SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 6996

Učenje dubokih konvolucijskih modela na malom broju primjera

Frano Rajič

Zagreb, prosinac 2022.

Učenje dubokih konvolucijskih modela na malom broju primjera

Sažetak

Klasifikacija slika je složen problem s mnogim zanimljivim primjenama. Konvolucijske neuronske mreže postigle su vrhunske rezultate u mnogim područjima računalnog vida, posebno u zadacima klasifikacije slika. Međutim, klasični pristupi učenja konvolucijskih modela zahtijevaju ogromne količine označenih podataka za svaki semantički razred. Taj zahtjev smo relaksirali dubokom konvolucijskom sijamskom neuronskom mrežom koja podržava učenje na malom broju primjera. Predloženi model se temelji na dubokim korespondencijskim ugrađivanjima koja primjerke istih semantičkih razreda preslikava u kompaktnu regiju mnogostrukosti najapstraktnije latentne reprezentacije. Prikazani su rezultati modela treniranog na Omniglot skupu podataka.

Ključne riječi: sijamske neuronske mreže, učenje na malom broju primjera, učenje u jednom pokušaju, duboki konvolucijski modeli

Few-shot learning of deep convolutional models

Abstract

Image classification is a complex problem with many interesting applications. Convolutional neural networks have achieved superior results in many areas of computer vision, especially in image classification tasks. However, classical approaches to learning convolutional models require huge amounts of labeled data for each semantic class. We relaxed this requirement with a deep convolutional Siamese neural network that supports learning from a small number of examples. The proposed model is based on deep correspondence embeddings that map specimens of the same semantic classes into a compact region of multiplicity of the most abstract latent representation. The results of the model trained on the Omniglot dataset are presented.

Keywords: Siamese neural networks, one-shot learning, deep convolutional models

Hvala ljudi.

Sadržaj

1. Uvod

Jedna od važnijih karakteristika ljudskog mozga je učenje iz samo nekoliko primjera. Čovjek treba vidjeti samo jedan Segway da bi bio u stanju razlikovati Segwayeve od drugih vozila poput skutera i jednocikla. [12] Slično tome, djeca mogu upoznati novu riječ iz samo jednog susreta s njom [3].

Novi koncepti se gotovo nikada ne uče u izoliranom okruženju. Prethodno iskustvo s drugim pojmovima u dotičnoj domeni može podržati ubrzano učenje novih koncepta, pokazujući onome koji uči ono što je bitno za generalizaciju. Stoga su mnogi autori kao put do učenja u jednom pokušaju predložili sljedeće: *prijenosno učenje*, *reprezentacijsko učenje* ili *učenje radi učenja*. [12]

Strojno učenje uspješno se koristi za postizanje vrhunskih performansi u različitim aplikacijama kao što su web pretraživanje, detekcija neželjene pošte, stvaranje titlova i prepoznavanje govora i slike. Međutim, ovi algoritmi često ne rade dobro kada su prisiljeni predvidjeti podatke za koje je dostupno malo nadziranih informacija. Poželjno svojstvo algoritama strojnog učenja je generalizacija bez potrebe za opsežnim ponovnim učenjem koje može biti skupo ili nemoguće zbog ograničenih podataka. [9]

Jedan posebno zanimljiv zadatak je klasificiranje uz uvjet da možemo vidjeti samo jedan primjer svakog mogućeg razreda prije nego što provedemo predikciju. To se naziva učenjem u jednom pokušaju [15] [12]. Učenje u jednom pokušaju se razlikuje od učenja s nula pokušaja u kojem model ne može pogledati niti jedan primjer ciljnih razreda [16].

Djeca osobito pokazuju jedinstvenu sposobnost za učenje u jednom pokušaju, što se očituje tijekom razvoja motoričkih i kognitivnih vještina u ranom djetinjstvu. Na primjer, mala djeca nevjerojatnom brzinom usvajaju jezik, svakodnevno uče nove riječi i mogu izvoditi zaključke temeljem jednostavnih jezičnih konstrukcija, razvijajući sofisticirana pravila o novim kategorijama riječi iz vrlo malo ili čak bez da su se s njima susreli prije. [22]

Problem učenja u jednom pokušaju može se izravno riješiti razvijanjem zna-

čajki ili postupaka zaključivanja specifičnih za domenu primjene. Sustavi koji uključuju ove metode izvrsno se snalaze u sličnim slučajevima, ali ne nude robusna rješenja koja se općenito mogu primijeniti na druge vrste problema. [10]

Sveukupno, istraživanje u području algoritama za učenje u jednom pokušaju je prilično nezrelo i primilo je samo ograničenu pažnju zajednice strojnog učenja. [10]

U okviru ovog završnog rada opisuje se složeni problem klasifikacije slika i postojeći pristupi, opisuje se Omniglot skup podataka i model konvolucijske neuronske mreže koji podržava učenje na malom broju primjera. Model se koristi za učenje u jednom pokušaju te se analiziraju eksperimentalni rezultati.

2. Pristupi klasifikaciji slika

2.1. Pregled

Među glavne probleme računalnog vida spada klasifikacija slika [5]. To je postupak kojim se slike svrstavaju u pripadne razrede [14]. Ljudski mozak može lako klasificirati slike, ali za računalo to nije jednostavno. Za obavljanje operacija klasificiranja razvijene su različite metode. Opći postupci klasifikacije mogu se podijeliti u dvije općenite kategorije na temelju korištene metode: nadzirana klasifikacija i nenadzirana klasifikacija [21].

Klasifikacija slika važan je i izazovan zadatak u različitim područjima primjene, uključujući biomedicinsko snimanje, biometriju, video nadzor, navigaciju vozila, navigaciju robota i daljinsko istraživanje. Klasifikacija slike je složen proces na koji mogu utjecati mnogi čimbenici. Budući da su klasifikacijski rezultati osnova za mnoge okolišne i socioekonomske primjene, znanstvenici i stručnjaci uložili su velike napore u razvoju naprednih klasifikacijskih pristupa i tehnika za poboljšanje točnosti klasifikacije. [5]

2.2. Konvolucijske neuronske mreže

Konvolucijske neuronske mreže postigle su vrhunske rezultate u mnogim područjima računalnog vida, posebno u zadacima klasifikacije slika [1, 11, 18, 17].

2.3. Sijamske neuronske mreže

Sijamske neuronske mreže prvi su put predstavili Bromley i LeCun početkom 90-ih kako bi riješili provjeru potpisa kao problem podudaranja slike [2]. Sijamska neuronska mreža sastoji se od podmreža blizanaca koje prihvaćaju različite ulaze, ali se pri njihovom spajanju izračunavaju neke metričke vrijednosti između apstraktnih latentnih reprezentacija koje su izračunale podmreže. Parametri između

podmreža su vezani, a to vezanje jamči da podmreže neće preslikati dvije izrazito slične slike na potpuno različite lokacije u prostoru značajki, a to ne mogu učiniti jer svaka podmreža računa istu funkciju. Kako su podmreže simetrične, tako za isti par slika nije važno jesmo li prvu sliku propustili kroz jednu ili drugu podmrežu. [10]

U ovom radu je korištena duboka sijamska neuronska mreža s konvolucijskim slojevima za učenje na malom broju primjera.

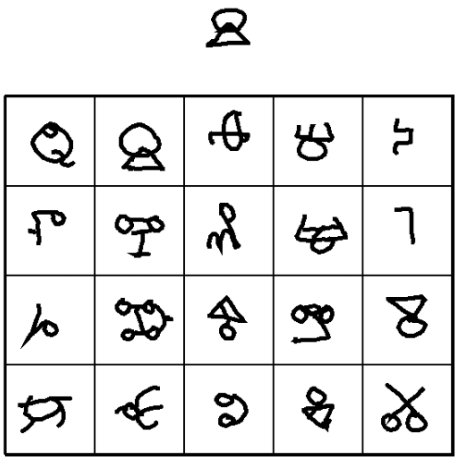
3. Omniglot skup podataka

3.1. Pregled skupa podataka

Omniglot skup podataka je skup slika koji su prikupili Brenden Lake i njegovi suradnici na MIT-u koristeći Amazonovog Mechanical Turk i time stvorili skup podataka koji je prigodan za učenje na malo primjera. Taj skup podataka se može promatrati kao "transponirani" MNIST jer umjesto 10 razreda (znamenaka) s tisućama primjera, koje ima MNIST, Omniglot sadrži preko 1600 različitih znakova gdje svaki znak sadrži 20 primjera [12]. Ovi znakovi potječu od 50 abeceda iz cijelog svijeta, uključujući bengalski, ćirilicu, hebrejski, sanskrt, tagalog, pa čak i sintetičke abecede koje se koriste za znanstveno-fantastične romane među kojima je klingonski. Iz svake abecede ima od 14 do najviše 55 različitih znakova. Svaki znak su jedanput nacrtali svaki od 20 crtača uključenih u crtanje preko Mechanical Turka. Na slici 3.1 su prikazani znakovi iz nekih abeceda.

Lake je originalno podatke nasumično podijelio u pozadinski skup od 30 abeceda i evaluacijski skup od 20 abeceda, s time da pozadinski skup uključuje šest najčešćih abeceda prema broju Googleovih pretraživanja [13]. Ovaj rad će koristiti pozadinski skup za formiranje podskupa za učenje, a evaluacijski skup će podijeliti tako da će prvih deset abeceda pripasti podskupu za validaciju, a drugih deset abeceda podskupu za testiranje, što je detaljno opisano kasnije u potpoglavlju 5.1.

Slika 3.1: Neke abecede sadržane u skupu podataka Omniglot



Slika 3.2: Primjer *20-way* klasifikacije u jednom pokušaju

3.2. Evaluacija učenja u jednom pokušaju

Kao način da empirijski evaluiramo uspješnost učenja u jednom pokušaju, Lake je koristio *20-way* unutar-abecednu klasifikaciju u jednom pokušaju. To je klasifikacija u kojoj ispitanik ima 20 primjeraka različitih razreda, od svakog po jedan primjer, te dobije novi primjerak koji do sada nije susreo, a treba ga klasificirati u jedan od tih 20 razreda. Primjer ovog zadatka je prikazan na slici 3.2. Razvijeni modeli će biti testirani na ovom zadatku.

4. Programska izvedba i vanjske biblioteke

4.1. Programski okviri dubokog učenja [19]

U ranijim danima, ljudi su trebali imati stručnost u C++-u i CUDA-i za implementaciju algoritama dubokog učenja. S velikim brojem organizacija koje su otvorile svoje programske okvire dubokog učenja, ljudi sa znanjem skriptnih jezika kao što je Python su započeli izgradnju i korištenje algoritama dubokog učenja. Neki od popularnih okvira dubokog učenja koji se danas koriste u industriji su TensorFlow, Caffe2, Keras, Theano, PyTorch, Chainer, DyNet, MXNet i CNTK.

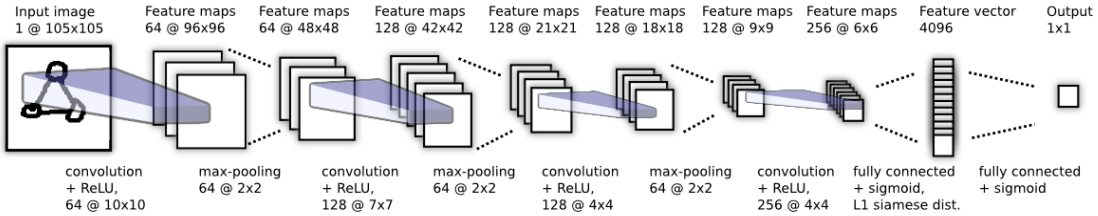
Usvajanje dubokog učenja ne bi bilo ogromno da nije bilo ovih okvira. Oni apstrahiraju mnogo osnovnih komplikacija i omogućuju usredotočenje na aplikacije. Razni okviri imaju svoje prednosti i nedostatke.

4.2. Biblioteka PyTorch

PyTorch je programski okvir za duboko učenje te paket za znanstveno računanje u Pythonu koji sadrži sve potrebno za jednostavnu implementaciju dubokog učenja uz akceleraciju računanja korištenjem grafičke procesorske jedinice. Aspekt znanstvenog računarstva je prije svega rezultat PyTorch-ove *tensor* biblioteke i operacija nad njom.

PyTorch tenzori imaju ugrađenu GPU podršku te se operacije nad tenzorima mogu izvoditi na grafičkim procesorskim jedinicama. Sasvim je jednostavno premjestiti tenzore na GPU jezgre kao i skinuti ih s njih, ako je podržan GPU sustava.

PyTorch u svojoj osnovi koristi dinamičke računske grafove za provođenje računanja na tenzorima u neuronskoj mreži. Kako bi se optimizirale neuronske mreže, tako je potrebno računati derivacije, a da bi to programski ostvarili,



Slika 4.1: Konvolucijska arhitektura od šest slojeva. Slika preuzeta iz [10]. Sijamski blizanac nije prikazan ali se spaja odmah nakon linearnog sloja s 4096 neurona. Pri spajanju podmreža se kao metrika računa L1 udaljenost.

programski okviri dubokog učenja koriste upravo računske grafove. Dinamički računski grafovi se stvaraju tijekom računanja, u odnosu na statičke koji su u potpunosti određeni još prije samih operacija. Novije istraživačke teme dubokog učenja zahtijevaju ili imaju veliku korist od korištenja dinamičkih grafova.

Jedna od prednosti korištenja PyTorch-a ili bilo kojeg drugog API-ja za neuronske mreže jest ta da je korištenje paralelizma unaprijed ugrađeno. Zbog toga se programer neuronske mreže može više usredotočiti na izgradnju neuronskih mreža, a manje na probleme s performansama.

PyTorch projekt su podržali Facebook i Microsoft.

4.3. Arhitektura dubokog klasifikacijskog modela

Kao arhitekturu prikladnu za učenje u jednom pokušaju je odabrana duboka sijemska neuronska mreža koju je predložio rad [10], a koja je prikazana na slici 4.1. Arhitekturu čine četiri konvolucijska sloja s različitim veličinama maski i dva potpuno povezana sloja:

1. Prvi konvolucijski sloj: 64x10x10 jezgre, korak iznosi 1
2. Drugi konvolucijski sloj: 128x7x7 jezgre, korak iznosi 1
3. Treći konvolucijski sloj: 128x4x4 jezgre, korak iznosi 1
4. Četvrti konvolucijski sloj: 128x2x2 jezgre, korak iznosi 1
5. Prvi potpuno povezani sloj: spaja ulaznih 9216 neurona na 4096 izlaznih
6. Drugi potpuno povezani sloj: spaja ulaznih 4096 neurona na jedan izlazni

Nakon svakog konvolucijskog sloja se najprije nalazi ReLU nelinearna aktivacijska funkcija i zatim sloj sažimanja s korakom 2 koji uzima najveći element. Na slici 4.2 su prikazani svi slojevi i svi parametri konvolucijske arhitekture. Podmreže imaju dijeljene parametre u sva četiri konvolucijska sloja, te u prvom linearnom sloju, dok drugi linearni sloj uslijedi nakon spajanja podmreža. Nakon potpuno povezanih slojeva su sigmoide aktivacijske funckije. Posljednju sigmoidu je moguće izostaviti iz implementirane mreže, što je u ovom radu napravljeno kako bi se koristio numerički stabilniji izračun binarne križne entropije kao funkcije gubitka.

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1          [-1, 64, 96, 96]           6,464
            Conv2d-2         [-1, 128, 42, 42]         401,536
            Conv2d-3         [-1, 128, 18, 18]         262,272
            Conv2d-4           [-1, 256, 6, 6]         524,544
            Linear-5                [-1, 4096]      37,752,832
            Linear-6                   [-1, 1]           4,097
================================================================
Total params: 38,951,745
Trainable params: 38,951,745
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.04
Forward/backward pass size (MB): 6.64
Params size (MB): 148.59
Estimated Total Size (MB): 155.27
----------------------------------------------------------------
```

Slika 4.2: Arhitektura mreže ispisana pomoću *torchsummary.summary()*

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1          [-1, 64, 96, 96]           6,464
       BatchNorm2d-2          [-1, 64, 96, 96]             128
            Conv2d-3         [-1, 128, 42, 42]         401,536
       BatchNorm2d-4         [-1, 128, 42, 42]             256
            Conv2d-5         [-1, 128, 18, 18]         262,272
       BatchNorm2d-6         [-1, 128, 18, 18]             256
            Conv2d-7           [-1, 256, 6, 6]         524,544
       BatchNorm2d-8           [-1, 256, 6, 6]             512
            Linear-9                [-1, 4096]      37,752,832
           Linear-10                   [-1, 1]           4,097
================================================================
Total params: 38,952,897
Trainable params: 38,952,897
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.04
Forward/backward pass size (MB): 13.25
Params size (MB): 148.59
Estimated Total Size (MB): 161.89
----------------------------------------------------------------
```

Slika 4.3: Arhitektura mreže ispisana pomoću *torchsummary.summary()* uz dodane *BatchNormalization* slojeve

4.4. Treniranje

4.4.1. Funkcija gubitka

Kao funkcija gubitka za zadatak binarne klasifikacije je korištena binarna križna entropija uz L_2 regularizaciju nad svim parametrima mreže za kažnjavanje visokih vrijednosti parametara. Neka je M mini-batch veličina, gdje i označava i-ti mini-batch. Neka je $\mathbf{y}\left(x_1^{(i)}, x_2^{(i)}\right)$ vektor duljine M koji sadrži oznake za dotični mini-batch uz pretpostavku da je $y\left(x_1^{(i)}, x_2^{(i)}\right) = 1$ uvijek kada vrijedi da su x_1 i x_2 istog razreda odnosno ako prikazuju isti znak, a da inače vrijedi $y\left(x_1^{(i)}, x_2^{(i)}\right) = 0$ [10]. Funkcija gubitka je sljedećeg oblika:

$$L\left(x_1^{(i)}, x_2^{(i)}\right) = y\left(x_1^{(i)}, x_2^{(i)}\right) \log p\left(x_1^{(i)}, x_2^{(i)}\right) + \left(1 - y\left(x_1^{(i)}, x_2^{(i)}\right)\right) \log\left(1 - p\left(x_1^{(i)}, x_2^{(i)}\right)\right) + \boldsymbol{\lambda}^T |\mathbf{w}|^2$$

4.4.2. Optimizacija

Za optimizaciju je korišten algoritam Adam [8] sa stopom učenja kao hiperparametrom.

4.4.3. Inicijalizacija parametara

Početna inicijalizacija vrijednosti parametara konvolucijskih odnosno linearnih slojeva je normalna razdioba preuzeta iz [10]. Težine konvolucijskih slojeva su inicijalizirane vrijednostima iz $\mathcal{N}(0, 0.01)$, a pristranost iz $\mathcal{N}(0.5, 0.01)$. Težine potpuno povezanih slojeva su inicijalizirane iz $\mathcal{N}(0, 0.2)$, a njihove pristranosti iz $\mathcal{N}(0, 0.01)$.

4.4.4. Postupak treniranja

Maksimalni broj epoha je ograničen na 500 epoha. Rano zaustavljanje je korišteno, ali bi se prilagodilo treniranom modelu.

4.4.5. Optimizacija hiperparametara

Postavljanje hiperparametara je provedeno ručnim podešavanjem vrijednosti.

4.4.6. Sprječavanje prenaučenosti

Sprječavanje prenaučenosti je borba. U nastavku su opisani načini kojima se u ovom radu bori protiv prenaučenosti.

Augmentacija podataka

Najlakši i često korišten način da se spriječi prenaučenost na slikama jest proširiti skup podataka dodavanjem transformacija koje sačuvaju oznaku primjera [11].

Regularizacija

Korištena je L_2 regularizacija nad svim parametrima mreže.

Distorzije

Koch je dodao distorzije tijekom evaluacije modela kako bi poboljšao performanse, u čemu je uspio [10]. Ovaj rad ih nije koristio.

4.5. Oblikovanje programske podrške

Programska izvedba je implementirana u programskom jeziku Python uz korištenje programskog okvira PyTorch. Ulazna točka u kojoj se podešavaju hiperparametri je program *demo.py*. U nastavku je opis napisanih Python paketa.

1. paket **base**
   (a) modul *base_trainer.py* koji sadrži apstraktni razred **BaseTrainer**. Funkcionalnosti BaseTrainer su:
      i. podešavanje pisanja u log datoteke
      ii. priprema instance razreda **SummaryWriter** iz paketa *torch.utils.tensorboard*, koja se onda može koristiti za zapisivanje log podataka za kasniju vizualizaciju alatom **TensorBoard**
      iii. implementacija treniranja do na funkciju *_train_epoch* koju nadjačava izvedeni razred, a koja se poziva na početku svake epohe treniranja
      iv. bilježenje statistike u log datoteku o metrikama mjerenim tijekom epohe
      v. spremanje kontrolnih točaka, odnosno pohrana trenutne verzije istreniranog modela na disk
      vi. praćenje i pohrana najboljeg modela na disk
2. paket **data_loader**

(a) modul *data_loader.py* koji sadrži svu funkcionalnost povezanu uz učitavanje skupa podataka i njegovo vizualiziranje. Razred **OmniglotVisualizer** omogućuje prikazivanje Omniglot *DataLoader*-a uz pomoć biblioteke matplotlib. Razred **OmniglotDataLoaderCreator** učitava i razvrstava sve slike iz skupa podataka preuzetog pomoću razreda Omniglot iz paketa **torchvision.datasets**.

3. paket **model**

   (a) modul *metric.py* u kojem je funkcionalnost računanja relevantnih metrika

   (b) modul *model.py* u kojem su smještene definicije modela neuronske mreže

4. paket **trainer**

   (a) modul *trainer.py* koji sadrži razred **OmniglotTrainer**. OmniglotTrainer nasljeđuje **base.BaseTrainer** ima sljedeće funkcionalnosti:

      i. implementira funkciju *_train_epoch*, *_valid_epoch* i *_oneshot_epoch* koje se bave svim koracima unutar tekuće epohe

      ii. računanje metrike relevantne za trenutnu epohu

      iii. zapisivanje relevantnih metrika za **TensorBoard**

      iv. ispisivanje statistike o napredovanju epohe

5. paket **utils**

   (a) modul *utils.py* koji sadrži funkcionalnosti korištene u ostalim paketima i među njima razred **MetricTracker** koji vodi brigu o pamćenju vrijednosti relevantnih metrika

Direktorij **data** služi za pohranu skupa podataka. Direktorij **logs** sadrži pohranjene modele i log datoteke. Direktorij **visuals** sadrži pohranjene vizualizacije skupa podataka i druge slike.

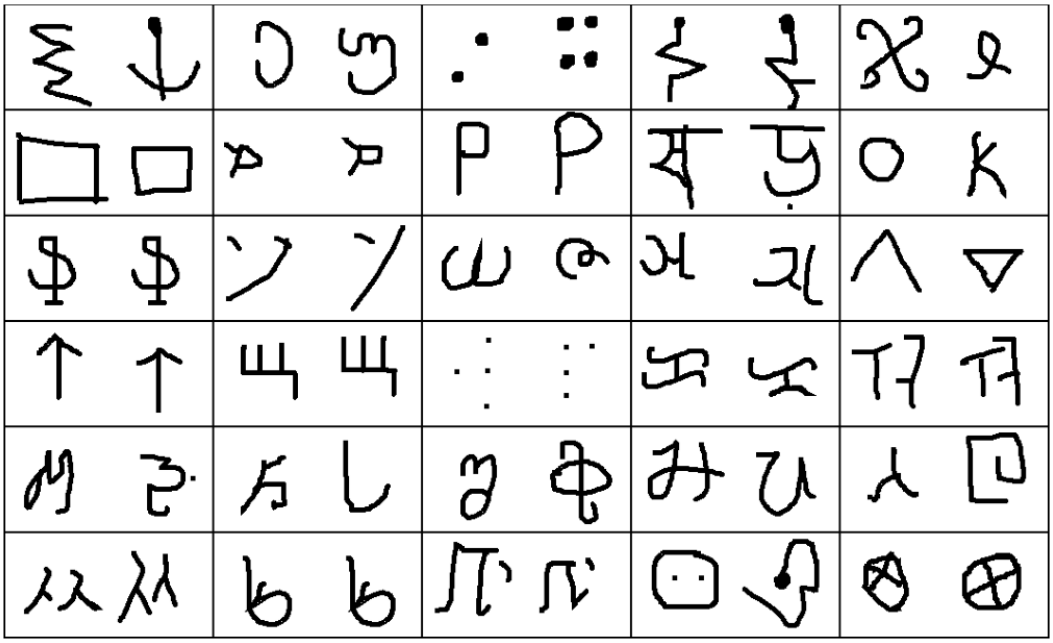
5. Eksperimentalni rezultati

5.1. Ispitni skupovi izvornih slika

Za učenje, validiranje i testiranje neuronske mreže, izvorni Omniglot skupa podataka je podijeljen na podskupove korištenjem pristupa opisanog u radu [10]. Za učenje su formirana tri različita podskupa od 30,000, 90,000 i 150,000 primjera. Svaki primjer sadrži dvije slike i pripadnu oznaku. Oznaka je 1 ako slike predstavljaju isti znak, odnosno 0 ako ne predstavljaju isti znak. Za oblikovanje podskupa za učenje je odvojeno prvih 30 od 50 abeceda i 12 crtača od 20. Podskup je kreiran iz izvornog Omniglot skupa podataka tako da su se ravnomjerno nad abecedama nasumično odabirali unutar-abecedni parovi znakova. Oznake su bile balansirane tako da je odabran jednak broj parova koji pokazuju isti odnosno različit znak. Za pojednostavljivanje odabira parova unutar određene abecede, [10] predlaže nasumičan odabir dvaju znakova i nasumičan odabir dvaju crtača, a zatim oblikovanje jednog para slika s prvim znakom od odabranih crtača i para slika od oba odabrana znaka i oba crtača. Primjeri parova nastalih ovim postupkom su prikazani na slici 5.1.

Ovim pristupom se osigurava ujednačen raspored primjera nad abecedama, iako raspored nad pojedinačnim znakovima unutar abecede nije podjednako raspoređen jer različite abecede imaju različit broj znakova. Dodavanjem afinih distorzija, izrađene su dodatne kopije napravljenih podskupova za učenje, tako da se za svaki od tri skupa stvori pripadna augmentirana verzija. Dodano je po osam transformacija za svaki primjerak, što rezultira pripadnim podskupovima veličine 270,000, 810,000, i 1,350,000 primjera. [10] Primjer slika nastalih primjenom afinih transformacija prikazan je na slici 5.2.

Dodatno, nad skupom za treniranje je izračunata srednja vrijednost i standardna devijacija nad svim pikselima, te su sve slike s kojima će se raditi bile standardizirane oduzimanjem srednje vrijednosti i dijeljenjem sa standardnom devijacijom. U tom aspektu je ovaj rad odstupio od rada [10].

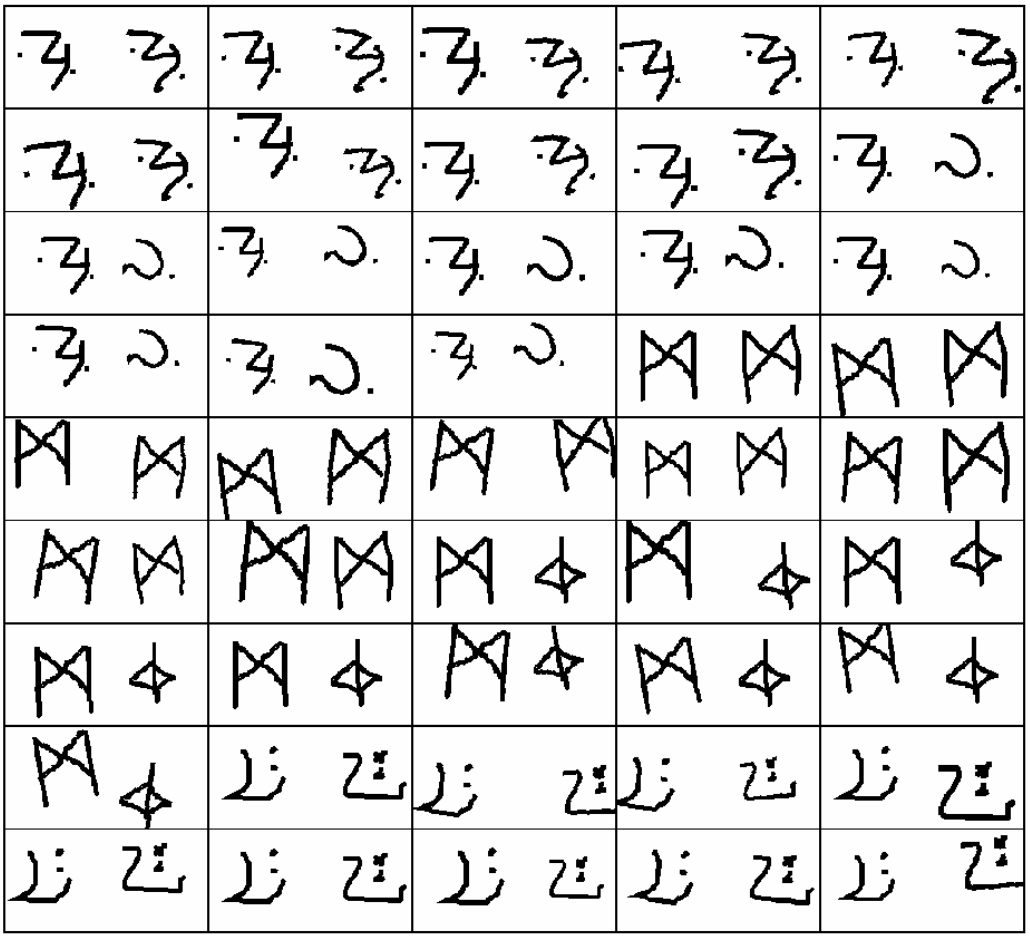


Slika 5.1: Primjeri parova slika koji su se pojavili u skupu za treniranje

Za praćenje performansi za vrijeme učenja, korištene su dvije strategije [10]. Prva je podskup za validaciju od 10,000 primjera koji su napravljeni od preostalih 10 od 20 abeceda i dodatna 4 crtača od preostalih 8 koji nisu korišteni prilikom učenja. Preostalih i posljednjih 10 abeceda i 4 crtača su ostavljeni za testiranje, i jednaka su onima koje je odabrao [13].

Druga strategija koristi iste znakove i crtače rezervirane za validacijski podskup da bi generirala 320 primjera *20-way* unutar-abecedne klasifikacije u jednom pokušaju i time oponašala ciljni zadatak skupa za testiranje. Koch je pokazao približnu korespondenciju između spomenute metrike i stvarne performanse na skupu za treniranje, zbog čega se ova metrika koristi kao temeljni kriterij za odabir najboljeg modela i rano zaustavljanje treniranja. [10]

Uspješnost treniranog modela se testira na 400 primjera *20-way* unutar-abecedne klasifikacije u jednom pokušaju. Testiranje se provodi nakon treniranja konvolucijske sijamske mreže, a za testiranje se odabire model koji je dao najbolje rezultate na validacijskom zadatku *20-way* klasifikacije u jednom pokušaju. Zadatak *20-way* klasifikacije u jednom pokušaju se razlikuje od onog na kojem je mreža trenirana. Sijamska mreža na ulazu prima jedan par slika pa je dotični zadatak potrebno prilagoditi mreži, što je opisano u nastavku. Pretpostavimo da je dana slika x koju treba klasificirati u jedan od 20 razreda. Za svaki od tih 20 razreda je dana po jedna slika. Slika x se propusti kroz mrežu sa svakim



Slika 5.2: Primjer slika nastalih primjenom afinih transformacija

primjerkom od dotičnih 20 razreda te se kao predikcija uzima onaj razred koji je na izlazu iz mreže dao najveće pobuđenje. Ovaj se postupak ponovi za svaki primjerak, odnosno 400 puta. Na slici 5.3 je prikazana vizualizacija 3 zadataka iz testiranja s pripadnim ocjenama sličnosti.

5.2. Eksperimentalni rezultati

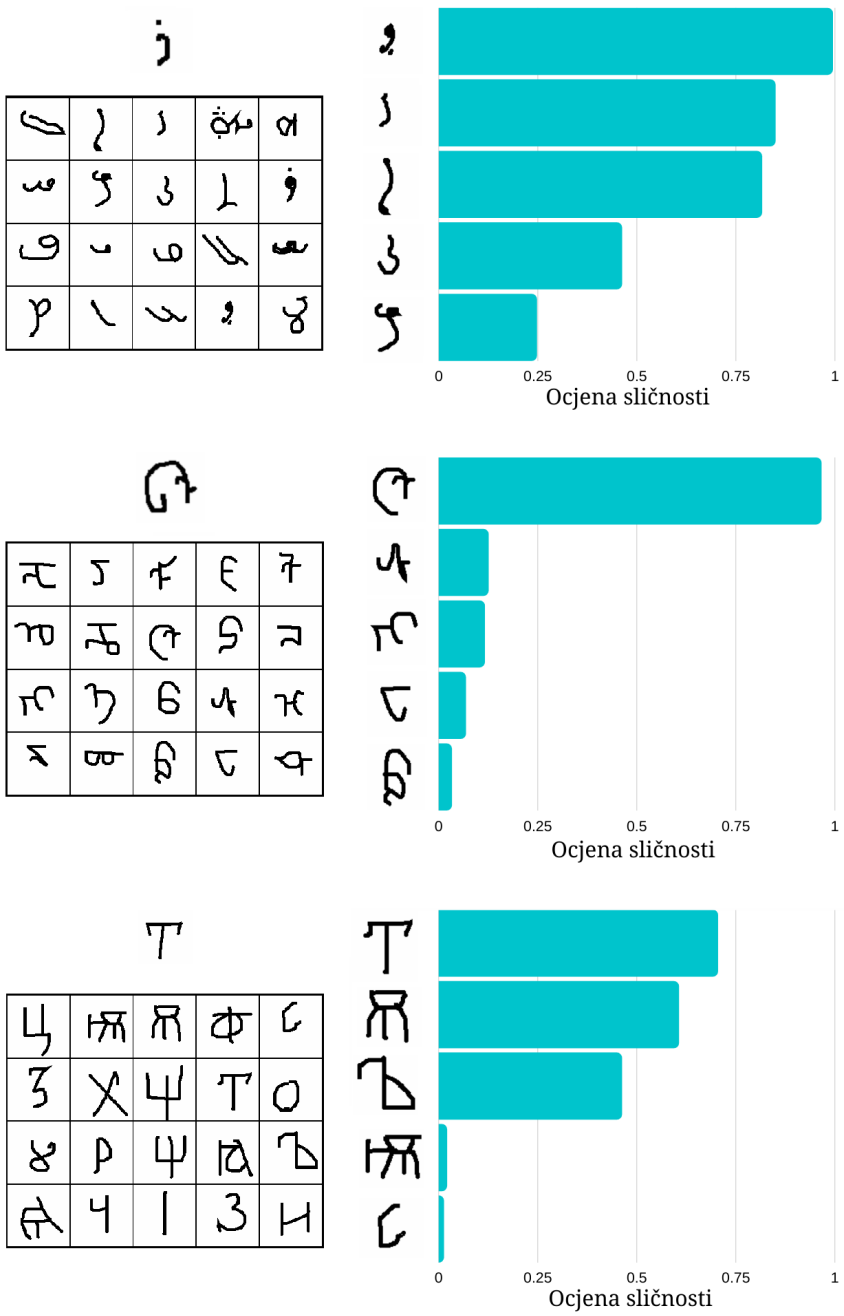
Treniranje je provedeno s različitim ručno podešenim vrijednostima hiperparametara. Hiperparametri su stopa učenja, stupanj L_2 regularizacije, dodavanje odnosno izostavljanje afinih transformacija, dimenzije skupa za treniranje (30,000, 90,000 i 150,000 slika), provođenje standardiziranja slika na temelju prosječne vrijednosti i devijacije piksela na skupu za učenje.

U tablici 5.1 se navode konačni rezultati provjere na podskupu za testiranje za svaki od šest mogućih skupova za učenje, gdje se navedena točnost testa navodi na najboljoj kontrolnoj točki. Kako se odabrana arhitektura razlikuje od [10] u nekolicini dodanih promjena, tako je u spomenutoj tablici prikazana usporedba s rezultatima originalnog autora. Na slikama 5.4, 5.5, 5.6 i 5.7 je usporedno prikazano prvih 50 epoha zabilježenih tijekom treniranja modela.

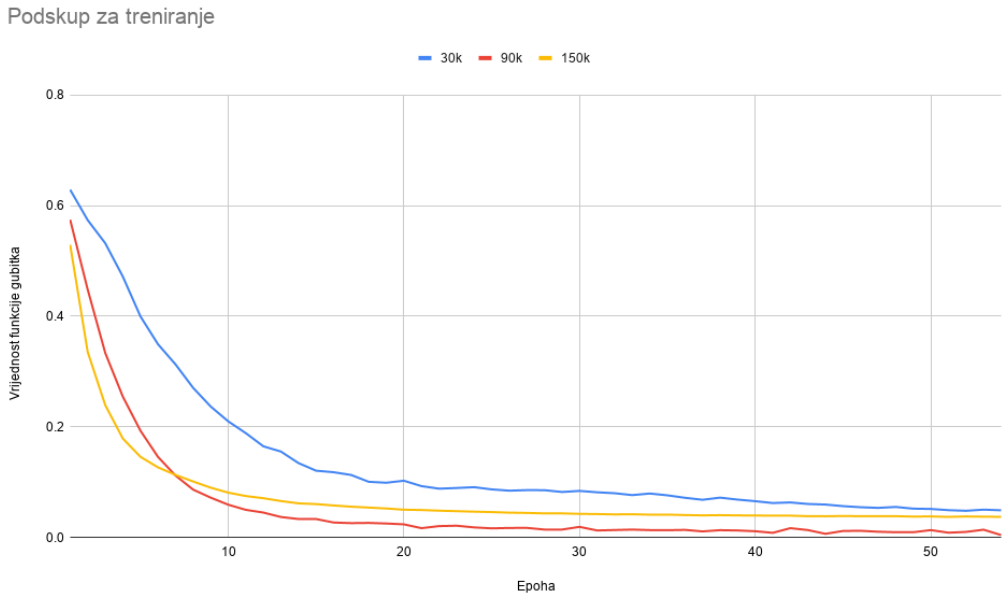
Tablica 5.1: Rezultati provjere na skupu za treniranje i usporedba s rezultatima iz [10]. Prvi stupac prikazuje točnost rada [10]. Drugi stupac prikazuje točnost modela treniranog u ovom radu. Redci određuju korišteni podskup za treniranje

Metoda	Točnost, Koch et al. [10]	Točnost
30k učenje		
bez afinih transformacije	90.61	51.00
8x afine transformacije	91.90	66.25
90k učenje		
bez afinih transformacije	91.54	62.00
150k učenje		
bez afinih transformacije	91.63	63.00

Važno je primijetiti da bi točnost modela, koji bi nasumično pogađao odgovor u zadatku *20-way* unutar-abecedne klasifikacije u jednom pokušaju, bila tek 5%. Ostvareni rezultati od preko 60% su daleko iznad toga, što znači da naučena mreža relativno uspješno obavlja zadatak. S druge strane, rezultati dobiveni u

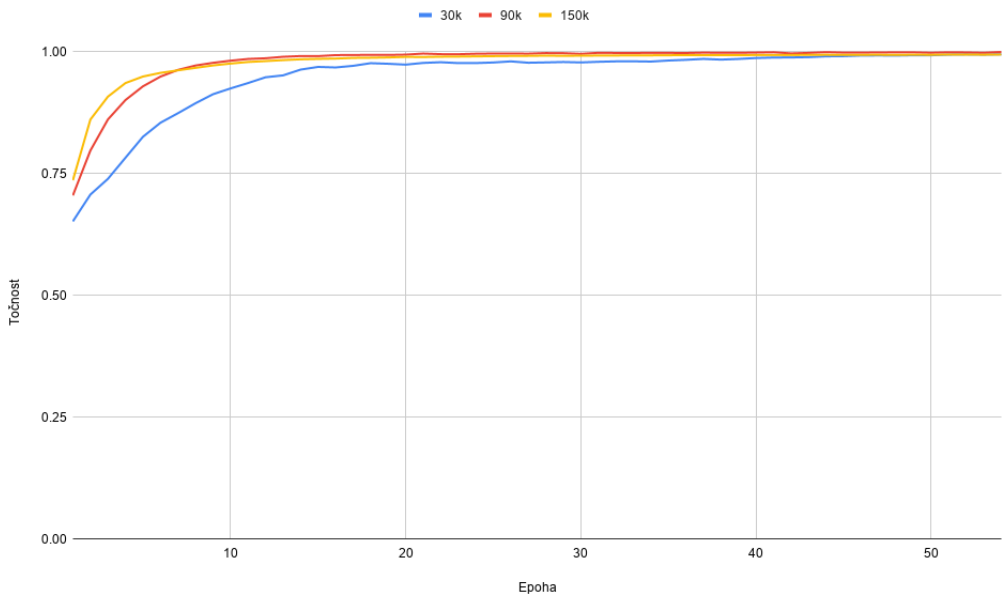


Slika 5.3: Primjeri pridodanih vrijednosti sličnosti za tri zadatka *20-way* unutarabecedne klasifikacije u jednom pokušaju iz podskupa za testiranje. S lijeve strane je klasifikacijski zadatak, a s desne strane ocjene sličnosti za 5 najsličnijih rezultata. U prva dva retka je primjer ispravne klasifikacije, a u trećem neispravne klasifikacije.

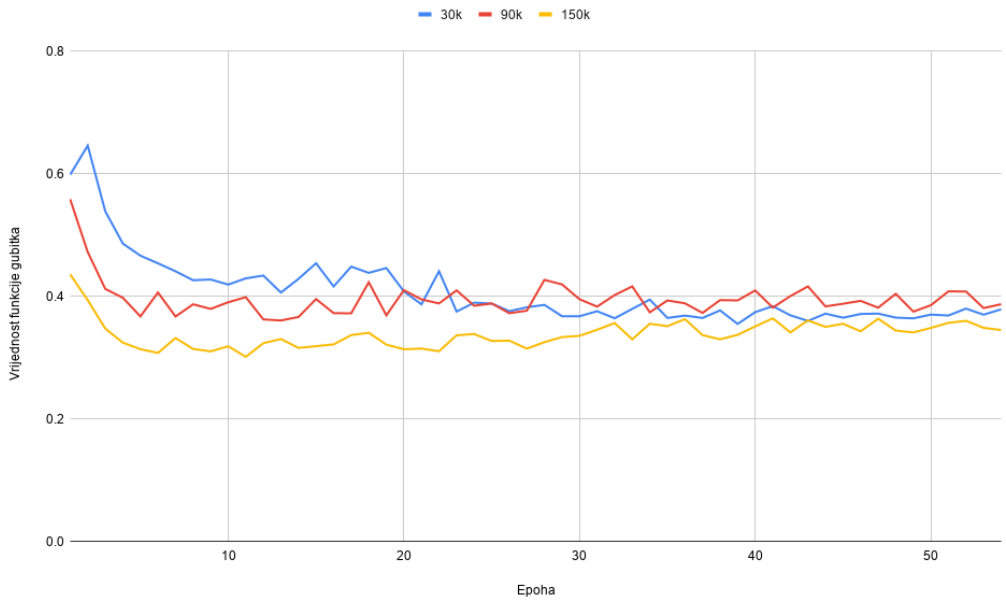


Slika 5.4: Vrijednosti funkcije gubitka na podskupu za učenje za tri različite veličine

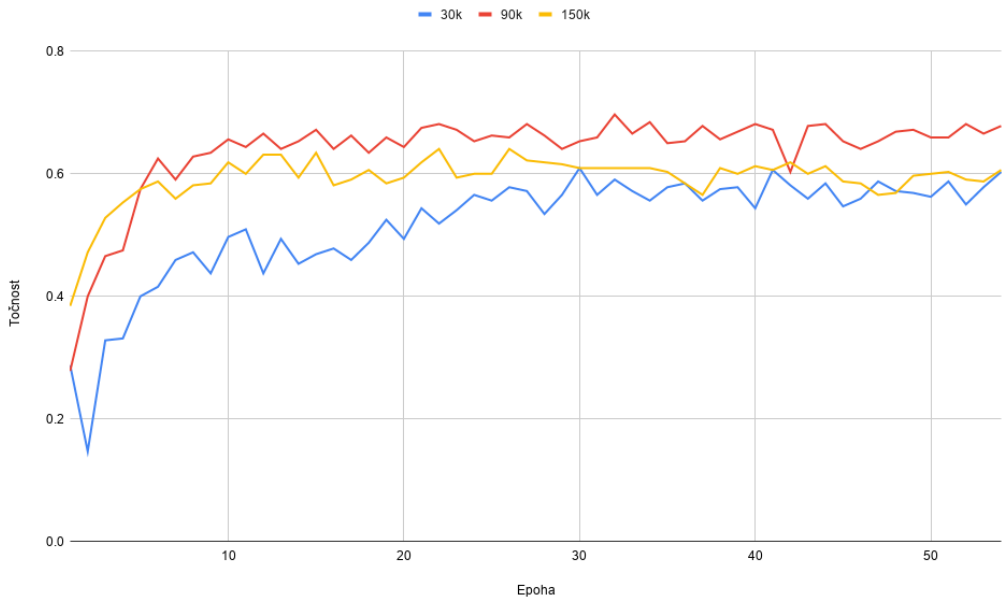
ovom radu se značajno razlikuju od rezultata u radu [10] koji su bolji za 20 do 40%. Ovo odstupanje nije razjašnjeno. Mogući razlozi odstupanja su neispravno tumačenje originalnog rada i prisutnost skrivene pogreške u implementaciji.



Slika 5.5: Točnost na podskupu za učenje za tri različite veličine podskupa



Slika 5.6: Vrijednosti funkcije gubitka na validacijskom podskupu za tri različite veličine podskupa za učenje



Slika 5.7: Točnost na validacijskom podskupu na zadatku *20-way* unutar-abecedne klasifikacije u jednom pokušaju za tri različite veličine podskupa za učenje

6. Zaključak

6.1. Pregled

Klasifikacija prirodnih slika važan je zadatak računalnog vida s mnogim zanimljivim primjenama. Konvolucijske neuronske mreže postigle su vrhunske rezultate u mnogim područjima računalnog vida, posebno u zadacima klasifikacije slika. Međutim, klasični pristupi učenja konvolucijskih modela zahtijevaju ogromne količine označenih podataka za svaki semantički razred.

Taj zahtjev je relaksiran dubokom konvolucijskom sijamskom neuronskom mrežom koja podržava učenje na malom broju primjera. Predloženi model se temelji na dubokim korespondencijskim ugrađivanjima koja primjerke istih semantičkih razreda preslikava u kompaktnu regiju mnogostrukosti najapstraktnije latentne reprezentacije.

Model je treniran na Omniglot skupu podataka. Naučeni modeli se evaluiraju na zadatku *20-way* unutar-abecedne klasifikacije u jednom pokušaju. Podskupovi za učenje, validaciju i testiranje su oblikovani iz Omniglot skupa podataka na isti način koji je opisan u radu [10], ali su postignuti niži rezultati.

6.2. Budući rad

Budući rad bi mogao uključiti treniranje duboke konvolucijske sijamske neuronske mreže na drugim skupovima podataka, osobito onim koji ne sadrže samo znakove, poput ImageNet-a [4].

Dodatno, u ovom radu se treniranju pristupa ručnim podešavanjem vrijednosti hiperparametara i ne koriste se rješenja koja bi automatski podešavala hiperparametre. Uz to, u daljnjem radu bi se moglo razmotriti eksperimentiranje s različitim dubokim arhitekturama.

Konačno, postoje gotovi modeli neuronskih mreža poput Resnet18 [6] i SENet [7] koji bi se mogli primijeniti za prijenos stečenog znanja odnosno predtreniranje

slojeva koji se nalaze prije spajanja blizanaca.

Literatura

[1] Yoshua Bengio. Learning deep architectures for ai. *Foundations and trends® in Machine Learning*, 2(1):1–127, 2009.

[2] Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, i Roopak Shah. Signature verification using a" siamese" time delay neural network. U *Advances in neural information processing systems*, stranice 737–744, 1994.

[3] Susan Carey i E. Bartlett. Acquiring a single new word. *Proceedings of the Stanford Child Language Conference*, 15:17–29, 01 1978.

[4] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, i Li Fei-Fei. Imagenet: A large-scale hierarchical image database. U *2009 IEEE conference on computer vision and pattern recognition*, stranice 248–255. Ieee, 2009.

[5] Pralhad Gavali i J Saira Banu. Deep convolutional neural network for image classification on cuda platform. U *Deep Learning and Parallel Computing Environment for Bioengineering Systems*, stranice 99–122. Elsevier, 2019.

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, i Jian Sun. Deep residual learning for image recognition. U *Proceedings of the IEEE conference on computer vision and pattern recognition*, stranice 770–778, 2016.

[7] Jie Hu, Li Shen, i Gang Sun. Squeeze-and-excitation networks. U *Proceedings of the IEEE conference on computer vision and pattern recognition*, stranice 7132–7141, 2018.

[8] Diederik P Kingma i Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[9] Gregory Koch. Siamese neural networks for one-shot image recognition. Magistarski rad. URL `http://www.cs.toronto.edu/~gkoch/files/msc-thesis.pdf`.

[10] Gregory Koch, Richard Zemel, i Ruslan Salakhutdinov. Siamese neural networks for one-shot image recognition. U *ICML deep learning workshop*, svezak 2. Lille, 2015.

[11] Alex Krizhevsky, Ilya Sutskever, i Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. U *Advances in neural information processing systems*, stranice 1097–1105, 2012.

[12] Brenden Lake, Ruslan Salakhutdinov, Jason Gross, i Joshua Tenenbaum. One shot learning of simple visual concepts. U *Proceedings of the annual meeting of the cognitive science society*, svezak 33, Lipanj 2011.

[13] Brenden M Lake, Russ R Salakhutdinov, i Josh Tenenbaum. One-shot learning by inverting a compositional causal process. U *Advances in neural information processing systems*, stranice 2526–2534, 2013.

[14] Shin-Jye Lee, Tonglin Chen, Lun Yu, i Chin-Hui Lai. Image classification based on the boost convolutional neural network. *IEEE Access*, 6:12755–12768, 2018.

[15] Fei-Fei Li, Rob Fergus, i Pietro Perona. One-shot learning of object categories. *IEEE transactions on pattern analysis and machine intelligence*, 28(4): 594–611, 2006.

[16] Mark Palatucci, Dean Pomerleau, Geoffrey E Hinton, i Tom M Mitchell. Zero-shot learning with semantic output codes. U *Advances in neural information processing systems*, stranice 1410–1418, 2009.

[17] Karen Simonyan i Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[18] Nitish Srivastava. Improving neural networks with dropout. *University of Toronto*, 182(566):7, 2013.

[19] Vishnu Subramanian. *Deep Learning with PyTorch: A practical approach to building neural network models using PyTorch*. Packt Publishing Ltd, 2018.

[20] Filip Tolić. *Sijamske neuronske mreže za one-shot prepoznavanje slika, Tehnička dokumentacija.* URL `http://www.zemris.fer.hr/~ssegvic/project/pubs/tolic19proj2.pdf`.

[21] Guotai Wang, Wenqi Li, Maria A Zuluaga, Rosalind Pratt, Premal A Patel, Michael Aertsen, Tom Doel, Anna L David, Jan Deprest, Sébastien Ourselin, et al. Interactive medical image segmentation using deep learning with image-specific fine tuning. *IEEE transactions on medical imaging*, 37(7): 1562–1573, 2018.

[22] Kenneth Yip i Gerald Jay Sussman. Sparse representations for fast, one-shot learning. 1997.