevamo, da sintetizira nove primerke, ki so podobni tistim v učnih podatkih.

Za ovrednotenje sposobnosti algoritma strojnega učenja ne glede na tip naloge, ki ga neki algoritem naslavlja, moramo za namen ovrednotenja oblikovati kvantitativno mero za uspešnost. Navadno je takšna kvantitativna mera P odvisna od naloge T, ki jo določeni algoritem strojnega učenja izvaja. V splošnem nas zanima, kako dobro neki algoritem strojnega učenja deluje nad podatki, ki jih v fazi učenja ni še nikoli procesiral, saj to določa, kako dobro bo tak algoritem deloval v realnem okolju. S tem namenom kvantitativne mere uspešnosti merimo nad tako imenovanimi testnimi podatki, ki so ločeni od podatkov, ki jih algoritem uporablja za učenje (učni podatki).

Algoritmi strojnega učenja so v splošnem kategorizirani v skupine na podlagi tega, s kakšno izkušnjo E so soočeni z učnimi podatki v procesu učenja [50]:

- **Nenadzorovano učenje** (angl. unsupervised learning): Pri tej skupini algoritmov so le-ti soočeni z učnimi podatki, ki vsebujejo mnogo značilnic, iz katerih se skušajo naučiti uporabnih lastnosti strukture podatkov. Nekateri algoritmi strojnega učenja te skupine opravljajo tudi naloge gručenja, katerih cilj je razdelitev podatkov na smiselne gruče podobnih primerkov.

- **Nadzorovano učenje** (angl. supervised learning): Algoritmi iz skupine nadzorovanega učenja so prav tako soočeni z učnimi podatki, ki vsebujejo mnogo značilnic, vendar so za razliko od algoritmov skupine nenadzorovanega učenja posamezni primerki znotraj učne množice pri nadzorovanem učenju opremljeni tudi z oznako oz. ciljnim razredom.

- **Spodbujevano učenje** (angl. reinforcement learning): Za razliko od prejšnjih dveh skupin so algoritmi iz skupine spodbujevanega učenja soočeni ne le zgolj z učnimi podatki, ampak tudi s povratno informacijo, ki učni sistem in podatke povezuje s povratno zanko. Povratna informacija pa je v procesu učenja podana kot nagrada ali kazen sistemu, odvisno od izida učenja.

2.2 Klasifikacija

Eden izmed osrednjih ciljev te doktorske disertacije je razvoj metode za reševanje klasifikacijskih nalog, katere primarna naloga je razvrščanje vhodnih podatkov, v

našem primeru slik, v ciljne razrede. V nadaljevanju tako podrobneje predstavimo nalogo klasifikacije ter klasifikacijske metode.

Naloga klasifikacije je razvrščanje oz. klasificiranje vhodnih, neoznačenih primerkov v vnaprej določen ciljni razred. Formalno je klasifikator model ali funkcija M, ki napoveduje oznako ciljnega razreda $\widehat{y}$ za vsak podan vhodni primerek x:

$$\widehat{y} = M(x), \tag{2.1}$$

kjer je $x = (x_1, x_2, \ldots, x_d)^T \in \mathbb{R}^d$ točka v d– dimenzionalnem prostoru in $\widehat{y} \in \{c_1, c_2, \ldots, c_k\}$ njen napovedan razred [51]. Kadar naslavljamo problem, v katerem je posamezni primerek lahko razvrščen med zgolj dva razreda, govorimo o binarnem klasifikacijskem problemu ali binarni klasifikaciji, če je posamezni primerek lahko razvrščen med več kot dva razreda, pa govorimo o večrazredni klasifikaciji.

Klasifikacijske metode (klasifikatorje) v splošnem razdelimo na šest skupin [52]: statistične metode (naivni Bayesov klasifikator, logistična regresija), odločitvena drevesa (evolucijska drevesa, CART, C4.5), odločitvena pravila, nevronske mreže (usmerjene, konvolucijske, s povratno zanko), metode podpornih vektorjev (SVM) in klasifikatorji na podlagi podobnosti (k-najbližjih sosedov).

Poleg deljenja klasifikacijskih metod glede na algoritem učenja lahko klasifikatorje delimo tudi na podlagi načina učenja modela. Večina klasifikacijskih metod gradi oz. uči model v fazi učenja in nato uporabi naučen model za napoved razredov specifičnih testnih instanc v fazi testiranja. Tipični predstavniki klasifikatorjev, ki jih učimo na tak način, so iz skupin odločitvenih dreves, odločitvenih pravil, nevronskih mrež in metod podpornih vektorjev. Tak način gradnje modela oz. učenja pogosto imenujemo tudi takojšnje ali nestrpno učenje (angl. eager learning). V nasprotju z metodami takojšnjega učenja pa poznamo tudi metode, ki se učijo na podlagi posameznih primerkov (angl. instance-based learning), pri katerih pa navadno ni ločnice med fazo učenja in fazo testiranja. Pri slednjem za vsak posamezni primerek, ki ga želimo klasificirati, zgradimo lokalni model za specifični testni primerek. Tipični predstavniki takšnega načina učenja so metode iz skupine klasifikatorjev na podlagi podobnosti [53]. Učenje na podlagi posameznih primerkov pogosto imenujemo tudi leno učenje (angl. lazy learning), saj ni izvedeno vnaprej in posledično nima modela, ampak se izvede šele ob podanem posameznem primerku [54].

Klasifikatorji spadajo v skupino metod, ki jih učimo z uporabo nadzorovanega učenja. Ker v doktorski disertaciji predstavljamo klasifikacijsko metodo, temelječo na konvolucijski nevronski mreži, ki jo učimo s pristopom takojšnjega učenja, v nadaljevanju formaliziramo posplošen proces takšnega učenja.

Če predpostavimo, da je model M definiran kot $Y = f(X) + \epsilon$, pri čemer ϵ predstavlja napako, se pri nadzorovanem učenju model poizkuša naučiti f z učenjem iz podanih primerkov oz. učne množice. Učno množico lahko definiramo kot nabor opazovanj τ, vključno z vsemi vhodi in izhodi, kjer velja $\tau = (x_i, y_i)$, za vsak $i, \ldots, N$, kjer N predstavlja število primerkov učne množice. Vhodne vrednosti posameznih opazovanj x_i so poslane v učenje učnemu algoritmu, ki na podlagi le-teh ustvarja izhodne vrednosti $\widehat{f(x_i)}$. Učni algoritem ima lastnost, da lahko prilagaja oz. spreminja odvisnost med vhodi in izhodi $\widehat{f}$ glede na razliko v rezultatu (napako) $y_i - \widehat{f(x_i)}$ med resničnimi in ustvarjenimi oz. napovedanimi izhodi [52]. Napoved za vsak nov x_i je tako odvisna od velikosti napake, ki pa jo lahko izrazimo z uporabo funkcije izgube. Pri oblikovanju funkcije $\widehat{f(x_i)}$, ki kategorizira x_i v enega izmed razredov, resnični razred ni znan. Znana je le napovedna porazdelitev $p(\widehat{f}|x_i)$ v učni množici. Če $L(y_i, \widehat{f(x_i)})$ predstavlja podporno funkcijo (funkcijo izgube) pri določanju napovedi $\widehat{f(x_i)}$, pri čemer je y_i resnični razred, potem je pričakovana funkcija izgube:

$$L\left(\widehat{f(x_i)}\right) = \sum_{y_i} L\left(y_i, \widehat{f(x_i)}\right)\ p\left(\widehat{f}|x_i\right), \tag{2.2}$$

optimalna napovedna funkcija $\widehat{f(x_i)}$ pa takšna, ki minimizira pričakovano vrednost funkcije izgube:

$$\hat{f}(x_i) = \underset{\hat{f}(x_i)}{\operatorname{argmin}}\ L\left(\hat{f}(x_i)\right) \tag{2.3}$$

Cilj učnega algoritma je torej poiskati takšno funkcijo $\widehat{f(x_i)}$, da bo vrednost $L(\widehat{f(x_i)}$ čim manjša [55]. Po koncu učnega procesa upamo, da so napovedani izhodi in resnični izhodi dovolj podobni, da so uporabni za vse mogoče nabore vhodov, ki jih je mogoče zaznati v praksi [52].

2.3 Vrednotenje klasifikacijskih modelov

Pri uporabi in razvoju poljubne klasifikacijske metode je ključnega pomena vrednotenje uspešnosti le-tega. Če za učenje izbrane klasifikacijske metode potrebujemo učno množico oz. učne podatke, pa za vrednotenje naučenega modela potrebujemo testno množico oz. testne podatke, s katerimi modela v procesu učenja nismo nikoli soočili. V ta namen so bile razvite različne metodologije vrednotenja in metrike uspešnosti, ki jih podrobneje predstavimo v naslednjih poglavjih.

2.3.1 Metodologije vrednotenja

V preteklosti so bile metodologije vrednotenja zastavljene na način, da so tako za učenje kot za testiranje uporabljale celotno podatkovno množico. Rezultati metrik uspešnosti pridobljenih na tak način, so bili seveda preoptimistični [56], saj je bil napovedni model ovrednoten nad podatki, s katerimi je bil v fazi učenja že soočen. Ta pojav pogosto imenujemo tudi odtekanje podatkov (angl. data leakage).

V izogib pojavu odtekanja podatkov je bila razvita bolj smiselna metodologija vrednotenja klasifikatorjev, metodologija izločitve (angl. hold-out evaluation) [57]. Pri tej metodologiji je podatkovna množica razdeljena na dva dela, učnega in testnega, pri čemer je navadno učni del večji kot testni. Čeprav je metodologija enostavna in onemogoča odtekanje podatkov, pa ima nekaj ključnih slabosti. Bistvena slabost te metodologije je, da podatkovna množica nikoli ni popolnoma raziskana oz. je klasifikator ovrednoten le na manjši podmnožici celotne podatkovne množice [53]. Lahko se namreč zgodi, da testna množica zajema zgolj primerke, ki so zelo enostavni ali zelo zahtevni, kar se posledično odraža na način, da so metrike uspešnosti zavajajoče. Dodaten problem predstavlja tudi možnost dogodka, da v učni množici niso vključeni vsi za učenje potrebni primerki (npr. v učni množici ni primerkov nekega razreda), kar predstavlja težavo pri fazi testiranja, saj napovedni model v fazi učenja s primerki določenega razreda ni bil soočen in posledično nima sposobnosti takšnega primerka v fazi testiranja razvrstiti v pravilno kategorijo.

Kot nadgradnja je bil razvit sistematični pristop, imenovan k-kratno prečno preverjanje (angl. k-fold cross-validation, K-CV) [58], pri katerem se metodologija izločitve ponovi večkrat. K-CV je uveljavljena metodologija, ki podatkovno množico razdeli na K enako velikih delov. Vsak izmed njih je uporabljen za vrednotenje napovednega modela, naučenega na preostalih $K-1$ delih. Ključna prednost takšnega pristopa je, da je vsak primerek v podatkovni zbirki ovrednoten natančno enkrat. Izbira vrednosti K predstavlja kompromis med pristranskostjo in spremenljivostjo, pri čemer manjše vrednosti K pripomorejo k večji pristranskosti (angl. bias), odvisno od spremembe uspešnosti klasifikatorja glede na primerke, vključene v učno množico, in od velikosti te množice. Za višje vrednosti K postane pristop bolj spremenljiv zaradi večje odvisnosti klasifikatorja od učnih podatkov, saj so si vsi učni nabori med seboj precej podobni. Za K sta običajno dobri vrednosti 5 ali 10, pogosto pa se uporablja tudi soroden pristop, kjer se 2-CV ponovi petkrat [53].

2.3.2 Metrike uspešnosti

Poleg izbire metodologije vrednotenja je pomembno izbrati tudi primerne metrike uspešnosti napovednega modela. Za vrednotenje klasifikacijskih modelov je primarni

Tabela 2.1 Matrika zmede za dva ciljna razreda.

		Napovedano	
		Pozitivno (c_1)	Negativno (c_2)
Resnično	Pozitivno (c_1)	TP	FP
	Negativno (c_2)	FN	TN

vir metrik učinkovitosti napovednih modelov matrika zmede (angl. confusion matrix) ali kontingenčna tabela (angl. contingency table). Primer matrike zmede za binarni klasifikacijski problem je predstavljen v tabeli 2.1.

Kadar imamo opravka z binarnim klasifikacijskim problemom, prvi razred c_1 označimo kot pozitiven razred, drugi razred c_2 pa kot negativen. Posamezna polja v matriki zmede za tak binarni klasifikacijski problem prikazujejo štiri vrednosti, ki predstavljajo število instanc glede na pridobljene rezultate klasifikacije:

- TP (angl. true positives): Število primerkov, ki jih je klasifikator pravilno označil kot pozitivne.
- FP (angl. false positives): Število primerkov, ki jih je klasifikator označil kot pozitivne, v resnici pa pripadajo negativnemu razredu.
- FN (angl. false negatives): Število primerkov, ki jih je klasifikator označil kot negativne, v resnici pa pripadajo pozitivnemu razredu.
- TN (angl. true negatives): Število primerkov, ki jih je klasifikator pravilno označil kot negativne.

Z uporabo rezultatov matrike zmede lahko izpeljemo množico uporabljenih metrik uspešnosti. Metriki, ki vrednotita napovedni model globalno, sta točnost (angl. accuracy), predstavljena v enačbi 2.4, in metrika delež napake (angl. error rate), predstavljena v enačbi 2.5. Metrika točnost, je najpogosteje uporabljana metrika za vrednotenje klasifikatorjev [59], ki predstavlja delež pravilnih napovedi glede na vse podane napovedi klasifikatorja, medtem ko metrika delež napake predstavlja delež napačnih napovedi izmed vseh podanih napovedi klasifikatorja. Pri vrednotenju napovednega modela z metriko točnost je cilj doseči čim višjo vrednost metrike znotraj intervala $[0, 1]$, po drugi strani pa je pri vrednotenju napovednega modela z metriko delež napake cilj doseči čim nižjo vrednost, prav tako znotraj intervala $[0, 1]$.

$$točnost = \frac{TP + TN}{TP + FP + TN + FN} \tag{2.4}$$

$$delež\ napake = \frac{FP + FN}{TP + FP + TN + FN} \tag{2.5}$$

V skupino drugih najpogosteje uporabljanih metrik za vrednotenje uspešnosti napovednih modelov spadajo metrike priklic (angl. recall), preciznost (angl. precision) in specifičnost (angl. specificity). Za razliko od prejšnjih dveh metrik, točnost in delež napake, ki vrednotita klasifikator globalno, je tem trem metrikam skupno, da se osredotočajo zgolj na pravilno razvrščene primerke testne množice. Vrednosti metrik so enako kot pri metriki točnost znotraj intervala $[0, 1]$, pri čemer je cilj doseči čim višjo vrednost.

Metrika priklic, pogosto poimenovana tudi senzitivnost (angl. sensitivity) ali delež pravilno razvrščenih pozitivnih primerkov (angl. true positive rate, TPR), predstavlja delež pravilno razvrščenih primerkov pozitivnega razreda glede na vse primerke pozitivnega razreda. Metrika je definirana v enačbi 2.6, kjer n_i predstavlja število primerkov pozitivnega razreda.

$$priklic = senzitivnost = TPR = \frac{TP}{TP + FN} = \frac{TP}{n_i} \tag{2.6}$$

Metrika preciznost, pogosto poimenovana tudi točnost pozitivno razvrščenih pozitivnih primerkov (angl. true positive accuracy), predstavlja delež pozitivnih primerkov, razvrščenih v pozitivni razred. Metrika preciznost je predstavljena v enačbi 2.7.

$$preciznost = \frac{TP}{TP + FP} \tag{2.7}$$

Metrika specifičnost, pogosto imenovana tudi delež pravilno razvrščenih primerkov negativnega razreda (angl. true negative rate, TNR), predstavlja delež pravilno razvrščenih primerkov negativnega razreda v negativni razred. Formalno je metrika specifičnost predstavljena v enačbi 2.8.

$$specifičnost = TNR = \frac{TN}{TN + FP} \tag{2.8}$$

Metrike uspešnosti napovednih modelov ne uporabljamo zgolj za številčno ovrednotenje napovednih klasifikacijskih modelov, ampak tudi za primerjavo različnih napovednih modelov med seboj. Čeprav je v ta namen najpogosteje uporabljana metrika točnost, pa je ta primerna zgolj, kadar imamo enakomerno porazdeljene primerke med ciljnimi razredi. Z drugimi besedami, če imamo razrede z bistveno različno porazdelitvijo primerkov, metrika točnost teži k vrednosti metrike priklic za številčno najbolj zastopan razred v testni množici [60]. V izogib temu problemu je bila razvita množica metrik [60, 61], mi pa se v nadaljevanju osredotočimo na dve:

vrednost Kappa (angl. Kappa value) in površina pod krivuljo ROC (angl. area under the curve, AUC).

Metrika vrednost Kappa [62] naslavlja omenjeno problematiko z vpeljavo pričakovane vrednosti metrike naključnega klasifikatorja. Naključni klasifikator posamezne primerke razvršča med razrede na podlagi porazdelitve razredov. Tako je vrednost Kappa izračunana na sledeč način [60]:

$$vrednost\ Kappa = \frac{točnost - P_c}{1 - P_C}, \tag{2.9}$$

kjer je verjetnost P_C izražena kot:

$$P_c = \sum_{m=1}^{M} (\frac{n_m}{n} \cdot \frac{n_m^{'}}{n}), \tag{2.10}$$

kjer M predstavlja število razredov, n_m število primerkov, pripadajočih m-temu razredu, in $n_m^{'}$ predstavlja število razvrščenih primerkov v m-ti razred. Metrika vrednost Kappa ima zalogo vrednosti na intervalu $[-1, 1]$, kjer 1 predstavlja optimalno vrednost, 0 predstavlja vrednost naključne klasifikacije, -1 pa najslabšo vrednost.

Pogosto uporabljano grafično orodje za vrednotenje klasifikacijskih modelov je izris krivulje ROC [60]. Najpogosteje je izris krivulje ROC uporabljan pri vrednotenju binarne klasifikacije. V tem kontekstu je graf ROC za binarne klasifikacijske probleme opredeljen kot dvodimenzionalni diagram, ki na osi x prikazuje delež napačno razvrščenih pozitivnih primerkov (angl. false positive rate, FPR), kar lahko definiramo tudi kot $1 -$ *specifičnost*, na osi y pa priklic klasifikatorja. Klasifikacijski modeli v splošnem vračajo verjetnost napovedi za posamezni razred, pri čemer je navadno prag za odločitve $0,5$. Krivuljo ROC pridobimo tako, da vrednost praga na intervalu $[0, 1]$ prestavljamo za majhen, diskreten korak ter v vsaki tako izbrani vrednosti pragu izračunamo priklic in vrednost $1 -$ *specifičnost* pri dani napovedi. Ker pa je ovrednotenje na podlagi takšne grafične predstavitve mnogokrat težavno, je bolj priročna in pogosto uporabljana metrika AUC, ki meri ploščino pod krivuljo ROC. Formalno lahko izračun površine oz. ploščine pod krivuljo AUC izrazimo kot [63]:

$$AUC(m) = \int_{\Theta} FPR(m_0) \cdot TPR(m_0) d\Theta, \tag{2.11}$$

kjer m predstavlja napovedni model in Θ predstavlja odločitveni prag.

Metrika AUC je uporabna zgolj pri binarnih klasifikatorjih, zato so bile razvite različne variante oz. razširitve metrike AUC [61] za uporabo pri večrazrednih klasifikacijskih problemih, kjer je število ciljnih razredov večje od 2. Ena izmed omenjenih variant metrike AUC je tudi metrika AUNP, predstavljena v delu [64], ki

izračunava AUC c-dimenzionalnega klasifikatorja v obliki uteženе vsote c binarnih klasifikatorjev, pri čemer upošteva predhodno verjetnost posameznega razreda P_C:

$$AUNP = \sum_{j_i}^{c} p(j)AUC(j, rest_j), \tag{2.12}$$

kjer $rest_j$ predstavlja vse razrede, različne od j.

2.4 Globoko učenje

Globoko učenje je specifično podpodročje strojnega učenja in predstavlja nov pristop učenja na podlagi predstavitve učnih podatkov na način, pri katerem je poudarjeno učenje iz sloja v sloj bolj smiselnih predstavitev podatkov. V splošnem globoko učenje ni strogo vezano na uporabo nevronskih mrež, je pa uporaba v navezavi z nevronskimi mrežami v praksi najpogostejša. Globoko učenje se ne sklicuje na globlje razumevanje delovanja, ki bi ga dosegli s takšnimi pristopi, ampak bolj na osnovno idejo zaporedno povezanih slojev, izmed katerih vsak sloj nagrajuje predhodno predstavitev vhodnih podatkov. Moderno globoko učenje v splošnem vključuje raznovrstne arhitekture nevronskih mrež, sestavljene iz več deset ali tudi več sto zaporednih slojev, ki se samodejno učijo iz podanih vhodnih podatkov. Število takšnih zaporednih slojev arhitekture nevronske mreže imenujemo tudi globina modela. Če v kategorijo globokega učenja uvrščamo pristope, ki uporabljajo mnogo zaporednih slojev umetne nevronske mreže, na drugi strani pristope, ki uporabljajo zgolj enega ali dva takšna sloja, pogosto uvrščamo v kategorijo plitkega učenja (angl. shallow learning).

V splošnem pristopi globokega učenja temeljijo na umetnih nevronskih mrežah. Umetna nevronska mreža je matematični model, katerega osnovna ideja temelji na delovanju bioloških nevronskih mrež, ki so sestavljene iz množice medsebojno povezanih umetnih nevronov, povezave med posameznimi umetnimi nevroni pa so utežene. Osnovni gradnik umetnih nevronskih mrež je nevron oz. perceptron. Perceptron za vhodni podatek x izračuna vrednost $f(x)$, ki označuje izhodno vrednost perceptrona, kar lahko formalno predstavimo kot:

$$y = \phi(\sum_{i=1}^{n}(x_i \cdot w_i + \theta_i), \tag{2.13}$$

kjer ϕ predstavlja aktivacijsko funkcijo, w vektor uteži povezav, θ pristranskost in n število vhodnih povezav perceptrona. Izhodna vrednost perceptrona je poleg vhodnih podatkov odvisna tudi od izbrane aktivacijske funkcije. Aktivacijska funkcija je pri modernih pristopih globokega učenja ključna oz. je uspešnost učenja v veliki

meri odvisna od pravilne izbire le-te. V splošnem naj bi bila aktivacijska funkcija nelinearna, kar nam z uporabo nevronskih mrež omogoča modeliranje nelinearnih odvisnosti. Med pogosteje uporabljenimi aktivacijskimi funkcijami so:

- Linearna funkcija (identiteta): formalno definirana kot $f(x) = x$, navadno uporabljena na vhodnem sloju.
- Sigmoidna funkcija (sigmoid): formalno definirana kot $f(x) = \frac{1}{1+e^{-x}}$, navadno uporabljena pri binarnih klasifikacijskih problemih.
- Funkcija hiperbolični tangens: formalno definirana kot $f(x) = tanh(x)$, predstavlja razširjeno obliko sigmoidne funkcije.
- Funkcija popravljena linearna enota (angl. rectified linear units, ReLU) [65]: formalno definirana kot $f(x) = max(0, x)$, predstavlja preprosto nelinearno preslikavo, ki navzdol omejuje linearno funkcijo. Je ena izmed najpopularnejših aktivacijskih funkcij zaradi enostavne implementacije, časovne učinkovitosti ter odpornosti do pojava ničelnega gradienta.
- Softmax funkcija: formalno definirana kot v enačbi 2.14, uporabljana predvsem na izhodnih slojih pri reševanju večrazrednih klasifikacijskih problemov.

$$f(x)_i = \frac{e^{x_i}}{\sum_{m=1}^{M} e^{x_m}} \tag{2.14}$$

Skupek nevronov oz. perceptronov z enako definicijo, ki so tipično medsebojno nepovezani, imenujemo sloj. Širina takšnega sloja je definirana s številom posameznih nevronov, zaporedje medsebojno povezanih slojev pa imenujemo model nevronske mreže. Najosnovnejši primer globokega učenja so usmerjene nevronske mreže (angl. feedforward neural networks) ali z drugim imenom večslojni perceptron (angl. multilayer perceptron, MLP), ki sestojijo iz vhodnega sloja (angl. input layer) na začetku, izhodnega sloja na koncu (angl. output layer) ter poljubno mnogo skritih slojev (angl. hidden layer) med njima. Formalno lahko izhod j-tega skritega sloja h izrazimo kot [50]:

$$h^{(j)} = \phi(W^{(j)}h^{(j-1)} + \theta^{(j)}), \tag{2.15}$$

kjer je ϕ nelinearna aktivacijska funkcija, $W^{(j)}$ matrika uteži na j-tem sloju, $\theta^{(j)}$ pristranskost in $h^{(0)} = x$.

2.4.1 Učenje nevronske mreže

Za učenje v prejšnjem poglavju predstavljenega večslojnega perceptrona v splošnem uporabljamo metodo vzvratnega razširjanja (angl. backpropagation) [66], ki vključuje tehniko, imenovano pravilo delta (angl. delta rule), poznano kot metodo gradientnega spusta (angl. gradient descent, GD), in izkorišča lastnosti verižnega pravila. Metoda vzvratnega razširjanja z uporabo GD iterativno minimizira funkcijo izgube s posodabljanjem parametrov (uteži) w in θ v nasprotni smeri glede na smer gradienta funkcije izgube. Velikost spremembe posodobitve parametrov je določena s stopnjo učenja α (angl. learning rate). Tak postopek ponavljamo iterativno, pri čemer učeni model nevronske mreže v vsaki ponovitvi učenja seznanimo z novim primerkom ali podmnožico učnih primerkov, s katerimi do tega trenutka še ni bil seznanjen.

Ideja algoritma vzvratnega razširjanja je, da algoritem za serijo vhodnih podatkov opravi prehod naprej in prehod nazaj. V fazi prehoda naprej se izračunajo napovedi modela glede na podane vhodne podatke ter vrednost funkcije izgube oz. napake, ki je nastala pri napovedi. V drugi fazi, pri prehodu nazaj, pa se s pomočjo parcialnih odvodov izračuna vpliv posamezne uteži na končno izračunano napako oz. vrednost funkcije izgube in na podlagi te priredi vrednosti uteži v modelu. Delovanje algoritma vzvratnega razširjanja je podrobneje predstavljeno v obliki psevdokoda v algoritmu 2.1.

Proces učenja z metodo vzvratnega razširjanja se začne z začetno nastavitvijo uteži w in parametrov pristranskosti θ modela nevronske mreže na naključno vrednost. Način nastavitve začetnih vrednosti lahko vpliva na hitrost konvergence, zato so bile v ta namen razvite različne strategije začetnih nastavitev, kot so Glorot-uniform [67], Glorot-normal [67] ali strategija ortogonalne matrike [68].

Vsak primerek učnega podatka ter pripadajoče oznake ciljnega razreda (označen z X) je procesiran na naslednji način: Najprej X pošljemo skozi nevrone vhodnega sloja brez kakršnihkoli sprememb. V nadaljevanju za nevrone skritih slojev izračunamo vhode I_j in izhode O_j. Vhodi nevronov skritih slojev in nevronov izhodnega sloja so izračunani kot linearna kombinacija njihovih vhodov. Za izračun vhodov nevronov posameznega sloja (skritega ali izhodnega) je torej treba zmnožiti izhod posameznega nevrona v predhodnem sloju s pripadajočimi utežmi in vse te zmnožke sešteti. Na podlagi tako izračunanih vhodov I_j z uporabo aktivacijske funkcije ϕ izračunamo izhod O_j. To izvedemo za vse nevrone skritih slojev kot tudi za nevrone izhodnega sloja, ki daje napoved. Sledi prehod nazaj oz. vzvratno popravljanje napake. V obratnem vrstnem redu, od izhodnega sloja pa do prvega skritega sloja, razširjamo napovedno napako s posodabljanjem uteži povezav w in parametrov pristranskosti θ. Za nevrone izhodnega sloja j je napaka izračunana s formulo:

$$Err_j = O_j(1 - O_j)(T_j - O_j), \tag{2.16}$$

kjer O_j predstavlja dejanski izhod, T_j pa resnično vrednost oznake učnega primerka X. Omeniti velja še, da v enačbi 2.16 zapis $O_j(1 - O_j)$ predstavlja odvod aktivacijske funkcije ϕ. Za izračun napake na posameznem nevronu skritega sloja j je upoštevana utežena vsota napak povezanih nevronov v prejšnjem sloju. Izračun napake lahko formalno predstavimo kot:

$$Err_j = O_j(1 - O_j) \sum_k Err_k w_{jk}, \tag{2.17}$$

kjer w_{jk} predstavlja uteži na povezavi med nevronom j ter k, pri čemer slednji spada v prejšnji (višji) sloj, in kjer Err_k predstavlja napako nevrona k. V procesu učenja tako izračunano napako pogosto imenujemo tudi učna napaka.

Uteži in parametri pristranskosti so spremenjeni in posodobljeni na način, da odražajo razširjano napako. Izračun za spremembo ter posodobitev uteži je prikazan v enačbah 2.18 in 2.19.

$$\Delta w_{ij} = (\alpha) Err_j O_i \tag{2.18}$$

$$w_{ij} = w_{ij} + \Delta w_{ij} \tag{2.19}$$

Pri spremembi uteži igra ključno vlogo tudi stopnja učenja $\alpha \in [0, 1]$, ki je navadno nizka konstanta vrednost in vpliva na relativno spremembo uteži. Če je stopnja učenja nastavljena na prenizko vrednost, se to odraža v počasnem učenju, v nasprotnem primeru pa lahko pride do pojava velike oscilacije med neprimernimi rešitvami, kar lahko posledično privede do nezmožnosti učenja modela. Pogosto se med izvajanjem učenja stopnja učenja spreminja. V ta namen so razvite različne strategije zmanjševanja stopnje učenja [69], v splošnem pa velja, da je primerna začetna vrednost stopnje učenja navadno $1 \cdot 10^{-3}$ [70].

Vrednosti parametrov pristranskosti so spremenjene in posodobljene z enačbami 2.20 in 2.21, kjer $\Delta\theta_j$ predstavlja spremembo parametra pristranskosti θ_j.

$$\Delta\theta_j = (\alpha) Err_j, \tag{2.20}$$

$$\theta_j = \theta_j + \Delta\theta_j, \tag{2.21}$$

Predstavljen postopek ponavljamo, dokler ne dosežemo dovolj majhne učne napake ali dokler ni dosežen kateri drug zaustavitveni pogoj. Navadno za zaustavitveni

pogoj uporabimo število ponovitev učenja oz. epoh, pri čemer se kot ena ponovitev učenja šteje, ko učeni model soočimo z vsemi učnimi primerki v učni množici. Takšno ponovitev učenja imenujemo tudi epoha.

Funkcija izgube

Ključna pri učenju z uporabo algoritma vzvratnega razširjanje je funkcija izgube, s katero izračunavamo napako napovedi, saj so na podlagi napake prirejene uteži v učenem modelu. Navadno pri naslavljanju problema klasifikacije uporabljamo metodo največjega verjetja (angl. maximum likelihood), iz česar sledi, da je vrednost funkcije izgube pri učenju enostavno enaka negativni vrednosti logaritemskega verjetja (angl. log-likelihood), kar je enakovredno opisu prečne entropije (angl. cross-entropy, *CE*) med dvema porazdeljenima merama. Formalno lahko prečno entropijo definiramo, kot je predstavljeno v enačbi 2.22, kjer q označuje dano distribucijo verjetnosti in p označuje »resnično« distribucijo [50].

$$CE = H(p, q) = -\sum_{x} p(x) \log(q(x)) \tag{2.22}$$

V doktorski disertaciji smo za učenje CNN uporabili eno izmed variacij te funkcije, imenovano kategorična prečna entropija (angl. categorical cross-entropy, *CCE*), ki jo lahko formalno izrazimo kot [71]:

$$CCE = -\frac{1}{M} \sum_{i=1}^{M} \sum_{j=1}^{C} y_{ij} log(p_{ij}) \tag{2.23}$$

kjer i označuje posamezne primerke izmed skupnih primerkov M, j označuje posamezne razrede izmed skupnih razredov C, y označuje oznako posameznega primerka, $p_{ij} \in (0, 1) : \sum_j p_{ij} = 1 \forall i, j$ pa predstavlja napoved za posamezni primerek.

Optimizacijski algoritmi

Optimizacijski algoritmi, namenjeni za učenje globokih nevronskih mrež, se od klasičnih optimizacijskih algoritmov razlikujejo na več načinov, saj strojno učenje navadno deluje posredno. V večini primerov nas pri strojnem učenju zanima metrika uspešnosti P, ki je izračunana nad testnimi podatki in je pogosto nerešljiva. Tako metriko P optimiziramo zgolj posredno. V postopku učenja minimiziramo funkcijo izgube $J(\theta)$ v upanju, da se bo s tem posledično izboljšala tudi metrika P. To je seveda v nasprotju s klasičnimi optimizacijskimi algoritmi, kjer je minimizacija J cilj sam po sebi. Cilj optimizacijskih algoritmov pri strojnem učenju je zmanjševanje pričakovane generalizirane napake s prilagajanjem učnih parametrov modela strojnega učenja.

Algoritem 2.1: Psevdokod algoritma vzvratnega razširjanja.

Vhod : Učna množica *UM* s pari primerkov in pripadajočim ciljnim razredom
Vhod : Stopnja učenja α
Vhod : Model nevronske mreže *M*
Vhod : Število epoh *e*
Rezultat : Naučen model *M*

begin
  inicializiraj_utezi_in_pristranskost(*M*);
  for $i = 1 \ldots, e$ **do**
    for *each X in UM* **do**
      // Prehod naprej
      for *each M.vhodni_sloj j* **do**
        $O_j = I_j$; // Izhod nevrona vhodnega sloja je enak vhodu
      end
      for *each unit j in (M.skriti_sloji or M.izhodni_sloj)* **do**
        // Izračun vhodov enot sloja j, glede na predhodni sloj i
        $I_j = \sum w_{ij} O_i + \theta_j$;
        $O_j = \phi(I_j)$; // Izračun izhodnih vrednosti enot
      end
      // Prehod nazaj
      for *each unit j in M.izhodni_sloj* **do**
        // Izračun napake
        $Err_j = O_j(1 - O_j)(T_j - O_j)$; // T_j = resnična vrednost
      end
      for *each unit j in vzvratno(M.skriti_sloji)* **do**
        // Izračuna napake glede na naslednji višji sloj k
        $Err_j = O_j(1 - O_j) \sum_k Err_k w_{jk}$;
      end
      for *each weight* w_{ij} *in M* **do**
        $\Delta w_{ij} = (\alpha) Err_j O_i$; // Sprememba uteži
        $w_{ij} = w_{ij} + \Delta w_{ij}$; // Posodobitev uteži
      end
      for *each bias* θ_j *in M* **do**
        $\Delta \theta_j = (\alpha) Err_j$; // Sprememba pristranskosti
        $\theta_j = \theta_j + \Delta \theta_j$; // Posodobitev pristranskosti
      end
    end
  end
end

Eden izmed najbolj uporabljanih optimizacijskih algoritmov je stohastični gradientni spust (angl. stochastic gradient descent, SGD), ki služi kot temelj večini naprednejših, novejših optimizacijskih algoritmov. Algoritem SGD, za razliko od klasičnega algoritma GD, kjer je izračunan gradient za celotno množico učnih podatkov, gradient izračunava za vsak par (primerek in ciljni razred) sproti. Če enačbo klasičnega algoritma GD za iskano funkcijo izgube $J(\theta)$ formalno definiramo, kot je predstavljeno v enačbi 2.24, lahko nadgrajeni algoritem SGD formalno definiramo z enačbo 2.25.

$$\theta = \theta - \alpha \Delta_\theta Err[J(\theta)] \tag{2.24}$$

$$\theta = \theta - \alpha \Delta_\theta J(\theta; x_i, y_i) \tag{2.25}$$

Računanje gradienta za vsak vhodni par podatkov je časovno zahtevno, dodatno pa je problem takšnega pristopa še občutljivost na varianco v podatkih, zato se v večini uporablja algoritem SGD v kombinaciji z minipaketi (angl. mini batch). Na ta način združimo prednosti obeh algoritmov, saj z minipaketi naključno izberemo podmnožico, nad katero izračunamo povprečno napako (enako kot pri GD), odvod povprečne napake ter posodobimo učne parametre. Proces nato ponavljamo, dokler v obliki minipaketov ne obdelamo celotne učne množice. Zaradi pristopa z uporabo minipaketov se zmanjša tudi občutljivost algoritma na varianco v podatkih. Algoritem SGD z uporabo minipaketne obdelave učnih podatkov je prikazan v enačbi 2.26, kjer predstavlja B število primerkov v posameznem minipaketu. Nastavitev velikosti minipaketa je v večini primerov odvisna od učnih podatkov in uporabljene arhitekture nevronske mreže, tipično pa je privzeto dobra začetna vrednost 32 [69].

$$\theta = \theta - \alpha \frac{1}{B} \sum_{i=1}^{B} (\phi(x_i) - y_k) x_k \tag{2.26}$$

Pogosto uporabljena nadgradnja algoritma SGD je tudi vpeljava metode momenta (angl. momentum) [72], ki pripomore k pospeševanju ter omili oscilacijo algoritma SGD. Algoritem momenta akumulira eksponentno pojemajočo drsečo sredino predhodnih gradientov ter pospešuje gibanje v njihovi smeri. Formalno algoritem momenta vpelje spremenljivko v, ki predstavlja hitrost, s katero se parametri pomikajo skozi parametrski prostor. Hitrost je nastavljena na vrednost eksponentno pojemajočega povprečja negativnega gradienta, s spremenljivko $\gamma \in [0, 1)$, pa definiramo, kako hitro bodo prispevki prejšnjih gradientov pojenjali. Z metodo momenta nadgrajen algoritem SGD je predstavljen v enačbi 2.27 in 2.28.

$$v = \gamma v + \alpha \nabla_\theta J(\theta) \tag{2.27}$$

$$\theta = \theta - v \tag{2.28}$$

V zadnjem času je eden izmed popularnejših optimizacijskih algoritmov algoritem Adam [73]. Njegova ključna lastnost je prilagajanje učne stopnje za vsak parameter. Poleg hrambe eksponentno pojemajočega povprečja kvadratov gradienta v algoritem Adam dodatno ohranja tudi pojemajoče povprečje preteklih gradientov m. Zaradi slednjega lahko delovanje algoritma Adam v splošnem opišemo kot gibanje težke krogle s trenjem. Izračunavanje povprečja preteklih gradientov vpliva na hitrost pomikanja parametrov, kar posledično algoritmu onemogoča, da bi zašel v majhna lokalna območja, in s tem navadno preide lokalne minimume [74]. Izračun povprečja gradientov ter povprečja kvadratov gradienta je predstavljen v enačbah 2.29 in 2.30, kjer g_t predstavlja izračun gradienta, β_1 in β_2 pa predstavljata faktor pojemka za m in v, ki sta približka prvega momenta ter drugega momenta gradientov.

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \tag{2.29}$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \tag{2.30}$$

Zaradi pojava pristranskosti m_t in v_t proti ničli, še posebej v začetnih korakih t, so avtorji vpeljali popravke za izračun približkov m_t in v_t, predstavljene v enačbah 2.31 in 2.32.

$$\widehat{m_t} = \frac{m_t}{1 - \beta_1^t} \tag{2.31}$$

$$\widehat{v_t} = \frac{v_t}{1 - \beta_2^t} \tag{2.32}$$

Za posodobitev parametrov je uporabljen izračun, predstavljen v enačbi 2.33. Avtorji algoritma [73] za vrednost ϵ predlagajo $1 \cdot 10^{-8}$, za faktorja pojemka β_1 ter β_2 pa $0,9$ in $0,999$. V splošnem je optimizacijski algoritem Adam manj občutljiv na izbiro parametrov ter dobro deluje za učenje globokih nevronskih mrež iz tega razloga ga tudi uporabimo v eksperimentalnem delu doktorske disertacije.

$$\theta_{t+1} = \theta_t - \frac{\alpha}{\sqrt{\widehat{v_t}} + \epsilon} \widehat{m_t} \tag{2.33}$$

Sloj normalizacije paketov

V naprednih arhitekturah CNN je pogosto uporabljen tudi sloj normalizacije paketov (angl. batch normalization layer). Sloj uporablja metodo normalizacije paketov [75], ki prilagaja učne parametre glede na zahtevnost učenja globokih modelov CNN. Globoki modeli so sestavljeni iz kompozicij mnogih funkcij oz. slojev, ki jim z izračunom gradienta določimo posodobitev vsakega učnega parametra oz. uteži, pod predpostavko, da se ostali sloji ne spremenijo. V praksi pa posodabljamo vse parametre v vseh slojih sočasno, pri čemer prej omenjena predpostavka ne drži, kar privede do učinka, da je izredno zahtevno določiti primerno stopnjo učenja, saj je vpliv posodobitve enega sloja močno odvisen od ostalih slojev [50].

Metoda normalizacije paketov vpeljuje eleganten način ponastavitve parametrov praktično kakršnekoli globoke nevronske mreže, s čimer bistveno zmanjša problem koordinacije posodabljanja parametrov preko več slojev. Normalizacija paketov je lahko uporabljena na vhodu kot tudi na skritih slojih nevronske mreže. Naj bo H minipaket aktivacij sloja, ki ga želimo normalizirati, v obliki matrike, kjer so v posameznih vrsticah aktivacije za posamezni primerek. Normalizacijo minipaketa izvedemo na način, da H nadomestimo s [50]:

$$H^{'} = \frac{H - \mu}{\sigma}, \tag{2.34}$$

kjer μ predstavlja vektor, vsebujoč povprečja posameznih nevronov, in σ vektor vsebujoč standardne odklone za posamezne nevrone. Izračun temelji na uporabi vektorjev μ in σ za vsako vrstico matrike H. Znotraj posamezne vrstice matrike H, se operacije izvajajo za posamezni element, kjer je $H_{i,j}$ normaliziran z odštevanjem μ_j in deljenjem s σ_j.

Normalizacija paketov tako vpeljuje tehniko normalizacije vhodov oz. izhodov posameznega sloja za vsak minipaket, kar vpliva na stabilnost učnega procesa ter na pohitritev učenja, saj smo z vpeljavo le-te korak bližje predpostavki, da se distribucija vhodnih podatkov med posodobitvijo uteži ne bo spreminjala, vsaj ne signifikantno.

Regularizacija

Eden izmed ključnih izzivov strojnega učenja je razviti algoritem, ki bo deloval dobro ne le na učni množici, pač pa tudi na novih vhodnih podatkih. Sposobnost dobrega delovanja na še ne videnih podatkih v splošnem imenujemo generalizacija. Sposobnost generalizacije navadno vrednotimo z uporabo testne množice podatkov in izračunom napake nad testno množico (testna napaka). Obstaja mnogo strategij, ki so oblikovane z namenom zmanjšanja testne napake, včasih tudi na račun povečanja učne napake. Takšne strategije in pristope v splošnem imenujemo regularizacija. [50]

Mnogo regularizacijskih pristopov temelji na omejevanju kapacitete modelov z dodajanjem kazni norme parametrov $\Omega(\theta)$ (angl. parameter norm penalty) ciljni funkciji izgube J. Tako regularizirano ciljno funkcijo izgube lahko označimo z $\tilde{J}$ ter jo formalno predstavimo kot:

$$\tilde{J}(\theta; x, y) = J(\theta; x, y) + \rho\Omega(\theta), \tag{2.35}$$

kjer $\rho \in [0, \infty)$ predstavlja parameter, ki določa relativni prispevek kazni Ω k ciljni funkciji J. Manjša vrednost ρ se tako izraža kot manjša stopnja regularizacije in obratno. Pri uporabi takšnega pristopa regularizacije za učenje nevronskih mrež je pomembno, da regulariziramo zgolj uteži w, in ne tudi parametrov pristranskosti θ, saj bi z regularizacijo slednjih lahko vpeljali signifikantno mero preslabega prileganja (angl. underfitting). Pojav preslabega prileganja se zgodi, kadar model ni zmožen doseči dovolj nizke učne napake.

V nasprotju s pojavom preslabega prileganja poznamo tudi pojav prekomernega prileganja (angl. overfitting), do katerega pride, kadar je razlika med učno napako in testno napako prevelika. Z drugimi besedami, kadar model nima sposobnosti znanja pridobljenega na učnih podatkih generalizirati na še ne videne testne podatke. Pri naslavljanju klasifikacijskih problemov je ena izmed pogosteje uporabljenih tehnik za preprečevanje prekomernega prileganja tehnika bogatenja podatkov (angl. dataset augmentation). S tem pristopom generiramo nove učne primerke z uporabo različnih transformacij nad obstoječimi učnimi primerki, in tako razširjamo učno množico, ki je pri učenju globokih nevronskih mrež ključnega pomena za doseganje visoke stopnje generalizacije.

Kadar učimo model z zadostno kapaciteto učnih predstavitev, ki lahko doseže prekomerno prileganje ciljni nalogi, pogosto opazimo, da se učna napaka skozi ponovitve učenja konstantno zmanjšuje, po drugi strani pa se validacijska napaka začne sčasoma zviševati. Validacijska napaka je napaka, ki jo po koncu vsake epohe izračunamo za podmnožico primerkov začetne učne množice, ki v fazi učenja ni uporabljena. To pomeni, da lahko pridobimo model z boljšo (nižjo) vrednostjo validacijske napake (in posledično možno nižjo vrednostjo testne napake) s ponastavitvijo parametrov na vrednosti v točki učenja, kjer je bila validacijska napaka najnižja. Če želimo to doseči, moramo v vsaki točki učenja, kjer se validacijska napaka zmanjša, shraniti vrednosti parametrov modela. Ko se učenje zaključi, povrnemo vrednosti parametrov na tiste, ki so dosegli najnižjo validacijsko napako. Učenje se zaključi, ko se validacijska napaka ne izboljša za vnaprej določeno število ponovitev učenja oz. epoh. Vnaprej določeno število ponovitev učenja pogosto imenujemo potrpežljivost (angl. patience) in označujemo z oznako p. Takšen pristop, formalno predstavljen v algoritmu 2.2, je v splošnem poznan kot zgodnja prekinitev učenja

(angl. early stopping) in je verjetno najpogosteje uporabljena oblika regularizacije pri učenju globokih nevronskih mrež [50].

Algoritem 2.2: Psevdokod zgodnje prekinitve učenja.

Vhod : Učna množica UM z pari primerkov in pripadajočim ciljni razredom
Vhod : Model nevronske mreže M
Vhod : Število epoh e
Vhod : Potrpežljivost p
Rezultat: Naučen model M

```
1  begin
       // Učno množico razdelimo na učno in validacijsko (navadno v
          razmerju 90:10)
2      UM_ucna, UM_validacijska = razdeli(UM);
       // Nastavimo začetne vrednosti začasnih spremenljivk
3      M_najboljsi = M;
4      E_val = ∞;
5      j = 0;
6      while j < e do
7          M.izvedi_ucenje(UM_ucna);
8          E'_val = M.izracunaj_napako(UM_validacijska);
9          if E'_val < E_val then
               // Priredimo vrednost začasnih spremenljivk na trenutno
                  najboljše vrednosti
10             E_val = E'_val;
11             M_najboljsi = M;
12             j = 0;
13         else
14             j = j + 1;
15         end
16     end
       // Po zaključenem učenju vrnemo model na najboljšega doseženega
17     M = M_najboljsi;
18 end
```

Pristop zgodnje prekinitve učenja je nevsiljiva oblika regularizacije, saj ne zahteva praktično nikakršnih sprememb v samem učnem algoritmu, ciljni funkciji ali učnih parametrih in v nasprotju s strategijo kaznovanja učnih parametrov posledično ne vpliva na učno dinamiko. Strategija se lahko uporabi samostojno ali v kombinaciji z drugimi regularizacijskimi strategijami.

Še ena v vrsti popularnih in učinkovitih regularizacijskih strategij, ki je pogosto uporabljana pri globokem učenju, je osip nevronov [76] (angl. dropout), ki predstavlja računsko nezahtevno metodo regularizacije za širok nabor modelov [50]. Metoda osipa nevronov preprečuje prekomerno prileganje in omogoča učinkovit način za

kombiniranje različnih arhitektur nevronskih mrež. Termin osip se navezuje na izpuščanje oz. osip nevronov (skritih ali vidnih) v nevronski mreži. Z osipom nevronov imamo v mislih njihovo začasno odstranitev iz nevronske mreže s pripadajočimi vhodnimi in izhodnimi povezavami. Izbira, katere nevrone odstranimo, je določena s parametrom naključne verjetnosti p, ki je navadno privzeto nastavljen na vrednost $0,5$. Uporaba metode osipa nevronov dejansko predstavlja vzorčenje zoženih nevronskih mrež iz osnovne nevronske mreže, saj z naključnim osipom nevronov nevronski mreži zmanjšamo (zožimo) njeno širino in na tak način ustvarimo novo nevronsko mrežo z manj nevroni. Nevronsko mrežo z n enotami si lahko predstavljamo kot zbirko 2^n možnih zoženih nevronskih mrež. Vse te nevronske mreže si delijo uteži, tako da je skupno število parametrov še vedno $O(n^2)$ ali manjše. Za vsak učni primerek se vzorči nova zožena nevronska mreža in izvede postopek učenja. Tako si lahko učenje nevronske mreže z osipom nevronov predstavljamo kot učenje množice 2^n zoženih nevronskih mrež z obsežnim deljenjem uteži med seboj, kjer se vsaka takšna vzorčena nevronska mreža uči zelo redko, če sploh [76]. V procesu vrednotenja oz. napovedovanja modela nevronske mreže, učenega z osipom nevronov, nobenega nevrona ne odstranimo, temveč skaliramo izhode posameznih slojev s faktorjem enakim izbrani verjetnosti osipa p. Na tak način uravnotežimo pojav, da je v fazi ovrednotenja aktivnih več nevronov, kot jih je bilo v fazi učenja [77].

2.4.2 Konvolucijske nevronske mreže

Konvolucijske nevronske mreže (angl. convolutional neural networks, CNN) so posebna zvrst nevronskih mrež, ki procesirajo podatke, ki imajo mreži podobno topologijo. Primeri takšnih vrst podatkov so časovne vrste, ki si jih lahko predstavljamo kot enodimenzionalne mreže, ki zajemajo podatke v nekem časovnem intervalu, in slike, ki si jih lahko predstavljamo kot dvodimenzionalne mreže slikovnih točk. Preboj arhitektur CNN se je začel leta 1989 s predstavitvijo arhitekture LeNet [78], pravi razcvet pa so doživele v zadnjem desetletju z razvojem programskih knjižnic in strojne opreme, ki omogočajo enostavno uporabo. Različne arhitekture CNN iz leta v leto prikazujejo izjemen uspeh na raznih področjih praktične uporabe, kjer pri določenih klasifikacijskih nalogah celo prekosijo sposobnosti ljudi [79, 80]. Ključna prednost CNN v primerjavi s predhodnimi metodami in pristopi je sposobnost samodejno prepoznati oz. izluščiti ključne značilnice brez pomoči domenskega strokovnjaka ali kakršnegakoli človeškega nadzora. Struktura CNN temelji, podobno kot pri klasičnih nevronskih mrežah, na nevronih človeških in živalskih možganov. Natančneje, v primeru CNN konvolucijski sloji simulirajo kompleksno zaporedje celic, ki tvorijo vizualni korteks v mačjih možganih [81].

V splošnem je klasična struktura CNN sestavljena iz številnih konvolucijskih slojev, ki se praviloma izmenjujejo z združevalnimi sloji. Tako sestavo konvolucijskih slojev in združevalnih slojev imenujemo tudi konvolucijska osnova (angl. convolutional base) in skrbi za uspešno ter učinkovito ekstrakcijo značilnic. Na koncu takšne konvolucijske osnove je navadno dodan eden ali več klasičnih polno povezanih slojev, ki opravljajo vlogo klasifikatorja tako pridobljenih značilnic.

Konvolucijski sloj

Konvolucijske nevronske mreže so ime dobile po operaciji konvolucije, ki je ključen gradnik vsake CNN. V najsplošnejši obliki je konvolucija operacija nad dvema funkcijama f in g, predstavljena v enačbi 2.36.

$$s(t) = (f * g)(t) = \int_{-\infty}^{\infty} f(x)g(t-x)dx \tag{2.36}$$

Iz vidika konvolucijskih nevronskih mrež funkcija f predstavlja vhod (npr. slika), funkcija g pa jedro (angl. kernel) ali filter. Rezultat takšne operacije je pogosto poimenovan mapa ali nabor značilnic (angl. feature map). Ključna ideja konvolucije je identifikacija značilnic v vhodnih podatkih z apliciranjem jedra čez vhodne podatke, pri čemer so tako vhodni podatki kot jedro v obliki večdimenzionalnih mrež oz. polj. Velikost jedra je lahko poljubna, navadno pa je manjša kot dimenzija vhodnih podatkov. Jedra so uporabljena za identifikacijo značilnic v vhodnih podatkih, ki v začetnih slojih prepoznavajo (v primeru slik) zgolj robove in enostavne oblike, v višjih slojih pa kompleksnejše lastnosti. To dosežemo z drsenjem jedra preko vhodnih podatkov in izračunom skalarnega oz. matričnega produkta (odvisno od dimenzij) med prekrivajočim se delom vhodnih podatkov in jedrom. Tak konvolucijski proces je grafično prikazan na sliki 2.2.

Pri uporabi konvolucije je pogosta praksa, da ob robovih vhodnih podatkov dodamo odmik (angl. padding) za primere, kadar je položaj jedra delno izven dimenzij vhodnih podatkov. Navadno to storimo z razširjanjem vhodnih podatkov in dodajanjem ničel ob robovih vhodnih podatkov. Poleg dodajanja odmika lahko povečamo tudi korak (angl. stride) pri drsenju jedra, ki določa, za koliko točk v vhodnih podatkih se bo jedro pomaknilo v vodoravni oz. navpični smeri. S povečevanjem koraka posledično zmanjšujemo dimenzijo izhodnega polja (nabora značilnic). Tak pristop nam omogoča izkoriščanje nespremenljivih lastnosti podatkov, kot sta prostorska lokalnost (angl. spatial locality) in ekvivariantne predstavitve (angl. equivariant representations) značilnic. Prostorska lokalnost se navezuje na razdaljo med ključnimi značilnicami, evivariantne predstavitve pa na lastnost, da je značilnice mogoče identificirati ne glede na položaj v vhodnih podatkih (npr. krog je

mogoče prepoznati kjerkoli na sliki). Obe lastnosti temeljita na rezultatu operacije konvolucije – drsenju jedra nad vhodnimi podatki. Posplošeno gledano je lastnost prostorske lokalnosti dosežena z uporabo jedra nad omejenimi podpodročji vhodnih podatkov, lastnost ekvivariantne predstavitve pa s pomikanjem oz. drsenjem jedra čez vhodne podatke. Takšen pristop tudi zmanjšuje število uteži v primerjavi s klasičnimi nevronskimi sloji, kjer je posamezna utež uporabljena zgolj enkrat, saj enako jedro drsi skozi celoten vhod. Ta koncept je poznan kot deljenje uteži oz. parametrov (angl. parameter sharing) in je prav tako ena izmed ključnih prednosti CNN. Medtem ko se tradicionalni popolnoma povezani sloji zanašajo na veliko število uteži oz. parametrov, ki opisujejo interakcijo med vsakim vhodnim in izhodnim nevronom, uporaba jedra manjših dimenzij od dimenzij vhoda znatno zmanjša število le-teh. To lastnost imenujemo redka povezljivost (angl. sparse connectivity ali angl. sparse interactions) [50].

V enačbi 2.36 je predstavljena splošna oblika operacije konvolucije, v nadaljevanju pa izpeljemo formalno enačbo za izračun nad dvodimenzionalnimi podatki. Če imamo diskretne vrednosti vhodnih podatkov, kar jih tipično imamo, lahko definiramo diskretno operacijo konvolucije kot:

$$s(t) = (f * g)(t) = \sum_{a=-\infty}^{\infty} f(x)g(t-x). \tag{2.37}$$

Pri uporabi CNN so vhodni podatki velikokrat večdimenzionalni, in tako so posledično večdimenzionalna tudi jedra. V teh primerih navadno želimo tudi uporabiti operacijo konvolucije v več kot samo eni smeri (npr. vodoravno in navpično). Če torej imamo na primer dvodimenzionalno sliko I na vhodu ter dvodimenzionalno jedro K, potem lahko operacijo konvolucije zapišemo kot:

$$S(i,j) = (I * K)(i,j) = \sum_{m} \sum_{n} I(i-m, j-n)K(m,n), \tag{2.38}$$

kjer je rezultat konvolucije izračunan za slikovno točko s položajem v vrstici i in stolpcu j, kjer je položaj središča jedra nad vhodno slikovno točko.

V splošnem za konvolucijskim slojem sledi, enako kot pri klasičnih slojih, operacija nelinearne aktivacijske funkcije ter zatem operacija združevanja (združevalni sloj). Pogosto takšnim zaporedjem enega ali več konvolucijskih slojev, ki jim sledi združevalni sloj, rečemo tudi konvolucijski blok.

Združevalni sloji

Glavna naloga združevalnih slojev je podvzorčenje naborov značilnic, ki jih pridobimo kot rezultat operacije konvolucije v prejšnjem koraku. Funkcija združevanja deluje

na način, da značilnice na določenih mestih nadomesti z rezultatom uporabljene statične funkcije bližnjih značilnic. Na primer, združevanje z uporabo maksimizacije (angl. max pooling) [82] nadomesti vrednosti značilnic z maksimalno vrednostjo značilnic znotraj okoliških značilnic. Okolica je navadno definirana podobno kot jedro pri konvoluciji in je v primeru dvodimenzionalne mape značilnic definirana, kot dvodimenzionalno polje, ki prav tako drsi z vnaprej določenim korakom preko mape značilnic. Poleg maksimizacije sta pogosto uporabljani funkciji združevanja še povprečenje ter globalno povprečenje [83]. Primer delovanja podvzorčenja z uporabo združevanja z omenjenimi funkcijami je prikazan na sliki 2.3.

Ne glede na uporabljeno funkcijo združevanja le-to v splošnem pripomore značilnice predstaviti kot invariantne za majhne zamike vhodnih podatkov [50]. Poenostavljeno, če vhodne podatke zamaknemo za majhno vrednost, se večina izhodov združevalnih funkcij ne bo spremenila. To je še posebej uporabna lastnost, če nam je bolj kot točna lokacija značilnice pomembno, da zaznamo, ali je določena značilnica sploh prisotna.

2.4.3 Arhitekture globokih konvolucijskih nevronskih mrež

Od preboja CNN leta 1989 pa do danes je bilo razvitih mnogo izboljšav samih arhitektur CNN. Izboljšave so bile narejene na različnih področjih CNN, od regularizacije pa do strukturnih sprememb. Ključno vlogo pri izboljšanju uspešnosti CNN pa so odigrale izboljšave procesnih enot ter oblikovanje novih različic konvolucijskih blokov. Avtorji v delu [30] arhitekturne inovacije CNN razdelijo v sedem kategorij:

- Arhitekture, temelječe na prostorskem izkoriščanju (angl. spatial exploitation): Zajemajo veliko število uteži, parametrov pristranskosti, slojev, nevronov, velikosti jedra itd. Uporaba različnih velikost jeder omogoča zajem in izluščevanje podrobnih, domensko specifičnih značilnic, ali pa bolj splošnih značilnic.

- Arhitekture, temelječe na izkoriščanju globine: Zaradi predpostavke, da lahko globlji modeli bolje in učinkoviteje predstavijo posamezne razrede ciljne funkcije, tovrstne arhitekture vsebujejo veliko število zaporedno veriženih slojev.

- Arhitekture, ki uporabljajo več poti povezav: Poleg klasičnih zaporednih povezav slojev vključujejo tudi povezave, ki zaobidejo enega ali več vmesnih slojev in na tak način omogočijo poseben pretok informacij preko slojev. S takšnimi povezavami naslavljajo problematiko bledenja gradienta in degradacije uspešnosti, do česar lahko pride zaradi povečane globine CNN.

- Arhitekture, temelječe na izkoriščanju širine: Čeprav lahko s povečano globino modelov CNN zajamemo raznolike značilnice, pa se s povečanjem globine

sposobnost učenja CNN ne poveča nujno. V ta namen so bile razvite širše in plitkejše arhitekture CNN.

- Arhitekture, temelječe na izkoriščanju naborov značilnic: Uporabljajo različne pristope k izbiri naborov značilnic, ustvarjenih v konvolucijskih slojih. Ker nabori značilnic igrajo ključno vlogo pri izboljšanju sposobnosti generalizacije, takšne arhitekture z uporabo uteži primernejše nabore značilnic favorizirajo, medtem ko manj primerne kaznujejo.

- Arhitekture, temelječe na spodbujanju oz. pospeševanju (angl. boosting) posameznega kanala: Sposobnost konvolucijskih jeder za uspešno ekstrakcijo značilnic je v veliki meri odvisna od vhodnih podatkov. Z namenom povečanja raznolikosti vhodnih podatkov takšne arhitekture vpeljujejo koncept spodbujanja posameznih kanalov, pri čemer z uporabo različnih metod na določenem kanalu vhodnega sloja umetno ustvarjajo vhodne podatke.

- Arhitekture, temelječe na pozornosti (angl. attention-based): Temeljijo na fenomenu človeškega vizualnega sistema, kjer ljudje sicer v vidnem polju zajemamo celoten prizor, pozorni pa smo zgolj na vsebinsko pomembne dele. Takšne arhitekture CNN poizkušajo poustvariti omenjen fenomen z vpeljavo posebnih blokov, ki zanemarjajo prostorsko lokalnost značilnic.

V nadaljevanju predstavimo tri arhitekture CNN, ki so predstavnice različnih kategorij in so v zadnjem času med bolj popularnimi. Predstavljene arhitekture tudi uporabimo v eksperimentalnem delu doktorske disertacije.

VGG16

Arhitektura VGG [46] je bila predstavljena leta 2014 s strani skupine raziskovalcev Visual Geometry Group (VGG) z univerze v Oxfordu in je ena izmed prepoznavnejših arhitektur, ki temeljijo na prostorskem izkoriščanju. Ključna lastnost te arhitekture je uporaba dimenzijsko majhnih konvolucijskih jeder s pomikom jedra za eno slikovno točko. Avtorji so predstavili več variant arhitekture VGG, med njimi je najpogosteje uporabljana *VGG16*. Arhitektura *VGG16*, predstavljena v tabeli 2.2, je sestavljena iz 16 slojev z utežmi, od katerih je 13 konvolucijskih slojev, preostali trije sloji pa so popolnoma povezani sloji in služijo kot klasifikator značilnic, izluščenih s predhodnimi konvolucijskimi sloji. Na vhodu arhitektura prejme slike dimenzij 224×224 slikovnih točk na treh barvnih kanalih. Konvolucijska osnova arhitekture *VGG16* je sestavljena iz petih blokov. Prva dva bloka sta sestavljena s po dvema konvolucijskima slojema z velikostjo jedra 3×3 ter korakom pomika jedra za eno slikovno točko v vsako dimenzijo in združevalnim slojem. Konvolucijski sloji v

Tabela 2.2 Struktura arhitekture *VGG16*.

Sloj/Blok	Struktura
Vhodni sloj	$224 \times 224 \times 3$ slikovnih točk
Blok 1	$\begin{bmatrix} \text{konvolucija}, & 64 \text{ jeder } 3 \times 3, & \text{korak } 1 \times 1 \\ \text{konvolucija}, & 64 \text{ jeder } 3 \times 3, & \text{korak } 1 \times 1 \\ \text{združevanje (maks.)}, & \text{okno } 2 \times 2, & \text{korak } 2 \times 2 \end{bmatrix}$
Blok 2	$\begin{bmatrix} \text{konvolucija}, & 128 \text{ jeder } 3 \times 3, & \text{korak } 1 \times 1 \\ \text{konvolucija}, & 128 \text{ jeder } 3 \times 3, & \text{korak } 1 \times 1 \\ \text{združevanje (maks.)}, & \text{okno } 2 \times 2, & \text{korak } 2 \times 2 \end{bmatrix}$
Blok 3	$\begin{bmatrix} \text{konvolucija}, & 256 \text{ jeder } 3 \times 3, & \text{korak } 1 \times 1 \\ \text{konvolucija}, & 256 \text{ jeder } 3 \times 3, & \text{korak } 1 \times 1 \\ \text{konvolucija}, & 256 \text{ jeder } 3 \times 3, & \text{korak } 1 \times 1 \\ \text{združevanje (maks.)}, & \text{okno } 2 \times 2, & \text{korak } 2 \times 2 \end{bmatrix}$
Blok 4	$\begin{bmatrix} \text{konvolucija}, & 512 \text{ jeder } 3 \times 3, & \text{korak } 1 \times 1 \\ \text{konvolucija}, & 512 \text{ jeder } 3 \times 3, & \text{korak } 1 \times 1 \\ \text{konvolucija}, & 512 \text{ jeder } 3 \times 3, & \text{korak } 1 \times 1 \\ \text{združevanje (maks.)}, & \text{okno } 2 \times 2, & \text{korak } 2 \times 2 \end{bmatrix}$
Blok 5	$\begin{bmatrix} \text{konvolucija}, & 512 \text{ jeder } 3 \times 3, & \text{korak } 1 \times 1 \\ \text{konvolucija}, & 512 \text{ jeder } 3 \times 3, & \text{korak } 1 \times 1 \\ \text{konvolucija}, & 512 \text{ jeder } 3 \times 3, & \text{korak } 1 \times 1 \\ \text{združevanje (maks.)}, & \text{okno } 2 \times 2, & \text{korak } 2 \times 2 \end{bmatrix}$
Polno povezan sloj	4096 nevronov, ReLU aktivacijska funkcija
Polno povezan sloj	4096 nevronov, ReLU aktivacijska funkcija
Polno povezan sloj	1000 nevronov, Softmax aktivacijska funkcija

prvem bloku imajo vsak po 64 jeder, v drugem bloku pa vsak po 128 jeder. Preostali trije bloki so sestavljeni s tremi zaporedno povezanimi konvolucijskimi sloji z enakimi nastavitvami kot v prvih dveh blokih ter združevalnim slojem na koncu. Konvolucijski sloji v tretjem bloku imajo vsak po 256 jeder, v četrtem in petem bloku pa vsak po 512 jeder. Vsi konvolucijskih sloji uporabljajo aktivacijsko funkcijo ReLU, vsi združevalni sloji pa uporabljajo maksimizacijsko metodo združevanja ob uporabi okna velikosti 2×2 slikovni točki ter pomiku za dve slikovni točki v obe dimenziji. Za konvolucijsko osnovo so polno povezani trije sloji, ki opravljajo vlogo klasifikatorja. Prva dva polno povezana sloja imata vsak po 4096 nevronov, ki so aktivirani z aktivacijsko funkcijo ReLU, zadnji (izhodni) polno povezan sloj pa vsebuje 1000 nevronov, ki so aktivirani z aktivacijsko funkcijo softmax. Predstavljena arhitektura zajema okrog 138 milijonov učnih parametrov in je posledično računsko precej zahtevna.

ResNet50

Družina arhitektur ResNet [47], predstavljena leta 2016, je nadaljevanje trenda razvoja arhitektur, ki izkoriščajo globino. Zaradi znanih težav z bledenjem gradienta in poslabšanja uspešnosti učenja pri vedno globjih modelih je bil cilj avtorjev oblikovati ekstremno globoko arhitekturo, ki za razliko od obstoječih ne bo trpela za omenjenimi pojavi. Ključna ideja pri arhitekturi ResNet je bila vpeljava koncepta obvodne povezave, poznane kot bližnjica ali preostala (angl. residual) povezava. Koncept je predstavljen na sliki 2.4, ki prikazuje klasično usmerjeno nevronsko mrežo z dodano preostalo povezavo.

S takšnimi povezavami lahko preskočimo oz. zaobidemo poljubno mnogo slojev, podatki, dostavljeni preko takšne povezave, pa so v primeru arhitektur *ResNet* preslikani linearno. Z uporabo tega pristopa dodatno ne povečujemo računske zahtevnosti, saj v procesu ni dodan oz. uporabljen noben dodatni parameter ali utež. Prav tako lahko arhitekture s takšnimi povezavami učimo na enak način kot klasične arhitekture nevronskih mrež. Zaradi uporabe obvodnih povezav arhitekture iz družine ResNet uvrščamo tudi v kategorijo arhitektur, ki izkoriščajo več poti povezav. Iz družine arhitektur *ResNet* izhaja pet različic CNN, ki se razlikujejo po številu slojev. V našem primeru smo izbrali srednjo različico s 50 sloji, *ResNet50*.

V tabeli 2.3 je podrobno predstavljena sestava arhitekture CNN *ResNet50*. Na vhodu arhitektura prejme slike dimenzij 224×224 slikovnih točk na treh barvnih kanalih. Sledi konvolucijski sloj s 64 jedri dimenzije 7×7 ter s premikom za dve slikovni točki v obe dimenziji. Zatem je združevalni sloj, ki za združevanje uporablja metodo povprečenja z oknom velikosti 3×3 ter premikom za dve slikovni točki v obe dimenziji. Sledijo štirje konvolucijski bloki z večkratnimi ponovitvami zaporednih konvolucijskih slojev. Pri vsaki prvi ponovitvi zaporedij v blokih *conv3_x*, *conv4_x* in *conv5_x* je korak pomika jeder v prvem konvolucijskem sloju namesto ene slikovne točke enak dvema slikovnima točka. Povečanje koraka v prvem konvolucijskem sloju vsakega izmed navedenih blokov služi za zmanjšanje dimenzije map značilnic. Po koncu konvolucijskih blokov sledi še združevalni sloj z metodo globalnega povprečenja ter polno povezan (izhodni) sloj s 1000 nevroni in uporabljeno aktivacijsko funkcijo softmax. V primerjavi z arhitekturo *VGG16* ima arhitektura *ResNet50* zgolj okrog 25,5 milijona učnih parametrov.

MobileNet

Arhitektura *MobileNet* [48] temelji na po globini ločljivi konvoluciji (angl. depthwise separable convolution), prvotno predstavljeni v članku [84], na kateri temeljijo arhitekture iz družine Inception. Pri arhitekturah CNN iz družine Inception je omenjeni pristop uporabljen za zmanjšanje računske zahtevnosti v prvih nekaj slojih.

Tabela 2.3 Struktura arhitekture *ResNet50*.

Sloj/Blok	Struktura
Vhodni sloj	$224 \times 224 \times 3$ slikovnih točk
Konvolucija conv1	64 jeder 7×7, korak 2×2
Združevanje	povprečenje, okno 3×3, korak 2×2
Blok conv2_x	$3 \times \begin{bmatrix} \text{konvolucija,} & 64 \text{ jeder } 1 \times 1, & \text{korak } 1 \times 1 \\ \text{konvolucija,} & 64 \text{ jeder } 3 \times 3, & \text{korak } 1 \times 1 \\ \text{konvolucija,} & 256 \text{ jeder } 1 \times 1, & \text{korak } 1 \times 1 \end{bmatrix}$
Blok conv3_x	$4 \times \begin{bmatrix} \text{konvolucija,} & 128 \text{ jeder } 1 \times 1, & \text{korak } 1 \times 1\ (2 \times 2) \\ \text{konvolucija,} & 128 \text{ jeder } 3 \times 3, & \text{korak } 1 \times 1 \\ \text{konvolucija,} & 512 \text{ jeder } 1 \times 1, & \text{korak } 1 \times 1 \end{bmatrix}$
Blok conv4_x	$6 \times \begin{bmatrix} \text{konvolucija,} & 256 \text{ jeder } 1 \times 1, & \text{korak } 1 \times 1\ (2 \times 2) \\ \text{konvolucija,} & 256 \text{ jeder } 3 \times 3, & \text{korak } 1 \times 1 \\ \text{konvolucija,} & 1024 \text{ jeder } 1 \times 1, & \text{korak } 1 \times 1 \end{bmatrix}$
Blok conv5_x	$3 \times \begin{bmatrix} \text{konvolucija,} & 512 \text{ jeder } 1 \times 1, & \text{korak } 1 \times 1\ (2 \times 2) \\ \text{konvolucija,} & 512 \text{ jeder } 3 \times 3, & \text{korak } 1 \times 1 \\ \text{konvolucija,} & 2048 \text{ jeder } 1 \times 1, & \text{korak } 1 \times 1 \end{bmatrix}$
Združevanje	globalno povprečenje
Polno povezan sloj	1000 nevronov, Softmax aktivacijska funkcija

Za razliko od arhitekture Inception, ki se osredotoča zgolj na prvih nekaj slojev, so si avtorji arhitekture *MobileNet* zastavili cilj razviti arhitekturo, ki bo s stališča računske zahtevnosti dovolj učinkovita, da bo uporabna na mobilnih napravah in vgrajenih sistemih. Po globini ločljiva konvolucija je razdeljena na dva dela: globinsko konvolucijo (angl. depthwise convolution) in točkovno (1×1) konvolucijo (angl. pointwise convolution). Pri globinski konvoluciji je za razliko od klasične konvolucije za vsak posamezni barvni kanal vhodne slike uporabljeno novo jedro. V nadaljevanju pa je uporabljena točkovna konvolucija za združitev v globinski konvoluciji pridobljenih naborov značilnic za vsak kanal v enega samega. Pri klasični konvoluciji poteka izluščevanje in združevanje vhodov v nabor izhodov v enem koraku, kar se odraža v večji računski zahtevnosti kot pri sestavljeni, po globini ločljivi konvoluciji. Klasični konvolucijski sloj kot vhod prejme $D_F \times D_F \times M$ veliko mapo značilnic F in ustvari $D_F \times D_F \times N$ veliko mapo značilnic G, kjer je D_F prostorska dimenzija (širina in višina) vhodne mape značilnic, M je število barvnih kanalov (vhodna globina), D_G je prostorska dimenzija izhodne mape značilnic, pri čemer je $D_G = D_F$ in N število izhodnih barvnih kanalov (izhodna globina). Tak sloj je parametriziran z uporabo konvolucijskega jedra K velikosti $D_K \times D_K \times M \times N$, kjer D_K predstavlja prostorsko dimenzijo jedra, za katerega predpostavljamo, da je kvadratno, M označuje število vhodnih kanalov ter N število izhodnih kanalov. Stroški strojnega

računanja (angl. computational cost) pri standardni konvoluciji so izračunani kot [48]:

$$D_K \cdot D_K \cdot M \cdot N \cdot D_F \cdot D_F. \tag{2.39}$$

Globinsko konvolucijo z enim jedrom za posamezni kanal lahko formalno izrazimo kot:

$$\widehat{G}_{k,l,m} = \sum_{i,j} \widehat{K}_{i,j,m} \cdot F_{k+i-1,l+j-1,m}, \tag{2.40}$$

kjer $\widehat{K}$ predstavlja jedro globinske konvolucije velikosti $D_K \times D_K \times M$ in kjer je m-to jedro v $\widehat{K}$ uporabljeno na m-tem kanalu v F za kreacijo m-tega kanala izhodne mape značilnic $\widehat{G}$. Stroški strojnega računanja za globinsko konvolucijo so predstavljeni v enačbi 2.41 ter so v primerjavi s standardno konvolucijo veliko bolj učinkoviti.

$$D_K \cdot D_K \cdot M \cdot D_F \cdot D_F. \tag{2.41}$$

Če globinsko konvolucijo združimo še s točkovno konvolucijo, pa dobimo stroške strojnega računanja:

$$D_K \cdot D_K \cdot M \cdot D_F \cdot D_F + M \cdot N \cdot D_F \cdot D_F \tag{2.42}$$

Z izražanjem konvolucije kot dvostopenjskega postopka filtriranja in kombiniranja pa lahko izpeljemo:

$$\frac{D_K \cdot D_K \cdot M \cdot D_F \cdot D_F + M \cdot N \cdot D_F \cdot D_F}{D_K \cdot D_K \cdot M \cdot N \cdot D_F \cdot D_F} = \frac{1}{N} + \frac{1}{D_K^2}. \tag{2.43}$$

Tako razvita učinkovita arhitektura *MobileNet* je podrobno predstavljena v tabeli 2.4.

Enako, kot pri prejšnjih dveh arhitekturah tudi arhitektura *MobileNet* na vhodu prejme slike dimenzij 224×224 slikovnih točk na treh barvnih kanalih. Zatem sledi klasični sloj konvolucije z jedrom velikosti 3×3 slikovne točke in pomikom jedra v obe dimenziji za 2 slikovni točki. Nato si izmenjaje sledijo sloji globinske konvolucije, označeni s *Conv dw*, ter sloji točkovne konvolucije, označeni s *Conv pw*. Na koncu sta enako kot pri arhitekturi *ResNet50* dodana združevalni sloj z uporabljeno metodo globalnega povprečenja ter na koncu izhodni sloj s 1000 nevroni in uporabljeno aktivacijsko funkcijo softmax. V primerjavi s predhodno predstavljenima arhitekturama CNN ima tako sestavljena arhitektura *MobileNet* zgolj dobre 4 milijone učnih parametrov, kar je 6-krat manj kot *ResNet50* in 32-krat manj kot *VGG16*.

Tabela 2.4 Struktura arhitekture *MobileNet*.

Sloj/Blok	Struktura
Vhodni sloj	$224 \times 224 \times 3$ slikovnih točk
Conv	32 jeder 3×3, korak 2×2
Conv dw	32 jeder 3×3, korak 2×2
Conv pw	64 jeder 1×1, korak 1×1
Conv dw	64 jeder 3×3, korak 2×2
Conv pw	128 jeder 1×1, korak 1×1
Conv dw	128 jeder 3×3, korak 1×1
Conv pw	128 jeder 1×1, korak 1×1
Conv dw	128 jeder 3×3, korak 2×2
Conv pw	256 jeder 1×1, korak 1×1
Conv dw	256 jeder 3×3, korak 1×1
Conv pw	256 jeder 1×1, korak 1×1
Conv dw	256 jeder 3×3, korak 2×2
Conv pw	512 jeder 1×1, korak 1×1
$4\times$	$\begin{bmatrix}\text{Conv dw}, & 512 \text{ jeder } 3 \times 3, & \text{korak } 1 \times 1 \\ \text{Conv pw}, & 512 \text{ jeder } 1 \times 1, & \text{korak } 1\times 1\end{bmatrix}$
Conv dw	512 jeder 3×3, korak 1×1
Conv pw	1024 jeder 1×1, korak 1×1
Združevanje	globalno povprečenje
Polno povezan sloj	1000 nevronov, Softmax aktivacijska funkcija

2.5 Učenje s prenosom znanja

Metode in pristopi nadzorovanega učenja so že dobro raziskano področje in uporabljani na različnih realnih problemih, vendar pa tovrstne metode in pristopi navadno delujejo dobro zgolj v primeru, kadar so učni in testni podatki predstavljeni z enakimi značilnicami in porazdeljeni z enako distribucijo. Hkrati je uspešnost takšnih metod in pristopov pogojena z zagotavljanjem dovolj velikih, visoko kakovostnih množic označenih učnih podatkov. Dodatno je proces zbiranja in priprave primernih podatkovnih zbirk otežen v medicini in njej podobnih področjih [13, 22, 24], kjer je že zaradi narave podatkov oz. občutljivosti le-teh zbiranje oteženo ali celo nemogoče. Omenjeni problem je postal eno izmed glavnih ozkih grl na poti do lažje in uspešnejše uporabe metod in pristopov strojnega učenja v praksi.

V zadnjem desetletju je bilo predlaganih več tehnik delno nadzorovanega učenja (angl. semi-supervised learning) [85–87] in aktivnega učenja (angl. active learning) [88, 89], ki pa problem naslavljata le delno, saj je predpostavljeno, da je učnih podatkov dovolj, le da niso vsi označeni. Za razliko od omenjenih tehnik pristop učenja s

prenosom znanja v splošnem omogoča, da se domene, naloge in distribucije podatkov pri učenju in testiranju razlikujejo. Ključna ideja pri učenju s prenosom znanja je, da s pomočjo obstoječih označenih podatkov ali obstoječih že naučenih modelov prenesemo znanje na drug podoben, ampak ne enak problem. Prve raziskave [90], ki so se osredotočale na učenje s prenosom znanja, segajo v leto 1995, vendar jih je mogoče zaslediti pod različnimi imeni, kot so induktivni prenos (angl. inductive transfer) [91], kumulativno učenje (angl. comulative learning) [92] ali večopravilno učenje (angl. multitask learning) [93], med katerimi je slednji še najbolj soroden učenju s prenosom znanja, kot ga poznamo sedaj. V splošnem lahko tehnike učenja s prenosom znanja definiramo kot izboljšanje učenja nove naloge z uporabo že naučenega obstoječega znanja, pridobljenega z učenjem druge naloge. S stališča učenja globokih modelov CNN lahko učenje s prenosom znanja predstavimo kot prenos učnih parametrov iz modela, naučenega za opravljanje specifične naloge, v drug (nov) model, katerega naloga je reševanje drugačnega, vendar sorodnega problema. Tak pristop učenja s prenosom znanja lahko glede na kategorizacijo, predstavljeno v raziskovalnem delu [31], uvrstimo v kategorijo induktivnega učenja s prenosom znanja (angl. inductive transfer learning). Formalno lahko induktivno učenje s prenosom znanja opišemo kot proces, katerega cilj je z učno ciljno nalogo T_T izboljšati učenje ciljne napovedne funkcije $f_T(\cdot)$ nad ciljno domeno D_T z uporabo znanja iz izvorne domene D_S z učno nalogo T_S.

Glede na dolgo zgodovino metod, tehnik in pristopov učenja s prenosom znanja, je le-to na področju globokega učenja dobilo večjo pozornost šele pred nekaj leti. Razlog lahko pripišemo splošnemu razcvetu globokega učenja in enormnemu porastu uporabe pristopov, metod in tehnik globokega učenja na različnih področjih, ki pa za učinkovito in uspešno delovanje potrebujejo velike količine podatkov. Dodatno se je pristop učenja s prenosom znanja, poleg naslavljanja problematike manka podatkov, izkazal kot učinkovitejši s stališča računske zahtevnosti.

V kontekstu globokega učenja pristopi, metode in tehnike za klasifikacijske naloge temeljijo na modelu, prednaučenem na splošni podatkovni množici, znanje katerega je nato uporabljeno za učenje novega ciljnega problema. V procesu predučenja modela je najpogosteje uporabljena podatkovna zbirka *ImageNet*, ki se je izkazala kot odlična podatkovna zbirka za učenje napovednih modelov, ki so kasneje uporabljeni kot »vir znanja«, saj ni specializirana za neko specifično domensko področje (vsebuje več kot 17.000 kategorij), kar posledično daje modelom, prednaučenim na tej podatkovni zbirki, bolj splošno »znanje« [35].

Učenje s prenosom znanja se v kontekstu globokega učenja v večini primerov uporablja na dva načina. Pri prvem načinu je prednaučena konvolucijska osnova globoke CNN uporabljena kot mehanizem za ekstrakcijo značilnic [40, 41], ki so nato uporabljene za vhodne podatke klasičnim metodam strojnega učenja, kot so

linearna regresija, naključna drevesa, metode podpornih vektorjev ipd. Drugi, v zadnjem času popularnejši pristop pa je z uporabo uglaševanja [38, 39, 42, 28] posameznih slojev prednaučene globoke CNN na nov ciljni domensko specifični problem. Najpogosteje uglaševanje slojev CNN (glej sliko 2.5) pri učenju s prenosom znanja poteka tako, da uteži konvolucijskih slojev prednaučenih modelov globokih CNN prenesemo v novo ustvarjeno ciljno globoko CNN, končne klasifikacijske polno povezane sloje pa pustimo naključno nastavljene z izbrano strategijo nastavitev uteži ali pa jih nadomestimo z lastnimi klasifikacijskimi sloji. Klasifikacijski sloji pri uporabi uglaševanja pri učenju s prenosom znanja niso ključni, saj so naučeni zgolj klasificirati značilnice primerkov izbrane specifične domene, kar pa nam pri uglaševanju na drugo domeno ne koristi. So pa zato toliko pomembnejši konvolucijski sloji prednaučenega modela oz. sloji znotraj konvolucijske osnove CNN, saj so pri klasifikaciji slik le-ti ključni gradniki, ki vsebujejo znanje oz. sposobnost izluščevanja pomembnih značilnic iz podanih vhodnih slik. Uglaševanje slojev pri učenju s prenosom znanja torej poteka na način, da izbrane sloje omogočimo za učenje oz. jih uglašujemo, medtem ko preostale onemogočimo za učenje oz. jih zamrznemo. Tako nadzorujemo, katero znanje iz prednaučenega modela bomo zadržali (zamrznjeni sloji) in katero znanje bomo prilagodili glede na ciljni problem. Posledično je za uspešno uporabo uglaševanja slojev pri učenju s prenosom znanja ključna izbira, katere sloje uglaševati in katere pustiti zamrznjene [43, 44]. Izbira namreč neposredno vpliva na učinkovitost prenosa oz. na prilagoditev znanja iz prednaučenega modela na izbran ciljni problem. Neprimerna izbira uglaševanih slojev se lahko tako posledično odraža kot nezmožnost učenja modela ali pa kot nizka napovedna točnost končnega naučenega modela [37, 38].

Za izbiro uglaševanih slojev globoke CNN pri učenju s prenosom znanja je bilo predstavljenih že več strategij. Od strategije, kjer so uglaševani vsi sloji izbrane arhitekture CNN [94], pa do strategije, kjer so za uglaševanje izbrani vsi sloji konvolucijske osnove brez klasifikacijskih slojev [95]. Ena izmed pogostejših strategij v zadnjem času pa je izbira uglaševanih slojev, temelječa na članku [43], ki naslavlja prenosljivost znanja in nakazuje, da je uglaševane sloje konvolucijske osnove smiselno izbirati od najvišjega konvolucijskega sloja (sloj najbližje izhodnemu) proti najnižjemu sloju (sloj najbližje vhodnemu) glede na podobnost med podatkovno množico, na kateri je bil model prednaučen, in ciljno podatkovno množico. Strategija namreč temelji na dognanjih, da imajo višji konvolucijski sloji sposobnost izluščevanja bolj domensko specifičnih značilnic, nižji sloji pa bolj abstraktnih, splošnih značilnic, ki jih je mogoče uporabiti na različnih domensko specifičnih problemih. Pri uporabi takšne strategije je zato predlagano, da se pri problemih, ki so bolj podobni izvornemu problemu, uglašuje višje sloje konvolucijske osnove. Za probleme, kjer pa se ciljni problem bolj razlikuje od izvornega, pa je dodatno treba poleg visokih konvolucijskih

slojev uglaševati tudi nižje sloje. Ker pa različne novejše raziskave [96, 97, 37] nakazujejo, da pri uporabi modernih, kompleksnejših arhitektur globokih CNN temu ni tako, se v večini primerov uporabe uglaševanja pri učenju s prenosom znanja uglaševani sloji izbirajo ročno oz. na podlagi preteklih izkušenj s posamezno CNN arhitekturo in domenskim problemom. Tak pristop je navadno časovno zahtevna naloga in ni nujno, da se v vseh primerih izkaže za ustreznega. Zaradi kompleksnosti problema izbire uglaševanih slojev globoke CNN, ki je odvisna tako od ciljnega problema kot tudi od uporabljene arhitekture CNN, je eden izmed glavnih ciljev doktorske disertacije naslovitev tega problema z uporabo prilagodljivega uglaševanja slojev CNN.

2.6 Uporaba optimizacijskih metod v strojnem učenju

V splošnem je optimizacija proces iskanja najboljše možne rešitve, pri katerem maksimiziramo ali minimiziramo pripadajočo ciljno funkcijo danega problema. Vsak optimizacijski problem je sestavljen iz treh osnovnih elementov: odločitvenih spremenljivk, ciljne funkcije in omejitev. Tehnike, metode in algoritme, ki naslavljajo optimizacijske probleme, lahko razdelimo v dve skupini: deterministične in nedeterministične metode. Prvi zagotavljajo najdbo optimalne rešitve, medtem ko je drugi ne. Dodatno je pri determinističnih algoritmih težava časovna učinkovitost, saj se časovna zahtevnost reševanja takšnih problemov povečuje eksponentno glede na dimenzijo problema. Po drugi strani pa nedeterministični algoritmi sicer ne zagotavljajo najdbe najbolj optimalne rešitve, najdejo pa v krajšem času nek približek optimalne rešitve [98]. Večina optimizacijskih problemov, s katerimi se soočamo v realnem svetu, spada v skupino zahtevnih problemov, ki so rešljivi v nedeterminističnem polinomskem času (angl. nondeterministical polyomial time, NP). Posledično so takšni problemi najpogosteje naslovljeni z uporabo nedeterminističnih algoritmov.

Pri strojnem učenju se pogosto soočamo z različnimi problemi, ki jih je mogoče nasloviti z uporabo optimizacijskih algoritmov. Najpogosteje so to problemi nastavitve hiperparametrov [99, 29, 28], izbire značilnic [100–102], izgradnje oz. načrtovanja najprimernejše arhitekture NN [103, 104] ali pa načrtovanja klasifikacijskih cevovodov [105]. Za naslovitev takšnih problemov so se v zadnjem času kot bolj uporabljani izkazali populacijski algoritmi [101, 28, 29], ki jih lahko razdelimo v dve skupini [106]: evolucijski algoritmi (angl. evolutionary algorithms) in algoritmi po vzoru iz narave (angl. nature-inspired algoritms). Tipični predstavniki evolucijskih algoritmov so genetski algoritem [107], diferencialna evolucija (angl. differential evolution, DE), evolucijsko programiranje [108], medtem ko so tipični predstavniki algoritmov po vzoru iz narave algoritem optimizacije z roji delcev (angl. particle

swarm optimization) [109], optimizacija s kolonijami mravelj (angl. ant colony optimization) [110], algoritem na osnovi obnašanja netopirjev (angl. bat algorithm) [111] itd.

2.6.1 Diferencialna evolucija

V tem poglavju se osredotočimo na algoritem diferencialne evolucije, saj je eden izmed najbolj popularnih in uspešnih algoritmov za reševanje kompleksnih problemov. Prvi zapis o algoritmu DE se je pojavil leta 1995 [112] v obliki tehničnega poročila avtorjev R. Storna in K. V. Price. Uspešnost algoritma se je prvič potrdila na prvem mednarodnem tekmovanju za evolucijsko optimizacijo leta 1996, ki je potekalo v povezavi z mednarodno konferenco IEEE International Conference on Evolutionary Computation (CEC) [106]. Vse od takrat se algoritem DE ter njegove izpeljanke in nadgradnje na mednarodnem tekmovanju CEC redno uvrščajo med najuspešnejše. Na tekmovanju leta 2020 sta bila med tremi najuspešnejšimi algoritmi dva temelječa na algoritmu DE, in sicer algoritem IMODE [113] ter j2020 [114]. Najnovejši rezultati iz leta 2021 pa še dodatno potrjujejo njegovo uspešnost, saj so se v štirih različnih kategorijah tekmovanja najvišje uvrstili trije na algoritmu DE temelječi algoritmi: jDE21 [115], NL-SHADE-RSP [116] ter MadDE [117].

V nadaljevanju predstavimo osnovni algoritem DE, ki ga v sklopu doktorske disertacije tudi uporabimo. Algoritem DE sestoji iz množice vektorjev realnih števil, ki predstavljajo velikost populacije (angl. number of population, NP), in treh osnovnih operatorjev: mutacija, križanje in selekcija. Vektorji realnih števil predstavljajo kandidatne rešitve oz. posameznike, ki jih lahko formalno definiramo, kot je predstavljeno v enačbi 2.44, kjer je vsak element posameznika znotraj intervala $x_{i,1}^{(t)} \in [x_i^{(L)}, x_i^{(U)}]$, medtem ko $x_i^{(L)}$ in $x_i^{(U)}$ določata spodnjo in zgornjo mejo i-te spremenljivke. Z D je označena dimenzija problema, z NP pa število posameznikov v populaciji, medtem ko je števec generacij označen s t.

$$\mathbf{x}_i^{(t)} = (x_{i,1}^{(t)}, \ldots, x_{i,D}^{(t)}), \quad \text{za } i = 1, \ldots, NP \tag{2.44}$$

Osnovna strategija DE je sestavljena iz operacij mutacije, križanja in selekcije. Operacijo mutacije lahko izrazimo kot:

$$u_i^{(t)} = x_{r1}^{(t)} + F \cdot (x_{r2}^{(t)} - x_{r3}^{(t)}), \quad \text{za } i = 1, \ldots, NP, \tag{2.45}$$

kjer $u_i^{(t)}$ označuje mutiran vektor, F predstavlja skalirni faktor, $r1$, $r2$ in $r3$ pa so naključno izbrane vrednosti iz uniformne porazdelitve v intervalu $1, \ldots, NP$.

Z namenom povečanja raznolikosti posameznih vektorjev je uporabljen operator križanja, ki ga lahko formalno predstavimo kot:

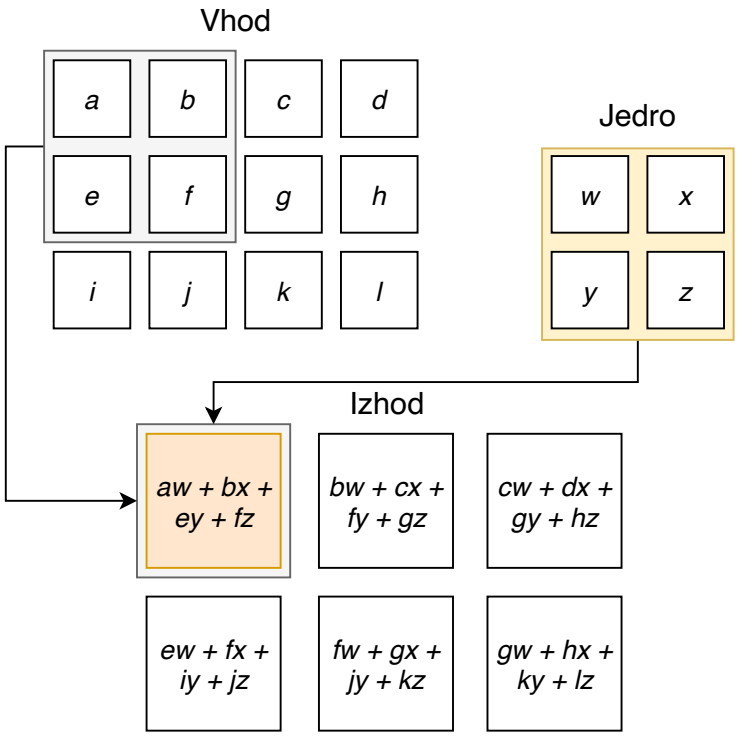
$$w_{i,j}^{(t+1)} = \begin{cases} u_{i,j}^{(t)} & \text{če } \text{rand}_j(0,1) \leq CR \vee j = j_{rand}, \\ x_{i,j}^{(t)} & \text{sicer,} \end{cases} \tag{2.46}$$

kjer $CR \in [0.0, 1.0]$ določa delež križanja vektorja pri čemer relacija $j = j_{rand}$ zagotavlja, da se poskusni vektor razlikuje od originalnega v vsaj enem elementu.

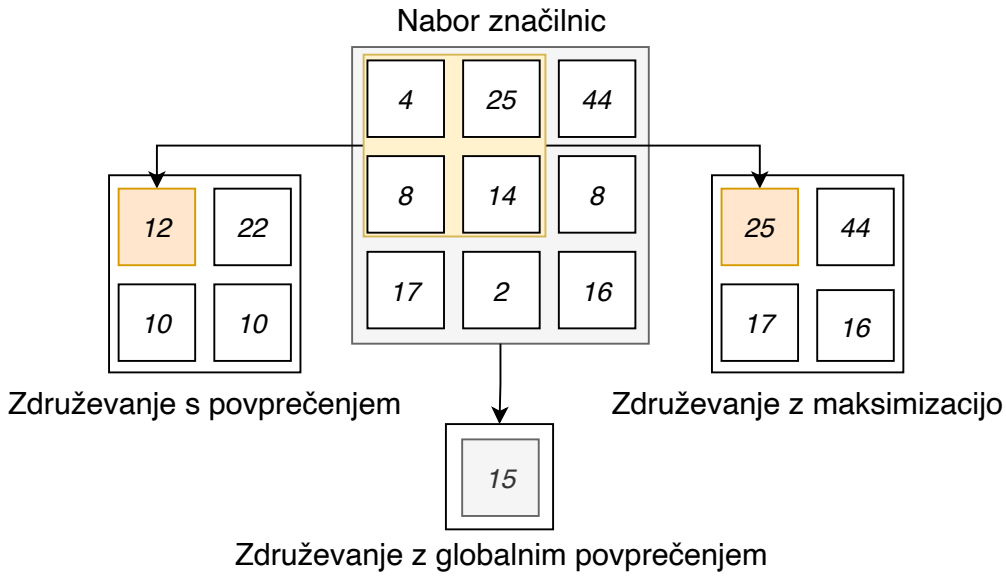
Kot končni operator pride na vrsto selekcija, ki je uporabljena z namenom odločitve, ali bo takšen ustvarjeni poskusni vektor tudi postal član generacije. Ta odločitev je sprejeta z uporabo pohlepnega merila (angl. greedy criterion), selekcijo pa lahko izrazimo kot:

$$x_i^{(t+1)} = \begin{cases} w_i^{(t)} & \text{če } f(w_i^{(t)}) \leq f(x_i^{(t)}), \\ x_i^{(t)} & \text{sicer,} \end{cases} \tag{2.47}$$

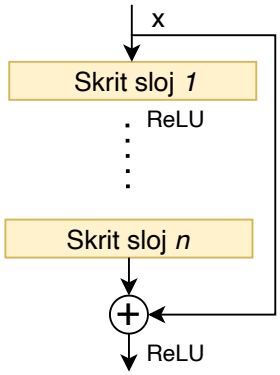
kjer $f(\cdot)$ označuje kriterijsko funkcijo, določeno za reševanje specifičnega problema, pri čemer poskusni vektor zamenja originalnega samo v primeru, da je njegova vrednost kriterijske funkcije manjša od vrednosti kriterijske funkcije originalnega vektorja.



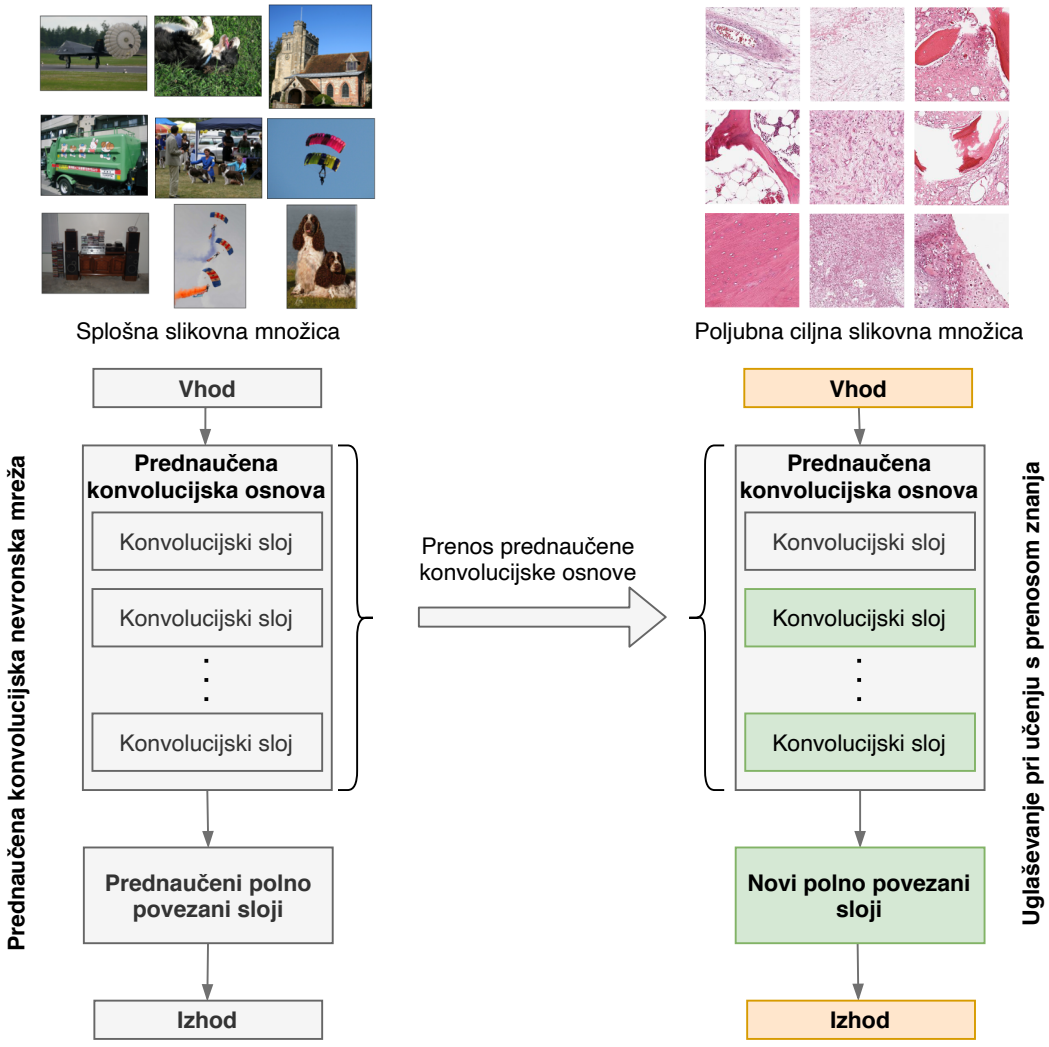
Slika 2.2 Prikaz operacije konvolucije nad dvodimenzionalnimi podatki s korakom drsenja jedra, nastavljenim na vrednost 1.



Slika 2.3 Prikaz delovanja podvzorčenja z uporabo združevanja.



Slika 2.4 Koncept obvodne povezave pri arhitekturi *ResNet50*.



Slika 2.5 Koncept uglaševanja globoke konvolucijske nevronske mreže pri učenju s prenosom znanja.

Poglavje 3

Analiza vpliva izbire slojev konvolucijske nevronske mreže za uglaševanje

"The important thing is not to stop questioning.
Curiosity has its own reason for existing."

- Albert Einstein

Poglavje predstavlja obstoječe raziskave, ki naslavljajo problematiko analize vpliva izbire slojev konvolucijske nevronske mreže za uglaševanje na uspešnost učenja. V nadaljevanju predstavimo obstoječe raziskave, zastavimo eksperiment za izvedbo empirične analize vpliva izbire slojev konvolucijske nevronske mreže na uspešnost učenja in predstavimo rezultate empirične analize. Na koncu dognanja in rezultate, pridobljene z uporabo empirične analize, primerjamo z izsledki obstoječih raziskav.

3.1 Obstoječe raziskave

Problematika izbire uglaševanih slojev ob uporabi uglaševanja pri učenju s prenosom znanja je v zadnjih letih v porastu. To gre pripisati predvsem splošni popularizaciji globokega učenja in raznovrstnim možnostim uporabe takšnih tehnik in metod na različnih področjih. Ključna prednost omenjenega pristopa pred ostalimi pristopi globokega učenja pa je zagotovo ta, da uglaševanje pri učenju s prenosom znanja za doseganje visokih napovednih točnosti ne potrebuje tako velike količine učnih podatkov kot bolj klasični pristopi. Dodatno je v splošnem uglaševanje pri učenju s prenosom znanja tudi manj časovno zahtevno, saj modela ne učimo od začetka, ampak ima ta že določeno predznanje. Poleg omenjenih prednosti pa nam uporaba

pristopa uglaševanja pri učenju s prenosom znanja prinaša tudi določene izzive. Eden izmed ključnih izzivov pri uporabi uglaševanja slojev konvolucijskih nevronskih mrež pri strojnem učenju s prenosom znanja je določitev oz. izbira, katere izmed slojev globoke nevronske mreže uglašujemo ter katere pustimo zamrznjene.

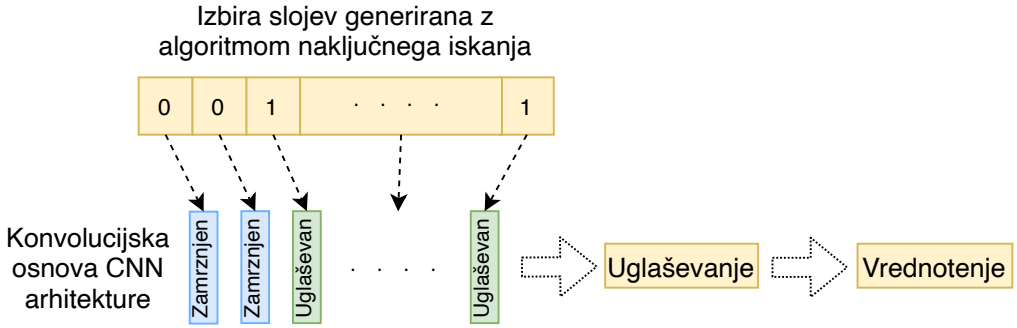
Ena zgodnejših raziskav [43], ki se ukvarja s prenosljivostjo značilnic pri globokih nevronskih mrežah, je bila objavljena leta 2014. Raziskava analizira vpliv prenosljivosti značilnic iz spodnjih, srednjih in zgornjih delov arhitektur nevronskih mrež, njeni izsledki pa služijo kot osnova oz. motivacija za strategijo izbire uglaševanih slojev. Na podlagi opravljene empirične analize je namreč mogoče zaznati, da spodnji sloji globoke konvolucijske nevronske mreže ohranjajo bolj abstraktne značilnice, ki so primerne za širši nabor domensko specifičnih problemov, medtem ko višji sloji večinoma ohranjajo značilnice, specifične za posamezen problem. Glede na te izsledke je torej smiselno uglaševanje slojev začeti pri višji slojih in jih po potrebi uglaševati proti vhodnemu sloju. Pa je temu res tako? Izsledki omenjene raziskave temeljijo na empirični analizi, v sklopu katere je bila uporabljena zgolj ena podatkovna zbirka (*ImageNet*) ter zgolj ena, za današnje čase precej enostavna arhitektura CNN. Na podlagi omenjene raziskave torej ni mogoče trditi, da predstavljeni izsledki držijo tudi za moderne arhitekture globokih CNN, kot so *MobileNet*, *ResNet* in druge.

V zadnjem času se je tako pojavilo nekaj študij [44, 96, 37], ki zavračajo to tezo in nakazujejo, da je uspešnost prenosa znanja odvisna od različnosti primarnega ciljnega problema, na katerem je bila nevronska mreža v osnovi naučena, od novega ciljnega problema. Nobena izmed omenjenih raziskav pa ne naslavlja dejanske analize vpliva izbire uglaševanih slojev konvolucijske nevronske mreže na uspešnost učenja.

3.2 Zasnova eksperimenta

Predstavljene raziskave sicer ponujajo vpogled v vpliv izbire uglaševanih slojev CNN na uspešnost učenja oz. prenosa znanja med različnimi ciljnimi problemi, vendar pa je ta vpogled precej omejen, saj predstavljene študije zajemajo le uporabo podmnožic podatkovne zbirke *ImageNet* in ne vključujejo uporabe različnih arhitektur konvolucijskih nevronskih mrež. Uporaba podmnožic podatkovne zbirke *ImageNet* prav tako ne ponuja večje neskladnosti med podmnožicami uporabljenimi za predučenje, in ciljnimi podmnožicami, uporabljenimi za učenje s prenosom znanja. Večja neskladnost med omenjenimi množicami ter uporaba različnih arhitektur konvolucijskih nevronskih mrež pa je pravzaprav nujna, če želimo pridobiti boljši pregled nad vplivom izbire uglaševanih slojev na uspešnost učenja, ki ga je potencialno mogoče generalizirati oz. posplošiti na različne ciljne domenske probleme ter različne arhitekture konvolucijskih nevronskih mrež.

Z namenom pridobitve boljšega pregleda in vpogleda na vpliv izbire uglaševanih slojev CNN na uspešnost učenja smo zastavili na algoritmu naključnega iskanja (angl. random search, RS) temelječ eksperiment, katerega konceptualna zasnova je prikazana na sliki 3.1. Z uporabo RS generiramo možne izvedljive rešitve, v našem primeru različne kombinacije slojev, izbranih za uglaševanje, ki jih nato naučimo ter njihovo uspešnost ovrednotimo z uporabo kategorične prečne entropije. Tako pridobljen nabor različnih kombinacij uglaševanih slojev globoke CNN s pripadajočimi vrednostmi uspešnosti v zadnjem delu analiziramo in skušamo identificirati morebitne vzročno-posledične vzorce med izbranimi sloji ter končno uspešnostjo tako naučenega napovednega modela.



Slika 3.1 Konceptualna zasnova eksperimenta, temelječega na algoritmu naključnega iskanja.

3.2.1 Algoritem naključnega iskanja

Algoritem naključnega iskanja in njegove variacije so eni od najpogosteje uporabljanih algoritmov za optimiziranje kompleksnih problemov. Kot nakazuje že ime samo, takšni algoritmi tipično, v svoji metodologiji vključujejo naključnost ali verjetnost, navadno v obliki psevdonaključnega generatorja števil. V literaturi je algoritem naključnega iskanja lahko imenovan tudi stohastični algoritem.

V nasprotju z deterministčnimi metodami, kot so algoritem razveji in omeji (angl. branch and bound algorithm) [118], intervalna analiza [119] in metoda tuneliranja [120], ki tipično zagotavljajo asimptotično konvergenco k optimumu, algoritem RS in njegove variante zagotavljajo konvergenco z verjetnostjo. Kompromis med različnimi pristopi pa se izkazuje v obliki računske zahtevnosti. Algoritem RS in njegove različice so izkazali svoj potencial, da lahko učinkovito rešujejo obsežne probleme na način, ki za deterministične algoritme ni mogoč. Medtem ko je znano, da je deterministična metoda za globalno optimizacijo NP-težka, obstajajo dokazi, da je mogoče stohastični algoritem v povprečju izvesti v polinomskem času [121]. Algoritmi RS so prav tako pogosto uporabljeni za reševanje stohastičnih optimizacijskih problemov oz. simulacijskih optimizacij, kjer ciljna funkcija in omejitve vključujejo naključnost.

Čeprav je dostopna množica naprednejših optimizacijskih algoritmov [122–124], smo v našem primeru izbrali algoritem RS zaradi njegove enostavnosti ter osnovne lastnosti, da iskalni prostor čim bolj enakomerno razišče, kar nam bo posledično omogočilo objektivnejši vpogled v rezultate analize vpliva izbire slojev konvolucijske nevronske mreže za uglaševanje.

V splošnem lahko globalni optimizacijski problem P definiramo kot:

$$min_{x \in S} f(x), \tag{3.1}$$

kjer x predstavlja vektor n spremenljivk, S pa označuje n-dimenzionalno izvedljivo množico vseh možnih rešitev optimizacijskega problema (iskalni prostor), za katero predpostavljamo, da ni prazna. Z f je označena funkcija realne spremenljivke definirana za vsak element nabora S. Cilj takšnega optimizacijskega problema je najti vrednost za vsak x, ki je element množice S, ki minimizira vrednost funkcije f. Funkcija f je lahko definirana tudi kot funkcija črne škatle (angl. black-box function), kar pomeni, da ni nujno eksplicitno matematično izražena, mora pa takšna funkcija vrniti vrednost za vsako vhodno vrednost x. Tako je lahko izvedljiv nabor vseh rešitev S predstavljen s sistemom nelinearnih enačb ali pa s pripadnostno funkcijo (angl. membership function), ki vrača vrednost, ali je podan $x \in S$ ali ne [121].

Splošni algoritem RS lahko predstavimo z zaporedjem ponavljajočih se elementov X_k za zaporedje $k = 0, 1, ...$, pri čemer lahko trenutni element X_k lahko predstavlja eno samo točko oz. vrednost ali pa skupek več vrednosti (navadno pri populacijsko temelječih algoritmih). V nadaljevanju so predstavljeni osnovni koraki splošnega algoritma RS [121]:

- **Korak 0.** Inicializacija parametrov algoritma Θ_0, začetnih točk $X_0 \subset S$ in indeksa $k = 0$.
- **Korak 1.** Generiranje kolekcije kandidatnih točk $V_{k+1} \subset S$ glede na specifični generator in pridruženo distribucijo vzorčenja.
- **Korak 2.** Posodobitev X_{k+1} glede na kandidatne točke V_{k+1}, prejšnje vrednosti ter parametre algoritma. Posodobijo se tudi parametri algoritma Θ_{k+1}.
- **Korak 3.** Če je zaustavitveni kriterij dosežen, se izvajanje zaključi, sicer se k poveča in izvajanje vrne na Korak 1.

3.2.2 Iskalni prostor

Če želimo algoritem RS uporabiti za namen pridobivanja različnih rešitev (kombinacij izbir uglaševanih slojev CNN), je treba najprej definirati iskalni prostor algoritma.

Iskalni prostor algoritma intuitivno vsebuje nabor možnih veljavnih rešitev za podan problem. V našem primeru iskalni prostor algoritma RS predstavljajo vse možne kombinacije izbranih uglaševanih slojev posamezne arhitekture CNN. Iskalni prostor lahko tako formalno izrazimo kot kombinacijo oz. zaporedje diskretnih iskalnih prostorov za vsak sloj konvolucijske osnove l uporabljene arhitekture CNN, kjer je dolžina takšnega zaporedja enaka številu slojev konvolucijske osnove izbrane arhitekture. Iz tega sledi, da je tak iskalni prostor $S = \{l_1, l_2, ..., l_n\}$ za $\forall l \in \{0, 1\}$, kjer l_i predstavlja vrednost, ali je uglaševanje za posamezni sloj omogočeno ($l_i = 1$) ali ne ($l_i = 0$).

3.2.3 Vrednotenje rešitev

Vsako izmed pridobljenih možnih veljavnih rešitev našega problema je treba ovrednotiti oz. definirati, na kak način bo posamezna rešitev ovrednotena. Glede na to, da naslavljamo problem izbire uglaševanih slojev globoke CNN za reševanje klasifikacijskih problemov, je smiselno, da v ta namen uporabimo metriko kategorična prečna entropija, ki se v splošnem uporablja za izračunavanje napake pri učenju CNN kot tudi za merjenje oz. vrednotenje uspešnosti učenja. Uporabimo lahko tudi katero izmed najpogosteje uporabljanih klasifikacijskih metrik, kot je točnost ali mera F1. Za uporabo slednjih se nismo odločili, ker vsaka izmed klasifikacijskih metrik v splošnem vpelje tudi določeno delež pristranskosti, medtem ko metrika kategorična prečna entropija predstavlja dejansko vrednost napake modela pri učenju in se uporablja tudi za optimizacijo samega učenja. Formalno je izpeljava metrike CCE predstavljena v enačbi 2.23, v nadaljevanju v enačbi 3.2 pa je formalno izražena kot kriterijska funkcija, ki jo uporabimo za vrednotenje pridobljenih veljavnih rešitev izbir uglaševanih slojev CNN.

$$L = CCE = -\frac{1}{M}\sum_{i=1}^{M}\sum_{j=1}^{C} y_{ij} log(p_{ij}) \tag{3.2}$$

3.2.4 Eksperiment

V prejšnjem poglavju smo podrobno predstavili podrobnosti zasnovanega eksperimenta, v nadaljevanju pa eksperiment predstavimo še v obliki psevdokoda v algoritmu 3.1.

Na vhodu sta podani učna in testna množica (*UM*, *TM*), na podatkovni zbirki *ImageNet* prednaučen model izbrane arhitekture CNN, maksimalno število ovrednotenih rešitev (posameznikov) oz. število preizkušenih kombinacij uglaševanih slojev p ter število epoh e, namenjenih za učenje. Po inicializaciji algoritma *RS*

Algoritem 3.1: Psevdokod eksperimenta temelječega na algoritmu naključnega iskanja

Vhod : Učna množica *UM*
Vhod : Testna množica *TM*
Vhod : Prednaučen model *M*
Vhod : Število ovrednotenih rešitev (posameznikov) *p*
Vhod : Število epoh *e*
Rezultat: Kandidatne rešitve *R* ter testne vrednosti funkcije izgube *L* za posamezno izbrano rešitev

```
1  begin
2      RS = inicializiraj_RS();
3      M_zacasni = M;
4      for i = 1,...p do
5          resitev = RS.pridobi_resitev();
           // Glede na rešitev omogoči uglaševanje posameznih slojev
6          M.nastavi_sloje(resitev);
7          for j = 1,...e do
8              M_zacasni.izvedi_ucenje(UM);
9          end
10         R.dodaj(resitev);
11         l = izracunaj_l(TM);
12         L.dodaj(l);
13     end
14     shrani_R_in_L();
15 end
```

pridobimo kandidatno *rešitev* kombinacije izbranih uglaševanih slojev CNN, ki jo preslikamo na model $M_{zacasni}$. To storimo tako, da za posamezno zaporedno vrednost znotraj *resitve*, za zaporedni sloj modela $M_{zacasni}$ omogočimo uglaševanje ali pa ne. Sledi izvedba učenja tako pripravljenega modela $M_{zacasni}$ nad učno množico *UM* za podanih *e* epoh. Po koncu učenja *resitev* dodamo v množico vseh pridobljenih rešitev *R* ter izračunamo vrednost funkcije izgube *l* modela $M_{zacasni}$ nad testno množico *TM*. Pridobljeno vrednost *l* dodamo v množico vseh testnih vrednosti funkcije izgube *L*. Ta postopek se ponavlja, dokler ni doseženo ovrednotenih rešitev *p*. Na koncu množico kandidatnih rešitev *R* in njihovih pripadajočih vrednosti funkcije izgube *L* shranimo za kasnejšo analizo. Rezultat predstavljenega algoritma sta tako množici s funkcijo izgube ovrednotenih kombinacij uglaševanih slojev izbrane arhitekture CNN *R* ter vrednosti funkcije izgube *L* modelov, naučenih na podlagi posamezne kandidatne rešitve.

3.3 Izvedba eksperimentov

Z namenom temeljitega ovrednotenja vpliva izbire slojev na uspešnost učenja oz. končno klasifikacijsko napovedno uspešnost smo izvedli devet eksperimentov, pri čemer smo uporabili tri različne podatkovne zbirke ter tri različne arhitekture konvolucijskih nevronskih mrež, ki so podrobneje predstavljene v nadaljevanju.

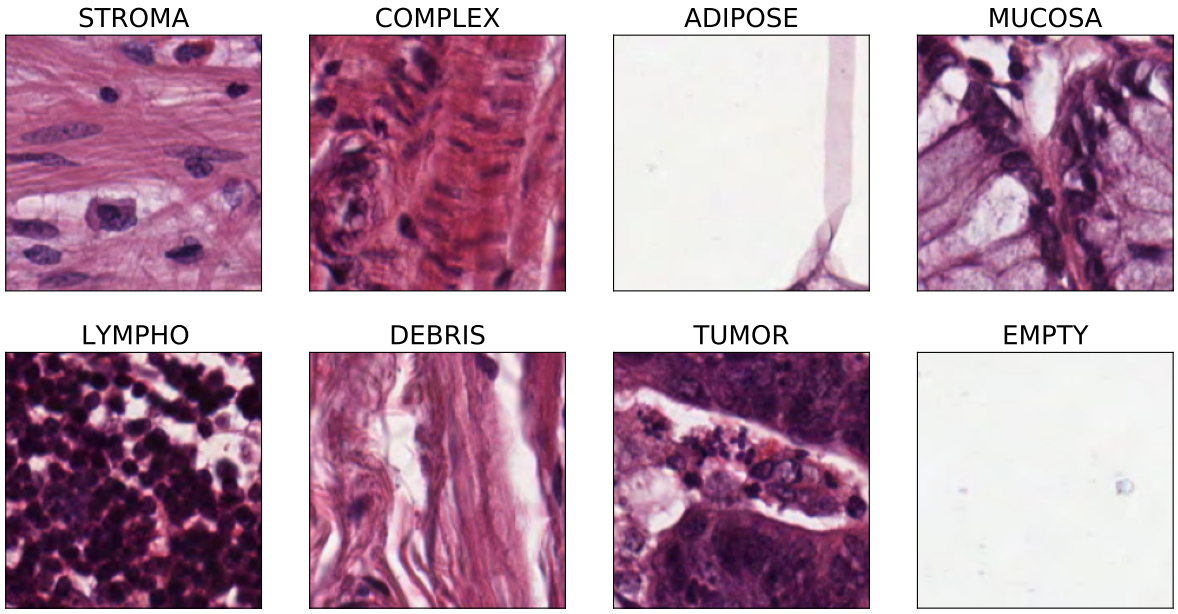
3.3.1 Podatkovne zbirke

Za podatkovne zbirke, nad katerimi so ovrednotene različne kombinacije izbir uglaševanih slojev globokih CNN, smo izbrali tri, ki se razlikujejo tako glede na vizualno predstavitev kot tudi glede na število ciljnih razredov in število primerkov posameznega razreda.

Colorectal histology

Prva podatkovna zbirka, predstavljena v delu [125], vsebuje anonimizirane slike diapozitivov s hemotoksinom in eosinom obarvanega kolorektalnega rakavega tkiva, zbrane s strani patološkega arhiva univerzitetnega kliničnega centra Mannheim Univerze v Heidelbergu v Nemčiji. Diapozitivi so bili najprej digitalizirani po postopku, opisanem v [126], nato pa ročno označeni s strani ekspertov. Tako je bila pridobljena zbirka 625 neprekrivajočih se slik tkiva velikosti 150×150 slikovnih točk. Zbirka vključuje lastnosti tekstur na različnih velikostih, ki variirajo od posameznih celic, velikosti približno 10 μm, do večjih struktur, kot so mukozne žleze (angl. mucosal glands), velikosti nad 50 μm.

Vsaka slika je bila s strani ekspertov označena oz. kategorizirana v enega izmed naslednjih osmih razredov. Oznako *TUMOR* imajo slike, ki predstavljajo epitelni tumor (angl. epithelial tumor). V razred *STROMA* so razvrščene slike, ki predstavljajo homogene kompozicije, ki vključujejo stromalni tumor (angl. stromal tumor), gladke mišice (angl. smooth muscle) ipd. Z oznako *COMPLEX* so označene slike, ki predstavljajo posamezne tumorne celice in/ali redke imune celice. Slike z oznako *LIMPHO* predstavljajo submukozne limfoidne folikle (angl. sub-mucosal lymphoid follicles). V kategorijo *DEBRIS* so razvrščene slike, ki predstavljajo nekrozo, krvavitve in sluz. Z oznako *EMPTY* so označeni primerki slik, ki predstavljajo ozadje oz. diapozitive brez tkiva. Oznaka *ADIPOSE* označuje slike, ki predstavljajo maščobno tkivo, oznaka *MUCOSA* pa slike, ki predstavljajo normalne mukozne žleze (angl. mucosal glands). Za vsakega izmed navedenih razredov podatkovna zbirka vsebuje 625 primerkov slik. Skupno torej podatkovna zbirka obsega 5.000 slik. Primerki posameznega razreda so predstavljeni na sliki 3.2.



Slika 3.2 Primerki slik različnih kategorij podatkovne zbirke histoloških slik *Colorectal histology*.

DeepWeeds

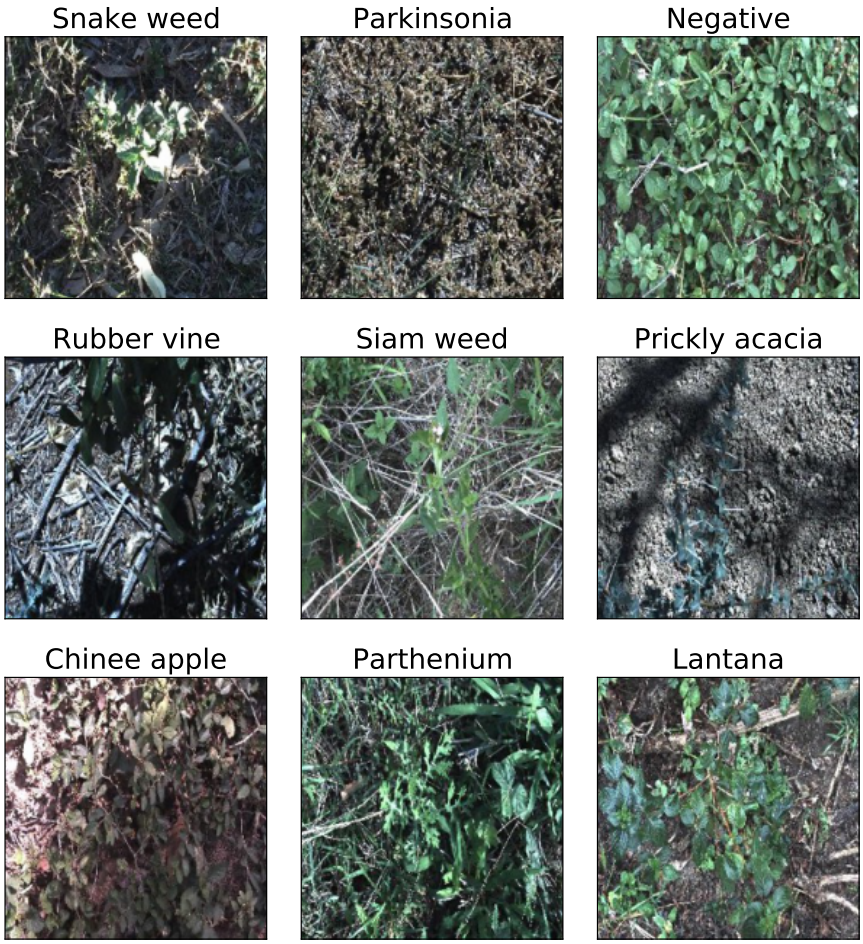
Podatkovna zbirka *DeepWeeds*, ki so jo avtorji predstavili v delu [127], zajema slike osmih različnih vrst rastlin, domorodnih za osem lokacij severne Avstralije. Slike posameznega plevela so bile zajete z višine enega metra, brez dodatne osvetlitve, v ločljivosti 1920×1080 slikovnih točk, ki so jih v nadaljnjem postopku predobdelave zmanjšali na velikost 224×224 slikovnih točk.

Vse slike so bile s strani ekspertov označene oz. kategorizirane v devet razredov: *Chinee apple* (lat. Ziziphus mauritiana), *Lantana* (lat. Lantana camara), *Parkinsonia* (lat. Parkinsonia aculeata), *Parthenium* (lat. Parthenium hysterophorus), *Prickly acacia* (lat. Vachellia nilotica), *Rubber vine* (lat. Cryptostegia grandiflora), *Siam weed* (lat. Chromolaena odorata), *Snake weed* (lat. Stachytarpheta spp.) in pa razred *Negative*, ki predstavlja slike, prikazujoče druge vrste plevela. Skupno podatkovna zbirka obsega 17.509 primerkov, porazdelitev primerkov med posameznimi razredi pa je prikazana v tabeli 3.1.

Primerki posameznih razredov podatkovne zbirke *DeepWeeds* so predstavljeni na sliki 3.3.

Tabela 3.1 Število primerkov slik posamezne kategorije podatkovne zbirke *DeepWeeds*.

Kategorija	Število primerkov
Snake weed	1016
Parkinsonia	1031
Negative	9106
Rubber vine	1009
Siam weed	1074
Prickly acacia	1062
Chinee apple	1125
Parthenium	1022
Lantana	1064



Slika 3.3 Primerki slik različnih kategorij podatkovne zbirke slik rastlin *DeepWeeds*.

UC Merced Land Use

Podatkovna zbirka *UC Merced Land Use*, predstavljena v delu [128], zajema ortografske posnetke različnih uporab površine nekaj regij Združenih držav Amerike, pridobljenih

iz nacionalnega geoprostorskega programa Združenih držav Amerike, United States Geological Survey(USGS) National Map. Skupno podatkovna zbirka vsebuje 2.100 primerkov ortografskih posnetkov, ki so razvrščeni v 21 kategorij (slika 3.4.

Slika 3.4 Primerki slik različnih kategorij podatkovne zbirke zračnih slik *UC Merced Land Use*.

Slike v kategoriji *agricultural* predstavljajo ortografske posnetke različnih polja. Slike, označene z *airplane*, predstavljajo zračne posnetke letal, medtem ko slike označene z oznako *runway* predstavljajo posnetke vzletno-pristajalne steze. V razred z oznako *harbor* so razvrščene slike pristanišč, v razred z oznako *parking lot* pa slike

parkirišč. Z *baseball diamond* so označeni posnetki igrišč za bejzbol, z oznako *tennis courts* teniška igrišča in z oznako *golf course* igrišča za golf. Z *beach* so označeni posnetki plaž, z oznako *mobile homepark* so označene slike mobilnih naselij in z oznako *buildings* posnetki večjih stavb. Oznaka *chaparral* označuje primerke slik, ki prikazujejo rastlinstvo širokolistnih zimzelenih grmovnic in majhnih dreves, oznaka *forest* pa primerke posnetkov, ki prikazujejo gozd. Z oznako *dense residential* so opredeljene slike gostega naselja, z oznako *medium density residential* srednje gosta naselja ter z oznako *sparse residential* redkeje poseljena naselja. Posnetki z oznako *freeway*, predstavljajo slike avtocest, posnetki, označeni z *intersection* predstavljajo cestna križišča, kot *overpass* pa so označeni posnetki, ki prikazujejo cestne nadvoze. V razred *river* so razvrščeni posnetki rek, v razred *storage tanks* pa posnetki večjih zalogovnikov oz. rezervoarjev. Za vsakega izmed 21 razredov je bilo ročno izbranih 100 slik velikosti 256×256 slikovnih točk.

Primerki posameznih razredov podatkovne zbirke *UC Merced Land Use* so predstavljeni na sliki 3.4.

3.3.2 Predobdelava podatkovnih zbirk

Pri nobeni izmed uporabljenih podatkovnih zbirk nismo uporabili nikakršne posebne tehnike oz. metode predobdelave podatkov. Vsem slikam iz uporabljenih podatkovnih zbirk smo zgolj spremenili velikost na enotnih 224×224 slikovnih točk, kar je enaka dimenzija, kot je privzeto definirana velikost slik na vhodnem sloju vseh treh uporabljenih arhitektur CNN. Dodatno smo po uveljavljenem postopku izvedli tudi normalizacijo vhodnih podatkov (slik), z uporabo uveljavljene metode normalizacije min-max, ki je formalno predstavljena v enačbi 3.3.

$$x' = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{3.3}$$

3.3.3 Arhitekture konvolucijskih nevronskih mrež

V eksperimentalnem delu smo izbrali tri arhitekture, ki se razlikujejo tako po številu konvolucijskih slojev kot enot v posameznem sloju ter mehanizmih, uporabljenih za ekstrakcijo značilnic. Izbrane arhitekture, *VGG16*, *ResNet50* in *MobileNet*, so podrobno predstavljene v poglavju 2.3.4.

Za potrebe izvedbe eksperimentalne študije smo uporabili prednaučene konvolucijske osnove (brez klasifikacijskih slojev) omenjenih arhitektur CNN, na katere smo povezali zaporedje klasifikacijskih slojev, predstavljenih v tabeli 3.2.

Tabela 3.2 Zaporedje klasifikacijskih slojev.

Sloj	Lastnosti
Združevalni sloj (globalno povp.)	-
Sloj normalizacije paketov	-
Polno povezan sloj	1024 nevronov, aktivacijska funk. ReLU
Polno povezan sloj	8/9/21 nevronov, aktivacijska funk. Softmax

3.3.4 Nastavitve eksperimentov

Z namenom pridobitve čim bolj objektivnega vpogleda na vpliv izbire uglaševanih slojev globoke CNN na klasifikacijsko točnost ter v izogib morebitnemu vplivu dejavnika naključnosti smo uporabili uveljavljeno metodologijo 10-kratnega prečnega preverjanja. Za vsak eksperiment smo podatkovno zbirko razdelili na 10 delov, pri čemer smo 9 delov zbirke uporabili za učenje, preostali del pa smo uporabili za testiranje. Takšen postopek smo ponovili skupno 10-krat, pri čemer smo za testni del vedno izbrali drug del.

V vsaki ponovitvi prečnega preverjanja, je algoritem RS ustvaril 50 različnih kandidatnih rešitev oz. izbir uglaševanih slojev uporabljene arhitekture CNN, vsaka izmed rešitev pa je bila učena z uporabljenimi vrednostmi učnih parametrov, predstavljenimi v tabeli 3.3. Vrednosti učnih parametrov ključno vplivajo na sposobnost oz. uspešnost učenja CNN, njihova določitev pa navadno predstavlja zahtevno nalogo, ki je običajno opravljena ročno na podlagi predhodnih izkušenj. Pristope, ki z uporabo različnih optimizacijskih algoritmov prilagajo vrednosti takšnih parametrov, pogosto imenujemo optimizacija hiperparametrov [27, 129, 29]. Mi se za tak pristop nismo odločili, saj bi s tem povečali iskalni prostor optimizacijskega algoritma in posledično dodatno povečali časovno zahtevnost izvedbe eksperimentov. Zato smo vrednosti parametrov nastavili na podlagi naših predhodnih raziskav, kjer so se takšne vrednosti izkazale kot primerne [38, 39].

Za učenje vsake kandidatne rešitve smo namenili 30 epoh, za optimizacijsko funkcijo je bil izbran algoritem SGD, katerega učni korak smo nastavili na $1 \cdot 10^{-4}$, moment pa na $0,9$. Velikost minipaketov smo pri arhitekturah *MobileNet* in *VGG16* nastavili na 16, medtem ko smo morali velikost minipaketov za arhitekturo *ResNet50* prilagoditi za vsako podatkovno zbirko posebej, saj smo se v nasprotnem primeru soočali s problematiko ekstremnega prileganja učni množici, kar je v večini primerov privedlo do pojava, da omenjena CNN nad posameznimi podatkovnimi zbirkami ni konvergirala. Za podatkovno zbirko *Colorectal histology* smo velikost nastavili na 128,

za podatkovno zbirko *DeepWeeds* na 256 ter za podatkovno zbirko *UC Merced Land Use* na 64.

3.4 Empirična analiza

V nadaljevanju predstavimo in analiziramo rezultate, ki smo jih pridobili z izvedbo eksperimentov nad tremi podatkovnimi zbirkami s tremi različnimi arhitekturami CNN. V prvem delu poglavja predstavimo pridobljene rezultate in postopek obdelave le-teh, v nadaljevanju pa analiziramo tako obdelane rezultate analiziramo iz več vidikov.

3.4.1 Obdelava rezultatov

Z izvedbo eksperimentov smo pridobili množico rezultatov v obliki parov izbire uglaševanih slojev in vrednosti funkcije izgube, ki je v našem primeru obenem tudi rezultat testne napake. Ker smo izvedli eksperimente s tremi arhitekturami CNN nad tremi različnimi podatkovnimi zbirkami in ker smo vsak eksperiment izvedli z uporabo 10-kratnega prečnega preverjanja, pri čemer smo za vsako ponovitev prečnega preverjanja uglaševali in ovrednotili 50 različnih izbir uglaševanih slojev, smo tako pridobili 4500 takšnih parov (500 parov za vsak eksperiment).

V nadaljevanju smo za vsak posamezni eksperiment pare razvrstili glede na doseženo vrednost funkcije izgube od najboljšega (najmanjša vrednost) do najslabšega. Tako razvrščene pare smo razdelili v tri skupine:

- *najboljše* rešitve, ki zajema 160 najboljših rešitev,
- *srednje* rešitve, ki zajema 180 srednje dobrih rešitev, in
- *najslabše* rešitve, ki zajema 160 najslabših rešitev.

Tabela 3.3 Nastavitve učnih parametrov za izvedbo eksperimentov analize vpliva izbire uglaševanih slojev konvolucijske nevronske mreže na uspešnost učenja.

Parameter	*VGG16*	*ResNet50*	*MobileNet*
Epoha	30	30	30
Velikost minipaketov	16	128/256/64	16
Optimizator	SGD	SGD	SGD
Učni korak	$1 \cdot 10^{-4}$	$1 \cdot 10^{-4}$	$1 \cdot 10^{-4}$
Moment	0,9	0,9	0,9

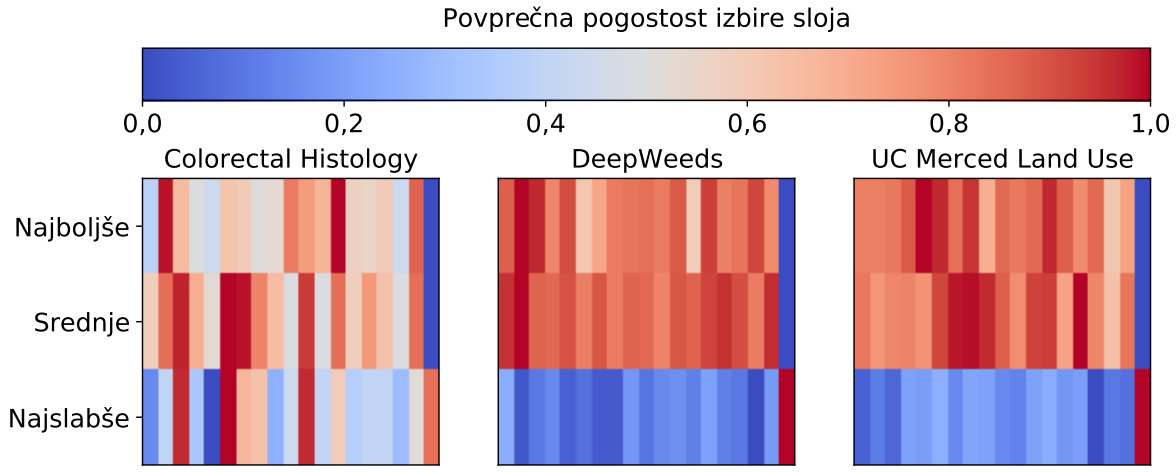
Za vsako skupino rešitev smo nato za vsak sloj posamezne arhitekture CNN izračunali povprečno pogostost izbire sloja, pridobljene vrednosti pa normalizirali na interval $[0, 1]$. V nadaljevanju smo enako izračunali tudi za bloke v posamezni arhitekturi CNN, kjer posamezeni blok predstavlja skupek določenih slojev konvolucijske osnove. Na tak način smo dodatno pridobili še povprečno pogostost izbire slojev znotraj posameznega bloka arhitekture CNN.

3.4.2 Analiza rezultatov

Pridobljene in obdelane rezultate v nadaljevanju grafično predstavimo in analiziramo. Rezultate analiziramo iz dveh vidikov, in sicer je v prvem delu izvedena analiza pogostosti izbire posameznega sloja glede na izbrano arhitekturo CNN ter nato analiza izbire posameznega bloka slojev glede na izbrano arhitekturo CNN. V nadaljevanju, na prikazanih grafičnih predstavitvah rezultatov, rdeče obarvani sloji prikazujejo visoko povprečno pogostost izbire posameznega sloja, medtem ko modro obarvani sloji prikazujejo nizko povprečno pogostost izbire posameznega sloja. Za posamezno podatkovno zbirko sloji skrajno levo predstavljajo začetne (najnižje) sloje konvolucijske osnove izbrane arhitekture CNN, medtem ko skrajno desni sloji predstavljajo končne (najvišje) sloje. Zaporedje slojev je pri vseh arhitekturah enako zaporedju implementacije posamezne arhitekture CNN.

Pogostost izbire posameznega sloja

Na sliki 3.5 je prikazana grafična predstavitev povprečne pogostosti izbire posameznih slojev arhitekture CNN *VGG16* nad tremi uporabljenimi podatkovnimi zbirkami. Iz slike je razvidno, da so se rešitve z več izbranimi sloji CNN izkazale za uspešnejše kot tiste z manj izbranimi sloji. Ta pojav je še najmanj izrazit pri podatkovni zbirki *Colorectal Histology*, kjer so tudi najslabše rešitve v splošnem imele za uglaševanje omogočenih več slojev kot pri ostalih podatkovnih zbirkah. Če se osredotočimo na izbiro vrhnjih (zadnjih) slojev, ugotovimo, da so pri vseh podatkovnih zbirkah za najboljše kot tudi za srednje dobre rešitve obarvani temno modro. To pomeni, da je povprečna pogostost izbire uglaševanja teh slojev nizka, kar je v nasprotju s priporočeno in pogosto uporabljano strategijo izbire slojev, pri kateri uglašujemo sloje od najvišjega proti najnižjemu. Pri srednjih in začetnih slojih drugih izstopajočih vzorcev ni mogoče razbrati. Izpostavimo lahko še pojav pri podatkovni zbirki *Colorectal Histology*, kjer imajo za razliko od preostalih dveh podatkovnih zbirk najboljše ovrednotene rešitve nizko povprečno pogostost izbire v prvem konvolucijskem sloju. Sicer visoka povprečna pogostost izbire začetnih slojev nakazuje, da se izvorna podatkovna zbirka (*ImageNet*) bolj razlikuje od podatkovnih zbirk *DeepWeeds* in

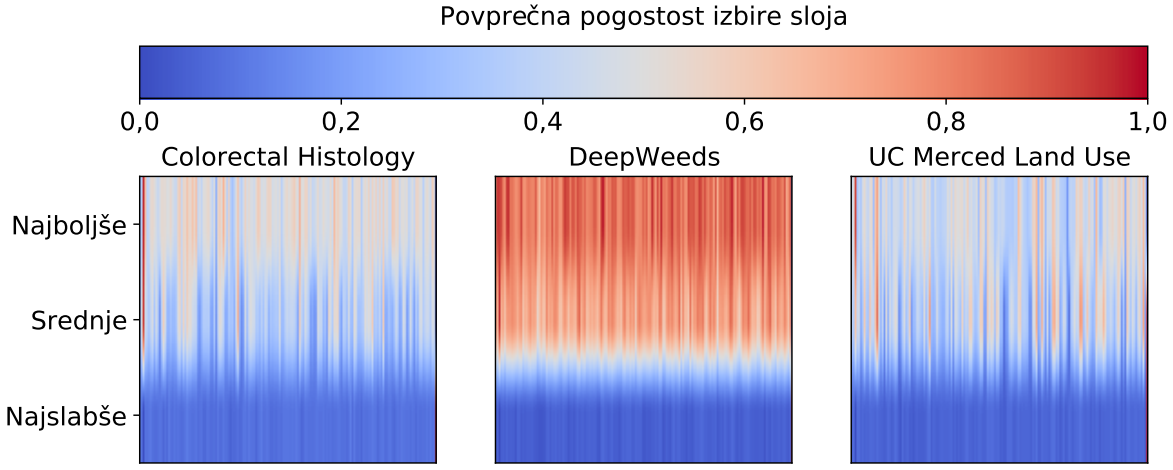


Slika 3.5 Pogostost izbire posameznega sloja arhitekture *VGG16* glede na podatkovno zbirko.

UC Merced Land Use kot od podatkovne zbirke *Colorectal Histology*. Začetni sloji konvolucijske nevronske mreže naj bi namreč izluščevali bolj splošne, abstraktne značilnice, vendar je bilo glede na prikazano v primeru podatkovnih zbirk *DeepWeeds* in *UC Merced Land Use* bilo treba za doseganje najboljših rezultatov uglaševati tudi začetne sloje.

Slika 3.6 prikazuje povprečno pogostost izbire posameznih slojev arhitekture *ResNet50* za posamezno podatkovno zbirko. Na prvi pogled je iz slike razvidno, da *ResNet50* arhitektura vsebuje precej več slojev kot arhitektura *VGG16*. V nadaljevanju je mogoče opaziti, da se rezultati nad podatkovno zbirko *DeepWeeds* precej razlikujejo od preostalih dveh zbirk. Pri podatkovni zbirki *DeepWeeds* je namreč pri najboljših in srednje dobrih rešitvah mogoče zaznati visoko povprečno pogostost izbire skozi večino slojev, medtem ko pri ostalih dveh zbirkah ni tako. Med začetnimi sloji pri podatkovnih zbirkah *Colorectal Histology* in *UC Merced Land Use* je mogoče v začetnih slojih zaznati višjo pogosto izbire, medtem ko za preostale sloje ni vidnih posebnih vzorcev. Z opazovanjem slabih rešitev je mogoče zaznati, da je kljub razlikam med srednjimi in najboljšimi rešitvami povprečna pogostost izbire nad vsemi podatkovnimi zbirkami praktično enaka in v splošnem nizka. Za razliko od preostalih dveh podatkovnih zbirk je pri podatkovni zbirki *DeepWeeds* mogoče zaznati ostrejšo ločnico med srednjimi in najslabšimi rešitvami.

Pogostost izbire posameznega sloja arhitekture *MobileNet* glede na posamezno podatkovno zbirko je predstavljena na sliki 3.7. Iz slike je razvidno, da je v tem primeru mogoče zaznati podoben pojav kot pri povprečni gostoti izbir slojev arhitekture *ResNet50* s to razliko, da je v tem primeru povprečna pogostost izbire slojev pri najboljši rešitvi višja pri podatkovnih zbirkah *Colorectal Histology* in *UC Merced Land*

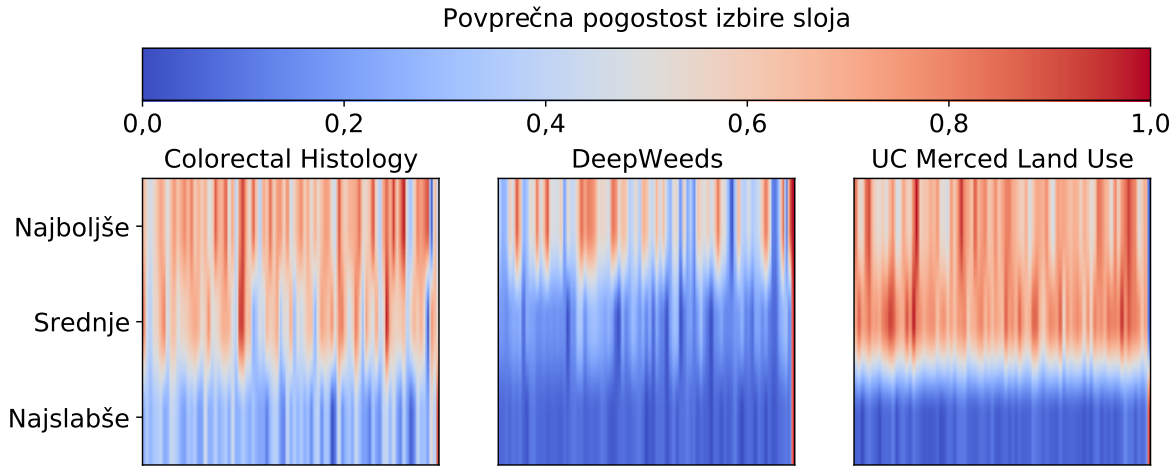


Slika 3.6 Pogostost izbire posameznega sloja arhitekture *ResNet50* glede na podatkovno zbirko.

Use. Povprečna pogostost izbire slojev pri arhitekturi MobileNet je nad omenjenima podatkovnima zbirkama precej podobna, saj so pri najboljših rešitvah sloji čez celotno konvolucijsko osnovo v povprečju pogosto izbrani z izjemo zadnjih slojev, kjer je ponovno razvidno, da se kombinacije izbir slojev, ki v povprečju pogosto vključujejo zadnje sloje, odrežejo slabše. Pri podatkovni zbirki *DeepWeeds* je mogoče opaziti zanimiv pojav, da večina rešitev v splošnem vsebuje malo uglaševanih slojev. Dodatno je zanimiv pojav tudi, da v tem primeru pogosta izbira zadnjih slojev za uglaševanje pozitivno vpliva na uspešnost modela. Pojav v splošnem majhnega povprečja pogostosti izbire slojev v kombinaciji s pogosto izbiro uglaševanja zadnjih slojev je morda mogoče pripisati temu, da arhitektura *MobileNet* prepoznava izvorno podatkovno zbirko *ImageNet* in zbirko *DeepWeeds* kot zelo podobni.

Pogostost izbire posameznega bloka slojev

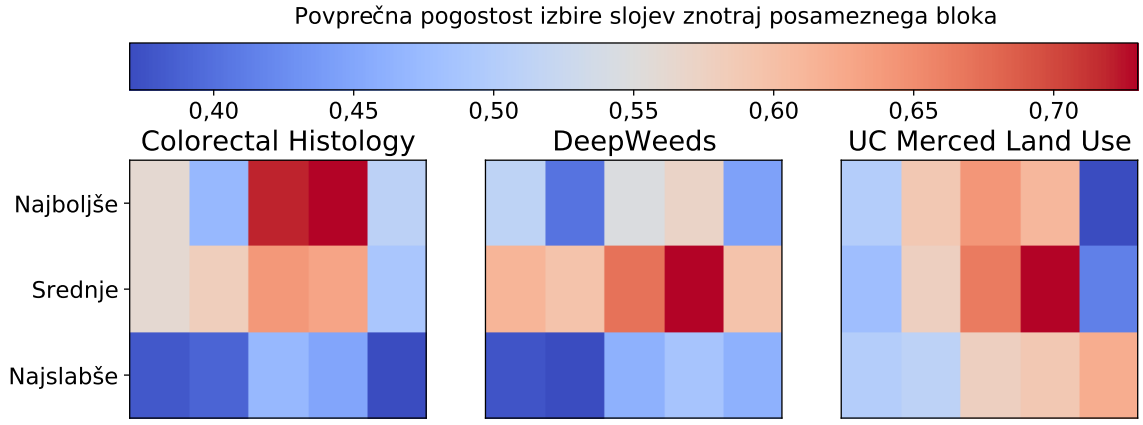
Glede na specifične rezultate, pridobljene za arhitekturi CNN *MobileNet* in *ResNet50*, ki ne izkazujejo posebnega vzorca pri pogostosti izbire, lahko takšno obnašanje pripišemo posebnostim arhitektur *MobileNet* in *ResNet50*. Omenjeni arhitekturi namreč za razliko od arhitekture *VGG16* nista v celoti sekvenčni, ampak vsebujeta tudi modernejše koncepte, pri katerih morda predstavitev povprečne pogostosti izbire ni nujno najboljša izbira. Zato smo izračunali še povprečno pogostost izbire po blokih konvolucijske osnove posamezne arhitekture CNN. Bloki so za vsako arhitekturo definirani, kot je prikazano v poglavju 2.4.3, pri čemer je treba omeniti, da posamezni blok konvolucijske arhitekture *MobileNet* predstavlja par slojev globinske konvolucije in točkovne konvolucije.



Slika 3.7 Pogostost izbire posameznega sloja arhitekture *MobileNet* glede na podatkovno zbirko.

Na sliki 3.8 je predstavljena povprečna pogostost izbire slojev znotraj posameznega bloka konvolucijske osnove arhitekture CNN *VGG16*. Iz slike je razvidno, da je povprečna pogostost izbire zadnjega bloka v splošnem pri najboljših rešitvah še vedno nizka. Pri srednje dobrih rešitvah je v primeru podatkovnih zbirk *Colorectal Histology* in *UC Merced Land Use* enako kot pri najboljših rešitvah, medtem ko je pri podatkovni zbirki *DeepWeeds* povprečna pogostost izbire povečana. Še posebej je zanimivo, da imajo pri tej podatkovni zbirki zgolj srednje dobre rešitve povišano povprečno pogostost izbire, medtem ko imajo najboljše in najslabše rešitve nižjo povprečno pogostost izbire. Pri podatkovni zbirki *UC Merced Land Use* lahko opazimo, da imajo zgolj srednji bloki najboljših in srednjih rešitev višjo povprečno pogostost izbire, medtem ko imajo najslabše rešitve najvišjo povprečno pogostost izbire v drugi polovici blokov. Če se osredotočimo na začetne bloke, lahko opazimo, da v splošnem ni zaznati bolj izstopajočih vzorcev pri pogostosti izbire. Izpostavimo lahko, da imajo prvi bloki pri podatkovnih zbirkah *Colorectal Histology* in *DeepWeeds* pri najboljših in srednjih rešitvah višjo povprečno pogostost izbire kot najslabše rešitve. Pri podatkovni zbirki *UC Merced Land Use* pa imajo v splošnem prvi bloki v vseh skupinah rešitev relativno nizko povprečno pogostost izbire.

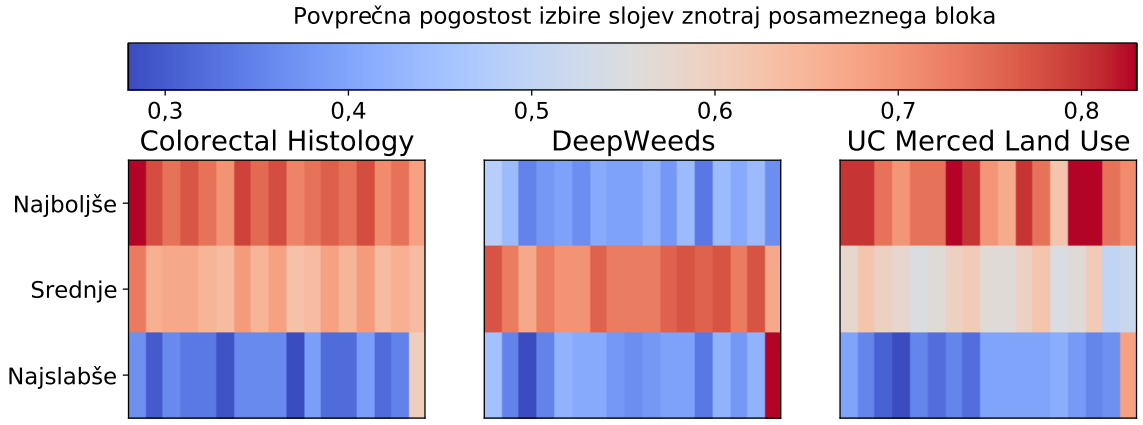
Povprečna pogostost izbire slojev znotraj posameznega bloka arhitekture *ResNet50* je predstavljena na sliki 3.9. Iz slike je razvidno, da nam blokovni prikaz povprečne pogostosti izbire slojev znotraj posameznega bloka omogoča boljši vpogled na izbiro slojev, saj so jasneje razvidne ločnice med posameznimi bloki konvolucijske osnove kot pa med posameznimi sloji. Opazimo lahko, da ni razvidnih splošnih vzorcev povprečne pogostosti izbire slojev znotraj posameznih blokov med različnimi podatkovnimi zbirkami. Če se osredotočimo na zadnji blok, lahko opazimo, da se



Slika 3.8 Pogostost izbire slojev arhitekture *VGG16* znotraj posameznega bloka glede na podatkovno zbirko.

pri vseh podatkovnih zbirkah v splošnem visoka povprečna pogostost izbire zgolj v zadnjem bloku izkaže kot slaba. Pri opazovanju najboljših rešitev lahko zaznamo zanimiv pojav, kjer so pri podatkovnih zbirkah *Colorectal Histology* in *UC Merced Land Use* večinoma vsi sloji znotraj vseh blokov pogosto izbrani, medtem ko je pri podatkovni zbirki *DeepWeeds* ravno obratno. Pri slednji imajo najboljše rešitve v splošnem najmanjšo povprečno pogostost izbire slojev skozi vse bloke. Dodatno je zanimiv tudi pojav, da se izbire med najboljšimi in najslabšimi rešitvami pri podatkovni zbirki *DeepWeeds* razlikujejo zgolj v zadnjem bloku, kjer je pri najslabših rešitvah povprečna pogostost izbire precej visoka, medtem ko je pri najboljših rešitvah nizka. Če primerjamo rezultate arhitekture *ResNet50* z rezultati arhitekture *VGG16* nad podatkovno zbirko *DeepWeeds*, lahko opazimo, da je obnašanje pri obeh arhitekturah v tem primeru podobno, saj v povprečni pogostosti izbire izstopajo zgolj srednje dobre rešitve, medtem ko so si najboljše in najslabše precej podobne. Če primerjamo rezultate arhitekture *ResNet50* nad podatkovnimi zbirkami *Colorectal Histology* in *UC Merced Land Use*, opazimo, da so si med seboj podobni. V splošnem so najboljše rešitve tiste z višjo povprečno pogostostjo izbire blokov čez celotno arhitekturo, medtem ko je pri srednje dobrih rešitvah mogoče zaznati zmanjšanje pogostosti izbire glede na predhodne. Najslabše rešitve pa imajo v splošnem najnižje vrednosti povprečne pogostosti izbire, pri čemer izstopa zgolj zadnji blok z višjo vrednostjo, kar ponovno nakazuje, da ni nujno vedno najbolj smiselno uglaševati višjih slojev arhitekture CNN.

Slika 3.10 prikazuje povprečno pogostost izbire slojev znotraj posameznega bloka konvolucijske osnove arhitekture *MobileNet*. V splošnem izrazitih vzorcev povprečne pogostosti izbire slojev znotraj bloka ni mogoče zaznati. Morda še najbolj izstopajoča je podobnost izbire blokov med podatkovnima zbirkama *Colorectal Histology* in *UC*

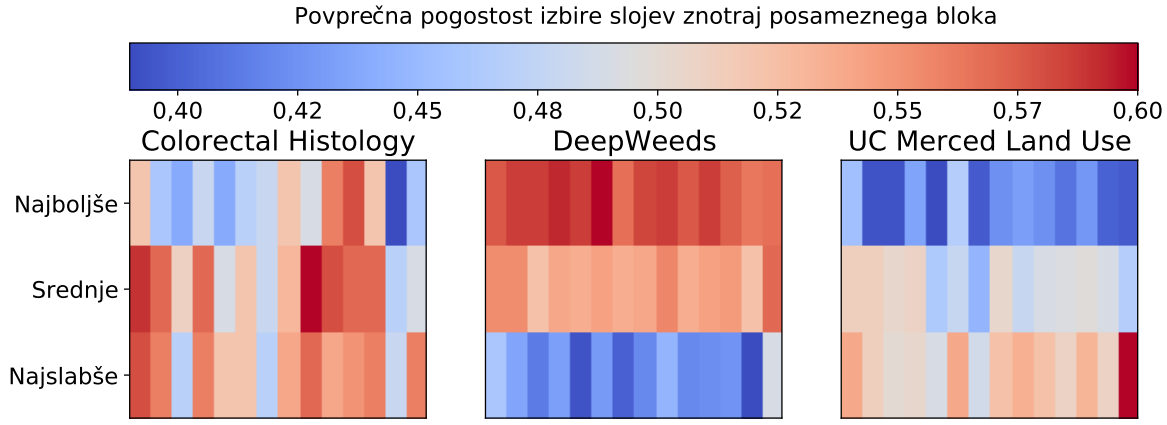


Slika 3.9 Pogostost izbire slojev arhitekture *ResNet50* znotraj posameznega bloka glede na podatkovno zbirko.

Merced Land Use. Pri primerjavi rezultatov omenjenih podatkovnih zbirk z rezultati arhitekture *ResNet50* je mogoče opaziti, da so povprečne pogostosti izbire slojev znotraj posameznih blokov skupin rešitev pravzaprav skoraj obratne. V primeru arhitekture *ResNet50* imajo višjo povprečno pogostost izbire slojev znotraj posameznega bloka najboljše in srednje dobre rešitve, medtem ko je pri arhitekturi *MobileNet* ravno nasprotno. Pri rezultatih arhitekture *MobileNet* ponovno izstopajo rezultati nad podatkovno zbirko *DeepWeeds*, vendar spet ravno obratno kot pri prejšnjih rezultatih. V tem primeru je razviden vzorec, da imajo najboljše in srednje dobre rešitve višjo povprečno pogostost izbire kot najslabše rešitve. Če se osredotočimo na zadnje bloke konvolucijske osnove, je mogoče zaznati, da je povprečna pogostost izbire pri podatkovnih zbirkah *Colorectal Histology* in *UC Merced Land Use* pri najboljših rešitvah nizka, medtem ko je pri najslabših rešitvah višja. Pri podatkovni zbirki *DeepWeeds* pa je povprečna pogostost izbire slojev znotraj zadnjega bloka konvolucijske osnove ravno obratna.

3.4.3 Diskusija

Z analizo rezultatov pogostosti izbire posameznih slojev konvolucijske osnove ter analizo pogostosti izbire slojev znotraj posameznih konvolucijskih blokov smo z uporabo različnih arhitektur CNN in podatkovnih zbirk skušali empirično ugotoviti, ali obstaja kakšen splošen vzorec kombinacije izbir slojev oz. blokov, ki v večini primerov daje dobre rezultate. Glede na rezultate analize lahko sklenemo, da splošnega vzorca, ki bi bil skupen vsem arhitekturam in podatkovnim zbirkam, ni mogoče zaznati. Pri arhitekturi *VGG16* je mogoče zaznati vzorec, ki v nasprotju s splošnimi smernicami izbire slojev [43] kaže, da je smiselno za uglaševanje izbirati



Slika 3.10 Pogostost izbire slojev arhitekture *MobileNet* znotraj posameznega bloka glede na podatkovno zbirko.

sloje, ki se ne nahajajo na koncu konvolucijske osnove, medtem ko pri uporabi naprednejših arhitektur CNN takšnega vzorca ni zaznati. Posledično vzorca, zaznanega pri arhitekturi *VGG16*, ne moremo posplošiti na vse arhitekture CNN. Iz tega lahko sklepamo, da je še posebej za naprednejše arhitekture CNN, ki vsebujejo veliko število raznovrstnih slojev, za reševanje problema izbire uglaševanih slojev smiselno uporabiti nekakšno metodo oz. mehanizem. Predpostavljamo, da bi s takšno metodo, ki bi samodejno optimizirala izbiro uglaševanih slojev glede na izbrano arhitekturo in ciljni problem, lahko dosegli višjo uspešnost napovednih modelov.

Poglavje 4

Prilagodljivo uglaševanje slojev konvolucijske nevronske mreže

In the long run, curiosity-driven research just works better... Real breakthroughs come from people focusing on what they're excited about.

- Geoffrey Hinton

Poglavje predstavlja razvito zasnovo prilagodljivega uglaševanja slojev konvolucijskih nevronskih mrež pri strojnem učenju s prenosom znanja in na tej zasnovi razvito metodo. Metoda temelji na optimizacijskem algoritmu diferencialne evolucije, ki ga izrabljamo za namen iskanja najbolj optimalne izbire uglaševanih slojev uporabljene arhitekture CNN. Razvita prilagodljiva metoda *DEFT* (Differential Evolution based Fine-Tuning) je na začetku predstavljena v obliki konceptualne zasnove, v nadaljevanju pa je podrobno razčlenjena v treh sklopih, pri čemer vsak izmed njih predstavlja enega izmed ključnih delov metode. Prvi, ključni del metode predstavlja mehanizem izbire slojev vključno z algoritmom diferencialne evolucije, drugi vrednotenje izbranih slojev, tretji pa učenje končnega modela.

4.1 Obstoječe raziskave

Področje prilagodljivega uglaševanja slojev CNN je dokaj neraziskano, saj problematika temelji na učenju s prenosom znanja, ki pa svoj razcvet doživlja šele v zadnjih letih.

Najsorodnejšo raziskavo so avtorji predstavili v članku [37], ki obravnavano problemsko področje naslavlja s samodejno izbiro posameznega sloja nevronske mreže za vsak posamezni učni primerek. To so dosegli z vpeljavo dodatne nevronske

mreže, ki skrbi za odločanje, ali bo posamezni učni primerek procesiran skozi uglaševan sloj ali skozi sloj zamrznjen za učenje ter uporabo arhitekture *ResNet* globoke konvolucijske mreže, katere lastnost je odpornost zamenjavo in preskakovanje posameznih blokov arhitekture.

V zadnjem letu je mogoče opaziti tudi raziskavi [130, 131], ki za namen prilagodljivega uglaševanja podobno kot avtorji prej predstavljenega članka vpeljujeta dodatno nevronsko mrežo, ki skrbi za odločanje, katere izmed slojev uglaševati in katere pustiti zamrznjene. Omenjenima raziskavama je skupno, da za dosego cilja izrabljata specifične arhitekture globoke konvolucijske mreže.

Za razliko od avtorjev sorodnih del, naša predlagana metoda deluje samodejno, prilagodljivo glede na podano specifično podatkovno množico. Prav tako za svoje delovanje ne izrablja specifičnih lastnosti arhitektur globokih nevronskih mrež in jo je kot tako mogoče uporabiti z različnimi arhitekturami konvolucijskih nevronskih mrež.

4.2 Konceptualna zasnova

Sama ideja za razvoj prilagodljive metode, ki naslavlja problem izbire uglaševanih slojev CNN pri učenju s prenosom znanja, izhaja iz ideje oz. pristopa reševanja problematike izbire značilnic. Na problemskem področju izbire značilnic je namreč eden izmed uveljavljenih pristopov uporaba različnih optimizacijskih algoritmov, kot so algoritem naključnega iskanja [121], genetski algoritem [107], diferencialna evolucija [132], ki v različnih raziskavah [102, 100] izkazujejo nadpovprečne rezultate. Pri reševanju problema izbire značilnic je pogost pristop preoblikovanja problema v optimizacijski problem, ki ga je mogoče reševati s prej omenjenimi optimizacijskimi algoritmi, vpeljan na način, da so posamezniki predstavljeni v obliki enodimenzionalnega polja oz. vektorja, katerega dolžina je enaka številu vseh značilnic, posamezna vrednost znotraj polja pa predstavlja, ali je posamezna značilnica vključena v proces učenja ali ne.

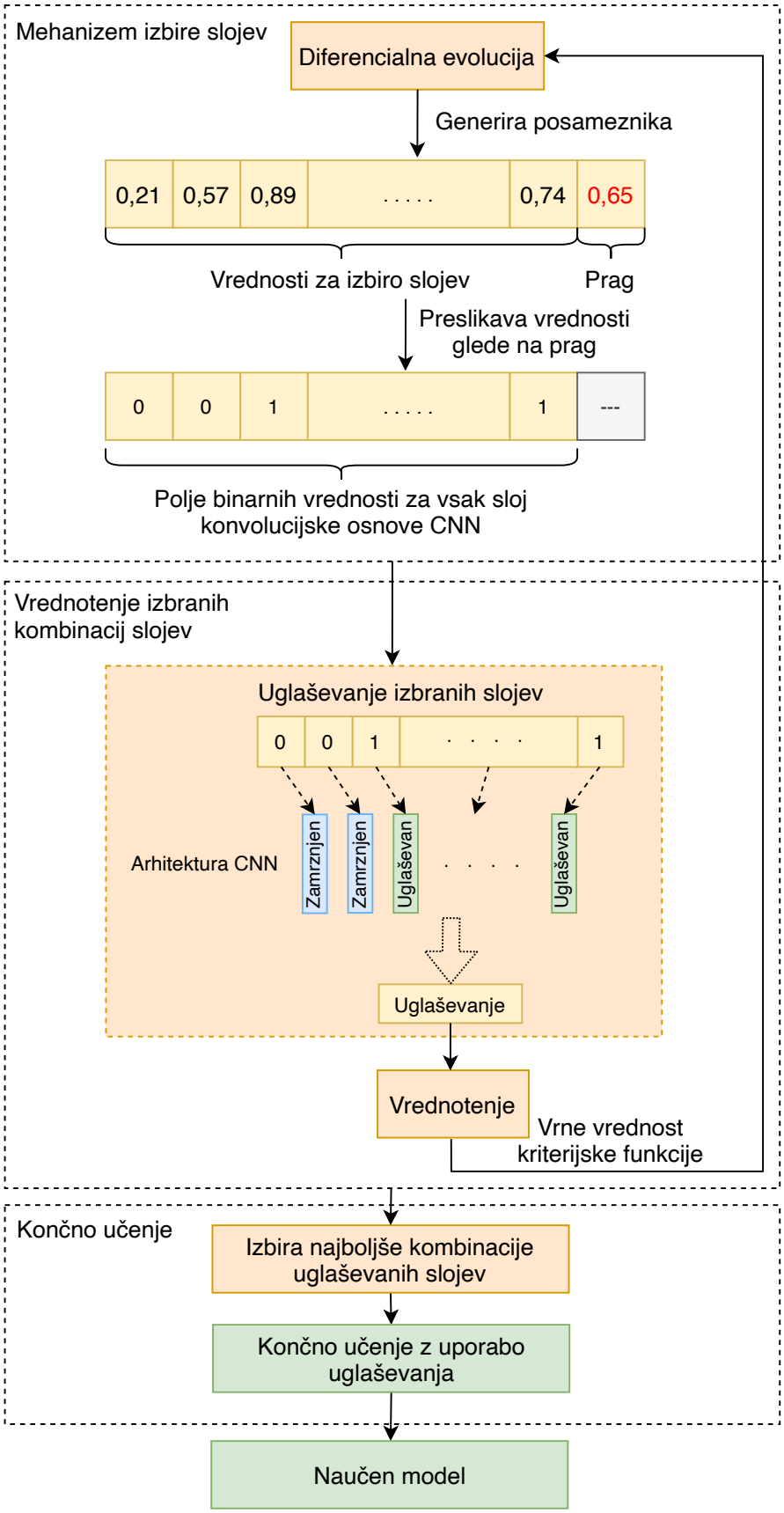
Problem izbire uglaševanih slojev CNN pri učenju s prenosom znanja lahko enostavno preslikamo na problem izbire značilnic, le da v tem primeru značilnice nadomestimo oz. zamenjamo s posameznim slojem konvolucijske osnove uporabljene arhitekture CNN. Če idejo malo podrobneje razčlenimo in si pogledamo konceptualni diagram metode za prilagodljivo uglaševanje slojev CNN na sliki 4.1, je ta sestavljena iz treh delov: mehanizma izbire slojev, vrednotenja izbranih kombinacij slojev ter končnega učenja.

Naloga mehanizma izbire slojev je zagotavljanje posameznih izvedljivih rešitev oz. kombinacij izbranih slojev za uglaševanje, ki so v nadaljevanju ovrednotene,

na podlagi ovrednotenja pa je ustvarjena naslednja izvedljiva rešitev. Mehanizem torej kot izhod zagotavlja polje binarnih vrednosti enake dolžine, kot je število slojev znotraj konvolucijske osnove uporabljene CNN, pri čemer posamezna vrednost znotraj polja odraža, ali je posamezni sloj izbran za uglaševanje ali ne.

Glede na rešitev, pridobljeno z uporabo mehanizma izbire slojev, je naloga mehanizma za vrednotenje izbranih kombinacij slojev pravilna preslikava binarnega polja vrednosti v uglaševane in zamrznjene sloje konvolucijske osnove izbrane arhitekture CNN. V primeru, da je posamezna vrednost v podanem binarnem polju enaka 1, je posamezni sloj izbran za uglaševanje, v nasprotnem primeru pa ni izbran za uglaševanje oz. ostane zamrznjen za učenje. Po opravljeni preslikavi podane rešitve na dejanske sloje izbrane arhitekture CNN sledi proces uglaševanja ter za tem vrednotenja tako pridobljenega napovednega modela. Pridobljena vrednost kriterijske funkcije se vrne v mehanizem izbire slojev, na podlagi katere je ustvarjena nova izvedljiva rešitev. Predstavljen proces se ponavlja s ciljem minimizacije vrednosti kriterijske funkcije, dokler ni dosežen zaključni pogoj – maksimalno število ovrednotenj. Maksimalno število ovrednotenj je podano kot parameter metode in ga je kot takšnega mogoče prilagajati glede na želje končnega uporabnika.

Po doseženem zaključnem pogoju sledi izbira najuspešnejše kombinacije uglaševanih in zamrznjenih slojev CNN, ki je v zadnjem koraku metode uporabljena za učenje končnega modela nad celotno učno množico. Tako pridobljen model je tudi končni rezultat razvite metode, katerega klasifikacijsko točnost navadno na koncu ovrednotimo nad testno množico in predstavlja uspešnost napovedovanja posamezne metode.

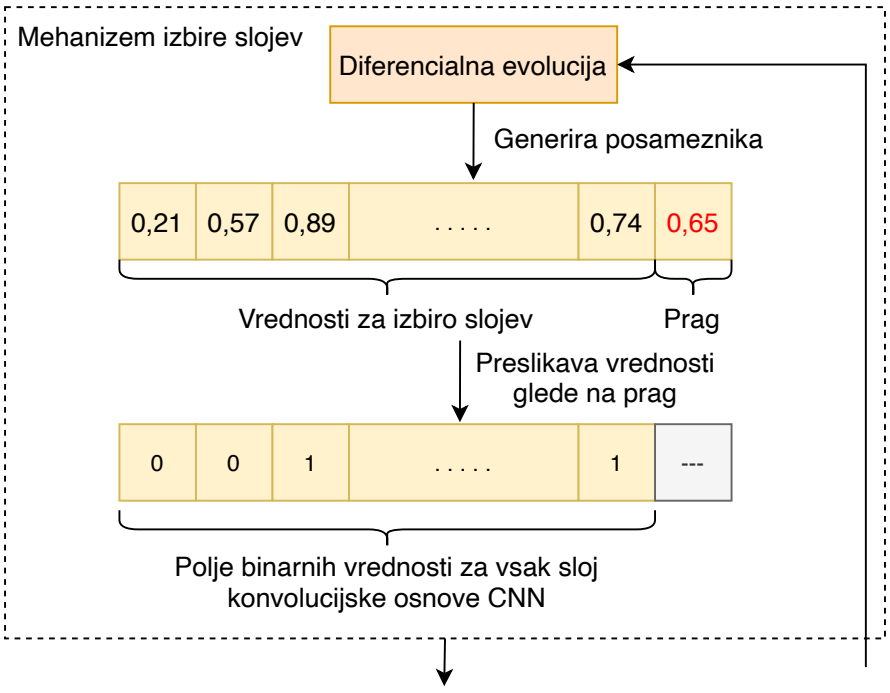


Slika 4.1 Konceptualna zasnova metode za prilagodljivo uglaševanje slojev konvolucijske nevronske mreže.

4.3 Mehanizem izbire slojev

Mehanizem izbire slojev predlagane metode, predstavljen na sliki 4.2, temelji na osnovnem algoritmu diferencialne evolucije, katerega predstavitev posameznikov je prilagojena za reševanje naloge izbire najprimernejših uglaševanih slojev CNN. Čeprav bi lahko v ta namen uporabili katero izmed nadgrajenih različic algoritma DE, kot so jDE21, NL-SHADE-RSP ali MadDE, ki so na tekmovanju CEC leta 2021 dosegle najvišje rezultate, smo se v našem primeru za osnovno različico DE odločili predvsem zaradi enostavnosti implementacije slednjega ter želje, da uspešnost delovanja metode preizkusimo na algoritmu, ki morda ni najuspešnejši, saj na ta način dodatno potrdimo delovanje metode same, in ne zgolj superiornosti izbranega algoritma.

Na začetku algoritem DE naključno inicializira posameznika (polje oz. vektor realnih števil), ki se nato preslika v binarno polje vrednosti. Polje binarnih vrednosti predstavlja posamezni sloj arhitekture CNN, posamezna vrednost znotraj polja pa določa, ali je posamezni sloj omogočen za uglaševanje ali ne. Na podlagi posameznika je nato napovedni model učen z uporabo uglaševanja pri učenju s prenosom znanja. Po koncu učenja je tak model ovrednoten z izbrano kriterijsko funkcijo, katere vrednost služi kot povratna informacija algoritmu DE, na podlagi katere je ustvarjen nov posameznik.



Slika 4.2 Prikaz delovanja mehanizma izbire uglaševanih slojev konvolucijske nevronske mreže.

Posamezniki so v metodi predstavljeni kot polje oz. vektor realnih vrednosti, ki jih lahko formalno predstavimo, kot je prikazano v enačbi 4.1, za vsak $i = 0, \ldots, NP$, kjer je za vsak sloj $x_{i,0}^{(t)}$ od $i = 0, \ldots, D$ izbrana vrednost na intervalu $[0, 1]$ in kjer D predstavlja skupno število vseh slojev konvolucijske osnove izbrane arhitekture CNN.

$$x_i^{(t)} = (x_{i,0}^{(t)}, \ldots, x_{i,D}^{(t)}, Th_i^{(t)}) \tag{4.1}$$

Z namenom pridobitve čim bolj raznolikih posameznikov pri preslikavi s strani DE pridobljenih posameznikov v binarna polja oz. vektorje smo uporabili metodo pomikajočega se pragu (angl. moving threshold method), predstavljenega v delu [133]. Mehanizem, uporabljen pri omenjeni metodi, deluje na način, da je prag oz. njegova vrednost del posameznika in je posledično določena dinamično, kar eliminira potrebo po ročnem nastavljanju vrednosti pragu. Uporaba takšnega pristopa nam zagotavlja ustvarjanje bolj raznolikih posameznikov, s čimer se lahko potencialno izognemo pojavu, da optimizacijski algoritem obstane v kakšnem izmed lokalnih optimumov. Sama preslikava je formalno definirana kot:

$$s_{i,j}^{(t)} = \begin{cases} 1, \text{če } x_{i,j}^{(t)} > Th^{(t)}, \\ 0, \text{sicer}, \end{cases} \tag{4.2}$$

kjer $s_{i,j}^{(t)}$ označuje binarno vrednost posamezne realne vrednosti posameznika, $x_{i,j}^{(t)}$ dejansko posamezno realno vrednost ter Th vrednost pragu.

Končno tako posamezni vektor s_i predstavlja i-to binarno polje preslikanega i-tega posameznika, kjer vrednost 0 za j-ti sloj odraža, da sloj ni bil izbran za uglaševanje, medtem ko vrednost 1 odraža, da je bil j-ti sloj izbran za uglaševanje. Dolžina takšnega preslikanega vektorja s_i je zmanjšana za 1 ($D - 1$).

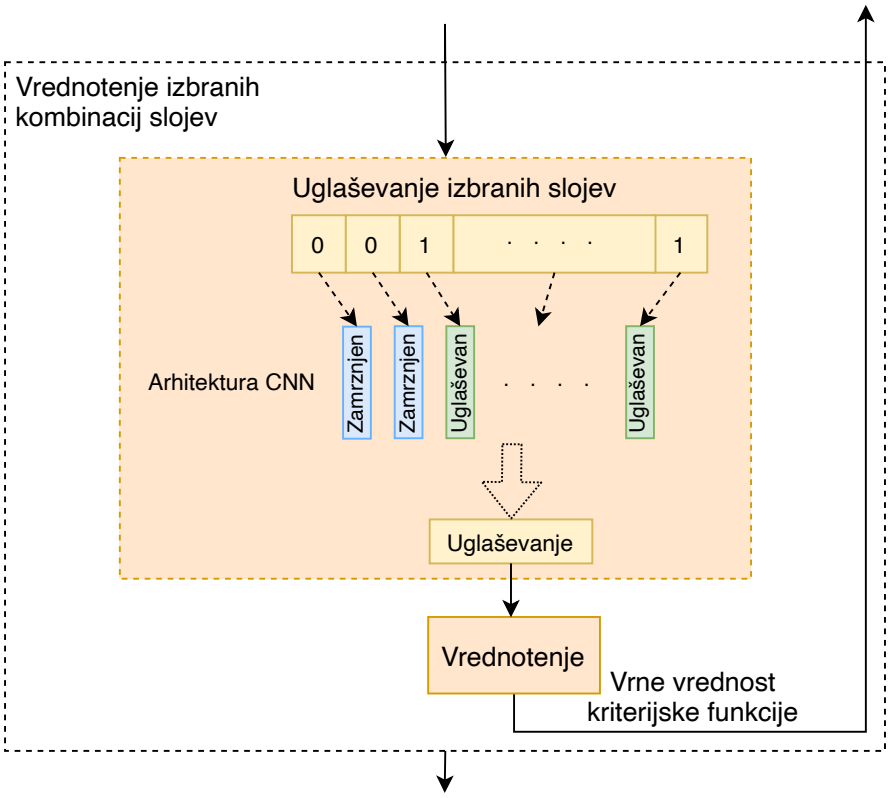
4.4 Vrednotenje izbranih kombinacij slojev

Za vsakega izmed pridobljenih posameznikov iz mehanizma izbire slojev je v delu metode, namenjenem vrednotenju izbranih kombinacij slojev (slika 4.3), najprej opravljena preslikava pridobljenega binarnega vektorja v uglaševane in zamrznjene sloje uporabljene arhitekture CNN. Po opravljeni preslikavi sledi proces uglaševanja pri učenju s prenosom znanja, pri čemer kot rezultat dobimo naučen model. V tem delu za učenje posameznih izbir slojev uporabimo le 80 % celotne učne množice, preostalih 20 % pa uporabimo za ovrednotenje naučenega modela. Naučen model v nadaljevanju ovrednotimo z izbrano kriterijsko funkcijo, rezultat katere nam v nadaljevanju služi kot povratna informacija uspešnosti posameznika algoritmu DE.

Uglaševanje je opravljeno s podanim maksimalnim številom epoh in z uporabo tehnike zgodnje prekinitve učenja. Uporaba omenjene tehnike metodi posredno tudi omogoča, da učenje ne tako obetavnih posameznikov oz. izbranih kombinacij uglaševanih slojev predčasno prekine in s tem prihrani čas, ki bi ga sicer porabila za učenje manj perspektivne rešitve do konca.

Maksimalno število epoh je parameter metode, ki ga je treba nastaviti ročno. V splošnem je nastavitev vrednosti parametra odvisna od uporabljene arhitekture CNN in ciljne podatkovne zbirke, vsekakor pa vrednosti tega parametra ni mogoče uniformno določiti za vse primere.

Za ovrednotenje napovedne uspešnosti pridobljenega napovednega modela, naučenega z uporabo uglaševanja pri učenju s prenosom znanja, smo uporabili kriterijsko funkcijo L, predstavljeno v poglavju 3.2.3.



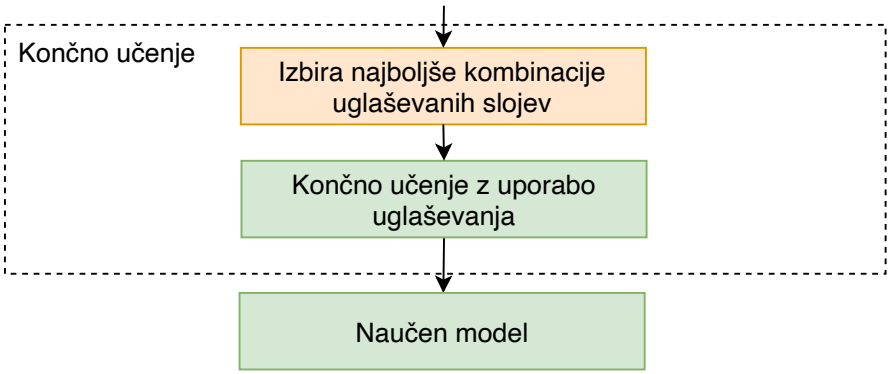
Slika 4.3 Postopek vrednotenja izbranih kombinacij uglaševanih slojev konvolucijske nevronske mreže.

4.5 Učenje končnega modela

Zadnji ključni del metode za prilagodljivo uglaševanje slojev CNN je učenje končnega modela, ki je predstavljeno na sliki 4.4. V tem delu metode je v prvem koraku treba

izbrati najboljšo izmed preizkušenih kombinacij uglaševanih slojev. Ker smo za kriterijsko funkcijo uporabili *CCE*, je najboljša preizkušena kombinacija uglaševanih slojev tista, ki je v procesu iskanja dosegla najnižjo vrednost kriterijske funkcije.

Izbrana najboljša preizkušena kombinacija uglaševanih slojev izbrane CNN je nato, za razliko od učenja pri vrednotenju izbranih kombinacij slojev, učena nad celotno učno množico. Tako naučen model je končni rezultat metode *DEFT*.



Slika 4.4 Končno učenje najboljše kombinacije izbranih uglaševanih slojev konvolucijske nevronske mreže.

4.6 Metoda DEFT

V prejšnjih poglavjih smo predstavili prilagodljivo uglaševanje slojev za izbrano arhitekturo CNN. V nadaljevanju pa je podrobno predstavljen algoritem razvite metode prilagodljivega uglaševanja slojev CNN.

Psevdokod metode je prikazan v algoritmu 4.1. Za vsako ponovitev 10-kratnega prečnega preverjanja metoda kot vhod prejme učno množico *UM*, na podatkovni zbirki *ImageNet* pred-naučen model *M*, število ovrednotenih posameznih izbir kombinacij uglaševanih slojev p, maksimalno število uporabljenih epoh e ter zaustavitveni pogoj zp. Rezultat metode *DEFT* je model *M*, naučen z najboljšo najdeno kombinacijo izbir uglaševanih slojev.

Na začetku je na vhodu podana učna množica *UM*, razdeljena na dve podmnožici UM_{ucna} in UM_{testna} v razmerju 80 : 20. Zatem sledi inicializacija algoritma *DE* ter spremenljivk $r_{najboljsa}$ ter l_{min}.

V nadaljevanju sledi iskanje najboljše rešitve oz. kombinacije izbir uglaševanih slojev CNN. Z uporabo algoritma DE pridobimo *rešitev*, na podlagi katere za uglaševanje omogočimo oz. onemogočimo sloje konvolucijske nevronske mreže modela $M_{zacasni}$. Sledi učenje nad množico UM_{ucna} z uporabo uglaševanja pri učenju s prenosom znanja za podanih e epoh. Po vsaki opravljeni epohi učenja se preveri, če je izpolnjen

zaustavitveni pogoj *zp*. Za zaustavitveni pogoj je privzeto definiran mehanizem zgodnje prekinitve učenja, ki je izpolnjen takrat, ko v treh zaporednih epohah ni zaznanega napredka oz. zmanjšanja vrednosti funkcije izgube na učno množico UM_{ucna}. Če je zaustavitveni pogoj *zp* izpolnjen, se učenje predčasno prekine, sicer pa se učenje nadaljuje. Po končanem učenju se izračuna vrednost funkcije izgube l modela $M_{zacasni}$ nad testno množico podatkov UM_{testna}. Če je vrednost l manjša od vrednosti l_{min}, je takšna rešitev izbrana kot trenutno najboljša najdena rešitev, vrednost $r_{najboljsa}$ se priredi vrednosti *resitev*, vrednost l_{min} pa se priredi vrednosti l. Nato se algoritmu DE kot vrednost kriterijske funkcije vrne vrednost l. Ta postopek se ponavlja, dokler števec ponovitev i ne doseže maksimalnega števila ovrednotenih posameznikov p.

Po doseženem maksimalnem številu ovrednotenih posameznikov p smo pridobili najboljšo *resitev* oz. izbiro uglaševanih slojev CNN, ki jo preslikamo na sloje modela M. Zatem izvedemo proces učenja z uporabo uglaševanja pri učenju s prenosom znanja ob uporabi celotne učne množice UM ter preverjanjem zaustavitvenega pogoja *zp* v vsaki epohi učenja. Po zaključku učenja kot rezultat metode *DEFT* dobimo naučen model M, ki ga lahko v nadaljevanju ovrednotimo nad testno množico izbrane podatkovne zbirke. Psevdoko postopka vrednotenja naučenega modela M je prikazan v algoritmu 4.2.

Algoritem 4.1: Psevdokod metode *DEFT*.

Vhod : Učna množica UM
Vhod : Pred-naučen model M
Vhod : Število ovrednotenih rešitev (posameznikov) p
Vhod : Število epoh e
Vhod : Zaustavitven pogoj zp
Rezultat: Z uglaševanjem naučen model M

```
1  begin
2      UM_ucna, UM_testna = razdeli(UM) ;      // Podano učno množico razdeli v
        razmerju 80:20
3      DE = inicializiraj_DE();
4      r_najboljsa = [] ;                          // Trenutno najboljša rešitev
5      l_min = ∞ ;     // Vrednost funkcije izgube trenutne najboljše rešitve
       // Iskanje najboljše kombinacije uglaševanih slojev
6      for i = 1 . . . , p do
7          resitev = DE.pridobi_resitev();
8          M_zacasni = M;
9          M.nastavi_sloje(resitev); // Omogoči uglaševanje posameznih slojev
10         for j = 1 . . . , e do
11             M_zacasni.izvedi_ucenje(UM);
12             if zaustavitven_pogoj_izpolnjen(zp) then
13                 break;
14             else
15                 continue;
16             end
17         end
18         l = izracunaj_L(UM_testna);
19         if l < l_min) then
20             r_najboljsa = resitev;
21             l_min = l;
22         else
23             continue;
24         end
25         DE.vrni_vrednost_kriterijske_funkcije(l);
26     end
       // Končno učenje najboljše kombinacije uglaševanih slojev
27     M.nastavi_sloje(r_najboljsa);
28     for j = 1 . . . , e do
29         M.izvedi_ucenje(UM);
30         if zaustavitven_pogoj_izpolnjen(zp) then
31             break;
32         else
33             continue;
34         end
35     end
36 end
```

Algoritem 4.2: Psevdokod vrednotenja modela pridobljenega z uporabo metode *DEFT*.

```
Vhod      : Testna množica TM
Vhod      : Z uporabo metode DEFT naučen model M
Rezultat  : Vrednosti klasifikacijskih metrik
1 begin
2     metrike = [];
3     napovedani_razredi = [];
4     for i ∈ TM do
          // Napoved testnega primerka shrani v polje napovedani_razredi
5         napovedani_razredi.dodaj(M.predict(i));
6     end
      // Izračun klasifikacijskih metrik
7     metrike = izracunaj_metrike(napovedani_razredi, TM.resnicni_razredi);
8 end
```

Poglavje 5

Zaznavanje primernosti izbir uglaševanih slojev konvolucijske nevronske mreže

"Everything must be made as simple as possible.
But not simpler."

- Albert Einstein

V tem poglavju je predstavljena metrika za zaznavanje primernosti izbire kombinacij uglaševanih slojev v zgodnjih fazah učenja. Na začetku je opisana sama ideja za razvoj metrike, temelječe na funkciji izgube. V nadaljevanju je metrika formalno predstavljena, njeno delovanje pa ovrednoteno na podlagi analize empirično pridobljenih rezultatov.

5.1 Ideja

V splošnem je časovna zahtevnost optimizacijskih algoritmov odvisna predvsem od časovne zahtevnosti izvajanja kriterijske funkcije, ta pa je pri uporabi predstavljene metode *DEFT* še posebej časovno zahtevna. Časovna zahtevnost je v našem primeru problematična zato, ker je treba za izračun vrednosti kriterijske funkcije uglasiti posamezno izbiro kombinacij uglaševanih slojev ter ovrednotiti uspešnost tako naučenega modela. Z uporabljenim mehanizmom predčasne zaustavitve učenja sicer potencialno prihranimo nekaj epoh, a tak mehanizem ne predstavlja zadovoljive rešitve, saj epohe prihranimo le v primeru, da pri učenju ni več zaznati napredka, do česar pa ni nujno, da pride.

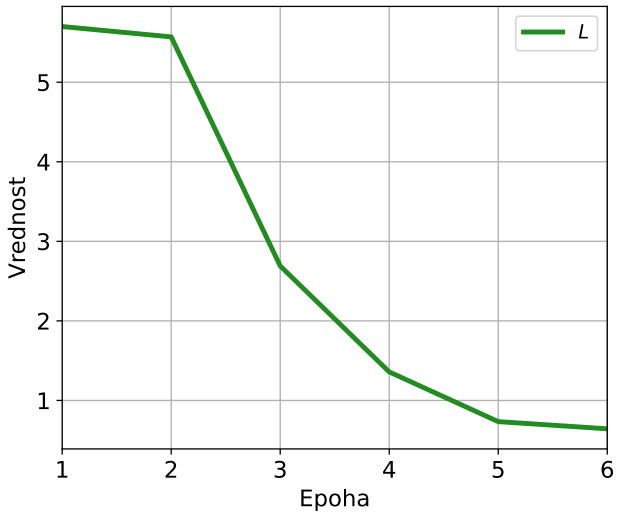
Na podlagi želje po zmanjšanju časovne zahtevnosti predstavljene metode *DEFT* se je porodila zamisel o razvoju metrike, ki bi bila sposobna v zgodnjih oz. čim zgodnejših fazah učenja globokih arhitektur CNN zaznati, ali je določeno izbiro uglaševanih slojev sploh smiselno učiti do konca (do maksimalnega števila epoh), ali je bolje, da se učenje predčasno ustavi, saj trenutna izbira glede na potek učenja ne obeta najboljšega končnega rezultata. Z drugimi besedami, cilj metrike je s čim manj porabljenimi epohami ugotoviti, ali je neko rešitev oz. izbiro uglaševanih slojev CNN smiselno učiti do konca ali ne.

Izhodišče za razvoj metrike temelji na opazovanju načina, kako raziskovalci ocenjujemo, ali učenje z uporabo neke kombinacije uglaševanih slojev ali kombinacije vrednosti učnih parametrov ali izbrane arhitekture CNN poteka oz. napreduje, kot želimo. Tipično napredek pri učenju ocenjujemo s spremljanjem vrednosti funkcije izgube oz. s spremljanjem grafa vrednosti funkcije izgube. Na ta način lahko že v zgodnjih fazah učenja zaznamo različne pojave pri strojnem učenju, kot so morebitno prekomerno prileganje, preslabo prileganje, nezmožnost konvergiranja modela ipd., ter takšno učenje predčasno prekinemo. Odločitve za predčasno prekinitev učenja pa ne sprejmemo zgolj glede na opazovanje absolutnih vrednosti uporabljene kriterijske funkcije, ampak poleg tega spremljamo tudi splošen trend vrednosti, hitrost padanja, morebitno naraščanje ali osciliranje vrednosti znotraj nekega intervala. Na podlagi predstavljenih opazovanj se je porodila ideja o razvoju metrike, temelječe na funkciji izgube (angl. loss derived metric, *LDM*), ki za razliko od klasične metrike, ki predstavlja zgolj absolutno vrednost, posredno vključuje tudi informacije o poteku učenja v predhodnih epohah.

5.2 Definicija

Razvoj metrike *LDM* je predstavljen po korakih, kot je bila razvita. Ker metrika temelji na funkciji izgube L, s tem tudi začnemo. Na sliki 5.1 je predstavljen primer grafa vrednosti funkcije izgube. Iz grafa je razvidno, da je neki model bil učen 6 epoh, graf vrednosti funkcije izgube pa je zvezno padajoč z minimumom v šesti epohi.

Z namenom vključitve predhodnih vrednosti funkcije izgube smo v prvem koraku razvoja metrike *LDM* vrednosti funkcije izgube normalizirali z uporabo min-max normalizacije. S tem korakom smo torej vrednosti funkcije izgube preslikali na vrednosti znotraj intervala $I = [0, 1]$. Posledično normalizirane vrednosti funkcije izgube v izbrani epohi NL_i predstavljajo relativno vrednost glede na absolutno maksimalno vrednost funkcije izgube L ter na ta način posredno vključujejo informacije o predhodnih vrednostih funkcije izgube. Graf na sliki 5.2 predstavlja primerjavo vrednosti



Slika 5.1 Graf vrednosti funkcije izgube L.

funkcije izgube L ter normalizirane vrednosti funkcije izgube (angl. normalized loss, NL).

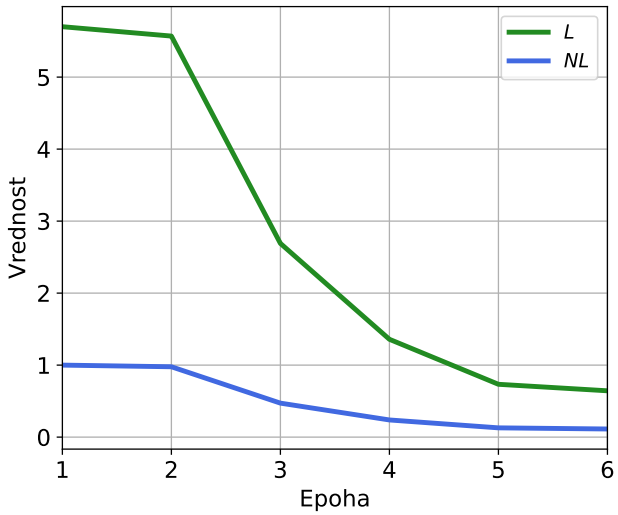
Z opisanim postopkom smo ustvarili enostavno, na funkciji izgube temelječo metriko NL, ki posredno vključuje informacije o predhodnih vrednostih funkcije izgube L. Ker tako izpeljana metrika ne vključuje informacij o splošnem trendu gibanja vrednosti funkcije izgube skozi posamezne epohe, se je porodila ideja o nadgradnji metrike, ki deluje po vzoru klasifikacijske metrike povprečnega območja pod krivuljo AUC.

V našem primeru način izračuna površine pod krivuljo ROC uporabimo za izračun površine pod funkcijo izgube NL. Formalno lahko površino p_{nl} pod normalizirano krivuljo funkcije izgube NL definiramo kot:

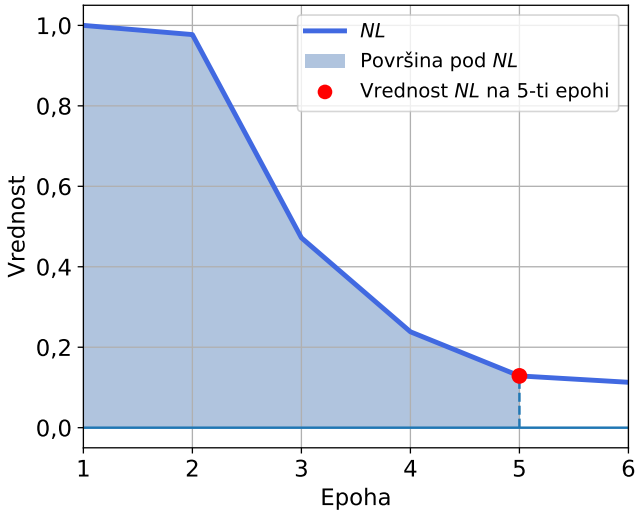
$$p_{NL} = \int_a^b NL(x)\,dx = \sum_{i=1}^{n} \frac{y_{i-1} + y_i}{2} h, \tag{5.1}$$

kjer a in b predstavljata prvo in zadnjo epoho, y_i vrednost NL v i-ti epohi, h pa korak, izražen kot $x_1 - x_0$. V našem primeru velja, da je $h = 1$, saj je definicijsko območje funkcije NL enako $D_{NL} = [1, n]$, pri čemer velja, da so točke znotraj D_{NL} ekvidistančne, torej velja $x_i - x_{i-1} = h$ za vsak $i = 1, 2, ..., n$ [134]. Grafična predstavitev površine pod krivuljo NL je podana na sliki 5.3.

Ker za izračun površine pod krivuljo NL potrebujemo vsaj dve vrednosti NL, je površino mogoče izračunati le od druge epohe naprej. Ker tako izračunana površina naravno narašča z vsako naslednjo epoho, v nadaljevanju izračunano površino izrazimo v deležu maksimalne površine. Maksimalna površina v tem primeru



Slika 5.2 Graf normalizirane vrednosti funkcije izgube *NL* in vrednosti funkcije izgube *L*.



Slika 5.3 Graf površine pod krivuljo normalizirane funkcije izgube na peti epohi.

predstavlja površino pod krivuljo NL, kjer je vrednost vseh točk enaka 1. Formalno lahko maksimalno površino p_m predstavimo kot:

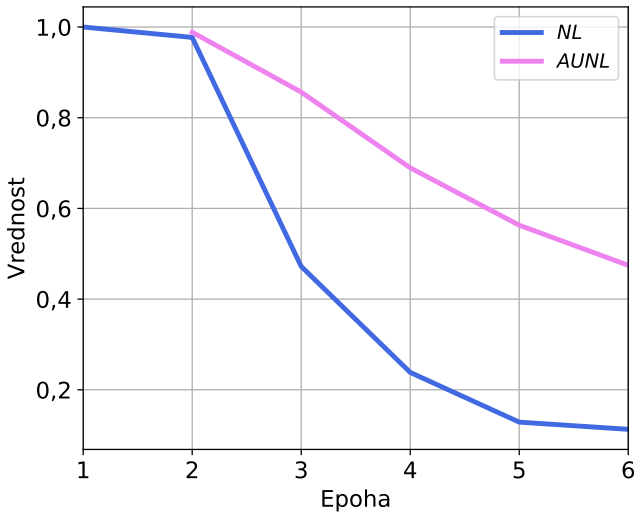
$$p_{max} = NL_{max} \cdot b, \tag{5.2}$$

kjer b predstavlja zadnjo epoho oz. izračun normalizirane vrednosti funkcije izgube, NL_{max} pa je maksimalna vrednost normalizirane vrednosti funkcije izgube, ki je enaka 1.

Na podlagi površine pod krivuljo lahko, kot je prikazano v enačbi 5.3, formalno izpeljemo metriko, temelječo na površini pod krivuljo funkcije izgube (angl. area under normalized loss, $AUNL$)

$$AUNL = \frac{p_{NL}}{p_{max}} \tag{5.3}$$

Na sliki 5.4 sta prikazani krivulji vrednosti metrik NL in $AUNL$. Iz prej omenjenega razloga, je prva vrednost metrike $AUNL$ prikazana šele v drugi epohi.



Slika 5.4 Graf vrednosti metrik NL in $AUNL$.

Z metriko $AUNL$ smo uspeli v samo vrednost metrike vključiti tudi informacije o splošnem trendu gibanja vrednosti normalizirane funkcije izgube, vendar pa bi želeli, da je ta trend malenkost bolj izrazit. Če primerjamo krivulji na sliki 5.4, je mogoče zaznati, da je med drugo in četrto epoho padanje vrednosti NL izrazitejše, pri vrednostih metrike $AUNL$ pa malo zaznavno. S tem ciljem smo šli v razvoju metrike še en korak naprej, in sicer smo kot indikator trenda vrednosti NL, smo vpeljali površino pod najbolj prilegajočim se polinomom prve stopnje (premico).

Polinom prve stopnje, ki se karseda najbolj prilega vrednostim *NL*, poiščemo z uporabo regresije in metode najmanjših kvadratov (angl. least square error, LSE). Torej lahko vrednosti metrike *NL* v vsaki epohi izrazimo v obliki točk:

$$(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n), \tag{5.4}$$

kjer x predstavlja posamezno zaporedno epoho, y pa pripadajočo vrednost metrik *NL* v tej epohi. Iskani model najbolj prilegajočega se polinoma prve stopnje lahko definiramo kot [135]:

$$y_i = a + bx_i + \epsilon_i; \qquad i = 1, 2, \ldots, n, \tag{5.5}$$

kjer velja

$$\epsilon_i \sim N(0, \sigma^2). \tag{5.6}$$

To pomeni, da so vse napake ϵ_i porazdeljene normalno, z matematičnim upanjem 0 in varianco σ^2. Z uporabo linearne regresije nato poiščemo takšni vrednosti za parametra a in b, da bo vrednost *LSE*, definirana v enačbi 5.7, najmanjša.

$$LSE := \sum_{i=1}^{n} \epsilon_i^2 = \sum_{i=1}^{n} (y_i - a - bx_i)^2 \tag{5.7}$$

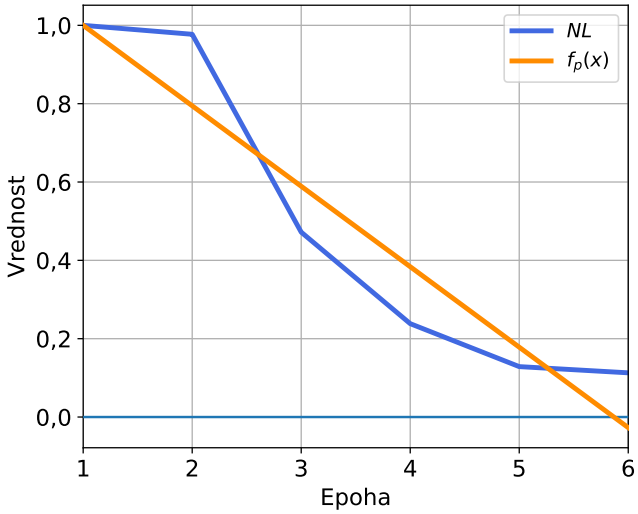
Tako pridobljen polinom prve stopnje $f_p = argminLSE$ je grafično predstavljen na sliki 5.5. Površino pod polinom prve stopnje p_{f_p}, predstavljeno na sliki 5.6, izračunamo na enak način kot pri izračunu površine pod normalizirano krivuljo funkcije izgube *NL*:

$$p_{f_p} = \int_a^b f_p(x)\, dx = \sum_{i=1}^{n} \frac{y_{i-1} + y_i}{2} h, \tag{5.8}$$

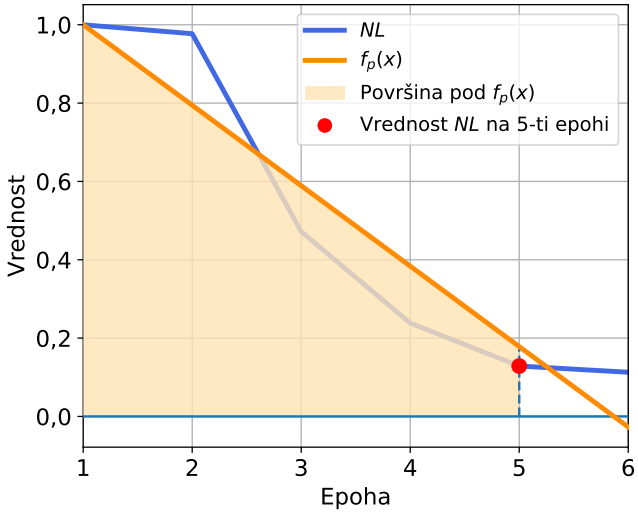
kjer a in b predstavljata prvo in zadnjo epoho, y_i vrednost f_p v i-ti epohi, h pa korak, izražen kot $x_1 - x_0$.

Enako kot pri površini pod normalizirano krivuljo funkcije izgube *NL* tudi v tem primeru površina z vsako nadaljnjo epoho narašča, kar je za naš primer uporabe neintuitivno, saj je za metriko, temelječo na funkciji izgube, podobno kot za vrednosti funkcije izgube, pričakovano, da skozi ponovitve pada. Zato končno metriko, temelječo na funkciji izgube *LDM*, izpeljemo, kot je prikazano v enačbi 5.9.

$$LDM = \frac{p_{f_p}}{p_{max}} \tag{5.9}$$

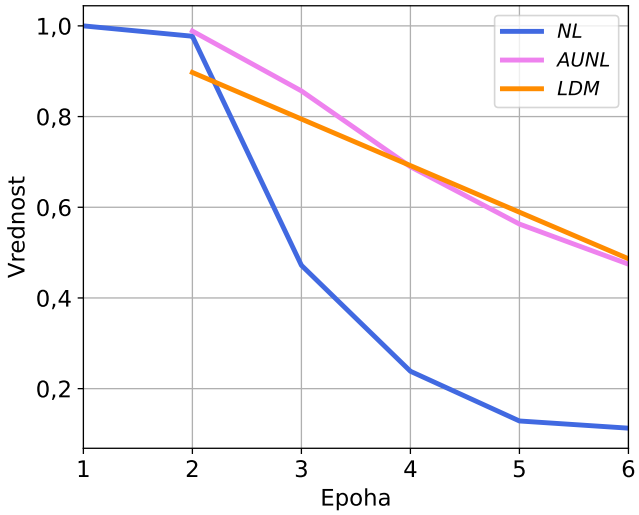


Slika 5.5 Graf najbolj prilegajočega se polinoma prve stopnje $f_(x)$.

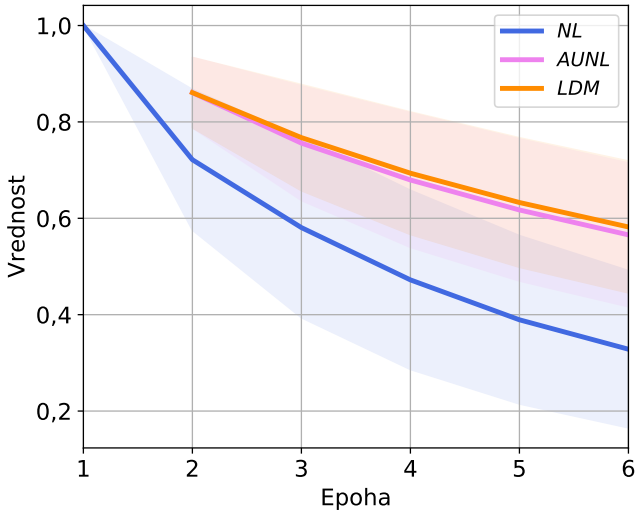


Slika 5.6 Graf površine pod najbolj prilegajočim se polinomom prve stopnje $f_p(x)$.

Na sliki 5.7 so prikazane krivulje vrednosti metrik *NL*, *AUNL* in *LDM*, iz katerihi je razvidno, da metrika *LDM* izkazuje bolj posplošen padajoč trend kot predhodna metrika *AUNL*, kar je bil tudi osnovni namen nadgradnje metrike *AUNL*. Slika 5.8 prikazuje empirično primerjavo krivulj empirično pridobljenih vrednosti prej omenjenih metrik, pri čemer črta prikazuje povprečno vrednost posamezne metrike v vsaki epohi, obarvano področje nad in pod njo pa standardni odklon.



Slika 5.7 Graf vrednosti metrik *NL*, *AUNL* in *LDM*.



Slika 5.8 Graf empiričnih vrednosti metrik *NL*, *AUNL* in *LDM*.

5.3 Zasnova eksperimenta

Z namenom temeljitega in objektivnega ovrednotenja v prejšnjem razdelku predstavljenih metrik, s čimer želimo utemeljiti izbiro ter preveriti smiselnost uporabe končne razvite metrike, smo zasnovali eksperiment po vzoru iz poglavja 3.

Podrobneje je zasnova eksperimenta predstavljena v algoritmu 5.1. Za vsako ponovitev 10-kratnega prečnega preverjanja podano učno množico *UM* razdelimo na dve podmnožici UM_{ucna} in UM_{testna} v razmerju 80 : 20. Večji del učne množice je namenjen uglaševanju posameznih izbir slojev, preostali, manjši del pa ovrednotenju naučenega modela. Z uporabo algoritma RS generiramo naključno kombinacijo izbir uglaševanih slojev, ki jih preslikamo na prednaučeni model. Preslikava poteka tako, da glede na pridobljeno rešitev posamezne sloje v modelu $M_{zacasni}$ omogočimo za uglaševanje, druge pa pustimo zamrznjene oz. jih onemogočimo za uglaševanje. Sledi uglaševanje oz. učenje modela z uporabo učne množice UM_{ucna}, del katere (10 %) je namenjen validacijski množici v procesu učenja. Po vsaki zaključeni epohi učenja izračunamo vrednosti metrik *NL*, *AUNL* in *LDM*. Ko učenje doseže maksimalno podano število epoh *e*, se izračuna še vrednost funkcije izgube *L* nad preostalim delom učne množice UM_{testna}. Vse izračunane metrike po vsaki končani ponovitvi shranimo, algoritem pa se izvaja, dokler ne dosežemo podanega maksimalnega števila ovrednotenih rešitev *p* oz. ovrednotimo *p* različnih kombinacij izbir uglaševanih slojev.

5.4 Izvedba eksperimentov

Po postopku, predstavljenem v prejšnjem poglavju, smo izvedli 9 eksperimentov, pri čemer smo uporabili tri različne podatkovne zbirke in tri različne arhitekture CNN. Podatkovne zbirke kot tudi arhitekture CNN smo izbrali enake kot pri analizi vpliva izbire slojev konvolucijske nevronske mreže na uspešnost učenja, predstavljeni v poglavju 3. Za podatkovne zbirke smo izbrali *Colorectal Histology*, *DeepWeeds* ter *UC Merced Land Use*, za arhitekture CNN pa *VGG16*, *ResNet50* in *MobileNet*. Izbrane podatkovne zbirke ter arhitekture CNN so podrobno predstavljene v poglavjih 3.3.1 in 2.3.4.

Enako kot pri analizi v poglavju 3 smo tudi v tem primeru podatkovne zbirke predobdelali po uveljavljenem postopku na enak način, kot je predstavljeno v poglavju 3.3.2. V eksperimentih smo uporabili na podatkovni zbirki *ImageNet* prednaučene konvolucijske osnove prej omenjenih arhitektur CNN, na katere smo povezali enako zaporedje klasifikacijskih slojev, kot je predstavljeno v tabeli 6.2 v poglavju 3.3.3.

Algoritem 5.1: Zasnova eksperimenta za ovrednotenje metrike *LDM*

Vhod : Učna množica *UM*
Vhod : Prednaučen model *M*
Vhod : Število ovrednotenih rešitev (posameznikov) *p*
Vhod : Število epoh *e*
Rezultat: Vrednosti metrik *NL*, *AUNL* ter *LDM* ter testne vrednosti funkcije izgube *L*

```
1  begin
2      UM_ucna, UM_testna = razdeli(UM) ;        // Podano učno množico razdeli v
         razmerju 80:20
3      RS = inicializiraj_RS();
4      M_zacasni = M;
5      for i = 1, ... p do
6          resitev = RS.pridobi_resitev();
           // Glede na rešitev omogoči uglaševanje posameznih slojev
7          M_zacasni.nastavi_sloje(resitev);
8          for j = 1, ... e do
9              M_zacasni.izvedi_ucenje(UM_ucna);
10             nl= izracunaj_nl();
11             NL.dodaj(nl);
12             aunl = izracunaj_aunl();
13             AUNL.dodaj(aunl);
14             ldm = izracunaj_ldm();
15             LDM.dodaj(izracunaj_ldm());
16         end
17         l = izracunaj_l(M_zacasni, UM_testna);
18         L.dodaj(l);
19         shrani_metrike();
20     end
21 end
```

Tudi v tem delu smo podobno kot pri analizi vpliva izbire uglaševanih slojev konvolucijske nevronske mreže na uspešnost učenja v poglavju 3 uporabili enake nastavitve učnih parametrov za izvedbo eksperimentov, ki so podrobneje predstavljeni v tabeli 3.3 v poglavju 3.3.4.

5.5 Empirična analiza

V tem poglavju predstavimo in analiziramo rezultate, ki smo jih pridobili z izvedbo eksperimentov nad tremi prej omenjenimi izbranimi podatkovnimi zbirkami in arhitekturami CNN. Cilj zastavljenih eksperimentov in analize pridobljenih rezultatov je ugotoviti, ali predstavljena metrika *LDM* pozitivno korelira s končno uspešnostjo posameznega napovednega modela. V primeru, da je zaznana pozitivna korelacija, je prav tako cilj ovrednotiti, kolikim izmed rešitev, ki so se izkazale za neperspektivne s stališča napovedne uspešnosti, je metrika *LDM* kljub temu dopustila nadaljevanje učenja. Nenazadnje je eden izmed ciljev analize tudi določiti smiseln prag vrednosti metrike *LDM* v določeni epohi učenja, ki bo služil kot privzeta vrednost metrike v nadaljnjih eksperimentih.

V prvem delu predstavimo pridobljene rezultate in postopek obdelave le-teh, v drugem delu pa predstavimo korelacijo metrik s končno uspešnostjo napovednega modela z več vidikov ter v tretjem delu ovrednotimo prepustnost metrike *LDM*.

5.5.1 Obdelava rezultatov

Z izvedbo eksperimentov smo pridobili množico rezultatov, ki za vsako ovrednoteno rešitev vsebujejo vrednosti predlaganih na funkciji izgube temelječih metrik (*NL*, *AUNL*, *LDM*) za vsako opravljeno epoho učenja ter vrednost funkcije izgube nad validacijsko podmnožico. Dodatno imamo za vsako ovrednoteno rešitev shranjeno tudi vrednost funkcije izgube nad testno množico; te vrednosti v nadaljevanju uporabimo izključno za namen post-hoc simulacije v poglavju 5.5.3, kjer ovrednotimo prepustnost metrike *LDM*. Vrednosti metrik, izračunanih po vsaki epohi, smo razdelili v množice, kjer so v posamezni množici zgolj vrednosti posameznih metrik v določeni epohi. S ciljem ugotoviti ali so tako pridobljene množice vrednosti metrik v posamezni epohi učenja v korelaciji z vrednostjo funkcije izgube nad validacijsko podmnožico, smo najprej uporabili Shapiro-Wilkov test [136], da bi ugotovili, ali so vrednosti metrik v posamezni množici porazdeljene normalno ali ne. Rezultati Shapiro-Wilkovega testa so pokazali, da v večini primerov ne moremo sprejeti hipoteze H_0 (s statistično značilnostjo $p < 0,01$), ki pravi, da je distribucija vrednosti v množicah normalno porazdeljena. Posledično smo uporabili Spearmanov koeficient korelacije rangov [137], s katerim smo izračunali korelacijo med posameznimi množicami vrednosti

metrik in na tak način ugotovili, katera izmed metrik je najprimernejša za uporabo v namen zaznavanja primernosti izbir slojev CNN z uporabo uglaševanja pri učenju s prenosom znanja.

Spearmanov koeficient korelacije rangov je definiran kot Pearsonov koeficient korelacije med rangi spremenljivk, ki ga lahko formalno izrazimo kot [138]:

$$r_s = \rho_{rg_X, rg_Y} = \frac{cov(rg_X, rg_Y)}{\sigma rg_Y \cdot \sigma rx_Y}, \tag{5.10}$$

kjer je n surovih vrednosti X_i, Y_i pretvorjenih v range rgX_i in rgY_i in kjer ρ označuje Pearsonov koeficient korelacije nad rangi spremenljivk. Kovarianca rangov spremenljivk je označena s $cov(rg_X, rg_Y)$, σ pa označuje standardni odklon rangov spremenljivk.

Zaradi velikega števila izračunov Spearmanovih koeficientov korelacije med posameznimi množicami vrednosti metrik v vsaki epohi smo v nadaljevanju pridobljene vrednosti koeficientov korelacije rangirali, kar nam omogoča celovit pregled korelacije med posameznimi množicami spremenljivk ter množico vrednosti funkcije izgube nad validacijsko množico podatkov v različnih epohah učenja.

5.5.2 Korelacija metrik z vrednostjo funkcije izgube nad validacijsko množico

Na podlagi pridobljenih rezultatov smo izračunali Spearmanove koeficiente korelacije med metrikami *NL*, *AUNL* in *LDM* ter validacijsko funkcijo izgube, ki so predstavljeni v prilogi A. Tako pridobljene vrednosti koeficientov korelacije smo nato rangirali, rezultate rangiranih vrednosti pa v nadaljevanju grafično predstavimo ter analiziramo iz vidika vseh izvedenih eksperimentov ter iz vidika eksperimentov, izvedenih s posamezno arhitekturo CNN.

Analiza nad vsemi eksperimenti

V analizo so vključene vse množice izračunanih koeficientov korelacije v vsaki epohi ne glede na arhitekturo CNN ali ciljno podatkovno zbirko, ki so v nadaljevanju rangirane. Vrednosti povprečnih rangov, ki so jih metrike dosegle v posamezni epohi, vključno s standardnim odklonom, so predstavljene v tabeli 5.1.

Povprečje in standardni odklon rangov sta izračunana nad množico vrednosti koeficientov korelacije, izračunanih nad vsemi 9 eksperimenti (3 arhitekture CNN in 3 podatkovne zbirke) od 2. do 30. epohe. Iz tabele je razvidno, da v drugi epohi vse metrike enakovredno korelirajo z validacijsko vrednostjo funkcije izgube, kar je pričakovano, saj lahko šele v drugi epohi vrednosti predstavljenih metrik prvič

Tabela 5.1 Vrednosti povprečnih rangov metrik ter standardnih odklonov v posamezni epohi.

Epoha	*NL* (povp. ± s.o.)	*AUNL* (povp. ± s.o.)	*LDM* (povp. ± s.o.)
2	**2.00 ± 0,00**	**2,00 ± 0,00**	**2,00 ± 0,00**
3	1,99 ± 0,43	1,89 ± 0,35	**2,11 ± 0,15**
4	1,96 ± 0,38	1,97 ± 0,29	**2,07 ± 0,18**
5	1,93 ± 0,39	1,95 ± 0,42	**2,12 ± 0,15**
6	1,90 ± 0,39	**2,05 ± 0,29**	**2,05 ± 0,19**
7	1,87 ± 0,40	**2,12 ± 0,35**	2,02 ± 0,12
8	1,89 ± 0,42	**2,08 ± 0,37**	2,03 ± 0,16
9	1,94 ± 0,36	1.98 ± 0,31	**2,08 ± 0,11**
10	1,95 ± 0,35	2.00 ± 0,27	**2,05 ± 0,16**
11	1,81 ± 0,43	2.09 ± 0,29	**2,10 ± 0,19**
12	1,89 ± 0,53	2.05 ± 0,34	**2,06 ± 0,25**
13	1,94 ± 0,62	**2,04 ± 0,46**	2,01 ± 0,19
14	1,93 ± 0,64	**2,10 ± 0,43**	1,97 ± 0,32
15	**2,07 ± 0,60**	2,05 ± 0,48	1,88 ± 0,17
16	**2,06 ± 0,57**	1,99 ± 0,45	1,94 ± 0,26
17	**2,08 ± 0,65**	2,08 ± 0,46	1,84 ± 0,24
18	2,00 ± 0,63	**2,03 ± 0,52**	1,97 ± 0,16
19	1,98 ± 0,61	**2,08 ± 0,50**	1,94 ± 0,15
20	1,98 ± 0,61	**2,09 ± 0,48**	1,93 ± 0,16
21	2,02 ± 0,53	**2,06 ± 0,46**	1,92 ± 0,15
22	**2,07 ± 0,59**	2,05 ± 0,47	1,88 ± 0,21
23	**2,04 ± 0,57**	2,01 ± 0,40	1,95 ± 0,25
24	**2,13 ± 0,47**	2,00 ± 0,34	1,87 ± 0,20
25	**2,08 ± 0,44**	**2,08 ± 0,31**	1,84 ± 0,25
26	**2,04 ± 0,54**	1,98 ± 0,46	1,97 ± 0,21
27	**2,14 ± 0,59**	1,97 ± 0,50	1,88 ± 0,22
28	**2,12 ± 0,59**	1,98 ± 0,51	1,89 ± 0,20
29	**2,11 ± 0,61**	2,04 ± 0,48	1,84 ± 0,22
30	**2,17 ± 0,63**	1,96 ± 0,49	1,87 ± 0,19

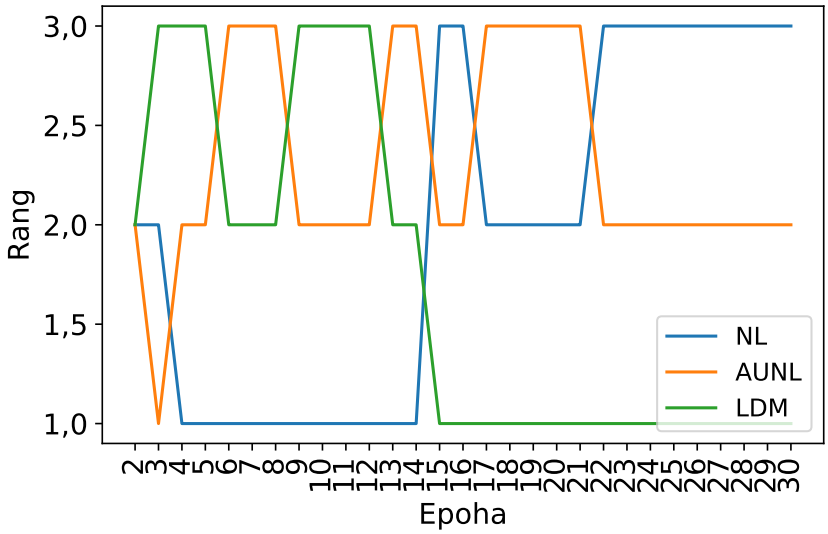
izračunamo. V naslednjih epohah lahko opazimo, da najvišji povprečni rang dosegata metriki *AUNL* ter *LDM*, od 15. epohe naprej pa se izmenjujeta metriki *AUNL* in *NL*.

Če se osredotočimo na nižje epohe, torej zgodnejšo fazo učenja, ter analiziramo dosežene povprečne rezultate od 3. do 15. epohe, vidimo, da je v tem delu metrika *LDM* dosegla najvišje povprečje v 7 epohah. V eni epohi si je najvišje povprečje delila z metriko *AUNL*, ki je v omenjenem intervalu dosegla najvišje povprečje rangov v 5 epohah, medtem ko je metrika *NL* najvišje povprečje dosegla zgolj v eni epohi. Dodatno je iz predstavljenih rezultatov mogoče tudi opaziti, da je standardni odklon povprečne vrednosti doseženih rangov pri metriki *LDM* v vseh epohah najnižji, kar pomeni, da so odstopanja korelacije metrike, ne glede na epoho oz. fazo učenja, majhna.

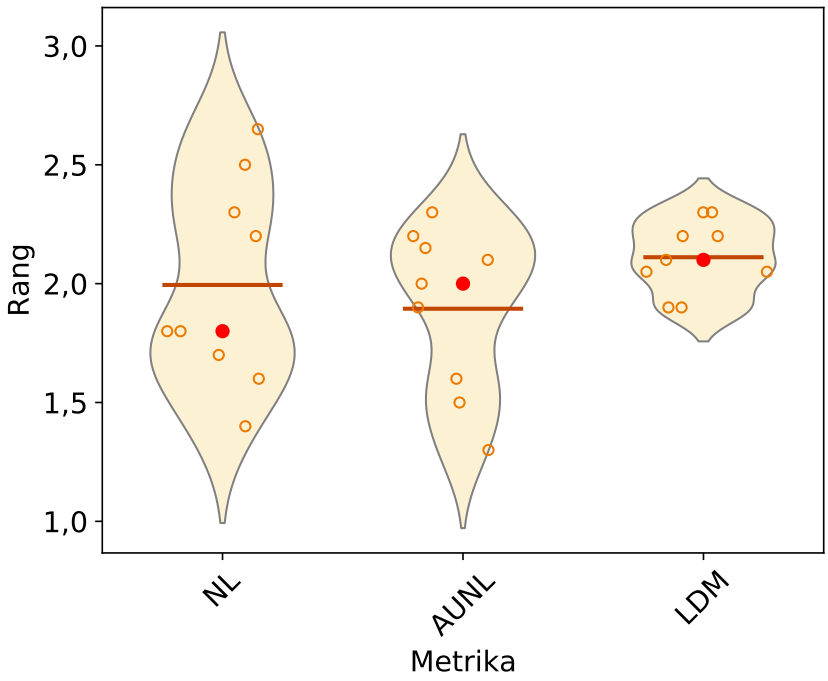
Slika 5.9 prikazuje rangirane povprečne vrednosti Spearmanovih koeficientov korelacije, predstavljene v tabeli 5.1. Kot je razvidno iz grafa, je od 3. do 5. epohe najvišje uvrščena metrika *LDM*, ki od 6. do 8. epohe zasede drugo mesto, prvo pa prevzame metrika *AUNL*, nato pa je ponovno najvišje rangirana do 12. epohe. Od 15. epohe naprej metrika *LDM* zasede najnižji rang, kar je v bistvu ravno nasprotno od metrike *NL*, ki začne višje range dosegati šele v kasnejših epohah. Slednje je sicer smiselno in pričakovano, saj metrika *NL* zgolj predstavlja normalizirano vrednost funkcije izgube na učnih podatkih. Posledično je pri kasnejših epohah višja korelacija takšne metrike s samo validacijsko vrednostjo funkcije izgube pričakovana. Prav tako je zanimivo rangiranje metrike *AUNL*, ki v začetnih fazah doseže tudi najnižji rang, nato pa čez celoten proces učenja variira med prvim in drugim rangom.

Na sliki 5.10 so prikazane vrednosti Spearmanovih koeficientov korelacije posamezne metrike v tretji epohi učenja. Vrednosti so predstavljene v obliki fižolnega grafikona (angl. bean plot) [139], ki je nadgradnja violinskega grafikona (angl. violin plot) [140]. Violinski grafikon si lahko predstavljamo kot nadgradnjo klasičnega grafikona kvantilov (angl. box-and-whisker plot) z dodano verjetnostno gostoto podatkov pri različnih vrednostih, ki je ponazorjena s širino grafikona pri različnih vrednostih. Fižolni grafikon pa prikazu gostote podatkov pri različnih vrednostih doda še izris dejanskih vrednosti. V našem primeru vodoravna rdeča črta na grafikonu predstavlja povprečno vrednost, rdeč krogec označuje mediano, z oranžnimi krogci (fižoli) pa so predstavljene dejanske dosežene vrednosti. Kot je razvidno iz slike, je najvišje povprečje rangov v tretji epohi dosegla metrika *LDM*, ki ima obenem tudi najmanjši standardni odklon, medtem ko je drugo najvišje povprečje rangov dosegla metrika *NL*, ki pa ima najvišji standardni odklon. Najslabše se je v tretji epohi odrezala metrika *AUNL* z nekoliko manjšim standardnim odklonom v primerjavi z metriko *NL*.

Če si pogledamo še vrednosti Spearmanovih koeficientov korelacije posameznih metrik v zadnji, 30. epohi učenja, prikazane na sliki 5.11, lahko vidimo, da je sicer

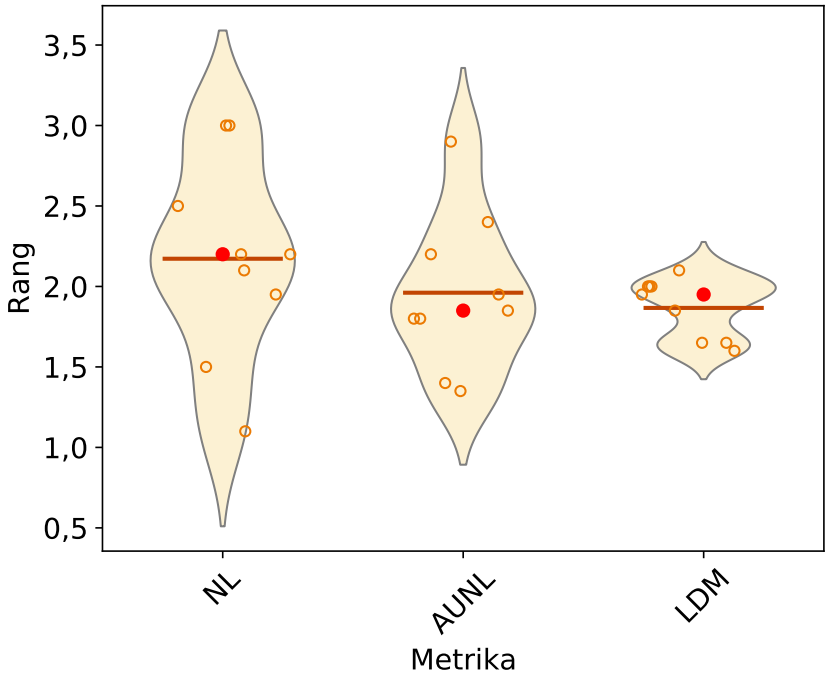


Slika 5.9 Rangi povprečnih vrednosti Spearmanovih koeficientov korelacije v posamezni epohi med metrikami in vrednostjo funkcije izgube nad validacijsko množico.



Slika 5.10 Grafikon vrednosti Spearmanovih koeficientov korelacije v tretji epohi med metrikami in vrednostjo funkcije izgube nad validacijsko množico.

metrika *LDM* dosegla v povprečju najnižji rang, je pa zadržala majhen standardni odklon. Metrika *NL*, ki se je sicer najboljše odrezala, ima obenem tudi najvišji standardni odklon, metrika *AUNL* pa se je uvrstila na drugo mesto z nekoliko manjšim standardnim odklonom kot metrika *NL*.



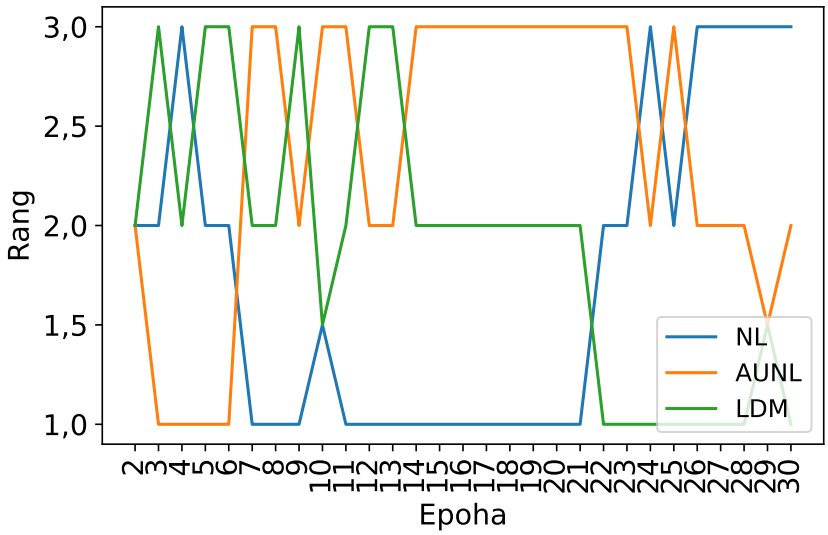
Slika 5.11 Grafikon vrednosti Spearmanovih koeficientov korelacije v trideseti epohi med metrikami in vrednostjo funkcije izgube nad validacijsko množico.

Analiza koeficientov korelacije eksperimentov za posamezno arhitekturo CNN

V prejšnjem delu smo predstavili ter analizirali rezultate in korelacije, izračunane nad vsemi izvedenimi eksperimenti, v nadaljevanju pa podrobneje predstavimo analizo koeficientov korelacije eksperimentov ob uporabi posamezne arhitekture CNN.

Slika 5.12, prikazuje range povprečnih vrednosti Spearmanovih koeficientov korelacije metrik v posamezni epohi za eksperimente, izvedene z uporabo arhitekture *VGG16*. Iz slike je razvidno, da je v prvi fazi učenja (do 15. epohe) metrika *LDM* največkrat dosegla najvišje povprečne vrednosti koeficientov korelacije, medtem ko v drugi fazi učenja prevladuje metrika *AUNL*. Metrika *NL* se je v tem primeru najslabše izkazala, saj je skozi celotno učenje največkrat dosegla najnižji rang.

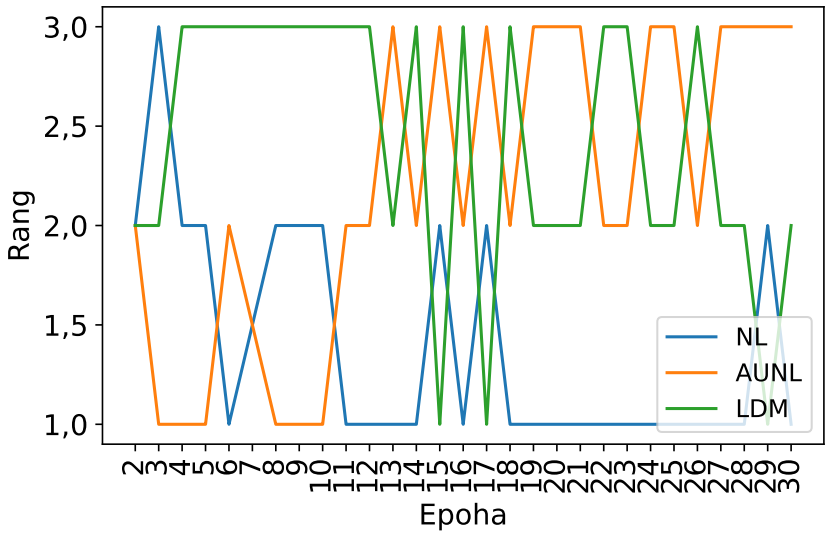
Rangi povprečnih vrednosti Spearmanovih koeficientov korelacije v posamezni epohi učenja za eksperimente, izvedene z uporabo arhitekture *ResNet50*, so prikazani na sliki 5.13. Če se osredotočimo na prvo fazo učenja, je razvidno, da je prevladovala



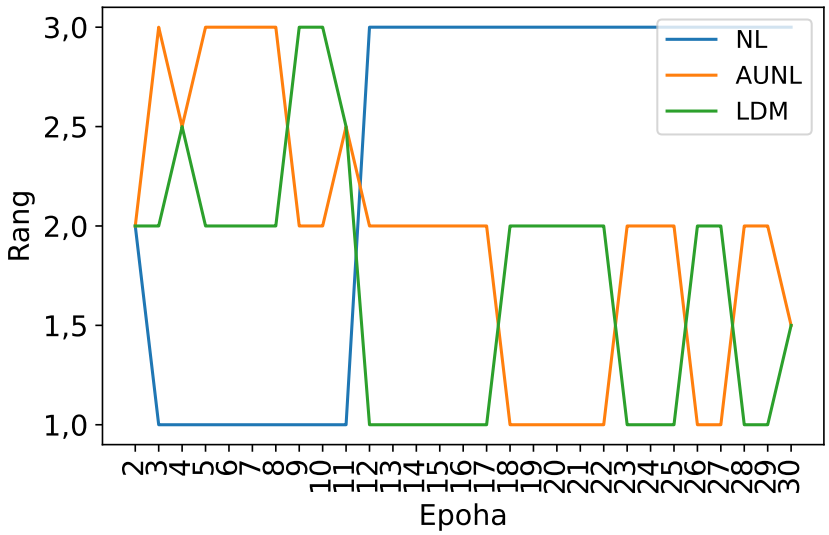
Slika 5.12 Rangi povprečnih vrednosti Spearmanovih koeficientov korelacije v posamezni epohi med metrikami in vrednostjo funkcije izgube nad validacijsko množico ob uporabi arhitekture *VGG16*.

metrika *LDM*, ki je do 15. epohe najvišji rang dosegla kar 11-krat. V splošnem se je najslabše odrezala metrika *NL*, ki pa je, zanimivo, v 3. epohi dosegla najvišji rang.

Na sliki 5.14 so prikazani rangi povprečnih Spearmanovih koeficientov korelacije v posamezni epohi učenja za eksperimente, izvedene z uporabo arhitekture *MobileNet*. Za razliko od preostalih dveh arhitektur CNN, se v tem primeru, metrika *LDM* v začetni fazi učenja ni tako dobro odrezala, saj je do 15. epohe dosegla najvišji rang v zgolj dveh epohah, dvakrat pa si je delila najvišji rang z metriko *AUNl*. Metrika *AUNL* se je v tem pogledu izkazala bolje, saj je dosegla najvišji rang v 5 epohah. Zanimivo pa je metrika *NL*, za razliko od prejšnjih dveh arhitektur, v tem primeru očitno dominirala od 12. epohe učenja naprej in konstanto dosegala najvišji rang.



Slika 5.13 Rangi povprečnih vrednosti Spearmanovih koeficientov korelacije v posamezni epohi med metrikami in vrednostjo funkcije izgube nad validacijsko množico ob uporabi arhitekture *ResNet50*.



Slika 5.14 Rangi povprečnih vrednosti Spearmanovih koeficientov korelacije v posamezni epohi med metrikami in vrednostjo funkcije izgube nad validacijsko množico ob uporabi arhitekture *MobileNet*.

Če povzamemo predstavljeno analizo Spearmanovih koeficientov korelacij, je za naš primer, kjer želimo zaznavati primernost izbir uglaševanih slojev arhitektur CNN v zgodnji fazi učenja, najprimernejša metrika *LDM*. Metrika *LDM* se je v splošnem izkazala kot najuspešnejša v zgodnji fazi učenja, obenem pa je stabilnost korelacije

z validacijsko funkcijo izgube potrdila z najmanjšimi standardnimi odkloni v vseh epohah učenja.

5.5.3 Ovrednotenje prepustnosti metrike LDM

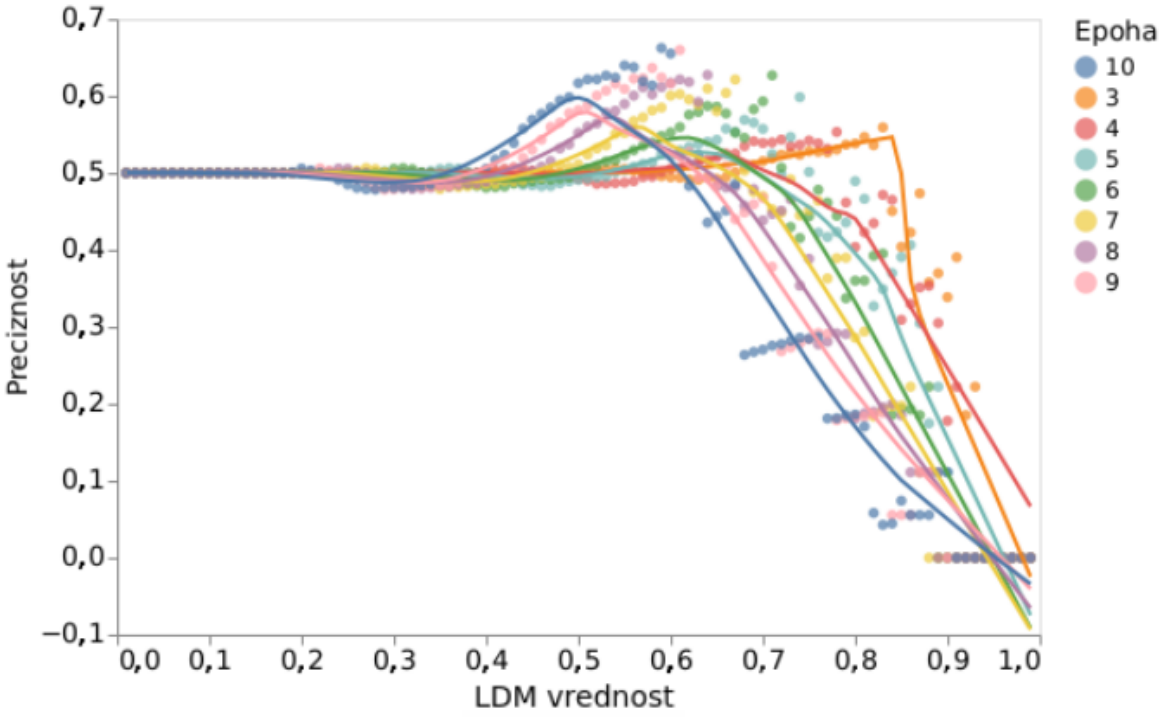
V prejšnjem delu smo kot najprimernejšo metriko za zaznavanje primernosti izbir uglaševanih slojev arhitekture CNN na podlagi opravljene analize izbrali metriko *LDM*. Za dejansko uporabo metrike kot odločevalca, katero izmed izbir uglaševanih slojev je smiselno učiti še naprej, je treba določiti pragove vrednosti metrike v posamezni epohi. Dodatno je smiselno ovrednotiti tudi prepustnost metrike. Cilj metrike je namreč, da v čim zgodnejši fazi učenja zazna izbire uglaševanih slojev, ki po koncu učenja predvidoma ne bodo dosegle visoke uspešnosti, in nam na tak način prihrani čas oz. nam omogoči, da lahko v enakem času ovrednotimo več različnih izbir slojev.

V ta namen smo izvedli simulacijo prepustnosti metrike *LDM*. Prepustnost smo definirali kot preciznost metrike *LDM*, ki smo jo izračunali za vsako epoho na intervalu $[3, 10]$. Pri tem smo prag (vrednosti metrike *LDM*), ki določa, ali nadaljujemo učenje posamezne izbire uglaševanih slojev, pomikali od 0 do 1 s korakom $0,01$. Za namen izračuna preciznosti metrike smo posamezne izbire uglaševanih slojev z njihovimi pripadajočimi vrednostmi funkcije izgube nad testno množico najprej razvrstili od najboljše do najslabše ter jih na polovici razdelili v skupino dobrih oz. slabih izbir. Preciznost smo nato izračunali z uporabo formule:

$$preciznost = \frac{TP}{TP + FP}, \tag{5.11}$$

kjer TP predstavlja število, koliko izbir iz skupine slabih smo zadržali oz. z njimi nismo nadaljevali učenja, FP pa število zadržanih izbir iz skupine dobrih. Cilj je seveda, da metrika zadrži čim več izbir iz skupine slabih ter čim manj iz skupine dobrih. Posledično višja vrednost izračunane preciznosti izkazuje boljšo sposobnost metrike, da predčasno zaustavi učenje izbir uglaševanih slojev, ki dosegajo končno slabšo napovedno uspešnost.

Rezultat opravljene simulacije prepustnosti metrike v posamezni epohi ob različnih vrednostih pragov je prikazan na sliki 5.15 v obliki razsevnega diagrama. Iz slike je mogoče razbrati doseženo prepustnost metrike pri različno nastavljenih vrednostih praga metrike *LDM* v posamezni epohi učenja. Krivulje na sliki predstavljajo glajene vrednosti preciznosti z uporabo metode lokalnega glajenja LOESS (angl. locally estimated scatterplot smoothing) [141]. Metrika najvišjo preciznost v 10 epohi učenja dosega ob vrednosti praga okrog $0,5$. Vrednost praga se za najvišje dosežene preciznosti z zniževanjem epoh postopoma povečuje. Če podrobneje



Slika 5.15 Preciznost metrike *LDM* glede vrednost v posamezni epohi.

analiziramo preciznost metrike v tretji epohi učenja, vidimo zanimiv pojav, ki se razlikuje od ostalih epoh. Preciznost pri tretji epohi namreč s povečevanjem vrednosti praga zvezno raste, po pragu z vrednostjo $0,83$ pa strmo pade, kar je v nasprotju z obnašanjem metrike v ostalih epohah, kjer do dosežene maksimalne preciznosti postopoma narašča ter zatem prične postopoma padati.

S predstavljeno simulacijo nad empirično pridobljenimi rezultati želimo nasloviti tudi problematiko izbire primerne vrednosti praga metrike *LDM* v kombinaciji z izbrano epoho, v kateri želimo preveriti, ali je smiselno nadaljevati z uglaševanjem posamezne izbire slojev. V ta namen smo iz rezultatov izračunov preciznosti, pridobljenih s simulacijo, izbrali za posamezno epoho 10 najboljših. Za te smo v nadaljevanju izračunali razpon pragov, povprečje, standardni odklon ter mediano, rezultati katerih so predstavljeni v tabeli 5.2. Iz tabele je razvidno, da je smiselno vrednost pragu, če želimo smiselnost nadaljevanja učenja preverjati v 3. epohi, nastaviti na vrednost med $0,73$ in $0,83$. Opazimo lahko, da se s povečevanjem epohe, v kateri želimo preverjati smiselnost, vrednost praga postopoma zmanjšuje. Zmanjševanje praga z večanjem epoh je seveda smiselno, saj je v začetnih epohah vrednost metrike *LDM* večja in ob uspešnem učenju skozi epohe postopoma pada.

Tabela 5.2 Najboljših 10 vrednosti (z najvišjo preciznostjo) metrike *LDM* v posamezni epohi.

Epoha	Razpon	Povprečje	Standardni odklon	Mediana
3	0,73–0,83	0,784	0,0304	0,785
4	0,68–0,78	0,726	0,0304	0,725
5	0,63–0,74	0,679	0,0348	0,675
6	0,60–0,71	0,651	0,0365	0,645
7	0,58–0,67	0,625	0,0287	0,625
8	0,55–0,64	0,595	0,0287	0,595
9	0,52–0,61	0,565	0,0287	0,565
10	0,50–0,60	0,547	0,0316	0,545

5.5.4 Diskusija

Z analizo Spearmanovih koeficientov korelacij smo ugotovili, da je za naš primer uporabe, kjer želimo zaznavati primernost izbir uglaševanih slojev arhitektur CNN v zgodnji fazi učenja, najprimernejša metrika *LDM*, ki se je v splošnem izkazala kot najuspešnejša v zgodnji fazi učenja, obenem pa je stabilnost korelacije z validacijsko funkcijo izgube potrdila z najmanjšimi standardnimi odkloni skozi celoten proces učenja.

V nadaljevanju smo z izvedbo simulacije nad empirično pridobljenimi rezultati ovrednotili prepustnost metrike *LDM* v posamezni epohi ob izbranih različnih vrednostih pragov ter izračunali razpon pragov, povprečje, standardni odklon in mediano nad 10 najboljšimi za posamezno epoho. Na tak način smo pridobili izhodišča za smiselno izbiro vrednosti pragov v posamezni epohi preverjanja smiselnosti nadaljevanja učenja. Če povzamemo, je na podlagi analize smiselna vrednost pragu v 3. epohi med $0,73$ in $0,83$, medtem ko je za preverjanje v 10. epohi smiselna izbira vrednosti med $0,5$ in $0,6$. Glede na zastavljeni cilj, da želimo čim prej zaznati neprimerne izbire uglaševanih slojev, smo za eksperimentalni del doktorske disertacije izbrali 3. epoho, v kateri želimo preverjati smiselnost izbire uglaševanih slojev ter vrednost praga $0,73$. S takšno izbiro smo v korist čim zgodnejše zaznave neprimernih izbir uglaševanih slojev žrtvovali doseganje višje preciznosti pri preverjanju v kasnejših epohah. V splošnem je izbira, v kateri epohi želimo preverjati smiselnost nadaljevanja učenja, ter posledično izbira vrednosti pragu, odvisna od zastavljenih ciljev. Če želimo čim hitreje zaznati manj primerne izbire ter posledično privarčevati pri porabljenem številu epoh in smo obenem pripravljeni tvegati, da bomo nehote izločili kakšno izmed potencialno dobrih rešitev, je smiselno izbrati nižje epohe in obratno.

Poglavje 6

Eksperimentalno ogrodje

"No amount of experimentation can ever prove me right; a single experiment can prove me wrong."

- Albert Einstein

Poglavje predstavlja podroben opis ter predstavitev eksperimentalnega ogrodja, ki služi za izvajanje eksperimentov z uporabo predlagane metode *DEFT* z in brez uporabe metrike *LDM*. Poleg predstavitve uporabljenih podatkovnih zbirk poglavje vsebuje tudi opis nastavitev parametrov metode *DEFT* in metrike *LDM*, uporabljene metode vrednotenja, metrike in statistične metode ter predstavitev eksperimentalnega okolja.

6.1 Podatkovne zbirke in priprava podatkov

Z namenom temeljitega ovrednotenja delovanja predstavljene metode prilagodljivega uglaševanja slojev konvolucijskih nevronskih mrež pri strojnem učenju s prenosom znanja z in brez uporabe metrike *LDM* smo izbrali tri čim bolj raznovrstne slikovne podatkovne zbirke z različnim številom kategorij oz. razredov ter primerkov posameznega razreda znotraj posamezne podatkovne zbirke. Zaradi karseda objektivnega ovrednotenja predlagane metode *DEFT* ter metrike *LDM* in v izogib potencialnemu pobegu informacij (angl. data leakage) podatkovnih zbirk iz predhodnih analiz nismo ponovno uporabili. Lastnosti izbranih podatkovnih zbirk so predstavljene v naslednjih poglavjih.

6.1.1 Osteosarcoma data

Podatkovna zbirka *Osteosarcoma data from UT Southwestern/UT Dallas for Viable and Necrotic Tumor Assessment*, predstavljena v delu [142], zajema s hemotoksinom in

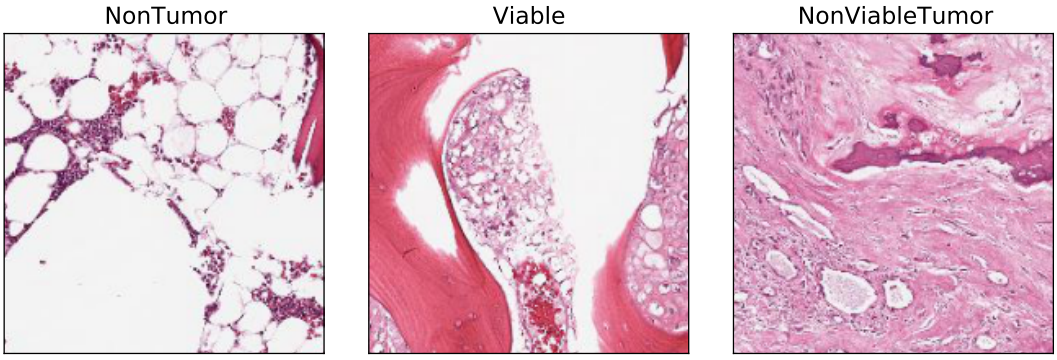
eosinom obarvane slike osteogenega sarkoma, ki je najpogostejši primarni rak kosti pri otrocih. Slike so bile zbrane s strani kliničnih znanstvenikov medicinskega centra University of Texas Southwestern Medical Center. Za kreiranje te podatkovne zbirke so bili uporabljeni arhivski primerki histoloških diapozitivov 50 pacientov, zdravljenih v otroškem medicinskem centru (Children's Medical Center) v Dallasu med letoma 1995 in 2015. Diapozitivi so bili digitalizirani tako, da je bilo 942 tkivnih rezin na stekelcu posnetih s tehniko zajema celotnega diapozitiva oz. celotne tkivne rezine na stekelcu (angl. whole slide imaging, WSI). Izmed 942 posnetkov WSI sta nato dva patologa ročno izbrala 40 takšnih, ki prikazujejo heterogenost tumorja in želene lastnosti. Iz vsakega izmed 40 posnetkov je bilo izbranih naključnih 30 slikovnih ploščic (podpodročij posameznega WSI) velikosti 1024×1024 slikovnih točk, kar se je odražalo v 1.200 slikovnih ploščicah. Izmed teh jih je bilo 56 izločenih, saj so prikazovale dele brez tkiva, zaznambe črnila ipd. Preostalih 1.144 slik je bilo označenih s strani dveh ekspertov, izmed katerih je vsak označil polovico slik z eno izmed treh označb, in sicer *NonTumor*, *Viable* in *NonViableTumor* [143].

Primerki posameznih razredov slik so prikazani na sliki 6.1, porazdelitev števila primerkov med posameznimi razredi pa je razvidna iz grafa na sliki 6.2. Slike, označene z oznako *ViableTumor*, prikazujejo vitalni tumor, kjer so jedra celic gosto združena, medtem ko slike, označene z *NonViableTumor* (nekrotični tumor), prikazujejo razpadla oz. razkrajajoča se jedra z manj intenzivno barvo kot pri vitalnem tumorju. V medicinskem smislu nekrotični tumor označuje umirajoče dele tumorskih celic, medtem ko vitalni tumor označuje slike tkiva, kjer so tumorske celice sposobne normalne rasti.

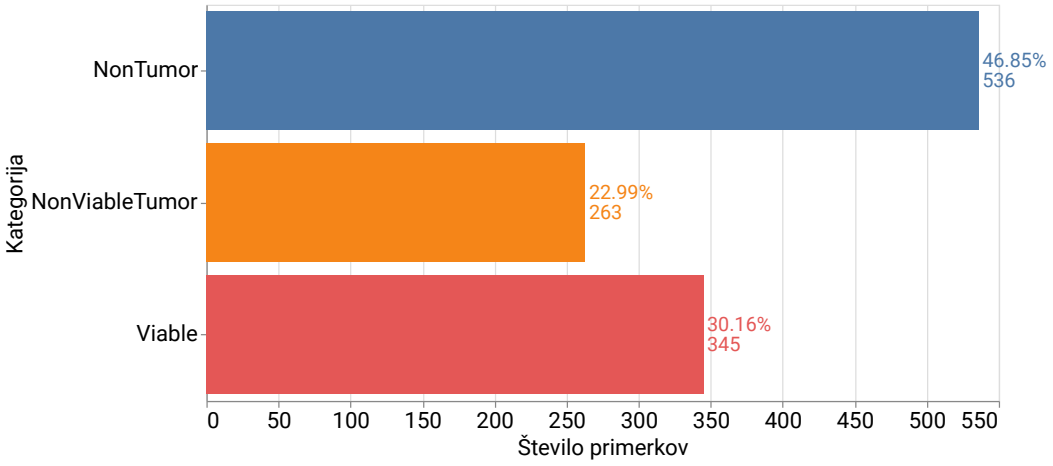
6.1.2 Športne slike

Podatkovna zbirka športnih slik je najnovejša izmed uporabljenih podatkovnih zbirk, predstavljena v delu [28], in zajema slike iger štirih različnih športov. Slike so bile samodejno zbrane iz različnih spletnih repozitorijev, zaradi česar so v različnih velikostih. Tako pridobljene slike so bile ročno preverjene in označene. Z ročnim preverjanjem vseh slik so avtorji zagotovili, da so slike dokaj sorazmerno porazdeljene med razrede in da zbrane slike ne vsebujejo zgolj raznih logotipov, risanih slik ipd., kar bi potencialno lahko vplivalo na uspešnost učenja ter razpoznavo. Na ta način je bilo zbranih skupno 1.359 slik, ki so razvrščene v štiri razrede: *american-football*, *field-hockey*, *rugby* ter *soccer*.

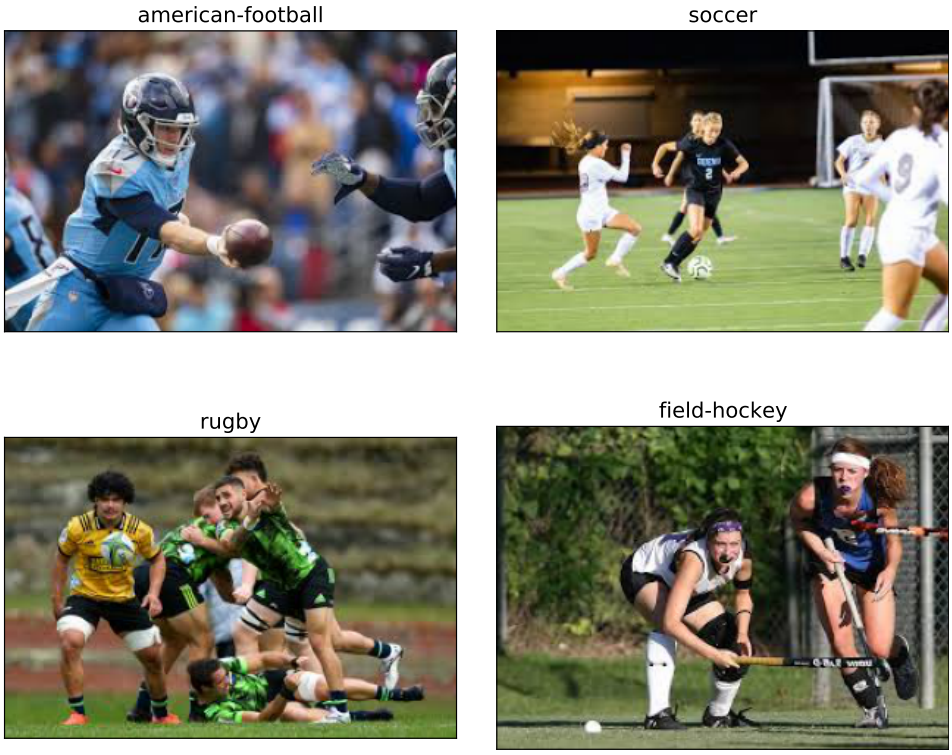
Primerki posameznih razredov slik so prikazani na sliki 6.3, porazdelitev števila primerkov med posameznimi razredi pa je razvidna iz grafikona na sliki 6.4. Slike z oznako *american-football* prikazujejo igro ameriškega nogometa, medtem ko slike označene z oznako *soccer* označujejo slike, ki prikazujejo igro klasičnega nogometa,



Slika 6.1 Primerki slik različnih kategorij podatkovne zbirke histoloških slik *Osteosarcoma*.



Slika 6.2 Porazdelitev primerkov posamezne kategorije podatkovne zbirke histoloških slik *Osteosarcoma*.

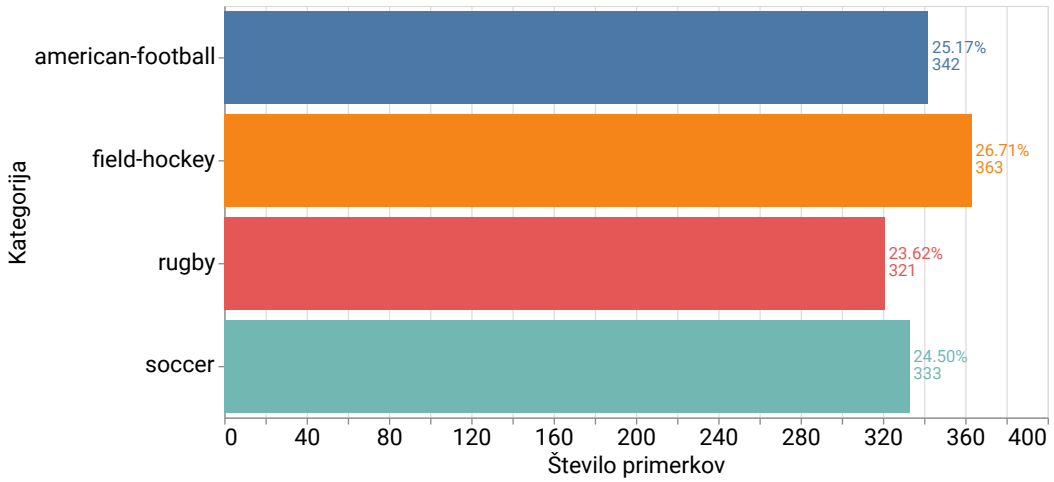


Slika 6.3 Primerki slik različnih kategorij podatkovne zbirke *Športne slike*.

kot ga poznamo pri nas. V kategoriji *rugby* so zbrane slike, ki prikazujejo igro ragbija, slike z oznako *field-hockey* pa prikazujejo igro hokeja na travi.

6.1.3 Rentgenske slike COVID-19

Podatkovna zbirka rentgenskih slik za odkrivanje bolezni COVID-19 je bila v osnovi pripravljena s strani avtorjev [144], ki so jim jo raziskovalci iz vsega sveta, v času od prve objave podatkovne zbirke do danes, pripomogli izboljšati tako kvantitativno kot kvalitativno. Ker se podatkovna zbirka nenehno dopolnjuje, velja omeniti, da smo jo pridobili 20. januarja 2021. Na ta datum je zajemala 929 rentgenskih slik različnega izvora, zaradi česar so zbrane slike v različnih formatih in velikostih. Pri pregledu podatkovne množice lahko opazimo, da je večina zbranih slik označena z oznako *COVID-19*, ostale oznake pa so v manjšini, kot je razvidno iz tabele 6.1.

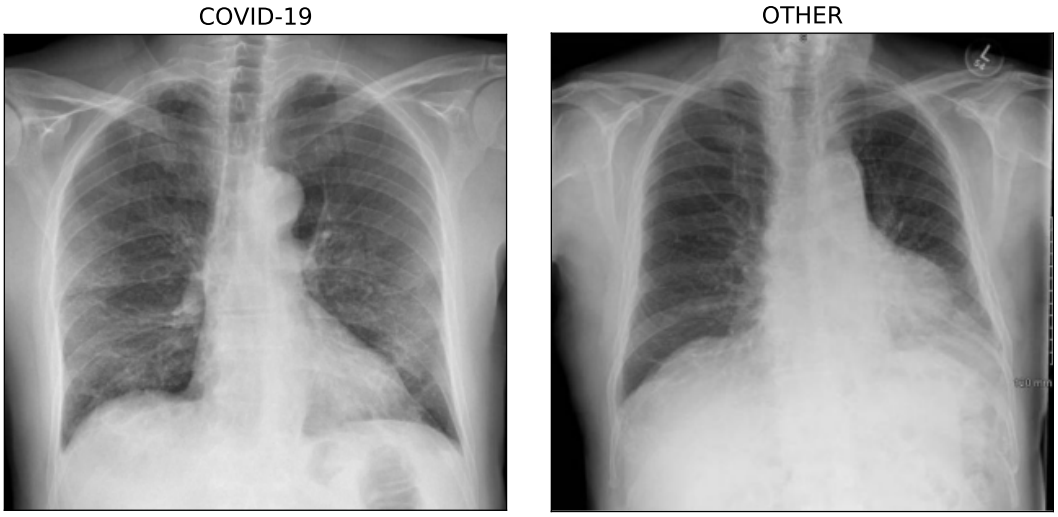


Slika 6.4 Porazdelitev primerkov posamezne kategorije podatkovne zbirke *Športne slike*.

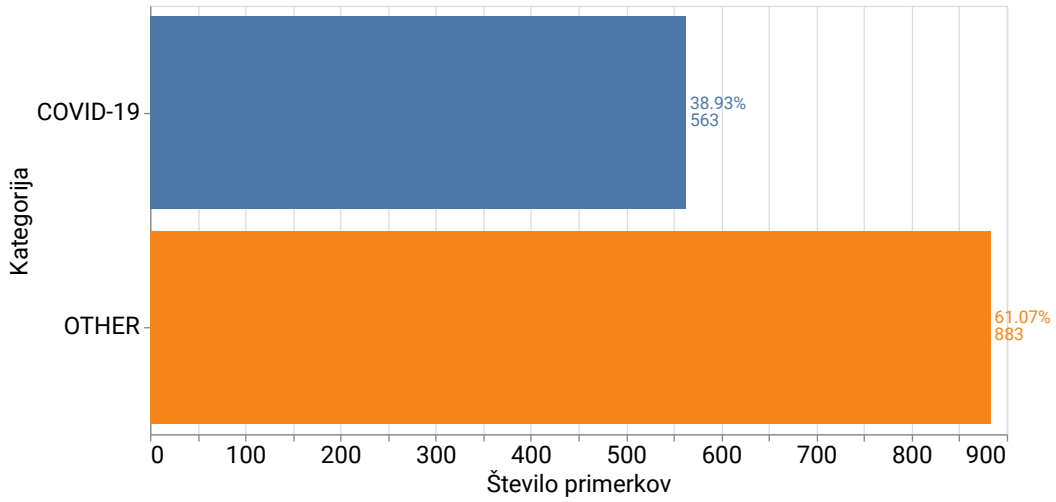
Tabela 6.1 Porazdelitev primerkov med razredi podatkovne zbirke rentgenskih slik *COVID-19*.

Razred	Število primerkov
COVID-19	563
Pneumonia	81
SARS	16
Pneumocystis	30
Streptococcus	22
No finding	22
Chlamydophila	3
E.Coli	4
Klebsiella	10
Legionella	10
Unknown	1
Lipoid	13
Varicella	6
Bacterial	4
Mycoplasma	11
Influenza	5
todo	83
Tuberculosis	18
H1N1	2
Aspergilliosis	2
Herpes	3
Aspiration	1
Nocardia	8
MERS-CoV	10
MRSA	1

Za namen razpoznave bolezni covid-19 iz rentgenskih slik smo obstoječe manjšinske kategorije, z izjemo kategorije *todo*, ki smo jo odstranili, združili v novo kategorijo, imenovano *OTHER*. Na tak način smo torej pridobili podatkovno zbirko s skupno 846 primerki rentgenskih slik, izmed katerih jih je 563 označenih z oznako *COVID*, preostalih 283 pa z oznako *OTHER*. Ker na tak način pripravljena podatkovna zbirka kljub vsemu ni preveč uravnotežena, smo iz obstoječe podatkovne zbirke RSNA Pneumonia Detection Challenge [145] naključno izbrali še 600 primerkov rentgenskih slik pljuč, označenih z oznako *Normal*, in jih dodelili k že obstoječim primerkom kategorije *OTHER*. Tako pripravljena podatkovna zbirka je še vedno relativno neuravnotežena, saj je sedaj več primerkov kategorije *OTHER*. Za takšno razširitev predstavljene podatkovne zbirke smo se odločili z namenom povečanja skupnega števila primerkov. Tako sestavljena podatkovna zbirka *COVID-19* vsebuje 1.446 primerkov slik, razmerje porazdelitve primerkov posamezne kategorije je predstavljeno v obliki stolpčnega diagrama na sliki 6.6, primerka posamezne kategorije pa sta predstavljena na sliki 6.5.



Slika 6.5 Primerki slik različnih kategorij podatkovne zbirke rentgenskih slik *COVID-19*.



Slika 6.6 Porazdelitev primerkov posamezne kategorije podatkovne zbirke rentgenskih slik *COVID-19*.

6.1.4 Predobdelava podatkov

Vsem slikam iz predstavljenih podatkovnih zbirk smo spremenili velikost na 224 × 224 slikovnih točk. Za to velikost smo se odločili, ker je to enaka velikost oz. dimenzija, kot je privzeto definirana na vhodnem sloju uporabljenih globokih arhitektur CNN. Poleg spremembe velikosti smo po postopku predstavljenem v poglavju 3.3.2, slike tudi normalizirali, s čimer je mogoča hitrejša konvergenca pri učenju napovednega modela.

6.2 Konvolucijske nevronske mreže in nastavitve

Z namenom temeljitega ovrednotenja razvite metode in metrike smo izbrali tri različne arhitekture, ki se razlikujejo tako po številu konvolucijskih slojev kot po številu enot v posameznem sloju ter mehanizmih, uporabljenih za ekstrakcijo značilnic. Enako, kot pri predhodnih eksperimentih smo tudi v tem primeru izbrali tri izmed pogosteje uporabljanih globokih arhitektur CNN: *VGG16*, *ResNet50* in *MobileNet*. Podrobnosti vsake izmed izbranih arhitektur so predstavljene v poglavju 2.4.3.

Za potrebe izvedbe eksperimentalne študije smo uporabili na podatkovni zbirki *ImageNet* prednaučene konvolucijske osnove omenjenih arhitektur CNN, na katere smo povezali zaporedje klasifikacijskih slojev, predstavljenih v tabeli 6.2.

Za razliko od prehodno predstavljenih eksperimentov smo v tem primeru dodali dodatne klasifikacijske sloje z namenom odprave učinka prekomernega prileganja v fazi učenja, saj so podatkovne zbirke, uporabljene v tem eksperimentalnem delu, manj obsežne in imajo manj številčne razrede. Za prvim polno povezanim slojem smo

Tabela 6.2 Zaporedje klasifikacijskih slojev.

Sloj	Lastnosti
Združevalni sloj (globalno povp.)	-
Sloj normalizacije paketov	-
Polno povezan sloj	1024 nevronov, aktivacijska funk. ReLU
Izpustni sloj	verjetnost osipa 0,8
Polno povezan sloj	1024 nevronov, aktivacijska funk. ReLU
Izpustni sloj	verjetnost osipa 0,5
Polno povezan sloj	3/4/2 nevronov, aktivacijska funk. Softmax

tako dodali še prvi osipni sloj z verjetnostjo osipa 0,8 ter nato še en polno povezan sloj s 1024 enotami in aktivacijsko funkcijo ReLU. Zatem smo dodali drugi osipni sloj z vrednostjo verjetnosti osipa 0,5 ter izhodni sloj, katerega število enot je enako številu razredov posamezne podatkovne zbirke. Uporabili smo aktivacijsko funkcijo softmax.

Pri eksperimentih, označenih z *baseline*, je bila izbira uglaševanih slojev opravljena ročno na podlagi priporočil [43] in preteklih izkušenj [39, 38, 28]. V primeru eksperimentov z oznako *baseline*, ki uporabljajo arhitekturo *VGG16*, so bili za uglaševanje izbrani sloji zadnjega konvolucijskega bloka. Pri uporabi arhitekture *ResNet50* so bili za uglaševanje prav tako izbrani sloji znotraj zadnjega konvolucijskega bloka, medtem ko so bili pri uporabi arhitekture *MobileNet* izbrani sloji znotraj zadnjih dveh parov globinske konvolucije (Conv dw) in točkovne konvolucije (Conv pw).

6.3 Nastavitve parametrov metode in metrike

Izbira vrednosti nastavitev učnih parametrov v eksperimentu uporabljenih metod je predstavljena v tabeli 6.3. Za vse uporabljene metode učenja globokih CNN smo uporabili enak optimizator, in sicer Adam s stopnjo učenja $1 \cdot 10^{-4}$. Prav tako smo pri vseh metodah velikost učnega paketa nastavili na 32. Število epoh smo za vse metode, razen za metodo klasičnega učenja, nastavili na 30. Ker metode, ki se poslužujejo uglaševanja pri učenju s prenosom znanja, v splošnem ne zahtevajo velikega števila epoh, smo zaradi objektivne primerjave metodi klasičnega učenja nastavili število epoh na 100. Pri vseh metodah je uporabljen mehanizem zgodnje prekinitve učenja, ki v primeru, da v treh zaporednih epohah ni zaznati izboljšanja funkcije izgube, učenje predčasno zaustavi. Pri metodi *DEFT* z uporabo metrike *LDM* je epoha, v kateri preverjamo, ali nadaljujemo učenje ali ne, nastavljena na 3, vrednost praga pa na $0,73$. Z drugimi besedami, metoda *DEFT* z uporabo metrike *LDM* bo zaustavila

učenje posamezne izbire uglaševanih slojev v primeru, da je v tretji epohi učenja vrednost metrike večja od $0,73$. Izbrano vrednost praga smo določili na podlagi izvedenega vrednotenja propustnosti metrike *LDM* v poglavju 5.5.3, podrobnejša utemeljitev pa je predstavljena v poglavju 5.5.4.

Tabela 6.3 Nastavitev učnih parametrov uporabljenih metod.

Parameter	klasično učenje	uglaševanje	*DEFT*	*DEFT* z *LDM*
Optimizator	Adam	Adam	Adam	Adam
Stopnja učenja	$1 \cdot 10^{-4}$	$1 \cdot 10^{-4}$	$1 \cdot 10^{-4}$	$1 \cdot 10^{-4}$
Velikost mini paketa	32	32	32	32
Epoha	100	30	30	30

Ker metoda *DEFT* temelji na algoritmu diferencialne evolucije, je treba primerno nastaviti tudi vrednosti algoritma DE. Izbira vrednosti za velikost populacije, skalirni faktor ter verjetnost križanja imajo velik vpliv na učinkovitost optimizacije. Vrednosti za parametra skalirni faktor in verjetnost križanja smo nastavili glede na priporočila iz literature [106], medtem ko smo velikost populacije in število ovrednotenj nastavili na podlagi predhodnih izkušenj [38, 29, 28] pri reševanju podobnih problemov. V tabeli 6.4 so predstavljene izbrane vrednosti parametrov metode *DEFT*. Dimenzija problema je nastavljena dinamično glede na uporabljeno arhitekturo CNN, kjer N predstavlja število slojev znotraj konvolucijskega bloka uporabljene arhitekture. Številu N je treba prišteti 1, saj je dodatna vrednost uporabljena kot prag, ki določa, kako se bo izvedla preslikava vsake pridobljene rešitve. Velikost populacije smo nastavili na 10, skalirni faktor na $0,5$ ter verjetnost križanja na $0,9$. Čeprav je velikost populacije v našem primeru nastavljena na manjšo od splošno priporočljive [106], se je v naših dosedanjih raziskavah [38, 28, 104] takšna izbira izkazala z uspešno. Število ovrednotenj kriterijske funkcije, ki določa, koliko izbir slojev bomo ovrednotili, smo nastavili na 50.

Tabela 6.4 Nastavitev parametrov metode *DEFT*.

Parameter	Vrednost
Dimenzija problema D	$N + 1$
Velikost populacije Np	10
Skalirni faktor F	0,5
Verjetnost križanja CR	0,9
Število ovrednotenj kriterijske funkcije	50

6.4 Metoda vrednotenja in metrike

Z namenom objektivnega vrednotenja uspešnosti predstavljene metode *DEFT* z metriko *LDM* v primerjavi z ostalimi metodami smo za vse eksperimente uporabili metodologijo 10-kratnega prečnega preverjanja, ki je v mnogo raziskavah predlagana kot najustreznejša [59, 146]. V procesu izvedbe 10-kratnega prečnega preverjanja smo izračunavali najpogosteje uporabljano metriko klasifikacijske uspešnosti – točnost. Poleg točnosti smo izračunavali tudi *AUNP*, ki temelji na metriki *AUC* in je namenjena za vrednotenje uspešnosti večrazredne klasifikacije, kot tudi vrednost Kappa. Sama metodologija vrednotenja in uporabljene metrike so predstavljene v poglavju 2.3.

6.5 Uporabljene statistične metode

Statistično analizo izvedemo v kombinaciji s pristopom preverjanja statistične značilnosti ničelne domneve (angl. null hypothesis significance testing, NHST) ter z uporabo pristopa, ki temelji na uporabi Bayesove analize. Najprej, po priporočilih Demšarja [59], preverimo porazdelitev podatkov (rezultatov) z uporabo Shapiro-Wilkovega testa ter v primeru, da porazdelitev podatkov ni normalna, uporabimo Friedmanov test [147], sicer pa test ANOVA za ponovljene meritve [148]. V izogib statistični napaki tipa I, uporabimo Holm-Bonferronijevo korekcijo [149] za zniževanje statistične značilnosti. Na tak način lahko zaznamo statistično značilne razlike med primerjanimi metodami glede na izbrano metriko. Podrobnejšo analizo primerjav uspešnosti med posameznimi metodami izvedemo z uporabo Bayesove analize, ki nam omogoča vrednotenje verjetnosti posamezne hipoteze. Pri statistični analizi uspešnosti klasifikatorjev nas namreč najpogosteje zanima, kakšna je verjetnost, da je en klasifikator boljši od drugega. Metode, ki temeljijo na pristopu NHST, med katere spada tudi Wilcoxonov test predznačenih rangov, nam odgovora na zastavljeno vprašanje ne morejo podati. Da bi pridobili odgovor na zastavljeno vprašanje, je treba poseči po metodah Bayesove analize.

Bayesovo analizo rezultatov izvedemo po priporočilih Benavolija in soavtorjev [150]. V raziskavi avtorji predstavijo pristop z uporabo Bayesove analize, saj je po njihovem mnenju ta za primerjavo klasifikatorjev strojnega učenja bolj smiselna kot klasični pristop. Klasični pristop temelji na preverjanju statistične značilnosti ničelne domneve. Ker pri strojnem učenju ni verjetnega razloga za domnevo, da dva različna klasifikatorja dosegata enako točnost, tak pristop morda ni najprimernejši. Prvi korak pri Bayesovi analizi je vzpostavitev opisnega matematičnega modela podatkov. V primeru parametričnega modela je to matematični model funkcije verjetja, ki zagotavlja verjetnost opazovanih podatkov za vsako kandidatno vrednost parametra

oz. parametrov $p(x|\theta)$. Drugi korak je vzpostavitev pričakovane kredibilnosti za vsako vrednost parametrov pred opazovanjem podatkov – predhodna (priorna) porazdelitev $p(\theta)$. Tretji korak je uporaba Bayesovega teorema, s pomočjo katerega lahko, glede na znano predhodno porazdelitev, izračunamo matematično verjetje opazovanih podatkov za vsako kandidatno vrednost parametrov ter v nadaljevanju z uporabo enačbe 6.1 izračunamo posteriorno porazdelitev $p(\theta|x)$ glede na dane podatke [150].

$$p(\theta|x) = \frac{p(\theta) \cdot p(x|\theta)}{p(x)} \tag{6.1}$$

V primeru klasifikatorjev posteriorna porazdelitev opisuje povprečne razlike med točnostjo dveh primerjanih klasifikatorjev. S poizvedbami nad posteriorno porazdelitvijo lahko ovrednotimo verjetnost domnev. Formalno lahko verjetnost, da je klasifikator K_1 boljši od K_2 ($P(K_1 > K_2)$), izračunamo z integralom posteriorne porazdelitve od nič do neskončno ali enakovredno s posteriorno verjetnostjo, da je povprečje razlik točnosti med klasifikatorjema K_1 in K_2 večje od nič. Posledično lahko izračunamo tudi verjetnost $P(K_2 > K_1) = 1 - P(K_1 > K_2)$ [150].

Za ugotovitev, ali sta primerjana klasifikatorja praktično enaka, Benavoli in soavtorji [150] predlagajo vpeljavo intervala regije praktične enakosti (angl. region of practical equivalence, ROPE) [151]. Z določitvijo takšnega intervala definiramo območje, znotraj katerega so povprečne razlike med primerjanima klasifikatorjema opredeljene kot praktično enake. Ko imamo interval ROPE definiran, lahko z uporabo posteriorja izračunamo naslednje verjetnosti [150]:

- $P(K_1 \ll K_2)$: posteriorna verjetnost, da je povprečje razlik točnosti praktično negativno. Z drugimi besedami, izračunamo integral posteriorja na intervalu $(-\infty, 0)$.
- $P(K_1 = K_2)$: posteriorna verjetnost, da je povprečje razlik točnosti dveh klasifikatorjev praktično enako. Z drugimi besedami, izračunamo integral posteriorja čez interval ROPE.
- $P(K_1 \gg K_2)$: posteriorna verjetnost, da je povprečje razlik točnosti praktično pozitivno, kar pomeni, da izračunamo integral na intervalu $(0,01, \infty)$.

Tako lahko s posteriorno porazdelitvijo z uporabo intervala ROPE ocenimo posteriorno verjetnost smiselne ničelne hipoteze (področje znotraj intervala ROPE) in podpremo značilne razlike med primerjanimi klasifikatorji, ki imajo tudi praktični pomen (področje izven intervala ROPE).

Bayesovo analizo za primerjavo parov klasifikatorjev nad več podatkovnimi množicami izvedemo z uporabo metode Bayesovega hierarhičnega koreliranega t-testa [152].

6.6 Eksperimentalno okolje

Vse uporabljene obstoječe in novo razvite metode so bile implementirane v programskem jeziku Python ob uporabi podpornih knjižnic. Knjižnice scikit-learn[153], Pandas [154] in Numpy [155] so bile uporabljene kot splošne podporne knjižnice, NiaPy [156] za optimizacijska algoritma RS in DE, Keras [157] in Tensorflow [158] za implementacijo in učenje CNN, PyCM [159] za izračun klasifikacijskih metrik, baycomp [150] in scipy [160] za statistično analizo rezultatov ter Altair [161] in matplotlib [162] za izris grafov.

Vsi eksperimenti so bili izvedeni z uporabo superračunalnika HPC Maister, pri čemer je imel vsak izvedeni eksperiment zagotovljenih osem procesorskih jeder s 64 GB pomnilnika in dvema grafičnima karticama Nvidia Tesla V100, vsako z 32 GB namenskega delovnega pomnilnika.

Poglavje 7

Analiza rezultatov

"A new idea must not be judged by its immediate results."

- Nikola Tesla

Poglavje predstavlja rezultate izvedenih eksperimentov. Pridobljene rezultate primerjanih metod primerjamo s stališča klasifikacijske uspešnosti in porabe virov. Poleg omenjenih primerjav rezultatov poglavje vsebuje tudi osnovno statistično analizo z uporabo klasičnega pristopa ter podrobno analizo z uporabo metode Bayesovega hierarhičnega koreliranega t-testa.

V eksperimentalnem delu smo izvedli sedem eksperimentov z uporabo treh različnih arhitektur CNN nad tremi različnimi podatkovnimi zbirkami. Vse vrednosti metrik v nadaljevanju predstavljajo povprečne vrednosti metrik, pridobljenih z uporabo metodologije 10-kratnega prečnega preverjanja. Rezultate izvedenih eksperimentov z uporabljenimi metodami v nadaljevanju označujemo sledeče:

- *klasicna*: Predstavlja rezultate metode klasičnega učenja globokih CNN.
- $uglasevanje_{vse}$: Predstavlja rezultate metode, kjer uporabljamo prednaučen model CNN ter uglašujemo vse sloje.
- $uglasevanje_{rocno}$: Predstavlja rezultate metode, kjer uporabljamo prednaučen model CNN in uglašujemo zgolj ročno izbrane sloje.
- $DEFT_{st-resitev}$: Predstavlja rezultate predstavljene metode *DEFT*, pri čemer zaustavitveni pogoj predstavlja maksimalno število ovrednotenih rešitev.
- $DEFT_{st-epoh}$: Predstavlja rezultate predstavljene metode *DEFT*, pri čemer zaustavitveni pogoj predstavlja maksimalno število epoh.

- $DEFT-LDM_{st-resitev}$: Predstavlja rezultate predstavljene metode *DEFT* z uporabo metrike *LDM*, pri čemer zaustavitveni pogoj predstavlja število ovrednotenih rešitev.
- $DEFT-LDM_{st-epoh}$: Predstavlja rezultate predstavljene metode *DEFT* z uporabo metrike *LDM*, pri čemer zaustavitveni pogoj predstavlja maksimalno število epoh.

7.1 Rezultati eksperimentov

V tem poglavju podrobno predstavimo rezultate izvedenih eksperimentov vseh primerjanih metod. Pridobljeni rezultati so predstavljeni po delih in v vsakem izmed delov so predstavljeni rezultati eksperimentov, izvedenih nad posamezno podatkovno zbirko.

Rezultati eksperimentov, izvedenih nad podatkovno zbirko *Osteosarcoma*

V tabeli 7.1 so prikazani vsi rezultati merjenih metrik primerjanih metod nad podatkovno zbirko *Osteosarcoma*. Rezultati v tabeli so razdeljeni glede na uporabljeno arhitekturo CNN ter prikazani v obliki povprečne vrednosti posamezne metrike čez vseh 10 ponovitev izvedenega 10-kratnega prečnega preverjanja. Odebeljeno so zapisane vrednosti posameznih metrik, kjer je bile ob uporabi posamezne arhitekture CNN dosežena najboljša vrednost.

Če se osredotočimo na rezultate, pridobljene z uporabo arhitekture *VGG16*, vidimo, da je bila iz stališča klasifikacijske točnosti najuspešnejša metoda $DEFT_{st-resitev}$, medtem ko je bila najmanj uspešna *klasicna* metoda, ki je obenem za učenje porabila najmanj epoh. V preostalih klasifikacijskih metrikah (*AUNP*, *Kappa*) se je prav tako najbolje odrezala metoda $DEFT_{st-resitev}$. Če primerjamo rezultate metod $DEFT_{st-resitev}$ in $DEFT_{st-epoh}$, ugotovimo, da je v vseh metrikah prva dosegla boljše rezultate kot $DEFT_{st-epoh}$, kar je presenetljivo, saj je imela slednja možnost ovrednotiti več različnih izbir uglaševanih slojev, ker njeno izvajanje ni bilo omejeno s številom ovrednotenih izbir slojev, ampak z maksimalnim številom porabljenih epoh. Podobno ugotovimo tudi, ko primerjamo metodi $DEFT-LDM_{st-resitev}$ in $DEFT-LDM_{st-epoh}$. Če primerjamo metodo $DEFT_{st-resitev}$ z metodo $DEFT-LDM_{st-resitev}$, vidimo, da je klasifikacijska uspešnost slednje primerljiva z metodo $DEFT_{st-resitev}$, medtem ko je za dosego primerljive uspešnosti porabila tako manj časa kot tudi epoh, kar nakazuje, da smo z uporabo metrike dosegli zmanjšanje porabe števila epoh metode *DEFT*. Pričakovano je klasifikacijska uspešnost metod *klasicna*, $uglasevanje_{vse}$ in $uglasevanje_{rocno}$ v splošnem nižja

kot pričakovana uspešnost metod *DEFT*. S stališča porabljenega časa pa so, pričakovano, prej omenjene metode boljše.

Tabela 7.1 Rezultati izvedenih eksperimentov nad podatkovno zbirko *Osteosarcoma*.

CNN	Metoda	Čas [s]	Točnost	AUNP	Kappa	Epoha
VGG16	*klasicna*	92,942	0,568	0,619	0,348	**13,900**
	$uglasevanje_{vse}$	72,579	0,822	0,848	0,706	19,222
	$uglasevanje_{rocno}$	**60,241**	0,895	0,918	0,834	23,222
	$DEFT_{st-resitev}$	2837,866	**0,907**	**0,931**	**0,856**	830,200
	$DEFT_{st-epoh}$	3756,092	0,872	0,902	0,800	1511,600
	$DEFT - LDM_{st-resitev}$	2007,309	0,900	0,922	0,843	733,800
	$DEFT - LDM_{st-epoh}$	4350,724	0,899	0,922	0,842	1517,600
ResNet50	*klasicna*	**83,451**	0,652	0,750	0,495	**10,600**
	$uglasevanje_{vse}$	111,023	0,749	0,818	0,613	18,300
	$uglasevanje_{rocno}$	97,351	0,776	0,841	0,654	19,300
	$DEFT_{st-resitev}$	5133,875	0,830	0,867	0,725	1054,000
	$DEFT_{st-epoh}$	6769,687	0,792	0,842	0,672	1518,300
	$DEFT - LDM_{st-resitev}$	4837,843	0,830	0,869	0,731	1020,500
	$DEFT - LDM_{st-epoh}$	7055,451	**0,872**	**0,904**	**0,801**	1520,300
MobileNet	*klasicna*	103,870	0,708	0,761	0,524	32,500
	$uglasevanje_{vse}$	64,018	0,829	0,868	0,731	21,600
	$uglasevanje_{rocno}$	**33,927**	0,652	0,772	0,498	**18,300**
	$DEFT_{st-resitev}$	2966,571	0,901	0,928	0,846	1025,900
	$DEFT_{st-epoh}$	4422,152	0,907	0,929	0,854	1519,200
	$DEFT - LDM_{st-resitev}$	2457,679	**0,914**	**0,936**	**0,866**	797,600
	$DEFT - LDM_{st-epoh}$	3759,590	0,913	**0,936**	0,864	1516,200

Rezultati, pridobljeni z uporabo arhitekture *ResNet50*, prikazujejo, da je bila najuspešnejša metoda s stališča klasifikacijskih metrik metoda $DEFT - LDM_{st-epoh}$, ki pa je obenem za učenje v povprečju porabila največ časa in epoh. Najslabše se je, podobno kot pri arhitekturi *VGG16*, odrezala metoda *klasicna*, ki je tudi v tem primeru za učenje v povprečju porabila najmanj epoh in časa. Če primerjamo uspešnosti metod $DEFT_{st-resitev}$ in $DEFT - LDM_{st-resitev}$, vidimo, da sta ponovno dosegli primerljive klasifikacijske rezultate, medtem ko je slednja za dosego primerljivih rezultatov v povprečju ponovno porabila manj časa in epoh. Metoda $DEFT_{st-epoh}$ je v povprečju ponovno dosegla slabše rezultate kot $DEFT_{st-resitev}$, pri čemer je za to porabila več časa in epoh kot prva, kar je podobno kot pri arhitekturi *VGG16*. V primeru metode $DEFT - LDM_{st-epoh}$ pa je ta, s stališča klasifikacijskih metrik, prekosila metodo $DEFT - LDM_{st-resitev}$. Tudi v tem primeru je klasifikacijska uspešnost metod *klasicna*, $uglasevanje_{vse}$ in $uglasevanje_{rocno}$ v splošnem nižja kot pri različicah metode

DEFT, po drugi strani pa sta čas izvajanja in število porabljenih epoh na strani prej omenjenih.

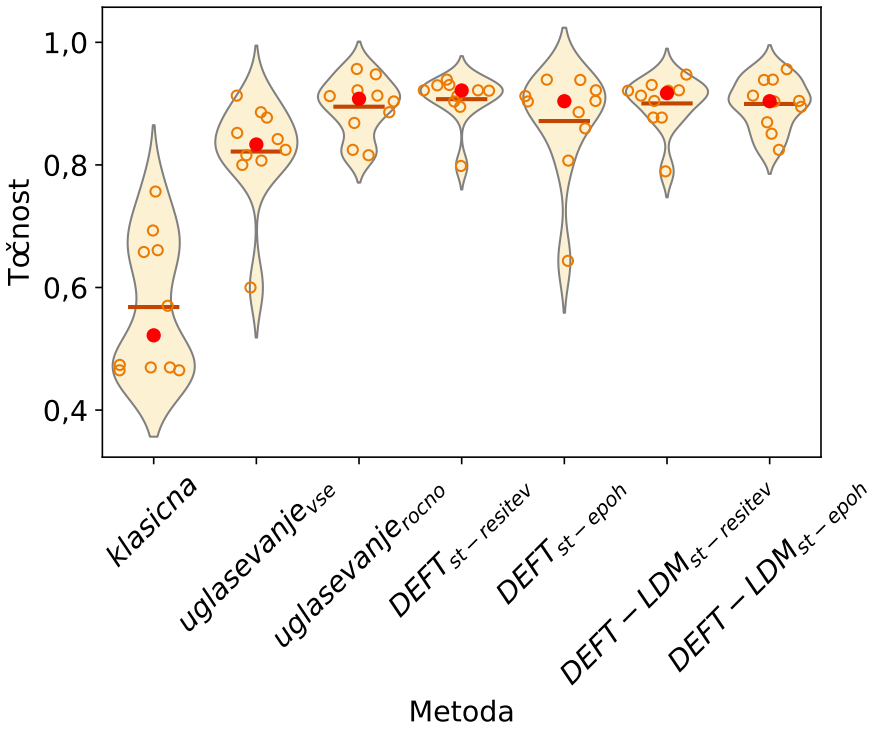
V splošnem so rezultati, pridobljeni z uporabo arhitekture *MobileNet*, podobni kot pri ostalih dveh arhitekturah CNN. Tako kot pri prejšnjih arhitekturah, so tudi v tem primeru metode *klasicna*, $uglasevanje_{vse}$ in $uglasevanje_{rocno}$ v podrejenem položaju. Najslabše se je iz stališča klasifikacijske uspešnosti odrezala metoda $uglasevanje_{rocno}$, ki je prav tako za učenje v povprečju porabila najmanj časa. Najvišjo točnost je v povprečju dosegla metoda $DEFT-LDM_{st-resitev}$, vendar so si s tega stališča vse različice *DEFT* metod precej podobne. Če primerjamo metodi $DEFT_{st-resitev}$ in $DEFT_{st-epoh}$, lahko opazimo, da je slednja dosegla malenkost višjo klasifikacijsko točnost, vendar na račun precej daljšega časa izvajanja in porabljenega večjega števila epoh. Po drugi strani pa metodi $DEFT - LDM_{st-epoh}$ ni uspelo preseči metode $DEFT - LDM_{st-resitev}$ v nobeni izmed klasifikacijskih metrik.

Splošno gledano, ne glede na uporabljeno arhitekturo CNN, je v povprečju najvišjo točnost dosegla metoda $DEFT - LDM_{st-resitev}$ ob uporabi arhitekture *MobileNet*, ki je prav tako v povprečju dosegala najvišje vrednosti metrik *AUNP* in *Kappa*. Po drugi strani pa se je daleč najslabše odrezala metoda *klasicna* ob uporabi arhitekture *VGG16*.

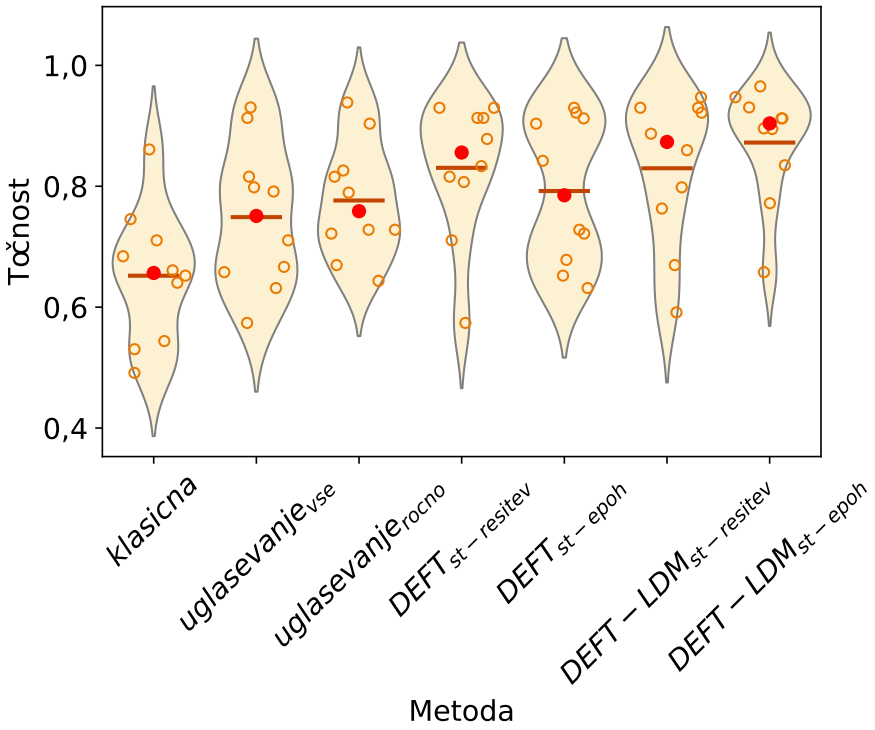
Slika 7.1 prikazuje grafikone klasifikacijske točnosti primerjanih metod ob uporabi arhitekture *VGG16*, ki poleg povprečnih vrednosti metrike točnost prikazujejo tudi srednje vrednosti, mediano in razpršenost rezultatov. Iz slike je razvidna velika razpršenost in velik standardni odklon točnosti pri metodah *klasicna*, $uglasevanje_{vse}$ in $DEFT_{st-epoh}$, medtem ko je pri preostalih primerjanih metodah standardni odklon manjši.

Na sliki 7.2 so prikazani rezultati vrednosti metrik za vseh 10 izvedenih prečnih preverjanj ob uporabi arhitekture *ResNet50*. Za razliko od rezultatov ob uporabljeni arhitekturi *VGG16* je v tem primeru mogoče opaziti veliko razpršenost doseženih vrednosti metrike točnost pri vseh metodah. Ta pojav lahko morda pripišemo velikemu številu slojev arhitekture *ResNet*, kar oteži izbiro najprimernejših uglaševanih slojev.

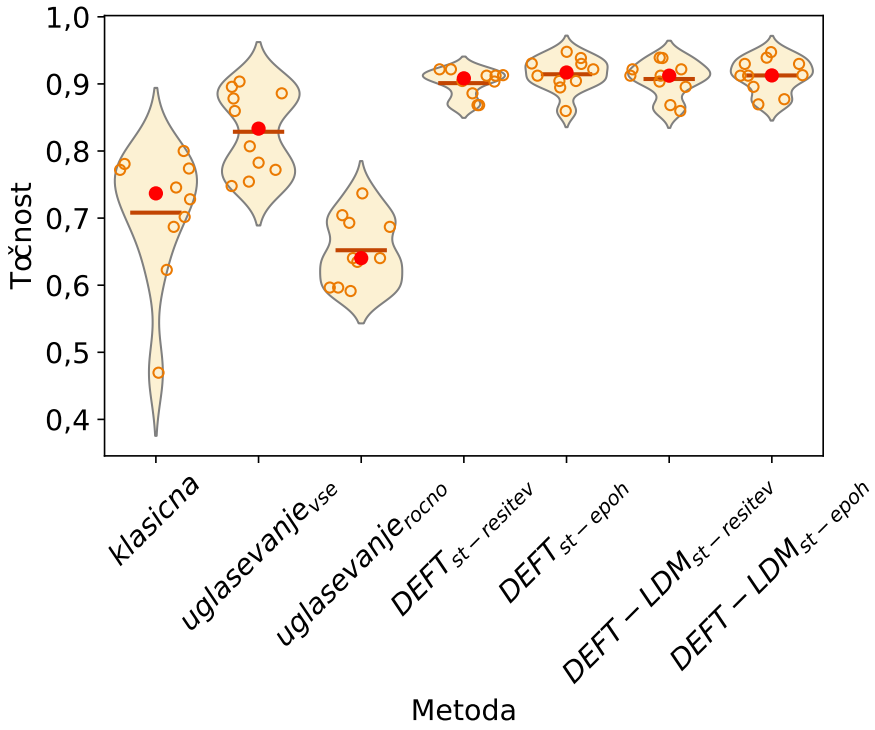
Za razliko od rezultatov točnosti primerjanih metod ob uporabi arhitekture *ResNet50* so rezultati, pridobljeni ob uporabi arhitekture *MobileNet*, predstavljeni na sliki 7.3, bolj izstopajoči. Iz slike je očitno, da so rezultati, pridobljeni z uporabo različic metode *DEFT*, boljši ter v splošnem z manjšim standardnim odklonom kot rezultati ostalih metod.



Slika 7.1 Točnost primerjanih metod ob uporabi arhitekture *VGG16* nad podatkovno zbirko *Osteosarcoma*.



Slika 7.2 Točnost primerjanih metod ob uporabi arhitekture *ResNet50* nad podatkovno zbirko *Osteosarcoma*.



Slika 7.3 Točnost primerjanih metod ob uporabi arhitekture *MobileNet* nad podatkovno zbirko *Osteosarcoma*.

Rezultati eksperimentov, izvedenih nad podatkovno zbirko *Športne slike*

Tabela 7.2 prikazuje vse rezultate merjenih metrik primerjanih metod nad podatkovno zbirko *Športne slike*. Rezultati so tudi v tem primeru razdeljeni glede na uporabljeno arhitekturo CNN ter prikazani v obliki povprečne vrednosti posamezne metrike. Odebeljeno so zapisane vrednosti posameznih metrik, kjer je bila, ob uporabi posamezne arhitekture CNN, dosežena najboljša vrednost.

Rezultati, pridobljeni z uporabo arhitekture *VGG16*, prikazujejo, da je v povprečju najvišjo točnost dosegla metoda $DEFT - LDM_{st-epoh}$, ki je obenem dosegla najvišje vrednosti preostalih klasifikacijskih metrik. Po drugi strani je omenjena metoda porabila za učenje največ časa. Iz vidika klasifikacijske točnosti so ji sledile preostale različice metode *DEFT*, med katerimi ni vidnih večjih odstopanj pri vrednosti klasifikacijskih metrik. Če primerjamo metodi $DEFT_{st-resitev}$ in $DEFT_{st-epoh}$, lahko opazimo, da je slednja dosegla za odtenek slabšo točnost, vendar je ob tem porabila bistveno več časa in epoh. Primerjava rezultatov metod $DEFT_{st-resitev}$ in $DEFT - LDM_{st-resitev}$ nam pokaže, da je slednja za dosego primerljivega rezultata porabila opazno manj časa in epoh. To ponovno nakazuje, da razvita metrika *LDM* pripomore k zmanjšanju časovne zahtevnosti izvajanja metode *DEFT*. Najslabše so se iz vidika klasifikacijske uspešnosti izkazale metode *klasicna*, $uglasevanje_{vse}$ in $uglasevanje_{rocno}$, ki so v splošnem za učenje porabile najmanj časa in epoh.

Tabela 7.2 Rezultati izvedenih eksperimentov nad podatkovno zbirko *Športne slike*.

CNN	Metoda	Čas	Točnost	AUNP	Kappa	Epoha
VGG16	*klasicna*	110,804	0,443	0,625	0,264	**17,470**
	$uglasevanje_{vse}$	104,187	0,690	0,794	0,587	26,400
	$uglasevanje_{rocno}$	**72,481**	0,795	0,863	0,726	25,100
	$DEFT_{st-resitev}$	4094,360	0,835	0,890	0,780	589,000
	$DEFT_{st-epoh}$	4440,858	0,834	0,889	0,778	1511,700
	$DEFT - LDM_{st-resitev}$	1945,304	0,834	0,889	0,778	524,200
	$DEFT - LDM_{st-epoh}$	9136,682	**0,845**	**0,897**	**0,794**	1503,100
ResNet50	*klasicna*	**95,077**	0,422	0,613	0,227	**11,700**
	$uglasevanje_{vse}$	125,022	0,568	0,713	0,425	18,500
	$uglasevanje_{rocno}$	122,041	0,676	0,785	0,569	20,700
	$DEFT_{st-resitev}$	5930,154	0,844	0,896	0,792	899,200
	$DEFT_{st-epoh}$	7541,511	0,852	0,902	0,803	1518,600
	$DEFT - LDM_{st-resitev}$	4447,403	0,840	0,894	0,787	643,800
	$DEFT - LDM_{st-epoh}$	9794,827	**0,854**	**0,903**	**0,805**	1519,000
MobileNet	*klasicna*	91,864	0,431	0,621	0,241	**10,900**
	$uglasevanje_{vse}$	101,914	0,846	0,898	0,795	24,900
	$uglasevanje_{rocno}$	**39,552**	0,788	0,860	0,718	19,700
	$DEFT_{st-resitev}$	3041,745	**0,887**	**0,925**	**0,850**	932,700
	$DEFT_{st-epoh}$	4092,665	0,883	0,922	0,844	1523,100
	$DEFT - LDM_{st-resitev}$	2021,641	0,876	0,918	0,834	403,700
	$DEFT - LDM_{st-epoh}$	7066,908	0,878	0,919	0,837	1517,600

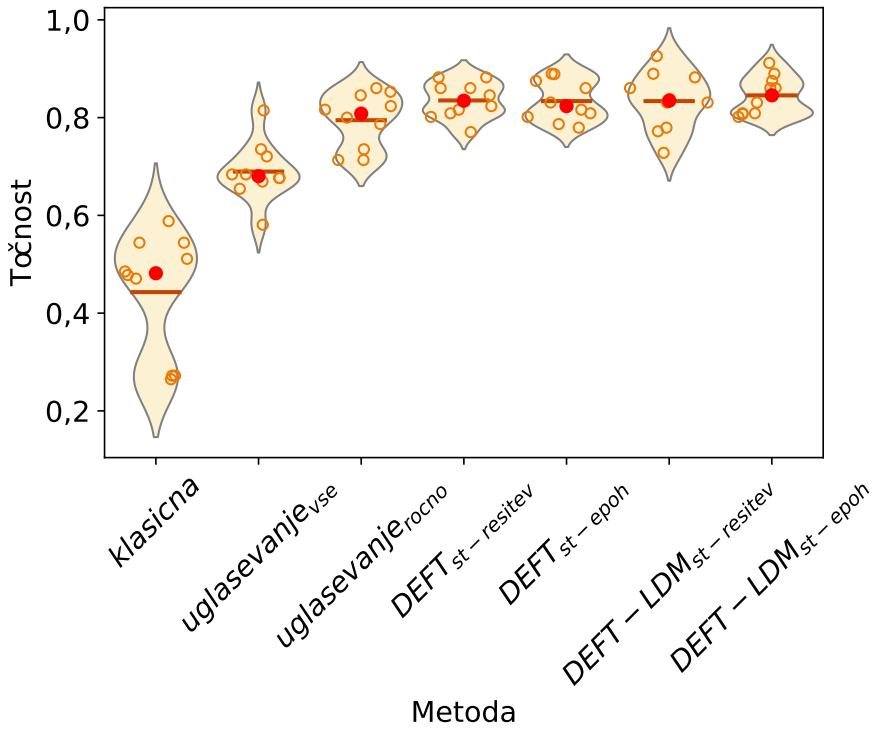
Ob uporabi arhitekture *ResNet* je najvišjo točnost v povprečju ponovno dosegla metoda $DEFT - LDM_{st-epoh}$, ki je tudi v tem primeru za učenje v povprečju porabila največ časa in epoh. Enako kot pri uporabi arhitekture *VGG16*, ji tudi v tem primeru s stališča klasifikacijske uspešnosti sledijo različice metode *DEFT*, najslabše pa so se odrezale metode *klasicna*, $uglasevanje_{vse}$ in $uglasevanje_{rocno}$. Tudi v tem primeru so slednje za učenje v povprečju porabile manj virov. Enako kot pri prejšnji primerjavi metod $DEFT_{st-resitev}$ in $DEFT - LDM_{st-resitev}$ se tudi tukaj izkaže, da slednja za dosego primerljive klasifikacijske uspešnosti v povprečju porabi manj časa in epoh. Če primerjamo rezultate točnosti metod $DEFT_{st-resitev}$ in $DEFT_{st-epoh}$, vidimo, da slednja doseže v povprečju boljšo klasifikacijsko točnost, vendar za to porabi občutno več virov. Enako velja tudi pri primerjavi metod $DEFT-LDM_{st-resitev}$ ter $DEFT-LDM_{st-epoh}$, z razliko, da slednja presega primerjano metodo v točnosti za večjo razliko.

V primeru uporabe arhitekture *MobileNet* je mogoče zaznati podoben trend kot pri preostalih dveh arhitekturah, z razliko, da je v povprečju najvišjo točnost dosegla metoda $DEFT_{st-resitev}$. Slednja se je tudi najbolje izkazala pri preostalih dveh klasifikacijskih metrikah *AUNP* in *Kappa*. Najslabše se je ponovno izkazala metoda

klasicna, ki je enako kot pri ostalih arhitekturah tudi pri tej porabila najmanj epoh. Z naskokom najmanj časa je za učenje porabila metoda $uglasevanje_{rocno}$. Različice metode *DEFT* so si iz vidika klasifikacijske uspešnosti tudi v tem primeru precej podobne, ponovno pa je mogoče opaziti, da različice, ki za zaustavitveni pogoj uporabljajo število epoh, v splošnem ne dosegajo bistveno boljših rezultatov kot preostali različici, čeprav v povprečju porabijo za učenje več časa in epoh.

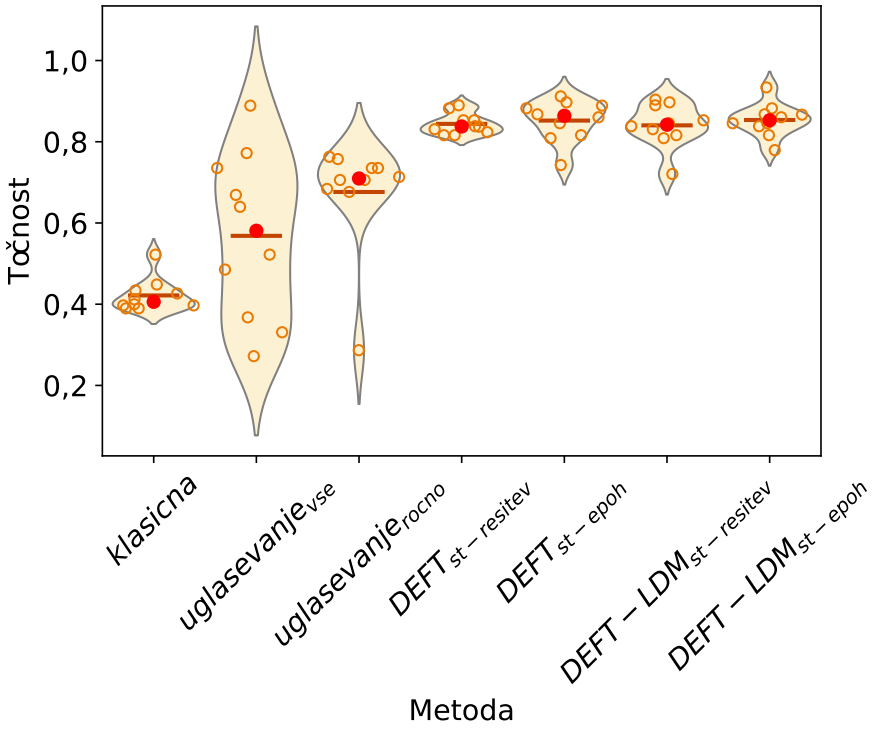
V splošnem je, ne glede na uporabljeno arhitekturo CNN, je v povprečju najvišjo točnost, dosegla metoda $DEFT_{st-resitev}$ ob uporabi arhitekture *MobileNet*. Na drugi strani je izstopala metoda *klasicna*, ki je v povprečju dosegla najslabšo klasifikacijo uspešnost. Ne glede na izbrano arhitekturo so se različice metode *DEFT* izkazale za uspešnejše od ostalih primerjanih metod.

Na sliki 7.4 so prikazani rezultati klasifikacijske metrike točnost za vse primerjane metode ob uporabi arhitekture *VGG16*. Iz slike je razvidno, da imajo v splošnem vse metode z izjemo metode *klasicna* majhen standardni odklon. Izmed različic metod *DEFT* s stališča velikosti standardnega odklona še najbolj izstopa metoda $DEFT - LDM_{st-resitev}$.



Slika 7.4 Točnost primerjanih metod ob uporabi arhitekture *VGG16* nad podatkovno zbirko *Športne slike*.

Slika 7.5 prikazuje rezultate klasifikacijske točnosti primerjanih metod ob uporabi arhitekture *ResNet50*. Na sliki z velikim standardnim odklonom izstopata metriki $uglasevanje_{vse}$ in $uglasevanje_{rocno}$, z razliko, da je pri slednji le ena izmed vrednosti zelo



Slika 7.5 Točnost primerjanih metod ob uporabi arhitekture *ResNet50* nad podatkovno zbirko *Športne slike*.

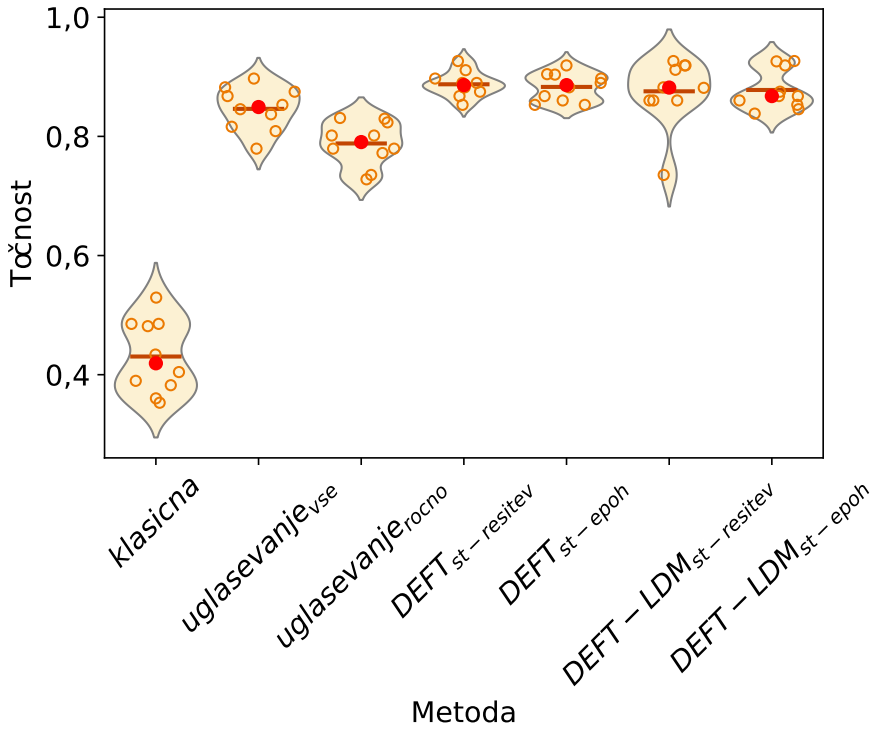
nizka, medtem ko so rezultati metode $uglasevanje_{vse}$ razpršeni. Preostale metode imajo majhen standardni odklon, pri čemer metoda *klasicna* dosega najnižjo povprečno točnost.

Rezultati klasifikacijske točnosti primerjanih metod ob uporabi arhitekture *MobileNet* so prikazani na sliki 7.6. Iz slike je razvidno negativno izstopanje metode *klasicna*, ki dosega daleč najslabše rezultate izmed vseh primerjanih metod. Pri preostalih metodah ni zaznati večjih odstopanj v smislu velikosti standardnega odklona. Še najbolj izmed preostalih metod izstopa $DEFT - LDM_{st-resitev}$, ki ima eno vrednost precej nižjo kot preostale vrednosti in ima posledično večji standardni odklon od preostalih različic metode *DEFT*.

Rezultati eksperimentov, izvedenih nad podatkovno zbirko *COVID-19*

Rezultati eksperimentov vseh primerjanih metod, izvedenih nad podatkovno zbirko *COVID-19*, so prikazani v tabeli 7.3. Tudi v tem primeru so rezultati razdeljeni glede na uporabljeno arhitekturo CNN ter prikazani v obliki povprečne vrednosti posamezne metrike. Odebeljeno so zapisane vrednosti posameznih metrik, kjer je bila ob uporabi posamezne arhitekture CNN dosežena najboljša vrednost.

Ob uporabi *VGG16* arhitekture lahko vidimo, da je najvišjo povprečno točnost dosegla metoda $DEFT - LDM_{st-epoh}$, ki je obenem za učenje porabila največ časa.



Slika 7.6 Točnost primerjanih metod ob uporabi arhitekture *MobileNet* nad podatkovno zbirko *Športne slike*.

S stališča klasifikacijske uspešnosti so ji podobno kot v prejšnjih primerih sledile različice metode *DEFT*. Najslabše se je odrezala metoda *klasicna*, najmanj časa in epoh pa je za učenje porabila metoda $uglasevanje_{rocno}$. Če se osredotočimo na obe različici metode *DEFT*, ki za zaustavitveni pogoj uporabljata maksimalno število epoh, lahko vidimo, da sta s stališča točnosti obe presegli preostali dve različici. Če primerjamo metodi $DEFT_{st-resitev}$ in $DEFT - LDM_{st-resitev}$, se ponovno izkaže, da je slednja dosegla primerljivo točnost z manj porabljenega časa in epoh.

Rezultati, pridobljeni z uporabo arhitekture *ResNet50*, prikazujejo, da je v povprečju, najvišjo točnost dosegla metoda $DEFT - LDM_{st-epoh}$, ki je tudi v tem primeru porabila največ časa za učenje. Najuspešnejši metodi iz vidika klasifikacijske uspešnosti ponovno sledijo različice metode *DEFT*. Izkaže se tudi, da različice metode *DEFT*, ki uporabljajo metriko *LDM*, v povprečju dosegajo primerljive rezultate ob krajšem času učenja in manj porabljenih epohah. Najslabšo točnost, za razliko od prejšnjih primerov, dosega metoda $uglasevanje_{rocno}$, ki obenem za učenje porabi najmanj časa in epoh.

V primeru uporabe arhitekture *MobileNet* je najvišjo povprečno točnost ponovno dosegla metoda $DEFT - LDM_{st-epoh}$, pri čemer je za dosego tega rezultata porabila največ epoh. Enako kot pri prejšnjih dveh uporabljenih arhitekturah ji tudi v tem primeru s stališča klasifikacijske uspešnosti sledijo različice metode *DEFT*. Ponovno

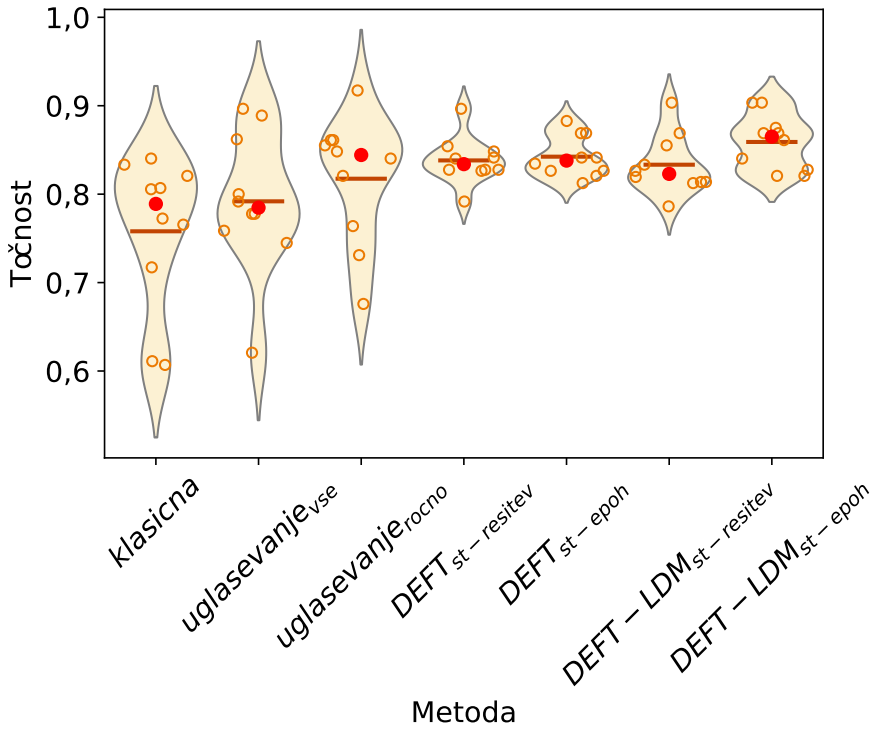
Tabela 7.3 Rezultati izvedenih eksperimentov nad podatkovno zbirko *COVID-19*.

CNN	Metoda	Čas	Točnost	AUNP	Kappa	Epoha
VGG16	*klasicna*	241,432	0,758	0,728	0,456	26,700
	$uglasevanje_{vse}$	175,988	0,792	0,770	0,552	26,900
	$uglasevanje_{rocno}$	**132,721**	0,817	0,810	0,609	**21,100**
	$DEFT_{st-resitev}$	2153,125	0,838	0,828	0,657	189,700
	$DEFT_{st-epoh}$	9357,520	0,842	0,836	0,669	1516,700
	$DEFT - LDM_{st-resitev}$	2005,556	0,833	0,829	0,652	174,300
	$DEFT - LDM_{st-epoh}$	18835,880	**0,859**	**0,858**	**0,708**	1512,500
ResNet50	*klasicna*	167,127	0,725	0,727	0,446	17,600
	$uglasevanje_{vse}$	445,084	0,653	0,662	0,331	**15,900**
	$uglasevanje_{rocno}$	**145,080**	0,623	0,660	0,315	**15,900**
	$DEFT_{st-resitev}$	7341,801	0,793	**0,789**	0,587	881,800
	$DEFT_{st-epoh}$	12663,816	0,809	0,785	0,582	1513,300
	$DEFT - LDM_{st-resitev}$	3056,481	0,775	0,774	0,550	259,100
	$DEFT - LDM_{st-epoh}$	16713,860	**0,811**	0,788	**0,589**	1511,500
MobileNet	*klasicna*	**86,914**	0,526	0,531	0,064	**9,700**
	$uglasevanje_{vse}$	153,030	0,746	0,689	0,402	23,100
	$uglasevanje_{rocno}$	161,926	0,688	0,605	0,239	26,300
	$DEFT_{st-resitev}$	7322,166	0,791	0,751	0,512	853,200
	$DEFT_{st-epoh}$	21439,846	0,806	0,769	0,561	1515,700
	$DEFT - LDM_{st-resitev}$	2535,436	0,794	0,761	0,538	180,000
	$DEFT - LDM_{st-epoh}$	9614,576	**0,819**	**0,790**	**0,598**	1519,500

se izkaže, da različice metode *DEFT* z uporabo metrike *LDM* dosežejo primerljive rezultate v krajšem času oz. z manj porabljenimi epohami. Najslabšo točnost je še enkrat več dosegla metoda *klasicna*, ki je ob tem za učenje porabila najmanj časa in epoh izmed vseh primerjanih metod.

V splošnem je, ne glede na uporabljeno arhitekturo CNN, najvišjo točnost, $0,859$, dosegla metoda $DEFT - LDM_{st-epoh}$ z uporabo arhitekture *VGG16*. Slednja je dosegla tudi najvišjo točnost pri vseh preostalih arhitekturah CNN. V splošnem najslabšo klasifikacijsko uspešnost je prikazala metoda *klasicna* ob uporabi arhitekture *MobileNet*.

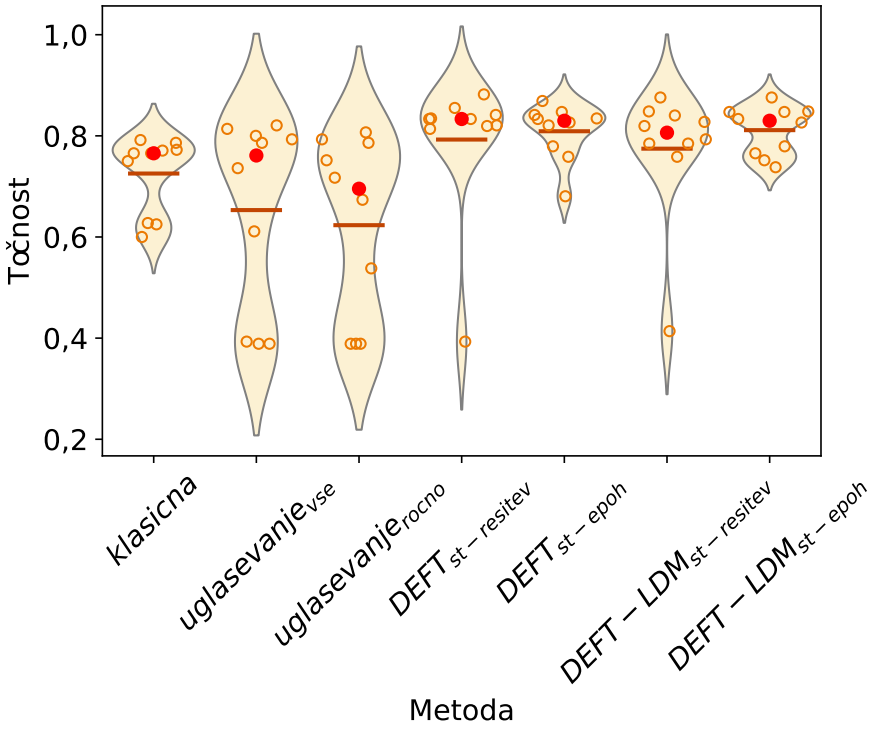
Rezultati primerjanih metrik ob uporabi arhitekture *VGG16* so prikazani na sliki 7.7. Iz slike je razviden velik standardni odklon in velika razpršenost rezultatov metod *klasicna*, $uglasevanje_{vse}$ in $uglasevanje_{rocno}$, medtem ko je pri preostalih primerjanih metodah razpršenost kot tudi standardni odklon manjši. Pri različicah metode *DEFT* ni vidnih večjih odstopanj iz vidika standardnega odklona, obenem pa tudi v povprečju dosegajo višjo klasifikacijsko uspešnost.



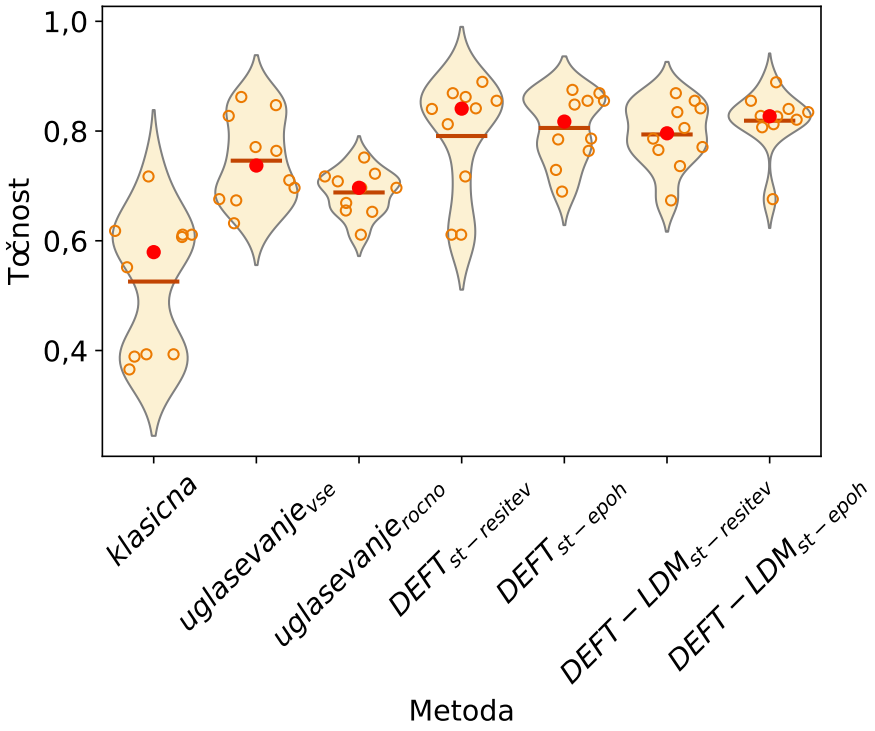
Slika 7.7 Točnost primerjanih metod ob uporabi arhitekture *VGG16* nad podatkovno zbirko *COVID-19*.

Na sliki 7.8 so prikazani rezultati primerjanih metod ob uporabi arhitekture *ResNet50*. Za razliko od rezultatov ob uporabljeni arhitekturi *VGG16* je v tem primeru v splošnem mogoče opaziti veliko razpršenost doseženih točnosti tudi pri različicah metode *DEFT*. Podoben pojav smo lahko opazili tudi pri analizi rezultatov točnosti nad podatkovno množico *Osteosarcoma*.

Slika 7.9 prikazuje dosežene rezultate točnosti primerjanih metod ob uporabi arhitekture *ResNet50*. Iz slike je razvidno, da metoda *klasicna* izstopa tako iz vidika doseženih točnosti kot tudi s stališča razpršenosti rezultatov. Preostale primerjane metode v splošnem dosegajo višjo klasifikacijsko uspešnost in pretirano ne odstopajo iz vidika velikosti standardnega odklona. Najmanjši standardni odklon pa dosega metoda $uglasevanje_{rocno}$.



Slika 7.8 Točnost primerjanih metod ob uporabi arhitekture *ResNet50* nad podatkovno zbirko *COVID-19*.



Slika 7.9 Točnost primerjanih metod ob uporabi arhitekture *MobileNet* nad podatkovno zbirko *COVID-19*.

7.2 Statistična analiza rezultatov

V nadaljevanju v prvem delu izvedemo osnovno statistično analizo z uporabo metod, ki temeljijo na preverjanju značilnosti ničelne domneve, nato pa še izvedemo podrobnejšo statistično analizo z uporabo metode Bayesove analize, natančneje Bayesovega hierarhičnega koreliranega t-testa.

V procesu izvedbe osnovne statistične analize smo najprej rezultate, pridobljene z izvedbo eksperimentov, organizirali v skupine glede na uporabljeno arhitekturo CNN. Za vsako množico rezultatov posamezne metode ob uporabi posamezne arhitekture CNN za posamezno metriko smo z uporabo Shapiro-Wilkovega testa preverili, ali je porazdelitev vrednosti normalna. Rezultati Shapiro-Wilkovega testa so pokazali, da v 36 izmed 105 opravljenih testih hipoteze H_0, ki pravi, da je porazdelitev vrednosti v množicah normalno porazdeljena, ne moremo sprejeti (s statistično značilnostjo $p < 0,01$). Posledično smo v naslednjem koraku za preverjanje statistično značilne različnosti primerjanih metod z vidika posamezne metrike uporabili Friedmanov test, pri katerem smo upoštevali Holm-Bonferronijevo korekcijo.

V tabeli 7.4 so predstavljeni rezultati opravljenega Friedmanovega testa, ki kažejo, da obstajajo statistično značilne razlike med primerjanimi klasifikatorji v vsaki izmed merjenih metrik ne glede na uporabljeno arhitekturo CNN.

Tabela 7.4 Rezultati Friedmanovega testa za vse metrike primerjanih metod ob uporabi različnih arhitektur CNN.

CNN	Metrika	χ^2	p-vrednost
VGG16	Čas	168,63	p < 0,001
	Točnost	78,53	p < 0,001
	AUNP	71,41	p < 0,001
	Kappa	76,39	p < 0,001
	Epoha	150,92	p < 0,001
ResNet50	Čas	165,32	p < 0,001
	Točnost	72,28	p < 0,001
	AUNP	66,12	p < 0,001
	Kappa	67,91	p < 0,001
	Epoha	165,32	p < 0,001
MobileNet	Čas	166,11	p < 0,001
	Točnost	106,32	p < 0,001
	AUNP	102,20	p < 0,001
	Kappa	103,41	p < 0,001
	Epoha	166,13	p < 0,001

Podrobnejšo statistično analizo izvedemo z uporabo Bayesovega hierarhičnega koreliranega t-testa, po priporočilu Benavolija in soavtorjev [150]. Uporaba metod Bayesove analize se v splošnem izkaže primernejša za vrednotenje oz. primerjavo uspešnosti med posameznimi metodami kot klasični post-hoc statistični testi, saj nas pri statistični analizi najpogosteje zanima, kakšna je verjetnost, da je en klasifikator boljši od drugega. Klasične post-hoc metode, kot je Wilcoxonov test, nam na to vprašanje ne morejo podati odgovora. Takšne metode namreč izračunavajo verjetnost, da bomo opazovano ali večjo razliko med dvema primerjanima klasifikatorjema pridobili ob predpostavki, da je ničelna domneva H_0 resnična, kar je pri primerjavi dveh klasifikatorjev praktično nemogoče [150].

Verjetnosti, da je posamezen klasifikator boljši od drugega, izračunamo z uporabo Bayesovega hierarhičnega koreliranega t-testa, s katerim so izračunane posteriorne verjetnosti za primerjave uporabljenih metod. Najprej izračunamo verjetnosti primerjav posameznih parov uporabljenih metod za dosežene vrednosti metrike točnost vsake izmed uporabljenih arhitektur CNN, zatem postopek ponovimo še za število porabljenih epoh. Statistično značilnost razlik posameznih primerjav parov metod *Metoda*1 in *Metoda*2 potrdimo na podlagi sledečih pravil:

- *Metoda*1 pri izbrani metriki dosega višje vrednosti od metode *Metoda*2, če je $P(Metoda1) > 0,95$.
- *Metoda*1 pri izbrani metriki dosega nižje vrednosti od metode *Metoda*2, če je $P(Metoda2) > 0,95$.
- *Metoda*1 pri izbrani metriki dosega enakovredne vrednosti kot *Metoda*2, če je $P(ROPE) > 0,95$.

V vseh izračunih posteriorne verjetnosti smo za regijo praktične ekvivalence dveh metod (ROPE) po priporočilih, podanih v [150], izbrali vrednost $0,01$.

Izračun verjetnosti klasifikacijske uspešnosti smo izvedli z uporabo povprečnih vrednosti metrike točnost posamezne metode nad vsako izmed uporabljenih podatkovnih množic. V tabeli 7.5 so predstavljene verjetnosti primerjav klasifikacijske uspešnosti za vse kombinacije parov metod z uporabo arhitekture *VGG16*. V tabeli so odebeljeno zapisane verjetnosti, za katere lahko potrdimo statistično značilnost.

Iz predstavljenih rezultatov je razvidno, da smo izmed 21 verjetnosti primerjav statistično značilnost potrdili pri 13 primerjavah. Iz posteriornih verjetnosti je mogoče razbrati, da je metoda $DEFT_{st-resitev}$ statistično značilno boljša od metod *klasicna*, $uglasevanje_{vse}$ ter $uglasevanje_{rocno}$. Klasifikacijska uspešnost metod $DEFT_{st-resitev}$ ter $DEFT - LDM_{st-resitev}$ je statistično značilno enaka. Pri preostalih parih primerjav različic metode *DEFT* izračunane verjetnosti niso statistično značilne. Pri primerjavi verjetnosti metod $DEFT_{st-resitev}$ in $DEFT_{st-epoh}$ statistične značilnosti ne moremo

Tabela 7.5 Verjetnosti primerjav klasifikacijske uspešnosti za vse kombinacije parov metod ob uporabi arhitekture *VGG16* nad vsemi tremi podatkovnimi zbirkami.

Metoda1	Metoda2	$P(Metoda1)$	$P(ROPE)$	$P(Metoda2)$
$klasicna$	$uglasevanje_{vse}$	0,0000	0,0085	**0,9915**
$klasicna$	$uglasevanje_{rocno}$	0,0000	0,0095	**0,9905**
$klasicna$	$DEFT_{st-resitev}$	0,0000	0,0095	**0,9905**
$klasicna$	$DEFT_{st-epoh}$	0,0000	0,0091	**0,9909**
$klasicna$	$DEFT-LDM_{st-resitev}$	0,0000	0,0085	**0,9915**
$klasicna$	$DEFT-LDM_{st-epoh}$	0,0000	0,0086	**0,9914**
$uglasevanje_{vse}$	$uglasevanje_{rocno}$	0,0000	0,0086	**0,9914**
$uglasevanje_{vse}$	$DEFT_{st-resitev}$	0,0000	0,0086	**0,9914**
$uglasevanje_{vse}$	$DEFT_{st-epoh}$	0,0000	0,0093	**0,9907**
$uglasevanje_{vse}$	$DEFT-LDM_{st-resitev}$	0,0000	0,0087	**0,9913**
$uglasevanje_{vse}$	$DEFT-LDM_{st-epoh}$	0,0000	0,0074	**0,9926**
$uglasevanje_{rocno}$	$DEFT_{st-resitev}$	0,0000	0,0382	**0,9618**
$uglasevanje_{rocno}$	$DEFT_{st-epoh}$	0,1564	0,2192	0,6244
$uglasevanje_{rocno}$	$DEFT-LDM_{st-resitev}$	0,0000	0,2135	0,7865
$uglasevanje_{rocno}$	$DEFT-LDM_{st-epoh}$	0,0000	0,1460	0,8540
$DEFT_{st-resitev}$	$DEFT_{st-epoh}$	0,4210	0,5790	0,0000
$DEFT_{st-resitev}$	$DEFT-LDM_{st-resitev}$	0,0000	**1,0000**	0,0000
$DEFT_{st-resitev}$	$DEFT-LDM_{st-epoh}$	0,0000	0,5864	0,4136
$DEFT_{st-epoh}$	$DEFT-LDM_{st-resitev}$	0,0000	0,7637	0,2363
$DEFT_{st-epoh}$	$DEFT-LDM_{st-epoh}$	0,0000	0,0914	0,9086
$DEFT-LDM_{st-resitev}$	$DEFT-LDM_{st-epoh}$	0,0000	0,3894	0,6106

potrditi, lahko pa ugotovimo, da sta metodi z verjetnostjo $P(ROPE) = 0,579$ praktično enakovredni, metoda $DEFT_{st-resitev}$ pa je z verjetnostjo $0,421$ boljša od metode $DEFT_{st-epoh}$.

Tabela 7.6 prikazuje verjetnosti primerjav klasifikacijske uspešnosti za vse kombinacije parov metod z uporabo arhitekture *ResNet50*. Statistično značilnost smo potrdili pri 13 izmed 21 verjetnosti primerjav. Tudi v tem primeru lahko potrdimo, da je metoda $DEFT_{st-resitev}$ statistično značilno boljša od metod *klasicna*, $uglasevanje_{vse}$ ter $uglasevanje_{rocno}$. Pri primerjavi verjetnosti parov metod $DEFT_{st-resitev}$ ter $DEFT-LDM_{st-resitev}$ ni mogoče potrditi statistične značilnosti, iz rezultatov posteriorne verjetnosti pa lahko razberemo, da sta metodi z verjetnostjo $P(ROPE) = 0,8405$ enakovredni. Z verjetnostjo $0,1595$ je metoda $DEFT_{st-resitev}$ boljša od metode $DEFT-LDM_{st-resitev}$.

Verjetnosti primerjav uspešnosti klasifikacijske točnosti za vse kombinacije parov metod ob uporabi arhitekture *MobileNet* so prikazane v tabeli 7.7. Statistično značilnost smo potrdili pri 16 izmed 21 verjetnosti primerjav. Iz rezultatov statistične analize je mogoče razbrati, da je metoda $DEFT_{st-resitev}$ statistično značilno boljša od metod *klasicna*, $uglasevanje_{vse}$ ter $uglasevanje_{rocno}$. Če primerjamo verjetnosti parov

Tabela 7.6 Verjetnosti primerjav uspešnosti klasifikacijske točnosti za vse kombinacije parov metod ob uporabi arhitekture *ResNet50* nad vsemi tremi podatkovnimi zbirkami.

Metoda1	Metoda2	$P(Metoda1)$	$P(ROPE)$	$P(Metoda2)$
$klasicna$	$uglasevanje_{vse}$	0,1063	0,0163	0,8774
$klasicna$	$uglasevanje_{rocno}$	0,1071	0,0160	0,8769
$klasicna$	$DEFT_{st-resitev}$	0,0000	0,0090	**0,9910**
$klasicna$	$DEFT_{st-epoh}$	0,0000	0,0097	**0,9903**
$klasicna$	$DEFT-LDM_{st-resitev}$	0,0000	0,0094	**0,9906**
$klasicna$	$DEFT-LDM_{st-epoh}$	0,0000	0,0089	**0,9911**
$uglasevanje_{vse}$	$uglasevanje_{rocno}$	0,1530	0,0480	0,7990
$uglasevanje_{vse}$	$DEFT_{st-resitev}$	0,0000	0,0086	**0,9914**
$uglasevanje_{vse}$	$DEFT_{st-epoh}$	0,0000	0,0087	**0,9913**
$uglasevanje_{vse}$	$DEFT-LDM_{st-resitev}$	0,0000	0,0095	**0,9905**
$uglasevanje_{vse}$	$DEFT-LDM_{st-epoh}$	0,0000	0,0093	**0,9907**
$uglasevanje_{rocno}$	$DEFT_{st-resitev}$	0,0000	0,0090	**0,9910**
$uglasevanje_{rocno}$	$DEFT_{st-epoh}$	0,0000	0,0359	**0,9641**
$uglasevanje_{rocno}$	$DEFT-LDM_{st-resitev}$	0,0000	0,0086	**0,9914**
$uglasevanje_{rocno}$	$DEFT-LDM_{st-epoh}$	0,0000	0,0085	**0,9915**
$DEFT_{st-resitev}$	$DEFT_{st-epoh}$	0,4983	0,2805	0,2212
$DEFT_{st-resitev}$	$DEFT-LDM_{st-resitev}$	0,1595	0,8405	0,0000
$DEFT_{st-resitev}$	$DEFT-LDM_{st-epoh}$	0,0000	0,2136	0,7864
$DEFT_{st-epoh}$	$DEFT-LDM_{st-resitev}$	0,5442	0,1097	0,3461
$DEFT_{st-epoh}$	$DEFT-LDM_{st-epoh}$	0,0000	0,5799	0,4201
$DEFT-LDM_{st-resitev}$	$DEFT-LDM_{st-epoh}$	0,0000	0,0370	**0,9630**

Tabela 7.7 Verjetnosti primerjav uspešnosti klasifikacijske točnosti za vse kombinacije parov metod ob uporabi arhitekture *MobileNet* nad vsemi tremi podatkovnimi zbirkami.

Metoda1	Metoda2	$P(Metoda1)$	$P(ROPE)$	$P(Metoda2)$
$klasicna$	$uglasevanje_{vse}$	0,0000	0,0095	**0,9905**
$klasicna$	$uglasevanje_{rocno}$	0,1101	0,0156	0,8743
$klasicna$	$DEFT_{st-resitev}$	0,0000	0,0093	**0,9907**
$klasicna$	$DEFT_{st-epoh}$	0,0000	0,0087	**0,9913**
$klasicna$	$DEFT-LDM_{st-resitev}$	0,0000	0,0082	**0,9918**
$klasicna$	$DEFT-LDM_{st-epoh}$	0,0000	0,0089	**0,9911**
$uglasevanje_{vse}$	$uglasevanje_{rocno}$	**0,9914**	0,0086	0,0000
$uglasevanje_{vse}$	$DEFT_{st-resitev}$	0,0000	0,0091	**0,9909**
$uglasevanje_{vse}$	$DEFT_{st-epoh}$	0,0000	0,0082	**0,9918**
$uglasevanje_{vse}$	$DEFT-LDM_{st-resitev}$	0,0000	0,0096	**0,9904**
$uglasevanje_{vse}$	$DEFT-LDM_{st-epoh}$	0,0000	0,0091	**0,9909**
$uglasevanje_{rocno}$	$DEFT_{st-resitev}$	0,0000	0,0088	**0,9912**
$uglasevanje_{rocno}$	$DEFT_{st-epoh}$	0,0000	0,0082	**0,9918**
$uglasevanje_{rocno}$	$DEFT-LDM_{st-resitev}$	0,0000	0,0086	**0,9914**
$uglasevanje_{rocno}$	$DEFT-LDM_{st-epoh}$	0,0000	0,0093	**0,9907**
$DEFT_{st-resitev}$	$DEFT_{st-epoh}$	0,0000	0,8410	0,1590
$DEFT_{st-resitev}$	$DEFT-LDM_{st-resitev}$	0,0510	0,8993	0,0497
$DEFT_{st-resitev}$	$DEFT-LDM_{st-epoh}$	0,0000	0,5809	0,4191
$DEFT_{st-epoh}$	$DEFT-LDM_{st-resitev}$	0,0460	**0,9540**	0,0000
$DEFT_{st-epoh}$	$DEFT-LDM_{st-epoh}$	0,0000	**0,9555**	0,0445
$DEFT-LDM_{st-resitev}$	$DEFT-LDM_{st-epoh}$	0,0000	0,5803	0,4197

metod $DEFT_{st-resitev}$ ter $DEFT-LDM_{st-resitev}$, ugotovimo, da statistične značilnosti ni mogoče potrditi. Metoda $DEFT_{st-resitev}$ je v primerjavi z metodo $DEFT-LDM_{st-resitev}$ enakovredna z verjetnostjo $0,8993$. Verjetnost, da je metoda $DEFT_{st-resitev}$ boljša od $DEFT-LDM_{st-resitev}$, je $0,051$, medtem ko je verjetnost, da je metoda $DEFT-LDM_{st-resitev}$ boljša od $DEFT_{st-resitev}$, enaka $0,0497$. Statistično značilno praktično ekvivalenco lahko potrdimo tudi med pari metod $DEFT_{st-epoh}$ in $DEFT-LDM_{st-resitev}$ ter $DEFT_{st-epoh}$ in $DEFT-LDM_{st-epoh}$.

Poleg posteriornih verjetnosti z uporabo metrike točnost smo za primerjane različice metode *DEFT* izračunali tudi verjetnosti primerjav za porabljeno število epoh in časa. Pri strojnem učenju epoha določa število ponovitev prehoda metode strojnega učenja čez celotno učno množico v procesu učenja. Iz tega vidika lahko epoho dojemamo kot neodvisno metriko algoritemske zahtevnosti uporabljene metode učenja nevronske mreže. Število epoh je v procesu učenja definirano kot učni parameter, na katerega ne vplivajo zunanji dejavniki, kot so zmogljivost strojne opreme, procesi operacijskega sistema in druge morebitne sistemske prekinitve. V

primerjavi s časom delovanja, ki je izmerjen za konkretno uporabljeno strojno opremo in je podvržen prej omenjenim zunanjim dejavnikom, metrika števila porabljenih epoh predstavlja bolj univerzalno mero za vrednotenje algoritemske zahtevnosti uporabljene metode strojnega učenja.

Z izračunanimi verjetnostmi primerjav porabljenega števila epoh lahko ugotovimo, katere izmed primerjanih metod statistično značilno porabijo več oz. manj epoh za učenje na podlagi česar lahko ugotovimo, ali uporaba metrike *LDM* boljše zaznava manj primerne izbire uglaševanih slojev CNN kot klasični pristop predčasnega ustavljanja učenja. Iz primerjav smo izločili primerjave z metodama $DEFT_{st-epoh}$ in $DEFT - LDM_{st-epoh}$, saj se pri slednjih kot zaustavitveni pogoj uporablja maksimalno število porabljenih epoh in kot taki vedno dosegata največje možno število epoh. Posledično primerjava teh dveh metod s stališča porabljenih epoh ni smiselna.

Tabela 7.8 prikazuje verjetnosti primerjav porabe epoh za vse uporabljene arhitekture CNN. Primerjani sta le različici metode *DEFT*, in sicer $DEFT_{st-epoh}$ in $DEFT - LDM_{st-epoh}$, saj nas primarno zanima zgolj primerjava njihove porabe epoh, iz česar lahko ugotovimo, ali uporaba metrike *LDM* pozitivno vpliva oz. zmanjšuje skupno porabo epoh predlagane metode *DEFT*. Statistično značilnost primerjanih metod smo potrdili nad vsemi tremi uporabljenimi arhitekturami CNN. Na podlagi tega lahko sklenemo, da *DEFT* ob uporabi metrike *LDM* statistično značilno porabi manj epoh učenja, kot metoda brez uporabe metrike *LDM*, ne glede na uporabljeno arhitekturo CNN. Iz tega sledi, da metrika *LDM* boljše zaznava manj primerne izbire uglaševanih slojev CNN kot klasični pristop predčasnega zaustavljanja učenja, saj metode, ki uporabljajo metriko *LDM*, statistično značilno porabijo manj epoh učenja kot tiste, ki je ne uporabljajo.

Tabela 7.8 Verjetnosti primerjav porabe epoh parov metod *DEFT* za vse uporabljene arhitekture CNN.

CNN	Metoda1 ($M1$)	Metoda2 ($M2$)	$P(M1)$	$P(ROPE)$	$P(M2)$
VGG16	$DEFT_{st-resitev}$	$DEFT - LDM_{st-resitev}$	**0,9913**	0,0087	0,0000
ResNet50	$DEFT_{st-resitev}$	$DEFT - LDM_{st-resitev}$	**0,9923**	0,0077	0,0000
MobileNet	$DEFT_{st-resitev}$	$DEFT - LDM_{st-resitev}$	**0,9913**	0,0087	0,0000

S ciljem ugotoviti, ali se je uporaba metrike *LDM* izkazala za uspešno rešitev problema časovne kompleksnosti metode *DEFT*, smo izračunali tudi posteriorne verjetnosti z uporabo metrike čas za različice metode *DEFT*. Tudi v tem primeru smo iz primerjav izločili primerjave metod $DEFT_{st-epoh}$ in $DEFT - LDM_{st-epoh}$ z metodami $DEFT_{st-resitev}$ in $DEFT - LDM_{st-resitev}$, saj se pri slednjih kot zaustavitveni pogoj uporablja maksimalno število porabljenih epoh in kot taki vedno dosegata največje možno

število epoh. Posledično metode s takšnim zaustavitvenim pogojem dosegajo daljše izvajalne čase, zaradi česar te primerjave niso smiselne.

Tabela 7.9 prikazuje verjetnosti primerjav porabljenega časa za vse arhitekture CNN. Statistično značilnost primerjav smo potrdili nad štirimi od šestih primerjav. Iz predstavljenih verjetnosti primerjav je razvidno, da ne glede na uporabljeno arhitekturo CNN, pri primerjavah parov metod $DEFT_{st-resitev}$ in $DEFT - LDM_{st-resitev}$, metoda brez uporabe metrike *LDM* porabi statistično značilno več časa za izvajanje. Pri primerjavah parov metod $DEFT_{st-epoh}$ in $DEFT - LDM_{st-epoh}$ je zanimivo, da za razliko od prejšnjih primerjav v splošnem metoda z uporabo metrike *LDM* porabi za izvajanje več časa kot brez uporabe metrike *LDM*. Povečanje časovne zahtevnosti pri metodah $DEFT - LDM_{st-epoh}$ lahko morda pripišemo večjemu številu ovrednotenih izbir uglaševanih slojev arhitekture CNN v primerjavi z metodami $DEFT_{st-epoh}$, kar privede do večkratnega izvajanja rutin za pripravo učenja modela ter za ovrednotenje modelov. Dodaten vpliv na izvajanje lahko ima tudi samo izvajalno okolje, nad katerim pa žal zaradi same narave oz. arhitekture superračunalnika žal nimamo nadzora. Glede na opravljeno statistično analizo, ne moremo trditi, da se je uporaba metrike *LDM* izkazala za uspešno rešitev problema časovne kompleksnosti metode *DEFT*, saj tega pri primerjavi parov metod $DEFT_{st-epoh}$ in $DEFT - LDM_{st-epoh}$ ni mogoče statistično potrditi.

Tabela 7.9 Verjetnosti primerjav porabe časa za pare metod *DEFT* nad vsemi uporabljenimi arhitekturami CNN.

CNN	Metoda1 ($M1$)	Metoda2 ($M2$)	$P(M1)$	$P(ROPE)$	$P(M2)$
VGG16	$DEFT_{st-resitev}$	$DEFT - LDM_{st-resitev}$	**0.9915**	0.0085	0.0000
	$DEFT_{st-epoh}$	$DEFT - LDM_{st-epoh}$	0.0000	0.0082	**0.9918**
ResNet50	$DEFT_{st-resitev}$	$DEFT - LDM_{st-resitev}$	**0.9914**	0.0086	0.0000
	$DEFT_{st-epoh}$	$DEFT - LDM_{st-epoh}$	0.1069	0.0152	0.8779
MobileNet	$DEFT_{st-resitev}$	$DEFT - LDM_{st-resitev}$	**0.9914**	0.0086	0.0000
	$DEFT_{st-epoh}$	$DEFT - LDM_{st-epoh}$	0.2565	0.0165	0.7270

Poglavje 8

Diskusija

"We can only see a short distance ahead, but we can see plenty there that needs to be done"

- Alan Turing

V tem poglavju povzamemo in interpretiramo rezultate, predstavljene v prejšnjih poglavjih, predstavimo omejitve raziskave ter podamo nadaljnje mogoče razširitve in možnosti uporabe predlaganih metod in metrik.

8.1 Interpretacija rezultatov

Raziskavo smo začeli z analizo vpliva izbire slojev konvolucijske nevronske mreže na uspešnost učenja. Za potrebe analize smo zasnovali eksperiment, temelječ na algoritmu naključnega iskanja, ki smo ga izvedli z uporabo treh različnih arhitektur CNN nad tremi različnimi podatkovnimi zbirkami. Pridobljene rezultate smo razdelili v tri skupine glede na klasifikacijsko uspešnost ter za vsako izmed skupin izračunali pogostost izbire posameznega sloja oz. bloka konvolucijske nevronske mreže. Z analizo rezultatov (poglavje 3.4) smo skušali ugotoviti, ali obstaja kakšen splošen vzorec kombinacije izbir slojev oz. blokov, ki bi v splošnem zagotavljal dobre rezultate. Ker splošnega vzorca pri izbiri uglaševanih slojev ob uporabi različnih arhitektur CNN nad različnimi podatkovnimi množicami v procesu analize rezultatov nismo uspeli zaznati, lahko **sprejmemo hipotezo H1**, ki se glasi:

(H1) Izbira uglaševanih slojev konvolucijske nevronske mreže je odvisna od ciljnega problema ter uporabljene arhitekture globoke konvolucijske nevronske mreže.

Delo smo nadaljevali z razvojem metode prilagodljivega uglaševanja slojev CNN, v sklopu katerega smo predlagali samodejno prilagodljivo metodo *DEFT*. Za potrebe

zaznavanja primernosti izbir uglaševanih slojev CNN, s katero bi lahko zmanjšali porabo epoh, potrebnih za učenje, z uporabo metode *DEFT*, smo predlagali tri različice na funkciji izgube temelječih metrik *NL*, *AUNL* in *LDM*, katerih delovanje smo v eksperimentalnem delu (poglavje 5.5) tudi ovrednotili. Za razvite metrike smo v okviru empirične analize izračunali Spearmanov koeficient korelacije med vrednostjo posamezne metrike in vrednostjo funkcije izgube v posamezni epohi učenja. Tako pridobljene koeficiente korelacije smo rangirali ter izračunali povprečne vrednosti rangov metrik (poglavje 5.5.2). Glede na v začetni fazi učenja najvišje dosežene povprečne range ter najmanjši standardni odklon smo kot najprimernejšo metriko za zaznavanje izbir uglaševanih slojev CNN izbrali metriko *LDM*. Na podlagi izračunanih korelacij metrike *LDM* z vrednostjo funkcije izgube v posamezni epohi učenja ter ovrednotenjem propustnosti metrike *LDM* (poglavje 5.5.3) lahko **sprejmemo hipotezo H2**, ki se glasi:

(H2) Razvita namenska metrika *LDM* omogoča zaznavo manj primernih izbir slojev konvolucijske nevronske mreže.

Sledila je predstavitev eksperimentalnega ogrodja, v katerem smo opredelili vse podrobnosti glede izvedbe eksperimentov z uporabo predlagane metode *DEFT* z in brez uporabe metrike *LDM*. Zatem smo predstavili rezultate izvedenih eksperimentov ter opravili statistično analizo. V sklopu statistične analize rezultatov smo v prvem delu izvedli analizo z uporabo metod, temelječih na preverjanju statistične značilnosti ničelne domneve, v katerem smo z uporabo Friedmanovega testa ugotovili, da so primerjane metode v vseh metrikah, ne glede na uporabljeno arhitekturo CNN, dosegle statistično različne rezultate. V nadaljevanju smo opravili podrobnejšo statistično analizo z uporabo Bayesovega hierarhičnega koreliranega t-testa, kjer smo na podlagi predhodne porazdelitve pridobljenih vrednosti metrike točnost izračunali posteriorno verjetnost primerjav uporabljenih metod ob uporabi posamezne arhitekture CNN. Zatem smo na enak način, vendar zgolj za različice metode *DEFT*, izračunali še posteriorno verjetnost za metriko porabljenega števila epoh in časa.

V sklopu opravljene statistične analize rezultatov izvedenih eksperimentov (poglavje 7.2) smo potrdili, da predlagana metoda *DEFT* z uporabo metrike *LDM*, ne glede na uporabljeno arhitekturo CNN, porabi statistično značilno manj epoh učenja kot metoda *DEFT* brez uporabe metrike *LDM*. V vseh primerjavah metode, ki ne uporabljajo metrike *LDM*, z verjetnostjo nad $0,99$ porabijo več epoh kot metode, ki metriko *LDM* uporabljajo. Posledično metrika *LDM* omogoča boljšo zaznavo manj primernih izbir uglaševanih slojev, saj učenje zaustavi v zgodnejših fazi kot klasični pristop predčasnega zaustavljanja učenja. Na podlagi opravljene analize metrike *LDM* (poglavje 5.5.3) in predstavljenih rezultatov statistične analize lahko **sprejmemo hipotezo H3**, ki se glasi:

(H3) Razvita metrika *LDM* boljše zaznava manj primerne izbire slojev konvolucijske nevronske mreže kot klasični pristop predčasnega ustavljanja učenja.

Statistična analiza rezultatov izvedenih eksperimentov je zajemala tudi primerjavo klasičnih pristopov učenja in predlagane metode *DEFT*. Na podlagi rezultatov statistične analize (poglavje 7.2) lahko ugotovimo, da predlagana metoda *DEFT*, ne glede na uporabljeno arhitekturo CNN, dosega večjo statistično značilno točnost kot primerjane klasične metode. Izračunana posteriorna verjetnost, da metoda *DEFT* doseže višjo točnost kot primerjane klasične metode, je v splošnem nad $0,99$. Na podlagi teh rezultatov lahko **sprejmemo hipotezo H4**, ki se glasi:

(H4) Razvita metoda prilagodljivega uglaševanja slojev konvolucijske nevronske mreže pri učenju s prenosom znanja dosega boljše rezultate v primerjavi s klasičnimi pristopi.

V statistični analizi smo predstavili tudi izračunane posteriorne verjetnosti klasifikacijske točnosti med različicami metode *DEFT* z in brez uporabe metrike *LDM*, iz katerih je razvidno, da sta metodi pri uporabi arhitekture *VGG16* statistično značilno enaki, pri uporabi arhitektur *ResNet50* in *MobileNet* pa statistično značilne enakosti nismo mogli potrditi. Pri uporabi arhitekture *ResNet50* je izračunana posteriorna verjetnost enakosti $0,8405$, pri uporabi arhitekture *MobileNet* pa $0,8993$. V splošnem lahko ugotovimo, da je klasifikacijska točnost primerjanih metod, ne glede na uporabljeno arhitekturo CNN, v vsaj $84,05\%$ primerov enaka. V sklopu statistične analize smo izračunali tudi posteriorne verjetnosti za primerjavo porabe epoh in časa primerjanih metod. Iz statistične analize metrike porabljenega števila epoh je razvidno, da metoda *DEFT* ob uporabi metrike *LDM* porabi statistično značilno manj epoh za učenje kot metoda brez uporabe metrike. Na podlagi statistične analize metrike porabljenega časa tega sicer ni mogoče trditi, čeprav je izmed šestih primerjav metoda *DEFT* z uporabo metrike *LDM* dosegla statistično značilno manjši čas v treh primerjavah. Pri primerjavi parov metod $DEFT_{st-epoh}$ in $DEFT - LDM_{st-epoh}$ se je izkazalo, da slednja porabi v splošnem več časa kot metoda $DEFT_{st-epoh}$. Čeprav metoda *DEFT* ob uporabi metrike *LDM* porabi statistično značilno manj epoh in čeprav je primerjava parov metod $DEFT_{st-resitev}$ in $DEFT - LDM_{st-resitev}$ pokazala statistično značilno razliko v prid metodi *DEFT* ob uporabi metrike *LDM*, lahko na podlagi vseh predstavljenih rezultatov sklenemo, da **ne sprejmemo hipoteze H5**, ki se glasi:

(H5) Razvita metoda z uporabo metrike *LDM* v krajšem času dosega primerljive ali boljše rezultate od razvite metode brez uporabe metrike *LDM*.

Z izračunom posteriorne verjetnosti primerjav parov metod smo ugotovili, da razvita metoda *DEFT* z uporabo metrike *LDM* v enakem času ne dosega boljših

rezultatov kot brez uporabe (poglavje 7.2). Izkaže se, da je ob uporabi arhitekture *VGG16* verjetnost, da metoda z uporabo metrike dosega boljše rezultate, enaka $0,9086$, medtem ko je ob uporabi arhitekture *MobileNet* potrjena statistično značilna enakost klasifikacijske uspešnosti. Ob uporabi arhitekture *ResNet50* je verjetnost, da sta metodi enako uspešni $0,5799$, verjetnost, da je uspešnejša metoda z uporabo metrike *LDM*, pa je enaka $0,4201$. S stališča časa, potrebnega za učenje posamezne primerjane metode, je na podlagi opravljene statistične analize mogoče ugotoviti, da metoda *DEFT* ob uporabi metrike *LDM* v vseh primerih ne dosega statistično značilno krajšega časa, medtem ko pa za učenje porabi statistično značilno manj epoh. Na podlagi predstavljenih izsledkov statistične analize lahko sklenemo, da **ne sprejmemo hipoteze H6**, ki se glasi:

(H6) Razvita metoda z uporabo metrike *LDM* v enakem času dosega boljše rezultate v primerjavi z metodo brez uporabe LDM.

8.2 Omejitve in možnosti aplikacije

V sklopu disertacije smo se zaradi obsežnosti raziskave omejili na uporabo konvolucijskih nevronskih mrež ter nadzorovanega učenja. V prihodnje bi lahko uporabo predlagane metode *DEFT* razširili na področje nenadzorovanega ali samonadzorovanega učenja kot tudi na druge naloge strojnega učenja. Predlagana metoda je namreč zasnovana tako, da jo je mogoče enostavno prilagoditi na različne arhitekture konvolucijskih nevronskih mrež, saj za svoje delovanje ne izrablja specifičnih lastnosti posameznih arhitektur. Eden izmed večjih in v zadnjem času bolj popularnih problemov, ki bi ga veljalo nasloviti z uporabo predlagane metode *DEFT* in metrike *LDM*, je zagotovo optimizacija hiperparametrov pri učenju globokih nevronskih mrež. V delu smo se zaradi lažje obvladljivosti raziskave omejili na izvedbo eksperimentov s tremi različnimi arhitekturami CNN (*VGG16*, *ResNet50* in *MobileNet*) nad tremi različnimi podatkovnimi zbirkami (*Osteosarcoma data*, *Športne slike* in *COVID-19*).

Nadaljnje delo bi bilo smiselno razširiti tudi na uporabo oz. primerjavo uporabe različnih optimizacijskih algoritmov v sklopu mehanizma izbire uglaševanih slojev metode *DEFT*. Trenutno namreč mehanizem temelji na osnovnem algoritmu DE. Z uporabo najuspešnejših algoritmov iz tekmovanja CEC, kot so jDE2021, NL-SHADE-RSP, MadDE in podobni, bi morda lahko uspešnost metode *DEFT* še dodatno izboljšali.

Predlagana metrika *LDM* se je izkazala kot uspešna za naslovitev problema algoritemske zahtevnosti metode *DEFT*, saj so eksperimenti ob uporabi metrike *LDM* za učenje porabili statistično značilno manj epoh kot eksperimenti brez uporabe metrike. V prihodnje bi lahko uporabo metrike *LDM* razširili tudi na sorodne

iterativne metode, ki temeljijo na optimizacijskih algoritmih. Podrobneje bi veljalo tudi analizirati učinkovitost preostalih dveh predlaganih metrik, *NL* in *AUNL*, ki smo ju razvili ob razvoju metrike *LDM*, saj sta že pri analizi, ki smo jo opravili v sklopu te raziskave, v določenih primerih dajali spodbudne rezultate.

Poglavje 9

Zaključek

"We crave for new sensations but soon become indifferent to them. The wonders of yesterday are today common occurrences."

- Nikola Tesla

V okviru doktorske disertacije smo si zastavili dva temeljna cilja: razvoj metode prilagodljivega uglaševanja slojev konvolucijske nevronske mreže ter razvoj lastne na funkciji izgube temelječe metrike, ki v zgodnjih fazah učenja omogoča zaznavo manj primernih izbir uglaševanih slojev konvolucijske nevronske mreže.

Zastavljena temeljna cilja sta nam služila kot vodilo pri izdelavi doktorske disertacije, ki smo jo začeli s pregledom področja strojnega učenja s poudarkom na klasifikacijskih metodah in vrednotenju klasifikacijskih modelov. V nadaljevanju smo predstavili področje globokega učenja ter pristop učenja s prenosom znanja. Nadaljevali smo z analizo vpliva izbire uglaševanih slojev konvolucijske nevronske mreže na uspešnost učenja, v sklopu katere smo pokazali, da je izbira uglaševanih slojev konvolucijske nevronske mreže odvisna od ciljnega problema ter uporabljene arhitekture globoke konvolucijske nevronske mreže, in s tem sprejeli hipotezo H1. Disertacijo smo nadaljevali s prilagodljivim uglaševanjem slojev konvolucijske mreže, v sklopu katerega smo predstavili metodo *DEFT*. Sledilo je zaznavanje primernosti izbir uglaševanih slojev konvolucijske nevronske mreže, kjer smo predstavili lastno na funkciji izgube temelječo metriko *LDM*, katere delovanje smo analizirali in ovrednotili ter na podlagi rezultatov sprejeli hipotezo H2, ki pravi, da razvita namenska metrika *LDM* omogoča zaznavo manj primernih izbir uglaševanih slojev konvolucijske nevronske mreže. Temu je sledila predstavitev eksperimentalnega ogrodja, kjer smo predstavili podatkovne zbirke, nastavitve, metode vrednotenja ter statistične metode. Sledil je del analize rezultatov, kjer smo predstavili rezultate in opravili statistično analizo, ki smo jo izvedli s kombinacijo klasičnih metod statistične analize

in napredne metode Bayesovega hierarhičnega koreliranega t-testa. Na podlagi rezultatov opravljene statistične analize smo sprejeli hipotezi H3 in H4. S tem smo potrdili, da predlagana metoda *DEFT* dosega višjo klasifikacijsko uspešnost kot klasične metode. Poleg tega smo potrdili tudi, da je metrika *LDM* primernejša za zaznavo manj primernih izbir uglaševanih slojev konvolucijske nevronske mreže. V sklopu statistične analize smo potrdili še, da predlagana metoda *DEFT* z uporabo metrike *LDM* dosega primerljive ali boljše rezultate klasifikacijske točnosti kot metoda *DEFT* brez uporabe metrike *LDM*. S stališča porabe časa nam opravljena statistična analiza ni podala enoznačnega odgovora, saj se je izkazalo, da metrika *LDM* pri uporabi v eksperimentih, kjer je kot zaustavitveni pogoj uporabljeno maksimalno število ovrednotenj uglaševanih izbir, omogoča, da v krajšem času dosežemo rezultate, primerljive z metodo brez uporabe metrike *LDM*. Nasprotno pa se je izkazalo pri eksperimentih, kjer je kot zaustavitveni pogoj uporabljeno maksimalno število porabljenih epoh. Po drugi strani se je, kot je razvidno iz statistične analize metrike porabljenega števila epoh, metrika *LDM* izkazala za uspešno, saj je ob uporabi le-te, število epoh statistično značilno manjše v vseh opravljenih primerjavah metod. Posledično hipoteze H5 ne sprejmemo. Izvedena statistična analiza je podala tudi odgovor na hipotezo H6, ki je prav tako ne sprejmemo, saj se je izkazalo, da predlagana metoda *DEFT* z uporabo metrike *LDM* v enakem času ne dosega statistično značilno višje klasifikacijske uspešnosti kot brez uporabe metrike.

Glede na cilje doktorske disertacije smo na začetku dela zastavili naslednjo tezo:

> *Z uporabo metode prilagodljivega uglaševanja slojev konvolucijske nevronske mreže pri učenju s prenosom znanja je mogoče uspešno nasloviti problem izbire slojev. Z vpeljavo zaznavanja manj primernih izbir slojev v tovrstno metodo lahko učinkovito zmanjšamo časovno zahtevnost ob doseganju primerljivih ali boljših rezultatov od primerjanih klasičnih metod učenja globokih konvolucijskih nevronskih mrež.*

V sklopu disertacije smo predlagali metodo prilagodljivega uglaševanja slojev konvolucijske nevronske mreže pri učenju s prenosom znanja (*DEFT*). Predlagana metoda v splošnem dosega višjo klasifikacijsko uspešnost kot primerjane klasične metode strojnega učenja, kar smo potrdili tudi s statistično analizo. Prav tako smo v sklopu disertacije predlagali na funkciji izgube temelječo metriko *LDM*, ki v zgodnji fazi učenja omogoča učinkovito zaznavo manj primernih izbir uglaševanih slojev konvolucijske nevronske mreže. Z uporabo predstavljene prilagodljive metode *DEFT* v kombinaciji z metriko *LDM* uspešno zmanjšamo porabljeno število epoh za izvedbo učenja ter v posameznih primerih tudi čas, potreben za izvedbo, ob doseganju primerljivih rezultatov, kar smo podprli tudi v sklopu statistične analize.

Na podlagi predstavljenih rezultatov lahko sklenemo, da celotno tezo doktorske disertacije sprejmemo. Glede na prikazane spodbudne rezultate predlagane metode in metrike lahko zaključimo, da metoda *DEFT* z uporabo metrike *LDM* učinkovito naslavlja problematiko ustrezne izbire uglaševanih slojev konvolucijskih nevronskih mrež, ob tem pa dosega visoko klasifikacijsko uspešnost.

Z opravljenimi raziskavami v sklopu doktorske disertacije smo ustvarili naslednje izvirne znanstvene prispevke:

(IZP1) Empirična analiza vpliva izbire uglaševanih slojev globoke konvolucijske nevronske mreže pri učenju s prenosom znanja (poglavje 3).

(IZP2) Razvita namenska metrika *LDM*, ki v zgodnjih fazah učenja omogoča zaznavo manj oz. bolj primernih izbir uglaševanih slojev konvolucijske nevronske mreže (poglavje 5).

(IZP3) Primerjava uspešnosti razvite metrike *LDM* s klasičnim pristopom predčasnega ustavljanja učenja (poglavje 7.2).

(IZP4) Razvita metoda prilagodljivega uglaševanja slojev konvolucijske nevronske mreže pri učenju s prenosom znanja (poglavje 4 in izvirni znanstveni članek [38]).

(IZP5) Podrobna primerjava in ovrednotenje rezultatov razvite metode z in brez uporabe metrike *LDM* s klasičnimi metodami in pristopi (poglavje 7).

Dodatek A

Spearmanovi koeficienti korelacije med metrikami

Tabela A.1 Vrednosti koeficientov korelacije v 2. epohi.

NL	AUNL	LDM
0.566229	0.566229	0.566229
0.572278	0.572278	0.572278
0.477201	0.477201	0.477201
0.425084	0.425084	0.425084
0.301281	0.301281	0.301281
0.558693	0.558693	0.558693
0.419168	0.419168	0.419168
0.680964	0.680964	0.680964
0.346025	0.346025	0.346025
0.742710	0.742710	0.742710

Tabela A.2 Vrednosti koeficientov korelacije v 3. epohi.

NL	AUNL	LDM
0.452298	0.538213	0.521528
0.467114	0.547412	0.532796
0.457403	0.475832	0.485419
0.569488	0.542682	0.567062
0.444294	0.402884	0.426602
0.570337	0.578525	0.578646
0.319169	0.382109	0.372438
0.560763	0.664941	0.653307
0.438589	0.419871	0.424486
0.748740	0.767397	0.765577

Tabela A.3 Vrednosti koeficientov korelacije v 4. epohi.

NL	AUNL	LDM
0.409839	0.483053	0.470477
0.500955	0.531280	0.524244
0.477824	0.500610	0.500610
0.567608	0.545593	0.550990
0.450832	0.446217	0.441217
0.703400	0.646693	0.671863
0.396083	0.397760	0.411063
0.379430	0.559758	0.517029
0.464999	0.463461	0.473589
0.644645	0.744075	0.720526

Tabela A.4 Vrednosti koeficientov korelacije v 5. epohi.

NL	AUNL	LDM
0.421792	0.461014	0.458898
0.537890	0.524972	0.537526
0.588891	0.520781	0.534353
0.565970	0.566274	0.571429
0.396730	0.448076	0.440512
0.678958	0.702429	0.706978
0.349353	0.408436	0.394741
0.000000	0.450292	0.409943
0.461217	0.466602	0.463461
0.595613	0.683553	0.670584

Tabela A.5 Vrednosti koeficientov korelacije v 6. epohi.

NL	AUNL	LDM
0.409092	0.455909	0.452298
0.536738	0.538254	0.553416
0.584035	0.573949	0.595116
0.573491	0.570822	0.571550
0.326666	0.418461	0.419230
0.601086	0.725718	0.713589
0.352931	0.391890	0.383897
0.000000	0.354205	0.326072
0.450448	0.466217	0.466409
0.287653	0.650561	0.624623

Tabela A.6 Vrednosti koeficientov korelacije v 7. epohi.

NL	AUNL	LDM
0.000000	0.432750	0.405232
0.314007	0.547169	0.544076
0.448065	0.585778	0.578680
0.525639	0.572824	0.583195
0.000000	0.404294	0.394358
0.512552	0.669679	0.618674
0.000000	0.356564	0.344378
0.000000	0.000000	0.000000
0.000000	0.451922	0.439871
0.000000	0.569447	0.396867

Tabela A.7 Vrednosti koeficientov korelacije v 8. epohi.

NL	AUNL	LDM
0.000000	0.352936	0.302508
0.000000	0.509264	0.457774
0.309114	0.574198	0.558136
0.422598	0.567729	0.550808
0.000000	0.362435	0.314935
0.000000	0.594839	0.557904
0.000000	0.332696	0.285296
0.000000	0.000000	0.000000
0.000000	0.412820	0.359679
-0.346522	0.000000	0.000000

Tabela A.8 Vrednosti koeficientov korelacije v 9. epohi.

NL	AUNL	LDM
0.000000	0.000000	0.000000
0.000000	0.378688	0.327744
0.000000	0.507209	0.451053
0.000000	0.524123	0.483246
0.000000	0.000000	0.000000
0.000000	0.501805	0.432787
0.000000	0.000000	0.000000
0.000000	0.000000	0.000000
0.000000	0.000000	0.000000
-0.556518	0.000000	0.000000

Tabela A.9 Vrednosti koeficientov korelacije v 10. epohi.

NL	AUNL	LDM
-0.448920	0.000000	0.000000
0.000000	0.000000	0.000000
0.000000	0.393901	0.322928
0.000000	0.406950	0.350184
0.000000	0.000000	0.000000
0.000000	0.346605	0.000000
0.000000	0.000000	0.000000
0.000000	0.000000	0.000000
0.000000	0.000000	0.000000
-0.624581	-0.279348	-0.356139

Tabela A.10 Vrednosti koeficientov korelacije v 11. epohi.

NL	AUNL	LDM
-0.600812	0.000000	-0.377216
-0.373310	0.000000	0.000000
0.000000	0.000000	0.000000
0.000000	0.000000	0.000000
-0.294500	0.000000	0.000000
-0.392501	0.000000	0.000000
-0.284417	0.000000	0.000000
-0.319250	0.000000	0.000000
-0.354389	0.000000	0.000000
-0.680376	-0.447833	-0.487878

Tabela A.11 Vrednosti koeficientov korelacije v 12. epohi.

NL	AUNL	LDM
-0.645291	-0.434119	-0.478695
-0.476553	0.000000	0.000000
-0.382127	0.000000	0.000000
-0.356789	0.000000	0.000000
-0.420724	0.000000	0.000000
-0.527325	0.000000	0.000000
-0.413249	0.000000	0.000000
-0.374353	0.000000	0.000000
-0.492236	0.000000	0.000000
-0.724540	-0.551700	-0.585488

Tabela A.12 Vrednosti koeficientov korelacije v 13. epohi.

NL	AUNL	LDM
-0.681436	-0.517295	-0.559132
-0.589239	0.000000	-0.307548
-0.472233	0.000000	0.000000
-0.381655	0.000000	0.000000
-0.571762	0.000000	0.000000
-0.651418	0.000000	-0.338297
-0.471394	0.000000	0.000000
-0.504138	0.000000	0.000000
-0.549563	0.000000	-0.284294
-0.760080	-0.614156	-0.670811

Tabela A.13 Vrednosti koeficientov korelacije v 14. epohi.

NL	AUNL	LDM
-0.712916	-0.599225	-0.627739
-0.663307	-0.370561	-0.413501
-0.618363	-0.288686	-0.358290
-0.554020	0.000000	0.000000
-0.599424	0.000000	-0.321153
-0.707166	-0.422780	-0.495194
-0.518865	-0.290718	-0.323753
-0.628520	-0.301217	-0.363142
-0.581342	-0.318076	-0.334871
-0.773260	-0.700276	-0.707898

Tabela A.14 Vrednosti koeficientov korelacije v 15. epohi.

NL	AUNL	LDM
-0.718927	-0.637825	-0.648906
-0.731340	-0.471359	-0.500228
-0.677573	-0.398882	-0.447940
-0.660562	-0.303727	-0.331928
-0.661847	-0.391089	-0.443525
-0.725608	-0.569791	-0.606666
-0.632974	-0.362712	-0.378419
-0.738855	-0.404919	-0.441831
-0.613262	-0.374166	-0.424294
-0.788803	-0.719161	-0.720412

Tabela A.15 Vrednosti koeficientov korelacije v 16. epohi.

NL	AUNL	LDM
-0.731186	-0.654634	-0.665716
-0.754978	-0.546624	-0.591746
-0.721331	-0.481559	-0.501108
-0.712689	-0.350851	-0.370440
-0.639898	-0.506345	-0.544550
-0.727116	-0.643540	-0.669133
-0.699996	-0.445551	-0.493454
-0.805756	-0.473084	-0.507564
-0.679671	-0.483332	-0.513653
-0.793384	-0.734064	-0.735088

Tabela A.16 Vrednosti koeficientov korelacije v 17. epohi.

NL	AUNL	LDM
-0.749287	-0.685763	-0.691988
-0.764960	-0.616854	-0.647543
-0.743115	-0.551412	-0.594120
-0.719599	-0.387846	-0.423144
-0.664274	-0.558076	-0.584678
-0.733846	-0.690724	-0.696789
-0.720343	-0.528557	-0.577802
-0.840987	-0.530462	-0.555898
-0.696578	-0.561153	-0.567563
-0.791728	-0.744758	-0.749308

Tabela A.17 Vrednosti koeficientov korelacije v 18. epohi.

NL	AUNL	LDM
-0.747232	-0.695101	-0.695101
-0.771268	-0.672044	-0.687025
-0.742063	-0.623630	-0.650276
-0.725226	-0.470267	-0.511266
-0.667307	-0.590448	-0.596217
-0.749411	-0.707706	-0.714620
-0.758485	-0.625817	-0.648567
-0.854864	-0.601165	-0.638500
-0.712859	-0.595191	-0.616794
-0.800117	-0.757386	-0.759661

Tabela A.18 Vrednosti koeficientov korelacije v 19. epohi.

NL	AUNL	LDM
-0.746043	-0.697716	-0.708922
-0.766422	-0.708070	-0.718198
-0.745300	-0.669700	-0.672066
-0.759353	-0.561179	-0.600237
-0.698283	-0.604614	-0.614165
-0.757530	-0.722928	-0.729357
-0.767656	-0.665672	-0.674894
-0.850096	-0.682392	-0.707881
-0.711613	-0.625960	-0.641922
-0.791671	-0.767624	-0.777636

Tabela A.19 Vrednosti koeficientov korelacije v 20. epohi.

NL	AUNL	LDM
-0.745896	-0.738806	-0.741919
-0.764474	-0.723899	-0.730570
-0.745723	-0.701202	-0.722993
-0.747771	-0.617582	-0.622919
-0.687897	-0.623845	-0.623845
-0.762364	-0.733603	-0.747066
-0.778435	-0.697141	-0.718270
-0.853817	-0.725068	-0.753254
-0.714292	-0.659358	-0.667755
-0.786998	-0.781390	-0.781390

Tabela A.20 Vrednosti koeficientov korelacije v 21. epohi.

NL	AUNL	LDM
-0.743585	-0.747397	-0.750510
-0.764109	-0.740213	-0.740213
-0.745443	-0.738432	-0.742168
-0.734011	-0.636808	-0.650090
-0.717258	-0.623845	-0.629614
-0.761724	-0.751009	-0.750887
-0.777442	-0.753205	-0.770310
-0.843273	-0.767268	-0.783715
-0.717321	-0.682178	-0.686794
-0.789278	-0.785599	-0.793904

Tabela A.21 Vrednosti koeficientov korelacije v 22. epohi.

NL	AUNL	LDM
-0.745466	-0.750510	-0.750510
-0.762102	-0.755557	-0.761622
-0.740194	-0.745903	-0.745903
-0.755359	-0.662705	-0.679686
-0.719905	-0.639678	-0.643076
-0.763841	-0.751858	-0.751858
-0.778589	-0.776570	-0.785011
-0.823785	-0.794079	-0.813646
-0.723446	-0.690191	-0.699870
-0.794551	-0.793904	-0.798000

Tabela A.22 Vrednosti koeficientov korelacije v 23. epohi.

NL	AUNL	LDM
-0.739463	-0.750510	-0.750510
-0.760797	-0.766353	-0.769324
-0.740604	-0.745903	-0.745903
-0.754657	-0.687267	-0.704431
-0.724192	-0.651986	-0.661665
-0.761110	-0.751858	-0.751494
-0.774364	-0.787973	-0.786184
-0.813300	-0.819833	-0.828823
-0.723204	-0.707050	-0.707178
-0.788253	-0.800502	-0.800502

Tabela A.23 Vrednosti koeficientov korelacije v 24. epohi.

NL	AUNL	LDM
-0.740840	-0.750510	-0.750510
-0.758956	-0.772660	-0.772660
-0.739947	-0.745903	-0.745903
-0.764982	-0.726992	-0.737666
-0.731526	-0.677755	-0.687819
-0.769064	-0.752100	-0.751373
-0.775844	-0.786184	-0.785625
-0.811188	-0.841991	-0.853255
-0.718616	-0.707178	-0.707178
-0.793240	-0.800502	-0.805508

Tabela A.24 Vrednosti koeficientov korelacije v 25. epohi.

NL	AUNL	LDM
-0.740590	-0.749265	-0.747522
-0.756180	-0.772660	-0.775874
-0.738696	-0.745903	-0.745903
-0.763576	-0.741972	-0.753192
-0.722325	-0.692691	-0.698717
-0.765733	-0.751373	-0.751858
-0.774648	-0.788923	-0.790433
-0.819137	-0.856745	-0.863091
-0.717940	-0.707178	-0.707178
-0.786162	-0.803801	-0.803801

Tabela A.25 Vrednosti koeficientov korelacije v 26. epohi.

NL	AUNL	LDM
-0.737823	-0.747522	-0.747522
-0.754826	-0.775874	-0.775874
-0.738107	-0.745903	-0.745903
-0.764499	-0.754284	-0.757498
-0.723935	-0.702627	-0.705768
-0.765205	-0.751736	-0.751736
-0.780061	-0.790656	-0.792613
-0.785137	-0.863091	-0.863091
-0.722362	-0.707178	-0.707178
-0.779081	-0.803801	-0.803801

Tabela A.26 Vrednosti koeficientov korelacije v 27. epohi.

NL	AUNL	LDM
-0.740354	-0.747522	-0.747522
-0.754983	-0.775874	-0.775874
-0.737482	-0.745903	-0.745903
-0.766985	-0.757498	-0.767141
-0.731354	-0.705383	-0.705383
-0.767658	-0.751494	-0.751494
-0.776499	-0.792613	-0.795575
-0.782652	-0.864836	-0.866369
-0.731437	-0.707178	-0.707178
-0.777424	-0.808352	-0.808352

Tabela A.27 Vrednosti koeficientov korelacije v 28. epohi.

NL	AUNL	LDM
-0.739804	-0.747522	-0.747522
-0.755433	-0.775874	-0.775874
-0.732045	-0.745903	-0.745903
-0.774175	-0.767141	-0.768111
-0.731482	-0.708652	-0.712306
-0.769389	-0.751494	-0.751494
-0.780848	-0.796022	-0.796022
-0.782520	-0.866369	-0.870389
-0.728177	-0.714870	-0.721152
-0.775055	-0.808352	-0.808352

Tabela A.28 Vrednosti koeficientov korelacije v 29. epohi.

NL	AUNL	LDM
-0.741442	-0.747522	-0.747522
-0.756108	-0.775874	-0.775874
-0.736875	-0.745903	-0.745903
-0.767321	-0.768111	-0.768111
-0.730497	-0.719806	-0.718524
-0.766222	-0.751494	-0.751494
-0.777407	-0.794904	-0.794904
-0.781938	-0.871922	-0.871922
-0.724813	-0.721152	-0.724037
-0.772846	-0.810855	-0.810855

Tabela A.29 Vrednosti koeficientov korelacije v 30. epohi.

NL	AUNL	LDM
-0.739160	-0.747522	-0.747522
-0.756406	-0.775874	-0.775874
-0.732752	-0.745903	-0.745903
-0.763961	-0.768111	-0.768961
-0.728202	-0.725063	-0.728460
-0.769459	-0.751494	-0.751494
-0.781655	-0.793451	-0.791439
-0.784706	-0.874196	-0.874196
-0.722917	-0.721986	-0.721217
-0.770332	-0.810855	-0.814609

Literatura

[1] J. Flisar and V. Podgorelec, "Identification of self-admitted technical debt using enhanced feature selection based on word embedding," *IEEE Access*, vol. 7, pp. 106475–106494, 2019.

[2] R. Gupta, S. Pal, A. Kanade, and S. Shevade, "Deepfix: Fixing common C language errors by deep learning," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 31, 2017.

[3] M. White, M. Tufano, C. Vendome, and D. Poshyvanyk, "Deep learning code fragments for code clone detection," in *2016 31st IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pp. 87–98, 2016.

[4] R. Kumar, Z. Xiaosong, R. U. Khan, I. Ahad, and J. Kumar, "Malicious code detection based on image processing using deep learning," in *Proceedings of the 2018 International Conference on Computing and Artificial Intelligence*, pp. 81–85, 2018.

[5] G. Polančič, S. Jagečič, and K. Kous, "An empirical investigation of the effectiveness of optical recognition of hand-drawn business process elements by applying machine learning," *IEEE Access*, vol. 8, pp. 206118–206131, 2020.

[6] N. Mehdiyev, J. Evermann, and P. Fettke, "A novel business process prediction model using a deep learning method," *Business & information systems engineering*, vol. 62, no. 2, pp. 143–157, 2020.

[7] P. Valter, P. Lindgren, and R. Prasad, "Advanced business model innovation supported by artificial intelligence and deep learning," *Wireless Personal Communications*, vol. 100, no. 1, pp. 97–111, 2018.

[8] K. Amin, S. Kapetanakis, K.-D. Althoff, A. Dengel, and M. Petridis, "Dynamic process workflow routing using deep learning," in *International Conference on Innovative Techniques and Applications of Artificial Intelligence*, pp. 132–142, 2018.

[9] A. L. Tarca, V. J. Carey, X.-w. Chen, R. Romero, and S. Drăghici, "Machine learning and its applications to biology," *PLoS Comput Biol*, vol. 3, no. 6, p. e116, 2007.

[10] B. Tang, Z. Pan, K. Yin, and A. Khateeb, "Recent advances of deep learning in bioinformatics and computational biology," *Frontiers in genetics*, vol. 10, p. 214, 2019.

[11] J. Zou, M. Huss, A. Abid, P. Mohammadi, A. Torkamani, and A. Telenti, "A primer on deep learning in genomics," *Nature genetics*, vol. 51, no. 1, pp. 12–18, 2019.

[12] H. Wu, C. Cao, X. Xia, and Q. Lü, "Unified deep learning architecture for modeling biology sequence," *IEEE/ACM transactions on computational biology and bioinformatics*, vol. 15, no. 5, pp. 1445–1452, 2017.

[13] R. Miotto, F. Wang, S. Wang, X. Jiang, and J. T. Dudley, "Deep learning for healthcare: review, opportunities and challenges," *Briefings in bioinformatics*, vol. 19, no. 6, pp. 1236–1246, 2018.

[14] O. Faust, Y. Hagiwara, T. J. Hong, O. S. Lih, and U. R. Acharya, "Deep learning for healthcare applications based on physiological signals: A review," *Computer methods and programs in biomedicine*, vol. 161, pp. 1–13, 2018.

[15] P. Mamoshina, A. Vieira, E. Putin, and A. Zhavoronkov, "Applications of deep learning in biomedicine," *Molecular pharmaceutics*, vol. 13, no. 5, pp. 1445–1454, 2016.

[16] M. Wainberg, D. Merico, A. Delong, and B. J. Frey, "Deep learning in biomedicine," *Nature biotechnology*, vol. 36, no. 9, pp. 829–838, 2018.

[17] C. Cao, F. Liu, H. Tan, D. Song, W. Shu, W. Li, Y. Zhou, X. Bo, and Z. Xie, "Deep learning and its applications in biomedicine," *Genomics, proteomics & bioinformatics*, vol. 16, no. 1, pp. 17–32, 2018.

[18] G. Vrbančič and V. Podgorelec, "Efficient ensemble for image-based identification of pneumonia utilizing deep cnn and sgd with warm restarts," *Expert Systems with Applications*, vol. 187, p. 115834, 2022.

[19] M. Žitnik, F. Nguyen, B. Wang, J. Leskovec, A. Goldenberg, and M. M. Hoffman, "Machine learning for integrating data in biology and medicine: Principles, practice, and opportunities," *Information Fusion*, vol. 50, pp. 71–91, 2019.

[20] G. Vrbančič and V. Podgorelec, "Automatic classification of motor impairment neural disorders from EEG signals using deep convolutional neural networks," *Elektronika ir Elektrotechnika*, vol. 24, no. 4, pp. 3–7, 2018.

[21] G. Vrbančič, V. Podgorelec, *et al.*, "Automatic detection of heartbeats in heart sound signals using deep convolutional neural networks," *Elektronika ir Elektrotechnika*, vol. 25, no. 3, pp. 71–76, 2019.

[22] T. Ching, D. S. Himmelstein, B. K. Beaulieu-Jones, A. A. Kalinin, B. T. Do, G. P. Way, E. Ferrero, P.-M. Agapow, M. Zietz, M. M. Hoffman, *et al.*, "Opportunities and obstacles for deep learning in biology and medicine," *Journal of The Royal Society Interface*, vol. 15, no. 141, p. 20170387, 2018.

[23] A. Akay and H. Hess, "Deep learning: current and emerging applications in medicine and technology," *IEEE journal of biomedical and health informatics*, vol. 23, no. 3, pp. 906–920, 2019.

[24] F. Wang, L. P. Casalino, and D. Khullar, "Deep learning in medicine—promise, progress, and challenges," *JAMA internal medicine*, vol. 179, no. 3, pp. 293–294, 2019.

[25] S. Pouyanfar, S. Sadiq, Y. Yan, H. Tian, Y. Tao, M. P. Reyes, M.-L. Shyu, S.-C. Chen, and S. Iyengar, "A survey on deep learning: Algorithms, techniques, and applications," *ACM Computing Surveys (CSUR)*, vol. 51, no. 5, pp. 1–36, 2018.

[26] X.-W. Chen and X. Lin, "Big data deep learning: challenges and perspectives," *IEEE access*, vol. 2, pp. 514–525, 2014.

[27] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization," in *25th annual conference on neural information processing systems (NIPS 2011)*, vol. 24, 2011.

[28] V. Podgorelec, Š. Pečnik, and G. Vrbančič, "Classification of similar sports images using convolutional neural network with hyper-parameter optimization," *Applied Sciences*, vol. 10, no. 23, p. 8494, 2020.

[29] G. Vrbančič, I. Fister Jr, and V. Podgorelec, "Parameter setting for deep neural networks using swarm intelligence on phishing websites classification," *International Journal on Artificial Intelligence Tools*, vol. 28, no. 06, p. 1960008, 2019.

[30] A. Khan, A. Sohail, U. Zahoora, and A. S. Qureshi, "A survey of the recent architectures of deep convolutional neural networks," *Artificial Intelligence Review*, vol. 53, no. 8, pp. 5455–5516, 2020.

[31] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2009.

[32] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu, "A survey on deep transfer learning," in *International conference on artificial neural networks*, pp. 270–279, 2018.

[33] R. Mehra *et al.*, "Breast cancer histology images classification: Training from scratch or transfer learning?," *ICT Express*, vol. 4, no. 4, pp. 247–254, 2018.

[34] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2009.

[35] M. Huh, P. Agrawal, and A. A. Efros, "What makes ImageNet good for transfer learning?," *arXiv preprint arXiv:1608.08614*, 2016.

[36] M. Hussain, J. J. Bird, and D. R. Faria, "A study on CNN transfer learning for image classification," in *UK Workshop on computational Intelligence*, pp. 191–202, 2018.

[37] Y. Guo, H. Shi, A. Kumar, K. Grauman, T. Rosing, and R. Feris, "Spottune: transfer learning through adaptive fine-tuning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4805–4814, 2019.

[38] G. Vrbančič and V. Podgorelec, "Transfer learning with adaptive fine-tuning," *IEEE Access*, vol. 8, pp. 196197–196211, 2020.

[39] G. Vrbančič, Š. Pečnik, and V. Podgorelec, "Identification of covid-19 x-ray images using CNN with optimized tuning of transfer learning," in *2020 International Conference on INnovations in Intelligent SysTems and Applications (INISTA)*, pp. 1–8, 2020.

[40] E. C. Orenstein and O. Beijbom, "Transfer learning and deep feature extraction for planktonic image data sets," in *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 1082–1088, 2017.

[41] M. Long, J. Wang, G. Ding, J. Sun, and P. S. Yu, "Transfer feature learning with joint distribution adaptation," in *Proceedings of the IEEE international conference on computer vision*, pp. 2200–2207, 2013.

[42] R. Poojary and A. Pai, "Comparative study of model optimization techniques in fine-tuned CNN models," in *2019 International Conference on Electrical and Computing Technologies and Applications (ICECTA)*, pp. 1–4, 2019.

[43] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?," in *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, (Cambridge, MA, USA), p. 3320–3328, 2014.

[44] H. Azizpour, A. S. Razavian, J. Sullivan, A. Maki, and S. Carlsson, "Factors of transferability for a generic convnet representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 9, pp. 1790–1802, 2016.

[45] N. Tajbakhsh, J. Y. Shin, S. R. Gurudu, R. T. Hurst, C. B. Kendall, M. B. Gotway, and J. Liang, "Convolutional neural networks for medical image analysis: Full training or fine tuning?," *IEEE Transactions on Medical Imaging*, vol. 35, no. 5, pp. 1299–1312, 2016.

[46] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *3rd International Conference on Learning Representations, ICLR 2015, Conference Track Proceedings*, 2015.

[47] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[48] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *Computing Research Repository (CoRR)*, vol. abs/1704.04861, 2017.

[49] T. Mitchell, *Machine Learning*. McGraw-Hill Education, 1997.

[50] Y. Bengio, I. Goodfellow, and A. Courville, *Deep learning*, vol. 1. MIT press Massachusetts, USA, 2017.

[51] M. J. Zaki and W. Meira Jr, *Data Mining and Machine Learning: Fundamental Concepts and Algorithms*. Cambridge University Press, 2019.

[52] T. Hastie, R. Tibshirani, and J. Friedman, *The elements of statistical learning: data mining, inference, and prediction*. Springer Science & Business Media, 2009.

[53] C. C. Aggarwa *et al.*, *Data Classification: Algorithms and Applications*. Chapman & Hall/CRC Data Mining and Knowledge Discovery Series, 2015.

[54] D. W. Aha, "Lazy learning," *Artificial Intelligence: Special issue editorial*, pp. 7–10, 1997.

[55] D. Barber, *Bayesian reasoning and machine learning*. Cambridge University Press, 2012.

[56] S. C. Larson, "The shrinkage of the coefficient of multiple correlation.," *Journal of Educational Psychology*, vol. 22, no. 1, p. 45, 1931.

[57] L. Devroye and T. Wagner, "Distribution-free performance bounds for potential function rules," *IEEE Transactions on Information Theory*, vol. 25, no. 5, pp. 601–604, 1979.

[58] M. Stone, "Cross-validatory choice and assessment of statistical predictions," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 36, no. 2, pp. 111–133, 1974.

[59] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *The Journal of Machine Learning Research*, vol. 7, pp. 1–30, 2006.

[60] D. Ballabio, F. Grisoni, and R. Todeschini, "Multivariate comparison of classification performance measures," *Chemometrics and Intelligent Laboratory Systems*, vol. 174, pp. 33–44, 2018.

[61] C. Ferri, J. Hernández-Orallo, and R. Modroiu, "An experimental comparison of performance measures for classification," *Pattern Recognition Letters*, vol. 30, no. 1, pp. 27–38, 2009.

[62] J. Cohen, "A coefficient of agreement for nominal scales," *Educational and psychological measurement*, vol. 20, no. 1, pp. 37–46, 1960.

[63] J. Jelenčič, *Napovedovanje vrednosti kriptovalut z globokim učenjem; Magistrsko delo*. Univerza v Ljubljani, Fakulteta za matematiko in fiziko, 2019.

[64] T. Fawcett, "Using rule sets to maximize ROC performance," in *Proceedings 2001 IEEE International Conference on Data Mining*, pp. 131–138, 2001.

[65] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," in *ICML*, 2010.

[66] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[67] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256, 2010.

[68] A. M. Saxe, J. L. McClelland, and S. Ganguli, "Exact solutions to the nonlinear dynamics of learning in deep linear neural networks," in *2nd International Conference on Learning Representations (ICLR)*, 2014.

[69] Y. Bengio, "Practical recommendations for gradient-based training of deep architectures," in *Neural networks: Tricks of the trade*, pp. 437–478, Springer, 2012.

[70] J. Han, M. Kamber, and J. Pei, *Data mining concepts and techniques, Third edition*, vol. 5. Elsevier, 2011.

[71] C. M. Bishop, *Pattern recognition and machine learning*. Springer Science+ Business Media, 2006.

[72] N. Qian, "On the momentum term in gradient descent learning algorithms," *Neural networks*, vol. 12, no. 1, pp. 145–151, 1999.

[73] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations (ICLR)*, 2015.

[74] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "Gans trained by a two time-scale update rule converge to a local nash equilibrium," *Advances in neural information processing systems*, vol. 30, 2017.

[75] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*, pp. 448–456, 2015.

[76] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[77] F. Chollet *et al.*, *Deep learning with Python*. Manning, New York, 2018.

[78] Y. Le Cun, L. Jackel, B. Boser, J. Denker, H. Graf, I. Guyon, D. Henderson, R. Howard, and W. Hubbard, "Handwritten digit recognition: applications of neural network chips and automatic learning," *IEEE Communications Magazine*, vol. 27, no. 11, pp. 41–46, 1989.

[79] S. Dodge and L. Karam, "A study and comparison of human and deep learning recognition performance under visual distortions," in *26th international conference on computer communication and networks (ICCCN)*, pp. 1–7, 2017.

[80] Y. C. Goh, X. Q. Cai, W. Theseira, G. Ko, and K. A. Khor, "Evaluating human versus machine learning performance in classifying research abstracts," *Scientometrics*, vol. 125, no. 2, pp. 1197–1212, 2020.

[81] D. H. Hubel and T. N. Wiesel, "Receptive fields, binocular interaction and functional architecture in the cat's visual cortex," *The Journal of physiology*, vol. 160, no. 1, pp. 106–154, 1962.

[82] Y.-T. Zhou and R. Chellappa, "Computation of optical flow using a neural network.," in *ICNN*, pp. 71–78, 1988.

[83] L. Alzubaidi, J. Zhang, A. J. Humaidi, A. Al-Dujaili, Y. Duan, O. Al-Shamma, J. Santamaría, M. A. Fadhel, M. Al-Amidie, and L. Farhan, "Review of deep learning: concepts, CNN architectures, challenges, applications, future directions," *Journal of Big Data*, vol. 8, no. 1, pp. 1–74, 2021.

[84] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015.

[85] A. Blum and T. Mitchell, "Combining labeled and unlabeled data with co-training," in *Proceedings of the eleventh annual conference on Computational learning theory*, pp. 92–100, 1998.

[86] O. Chapelle, B. Scholkopf, and A. Zien, "Semi-supervised learning," *IEEE Transactions on Neural Networks*, vol. 20, no. 3, pp. 542–542, 2009.

[87] J. E. Van Engelen and H. H. Hoos, "A survey on semi-supervised learning," *Machine Learning*, vol. 109, no. 2, pp. 373–440, 2020.

[88] D. D. Lewis and W. A. Gale, "A sequential algorithm for training text classifiers," in *SIGIR'94*, pp. 3–12, 1994.

[89] B. Settles, "Active learning literature survey," tech. rep., University of Wisconsin-Madison Department of Computer Sciences, 2009.

[90] J. Y. Ching, A. K. C. Wong, and K. C. C. Chan, "Class-dependent discretization for inductive learning from continuous and mixed-mode data," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 17, no. 7, pp. 641–651, 1995.

[91] S. Dumais, J. Platt, D. Heckerman, and M. Sahami, "Inductive learning algorithms and representations for text categorization," in *7th International Conference on Information and Knowledge Management*, pp. 148–152, January 1998.

[92] X. Zhu and X. Wu, "Class noise handling for effective cost-sensitive learning by cost-guided iterative classification filtering," *IEEE Transactions on Knowledge & Data Engineering*, vol. 18, no. 10, pp. 1435–1440, 2006.

[93] Q. Yang, C. Ling, X. Chai, and R. Pan, "Test-cost sensitive classification on data with missing values," *IEEE Transactions on Knowledge & Data Engineering*, vol. 18, no. 5, pp. 626–638, 2006.

[94] M. Long, Y. Cao, J. Wang, and M. Jordan, "Learning transferable features with deep adaptation networks," in *International conference on machine learning*, pp. 97–105, 2015.

[95] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 580–587, 2014.

[96] L. Xuhong, Y. Grandvalet, and F. Davoine, "Explicit inductive bias for transfer learning with convolutional networks," in *International Conference on Machine Learning*, pp. 2825–2834, 2018.

[97] Z. Yang, J. J. Zhao, B. Dhingra, K. He, W. W. Cohen, R. Salakhutdinov, and Y. LeCun, "GLoMo: Unsupervisedly learned relational graphs as transferable representations," *Computing Research Repository (CoRR)*, vol. abs/1806.05662, 2018.

[98] S. A. Doumari, H. Givi, M. Dehghani, Z. Montazeri, V. Leiva, and J. M. Guerrero, "A new two-stage algorithm for solving optimization problems," *Entropy*, vol. 23, no. 4, p. 491, 2021.

[99] L. Yang and A. Shami, "On hyperparameter optimization of machine learning algorithms: Theory and practice," *Neurocomputing*, vol. 415, pp. 295–316, 2020.

[100] Q. Al-Tashi, S. J. Abdulkadir, H. M. Rais, S. Mirjalili, and H. Alhussian, "Approaches to multi-objective feature selection: A systematic literature review," *IEEE Access*, vol. 8, pp. 125076–125096, 2020.

[101] L. Brezočnik, I. Fister, and V. Podgorelec, "Swarm intelligence algorithms for feature selection: A review," *Applied Sciences*, vol. 8, no. 9, 2018.

[102] J. Tang, S. Alelyani, and H. Liu, "Feature selection for classification: A review," *Data classification: Algorithms and applications*, p. 37, 2014.

[103] A. Berthelier, T. Chateau, S. Duffner, C. Garcia, and C. Blanc, "Deep model compression and architecture optimization for embedded systems: A survey," *Journal of Signal Processing Systems*, vol. 93, no. 8, pp. 863–878, 2021.

[104] G. Vrbančič, I. Fister Jr, and V. Podgorelec, "Designing deep neural network topologies with population-based metaheuristics," in *Central European Conference on Information and Intelligent Systems*, pp. 163–170, 2018.

[105] I. Fister, *Avtomatsko načrtovanje in vrednotenje klasifikacijskih cevovodov v bioinformatiki; Magistrsko delo*. Univerza v Mariboru, Fakulteta za zdravstvene vede, 2019.

[106] S. Das and P. N. Suganthan, "Differential evolution: A survey of the state-of-the-art," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 1, pp. 4–31, 2011.

[107] Z. Michalewicz, *Genetic algorithms+ data structures= evolution programs*. Springer Science & Business Media, 2013.

[108] A. E. Eiben, J. E. Smith, *et al.*, *Introduction to evolutionary computing*, vol. 53. Springer, 2003.

[109] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95-international conference on neural networks*, vol. 4, pp. 1942–1948, 1995.

[110] M. Dorigo, M. Birattari, and T. Stutzle, "Ant colony optimization," *IEEE computational intelligence magazine*, vol. 1, no. 4, pp. 28–39, 2006.

[111] X.-S. Yang and A. H. Gandomi, "Bat algorithm: a novel approach for global engineering optimization," *Engineering computations*, 2012.

[112] K. Price and R. Storn, "Differential evolution-a simple and efficient adaptive scheme for global optimization over continuous space," *Technical Report, International Computer Science Institute*, 1995.

[113] K. M. Sallam, S. M. Elsayed, R. K. Chakrabortty, and M. J. Ryan, "Improved multi-operator differential evolution algorithm for solving unconstrained problems," in *2020 IEEE Congress on Evolutionary Computation (CEC)*, pp. 1–8, 2020.

[114] J. Brest, M. S. Maučec, and B. Bošković, "Differential evolution algorithm for single objective bound-constrained optimization: Algorithm j2020," in *2020 IEEE Congress on Evolutionary Computation (CEC)*, pp. 1–8, 2020.

[115] J. Brest, M. S. Maučec, and B. Bošković, "Self-adaptive differential evolution algorithm with population size reduction for single objective bound-constrained optimization: Algorithm j21," in *2021 IEEE Congress on Evolutionary Computation (CEC)*, pp. 817–824, 2021.

[116] V. Stanovov, S. Akhmedova, and E. Semenkin, "Nl-shade-rsp algorithm with adaptive archive and selective pressure for cec 2021 numerical optimization," in *2021 IEEE Congress on Evolutionary Computation (CEC)*, pp. 809–816, 2021.

[117] S. Biswas, D. Saha, S. De, A. D. Cobb, S. Das, and B. A. Jalaian, "Improving differential evolution through bayesian hyperparameter optimization," in *2021 IEEE Congress on Evolutionary Computation (CEC)*, pp. 832–840, 2021.

[118] A. H. Land and A. G. Doig, "An automatic method of solving discrete programming problems," *Econometrica*, vol. 28, no. 3, pp. 497–520, 1960.

[119] E. Hansen and G. W. Walster, *Global optimization using interval analysis: revised and expanded*, vol. 264. CRC Press, 2003.

[120] A. V. Levy and S. Gómez, "The tunneling method applied to global optimization," *Numerical optimization*, vol. 1981, pp. 213–244, 1985.

[121] Z. B. Zabinsky, "Random search algorithms," *Wiley encyclopedia of operations research and management science*, 2010.

[122] I. Fister, I. Fister Jr, X.-S. Yang, and J. Brest, "A comprehensive review of firefly algorithms," *Swarm and Evolutionary Computation*, vol. 13, pp. 34–46, 2013.

[123] M. Abdel-Basset, L. Abdel-Fatah, and A. K. Sangaiah, "Chapter 10 - metaheuristic algorithms: A comprehensive review," in *Computational Intelligence for Multimedia Big Data on the Cloud with Engineering Applications* (A. K. Sangaiah, M. Sheng, and Z. Zhang, eds.), Intelligent Data-Centric Systems, pp. 185–231, Academic Press, 2018.

[124] U. Škvorc, T. Eftimov, and P. Korošec, "Cec real-parameter optimization competitions: Progress from 2013 to 2018," in *2019 IEEE Congress on Evolutionary Computation (CEC)*, pp. 3126–3133, IEEE, 2019.

[125] J. N. Kather, C.-A. Weis, F. Bianconi, S. M. Melchers, L. R. Schad, T. Gaiser, A. Marx, and F. G. Zöllner, "Multi-class texture analysis in colorectal cancer histology," *Scientific reports*, vol. 6, no. 1, pp. 1–11, 2016.

[126] J. N. Kather, A. Marx, C. C. Reyes-Aldasoro, L. R. Schad, F. G. Zöllner, and C.-A. Weis, "Continuous representation of tumor microvessel density and detection of angiogenic hotspots in histological whole-slide images," *Oncotarget*, vol. 6, no. 22, p. 19163, 2015.

[127] A. Olsen, D. A. Konovalov, B. Philippa, P. Ridd, J. C. Wood, J. Johns, W. Banks, B. Girgenti, O. Kenny, J. Whinney, *et al.*, "Deepweeds: A multiclass weed species image dataset for deep learning," *Scientific reports*, vol. 9, no. 1, pp. 1–12, 2019.

[128] Y. Yang and S. Newsam, "Bag-of-visual-words and spatial extensions for land-use classification," in *Proceedings of the 18th SIGSPATIAL international conference on advances in geographic information systems*, pp. 270–279, 2010.

[129] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization.," *Journal of machine learning research*, vol. 13, no. 2, 2012.

[130] X. Yang, Q. Liu, R. Su, R. Tang, Z. Liu, and X. He, "Autoft: Automatic fine-tune for parameters transfer learning in click-through rate prediction," *arXiv preprint arXiv:2106.04873*, 2021.

[131] L. Chen, F. Yuan, J. Yang, X. Ao, C. Li, and M. Yang, "A user-adaptive layer selection framework for very deep sequential recommender models," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, pp. 3984–3991, 2021.

[132] R. Storn and K. Price, "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces," *Journal of global optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[133] D. Fister, I. Fister, T. Jagric, I. Fister Jr., and J. Brest, "A novel self-adaptive differential evolution for feature selection using threshold mechanism," in *IEEE SSCI2018 Symposium Series on Computational Intelligence*, pp. 17–24, 2018.

[134] P. Žigert Pleteršek, *Matematika II*. Univerza v Ljubljani, Fakulteta za kemijo in kemijsko tehnologijo, 2013.

[135] S. Boh, *Regresija: delo diplomskega seminarja*. Univerza v Ljubljani, Fakulteta za matematiko in fiziko, 2013.

[136] S. S. Shapiro and M. B. Wilk, "An analysis of variance test for normality (complete samples)," *Biometrika*, vol. 52, no. 3/4, pp. 591–611, 1965.

[137] C. Spearman, "The proof and measurement of association between two things," *The American Journal of Psychology*, vol. 15, no. 1, pp. 72–101, 1904.

[138] J. C. Caruso and N. Cliff, "Empirical size, coverage, and power of confidence intervals for Spearman's rho," *Educational and Psychological Measurement*, vol. 57, no. 4, pp. 637–654, 1997.

[139] P. Kampstra, "Beanplot: A boxplot alternative for visual comparison of distributions," *Journal of statistical software*, vol. 28, no. 1, pp. 1–9, 2008.

[140] J. L. Hintze and R. D. Nelson, "Violin plots: a box plot-density trace synergism," *The American Statistician*, vol. 52, no. 2, pp. 181–184, 1998.

[141] R. J. Carroll and D. Ruppert, *Transformation and Weighting in Regression*. Chapman & Hall, Ltd., 1988.

[142] P. Leavey, A. Sengupta, D. Rakheja, O. Daescu, H. B. Arunachalam, and R. Mishra, "Osteosarcoma data from ut southwestern/UT Dallas for viable and necrotic tumor assessment [data set]," 2019.

[143] H. B. Arunachalam, R. Mishra, O. Daescu, K. Cederberg, D. Rakheja, A. Sengupta, D. Leonard, R. Hallac, and P. Leavey, "Viable and necrotic tumor assessment from whole slide images of osteosarcoma using machine-learning and deep-learning models," *PloS ONE*, vol. 14, no. 4, p. e0210706, 2019.

[144] J. P. Cohen, P. Morrison, and L. Dao, "COVID-19 image data collection," *arXiv 2003.11597*, 2020.

[145] "RSNA Pneumonia Detection Challenge Dataset." Radiological Society of North America, 2018.

[146] J. Demšar, "On the appropriateness of statistical tests in machine learning," in *Workshop on Evaluation Methods for Machine Learning in conjunction with ICML*, p. 65, 2008.

[147] M. Friedman, "The use of ranks to avoid the assumption of normality implicit in the analysis of variance," *Journal of the american statistical association*, vol. 32, no. 200, pp. 675–701, 1937.

[148] R. A. Fisher, "Statistical methods for research workers," in *Breakthroughs in statistics*, pp. 66–70, Springer, 1992.

[149] S. Holm, "A simple sequentially rejective multiple test procedure," *Scandinavian journal of statistics*, pp. 65–70, 1979.

[150] A. Benavoli, G. Corani, J. Demšar, and M. Zaffalon, "Time for a change: a tutorial for comparing multiple classifiers through bayesian analysis," *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 2653–2688, 2017.

[151] J. K. Kruschke and T. M. Liddell, "The bayesian new statistics: two historical trends converge," *SSRN Electronic Journal*, vol. 2606016, 2015.

[152] G. Corani, A. Benavoli, J. Demšar, F. Mangili, and M. Zaffalon, "Statistical comparison of classifiers through Bayesian hierarchical modelling," *Machine Learning*, vol. 106, no. 11, pp. 1817–1837, 2017.

[153] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[154] W. McKinney, "Data structures for statistical computing in python," in *Proceedings of the 9th Python in Science Conference*, pp. 51 – 56, 2010.

[155] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. Fernández del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, "Array programming with NumPy," *Nature*, vol. 585, p. 357–362, 2020.

[156] G. Vrbančič, L. Brezočnik, U. Mlakar, D. Fister, and I. Fister Jr., "NiaPy: Python microframework for building nature-inspired algorithms," *Journal of Open Source Software*, vol. 3, 2018.

[157] F. Chollet *et al.*, "Keras." https://keras.io, 2015.

[158] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas,

O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. Software available from tensorflow.org.

[159] S. Haghighi, M. Jasemi, S. Hessabi, and A. Zolanvari, "PyCM: Multiclass confusion matrix library in Python," *Journal of Open Source Software*, vol. 3, p. 729, may 2018.

[160] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, *et al.*, "Scipy 1.0: fundamental algorithms for scientific computing in python," *Nature methods*, vol. 17, no. 3, pp. 261–272, 2020.

[161] J. VanderPlas, B. Granger, J. Heer, D. Moritz, K. Wongsuphasawat, A. Satyanarayan, E. Lees, I. Timofeev, B. Welsh, and S. Sievert, "Altair: Interactive statistical visualizations for Python," *Journal of Open Source Software*, vol. 3, no. 32, p. 1057, 2018.

[162] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing in Science Engineering*, vol. 9, no. 3, pp. 90–95, 2007.