Prepoznavanje osoba temeljem hoda primjenom dubokog učenja

Ivović, Leon

SVEUČILIŠTE U RIJECI
TEHNIČKI FAKULTET
Sveučilišni prijediplomski studij računarstva

Završni rad

Prepoznavanje osoba temeljem hoda primjenom dubokog učenja

Rijeka, rujan 2023.

Leon Ivović
0069089815

SVEUČILIŠTE U RIJECI
TEHNIČKI FAKULTET
Sveučilišni prijediplomski studij računarstva

Završni rad

Prepoznavanje osoba temeljem hoda primjenom dubokog učenja

Mentor: prof. dr. sc. Kristijan Lenac
Komentor: v. asist. dr. sc. Diego Sušanj

Rijeka, rujan 2023.

Leon Ivović
0069089815

Umjesto ove stranice umetnuti zadatak za završni ili diplomski rad

Izjava o samostalnoj izradi rada

Izjavljujem da sam samostalno izradio ovaj rad.

Rijeka, rujan 2023.

_____

Leon Ivović

Sadržaj

Popis slika

Popis tablica

Poglavlje 1

Uvod

Hod svake osobe je jedinstven te možemo koristiti to svojstvo hoda za prepoznavanje i razlikovanje osoba. Prepoznavanje hoda ima brojne prednosti u odnosu na ostale biometrijske metode identifikacije osoba. Ono omogućuje identifikaciju osobe iz puno veće udaljenosti i bez potrebe za direktnim kontaktom u usporedbi s metodama prepoznavanja lica, očiju ili otiska prstiju. Zbog svoje prirode hod neke osobe je teže oponašati, te ga se može prepoznati neovisno o odjeći i nošenju tereta. Prepoznavanje hoda je moguće i pri niskim rezolucijama pa nije potrebna ni posebna tehnologija.

Neki nedostatci postojećih metoda prepoznavanja hoda su manja točnost u odnosu na metode poput prepoznavanja šarenice ili otiska prstiju i smanjena točnost pri upotrebi u vanjskom svijetu. Istraživanja pokazuju da u vanjskim, nekontroliranim uvjetima pouzdanost prepoznavanje hoda pada za oko 40% u odnosu na unutarnje uvjete[7]. Uzrok tako velikog pada u pouzdanosti su razne smetnje prisutne u vanjskom svijetu, na primjer, promjene svjetlosti i pozadine. Problem mogu predstavljati i veće varijacije u izgledu ljudskog tijela. Za razliku od ostalih biometrijskih značajki osobe koje većinom ostaju stalne kroz život, naš hod se mijenja kroz vrijeme starenjem. Mogućnost prepoznavanja i praćenja osoba bez kontakta i bez pristanka u otvara etička pitanja i ugrožava privatnost. Područje strojnog učenja se brzo razvija, usavršavaju se stare i razvijaju nove metode prepoznavanja hoda sa sve većom pouzdanosti i točnosti. Stvaraju se i novi modeli i skupovi podataka za točnije prepoznavanje hoda i u vanjskim uvjetima.

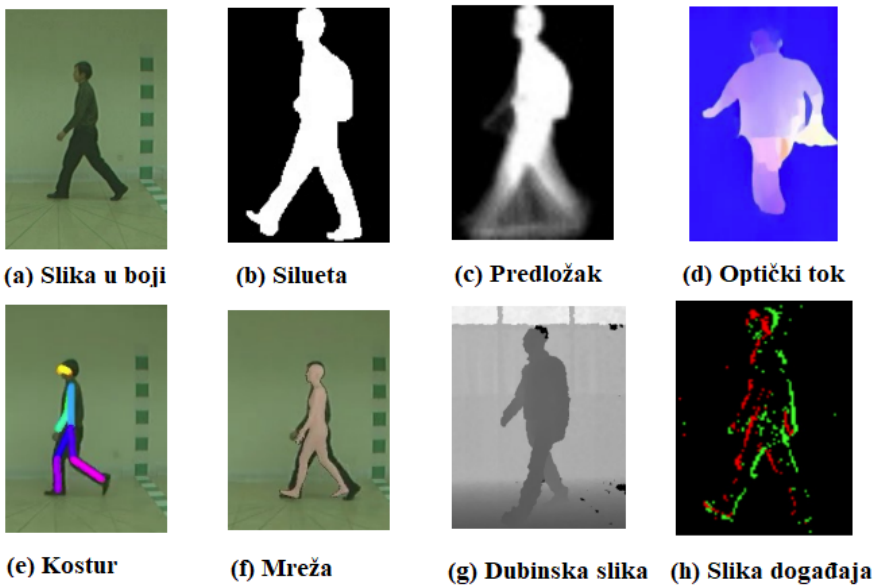
Poglavlje 2

Priprema podataka

Postoje razne vrste ulaznih podataka na kojima se treniraju modeli za prepoznavanje hoda. Najčešće se koriste jednostavne kamere koje snimaju videozapise ili slike u boji. Ako se koriste siluete osoba, nije nužna visoka rezolucija ulaznih podataka. Za snimanje u uvjetima u kojima obične kamere nisu dovoljne, na primjer snimanje noću, mogu se koristiti i infracrvene kamere, no u tom slučaju siluete ljudi su mutnije te teže prepoznatljive.

Kako bi izbjegli ovisnost izlaza i pristranost modela o odjeći koju osoba nosi najčešće se koriste crno-bijele slike. Slike se također prethodno obrade raznim metodama uklanjanja pozadina, time dobivamo siluetu osobe. Ovim metodama se gube određeni detalji iz originalne slike, ti detalji bi se mogli koristiti u naprednijim metodama za dodatno povećanje točnosti, no siluete su ipak korisne te se koriste zbog svoje jednostavnosti. Radi dodatnog uklanjanja mogućih varijacija između različitih slika iste osobe i povećavanja točnosti, siluete se mogu kombinirati u Energetsku Sliku Hoda (eng. Gait Energy Image).

Prethodno opisani ulazni podatci i metode se većinom koriste za dvodimenzionalne slike i modele, no u novije vrijeme se pojavljuju i trodimenzionalne podaci i dubinske slike koje sadrže informaciju o udaljenosti točaka na slici od kamere. Trenutno glavni nedostatak ove metode je ograničena udaljenost (oko 10 metara) na kojoj dubinske kamere mogu pouzdano snimiti udaljenost[8].



Slika 2.1 Priakz nekih od različitih vrsta ulaznih podataka. (a) slika u boji, (b) silueta, (c) predložak, (d) optički tok, (e) kostur, (f) mreža, (g) dubinska slika, (h) slika događaja

2.1 Skupovi podataka

Postoji nekoliko skupova podataka (eng. dataset) koji se koriste za prepoznavanje hoda. Neki od najvećih i najčešće korištenih su: CASIA, OUMVLP, GREW i Gait3D.

CASIA-B i OUMVLP su nešto stariji, veliki skupovi podataka za unutarnje uvjete, GREW je novi veliki skup podataka za vanjske uvjete. Gait3D je noviji skup podataka s 4 000 subjekata i podacima u tri dimenzije. Stvaranje skupa podataka za prepoznavanje hoda je teže nego stvaranje skupa podatka za ostale metode identifikacije, potrebno je više kamera koje snimaju subjekte iz više kutova, te veći prostor za pohranu snimljenih podataka[8]. Skupovi podataka su uobičajeno podijeljeni u 3 grupe: *train*, *query* i *gallery*. *Train* se koristi za treniranje modela, a *gallery* i *query* za testiranje i provjeru. Za potrebe ovog završnog rada dobio sam pristup skupovima podataka CASIA-B i OUMVLP.

2.1.1 CASIA-B

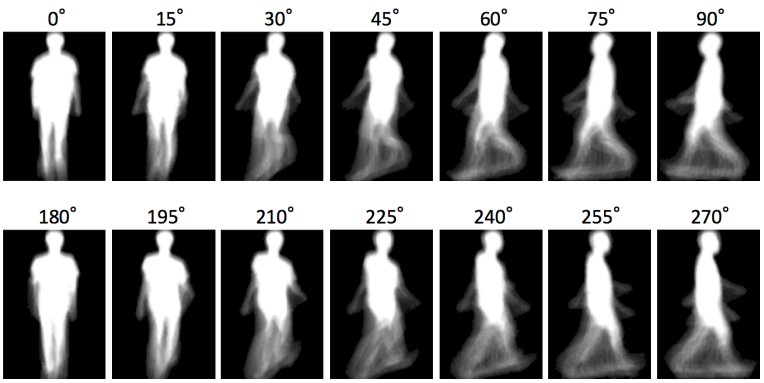
CASIA (eng. Chinese Academy of Sciences, Institute of Automation) sadržava četiri skupa podataka, A, B, C i D. U ovom projektu korišten je skup podataka CASIA-B iz 2005. godine. Sadržava 124 subjekta snimljenih iz 11 pogleda, te uključuje varijacije poput kuta pogleda, odjeće, nošenja kaputa i torbe[1, 9].

Slika 2.2 Prikaz postupka snimanja slika za CASIA-B skup podataka[1].

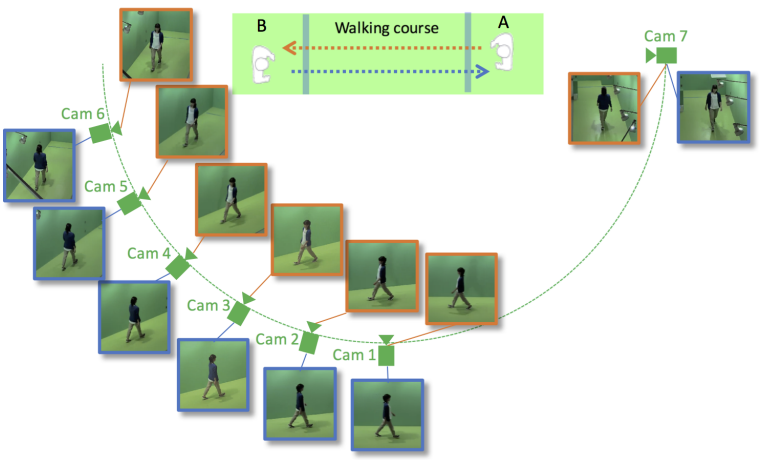
Slika 2.3 Primjer GEI (eng. Gait Energy Image) slike jednoga subjekta iz CASIA-B skupa podataka[1].

2.1.2 OUMVLP

OUMVLP (eng. OU-ISIR Multi-View Large Population Dataset) je drugi veliki, često korišteni skup podataka. Sadrži 10 307 subjekata snimljenih iz 14 pogleda. Subjekti su snimljeni pomoću 7 kamera postavljenih u četvrtinu kruga u čijem se centru nalazi područje na kojem su subjekti hodali [2].



Slika 2.4 Primjeri obrađenih slika iz OUMVLP skupa podataka[2].



Slika 2.5 Postavljenih kamera korištenih za snimanje OUMVLP skupa podataka[2].

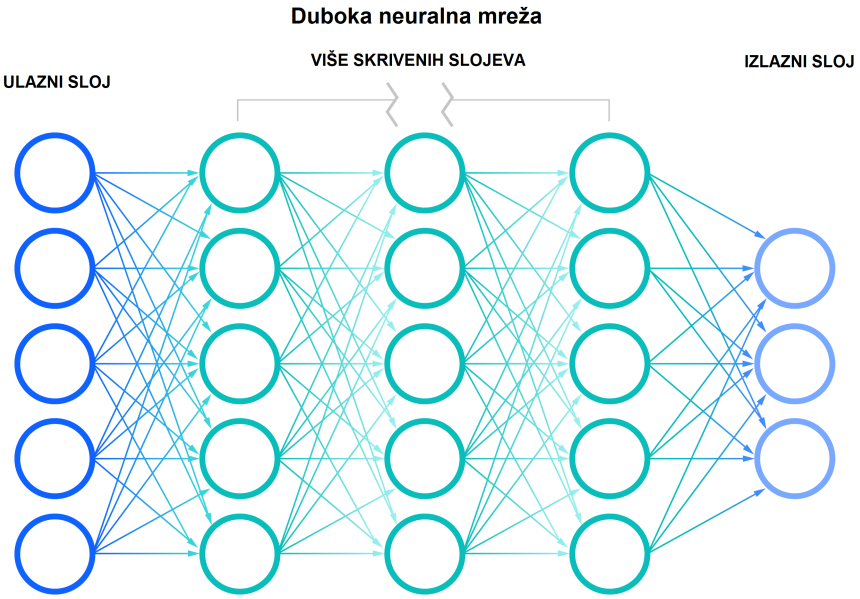
Poglavlje 3

Duboko učenje

3.1 Neuralne mreže

Duboko učenje je jedna od grana strojnog učenja koja koristi neuralne mreža od tri ili više slojeva. Ideja i struktura neuralnih mreža dolazi iz pokušaja simulacije neurona u ljudskom mozgu. Duboke neuralne mreže sastoje se od jednog ulaznog, jednog ili više skrivenih, te jednog izlaznog sloja (Slika 3.1). Pomoću ovih slojeva neuralne mreže imaju mogućnost analize i učenja na velikoj količini podataka.

Slojevi se sastoje jednog ili više čvorova, a svaki čvor ima određenu težinsku, graničnu i izlaznu vrijednost. Čvorovi u ulaznom sloju primaju vrijednosti iz ulaznih podataka. Broj ulaznih čvorova ovisi o tipu ulaznih podataka, na primjer ako kao ulaz imamo crno-bijelu sliku veličine 10 x 5 piksela možemo koristiti 50 ulaznih čvorova, po jedan za svaki pikesl. Ako je slika u boji i u RGB formatu, potrebno je 150 čvorova. Čvorovi susjednih slojeva su povezani vezama, svaka veza ima težinu koja određuje njezinu važnost. Ulazne vrijednosti nekog čvora se zbrajaju uzimajući u obzir težinu, te se ovaj zbroj koristi u aktivacijskoj funkciji koja određuje hoće li se čvor aktivirati. Dodatno čvor će se aktivirati samo ako je izlaz aktivacijske funkcije veći od čvorove granične vrijednosti. Aktivirani čvor šalje izlaznu vrijednost na sljedeći sloj. Izlazni sloj sadrži onoliko čvorova koliko imamo subjekata ili klasa koje želimo raspoznati, a vrijednosti ovih čvorova predstavljaju vjerojatnost da ulazni podaci odgovaraju izlaznom subjektu[3].



Slika 3.1 Primjera slojeva u neuralnoj mreži. Ulazni sloj je prikazan lijevo, nakon njega slijede više skrivenih slojeva, posljednji sloj je izlazni[3].

3.1.1 Konvolucijske neuralne mreže

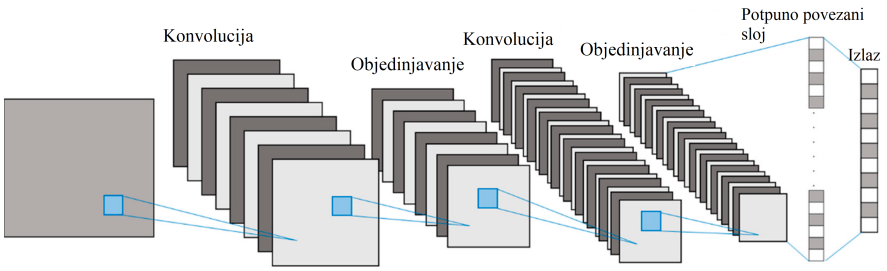
Podvrsta dubokih neuralnih mreža koja se najčešće koristi za računalni vid i analizu slika je konvolucijska neuralna mreža. Ova vrsta neuralne mreže sadrži tri vrste skrivenih slojeva: objedinjujući, konvolucijski i potpuno povezani sloj. Konvolucijski sloj je uvijek prvi sloj, te čini temeljni dio konvolucijske neuralne mreže.

Konvolucijski sloj koristi više filtera, filtar je često definiran pomoću 3 x 3 matrice, te svaki filtar može prepoznati određen uzorak u ulaznim podacima. Primjenom matematičke operacije konvolucije na ulaznim podacima, filteri prolaze kroz ulazne podatke i provjeravaju je li u njima prisutan odgovarajući uzorak. Nakon prvog konvolucijskog sloja mogu slijediti dodatni konvolucijski ili objedinjujući slojevi. Prvi konvolucijski sloj je sposoban prepoznati osnovne uzorke, a svaki sljedeći konvolucijski sloj je sposoban prepoznati apstraktnije i složenije uzorke.

Objedinjujući sloj (eng. pooling layer) služi za smanjenje rezolucije čime se sma-

njuje broj ulaznih parametara i kompleksnost mreže. Ovaj sloj također filtrom prelazi kroz podatke iz konvolucijskog sloja, no za razliku od konvolucijskog sloja, ovi filtri nemaju težinske vrijednosti. Postoje dva glavna tipa objedinjavanja: maksimalno i prosječno. Maksimalno objedinjavanje koristi filtar koji pronalazi najveću u ulazu te nju prosljeđuje prema izlazu, prosječno objedinjavanje računa i vraća prosječnu vrijednost ulaza.

Posljednji sloj je uvijek potpuno povezani sloj koji služi za klasifikaciju. Svaki izlazni čvor u ovom sloju je povezan s čvorom u prethodnom sloju. [10]



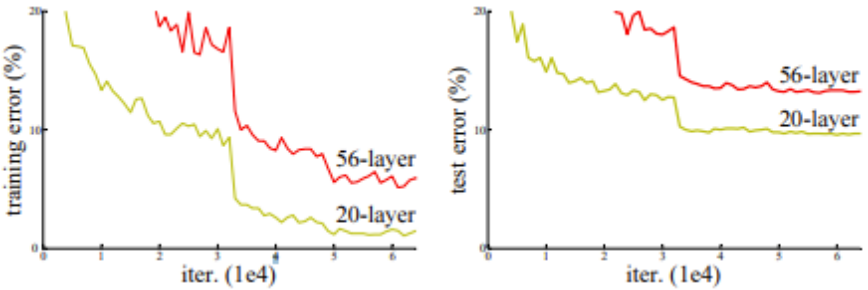
Slika 3.2 Primjera konvolucijske neuralne mreže[4].

3.2 Modeli

Neki od najčešće korištenih modela konvolucijskih neuralnih mreža su ResNet, VGGNet, ALexNet i GoogLeNet. U svojim treniranjima odlučio sam koristiti ResNet i VGGNet, često se koriste te pritom postižu dobre performanse.

Općenito se dodavanjem više slojeva u neuralnoj mreži može povećati točnost, no kod takvih vrlo dubokih neuralnih mreža pojavljuje se problem degradacije. Opaženo je povećavanje točnosti do jednog trenutka nakon kojega točnost naglo pada.

Ovaj problem je uzrokovan nestajućim gradijentima, te je posebno izražen kod mreža koje koriste aktivacijske funkcije koje imaju interval gradijenata od 0 do 1 poput hiperbolne i sigmoidne funkcije. Gradijent predstavlja brzinu rasta i promjene funkcije gubitaka (više o funkciji gubitaka u 4.4.1). Povećanjem broja slojeva u neuralnoj mreži, opada gradijent funkcije gubitaka, te kada se on približi nuli neuralnu



Slika 3.3 Prikaz grešaka kod treniranje i testiranja na CIFAR-10 skupu podataka sa 20 (označeno žuto) i 56 (označeno crveno) slojeva običnih mreža. Dublja mreža ima veću pogrešku treniranja i testiranja[5].

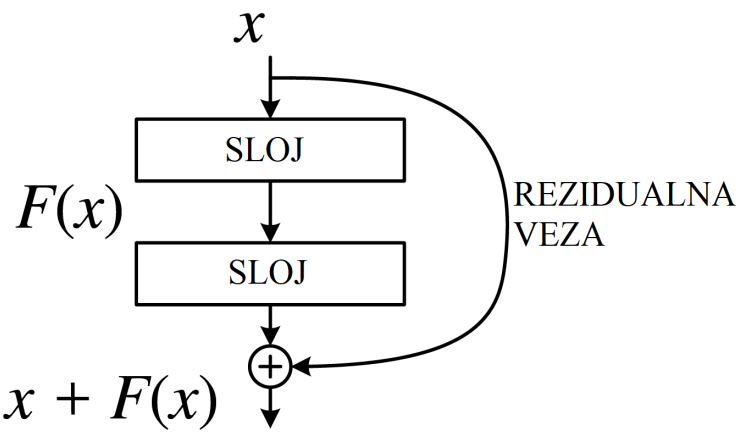
mrežu postaje teško trenirati. Gradijent se određuje preko širenja unazad (eng. backpropagation) koje se kreće od zadnjeg prema početnom sloju te podešava parametre čvorova. Širenja unazad prilagođava težišne i granične vrijednosti ovisno o funkciji gubitaka i razlici između izlaznih i ispravnih podataka sa ciljem poboljšanja predviđanja modela. Širenje unazad pritom koristi množenje derivacija koje kroz veliki broj slojeva i množenja malih derivacija uzrokuje smanjenje gradijenta prema nuli[11].

3.2.1 ResNet

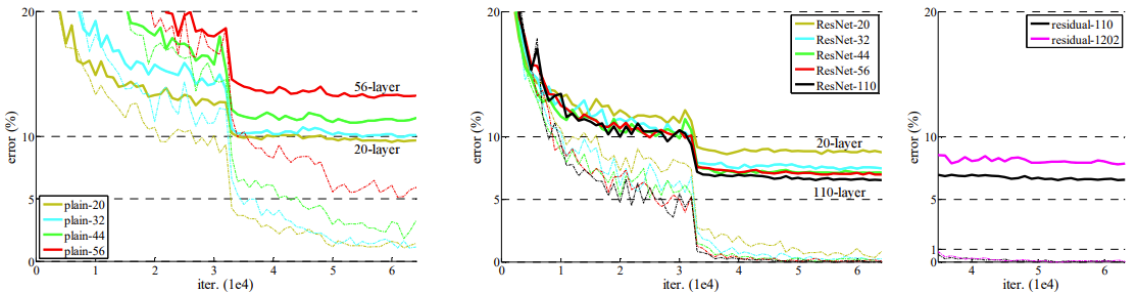
ResNet (eng. Residual Neural Network) je model koje je razvio istraživački tim iz Microsofta 2015. godine. ResNet je jedno od rješenja problema nestajućih gradijenta kod vrlo dubokih neuralnih mreža. Ovo postiže korištenjem rezidualnih veza prema ranijim slojevima. Rezidualne veze preskaču nekoliko slojeva te time izbjegavaju aktivacijsku funkciju koja uzrokuje smanjenje gradijenata. Postoji više verzija ResNeta ovisno o dubini (broju slojeva), za potrebe završnog rada koristio sam ResNet-18 (18 slojeva) i ResNet-50 (50 slojeva)[5].

3.2.2 VGGNet

VGGNet (eng. Visual Geometry Group Network) razvili su istraživači sa Sveučilišta u Oxfordu 2014. godine. Često se koristi zbog jednostavnije arhitekture i dobrih



Slika 3.4 Prikaz rezidualne veze koja direktno povezuje 2 sloja i preskače jedan među sloj[5].



Slika 3.5 Treniranje na CIFAR-10 skupu podataka. Isprekidane linije predstavljaju grešku treniranja, a cijele linije predstavljaju grešku testiranja. Lijevo je prikazana obična mreža, u sredini ResNet mreže od 30 do 110 slojeva, desno ResNet sa 110 i 1202 sloja. ResNet ostvaruje manju grešku od obične mreže sa istim brojem slojeva[5].

rezultate koje postiže. Za potrebe završnog rada koristio sam VGGNet16 dubine 16 slojeva. Ovaj tip VGGNet-a sadrži 13 konvolucijskih slojeva i 3 potpuno povezana sloja[12].

Slika 3.6 Prikaz slojeva VGGNeta[6].

Poglavlje 4

Priprema za treniranje

4.1 Predobrada skupova podataka

Nakon pripreme skupova podataka potrebno ih je reorganizirati, te preimenovati radi jednostavnijega treniranja. Skupovi podataka koje sam koristio sadržavali su slike organizirane u 3 direktorija: *train*, *query* i *gallery*. Svaki od ovih direktorija je sadržavao direktorije nazvane po identifikacijskom broju subjekta. U slučaju CASIA-B skupa podataka *query* i *gallery* sadržavaju 50 subjekata, a *train* 74 subjekta. OUMVLP skup podataka sadržavao je 5154 subjekata u *query* i *gallery* te 5153 subjekata u *train* direktoriju. Svaki od direktorija subjekata je sadržavao određen broj slika istog subjekta iz različitih kutova te s različitih stanja. Ime slike sastojalo se od stanja hodanja, kuta pogleda i rednoga broja. Stanje hodanja može biti normalno, s torbom ili s kaputom.

Slike sam reorganizirao tako da su slike svih subjekata neke kategorije u istom direktoriju. Nova imena slika sastojala su se od 4 (CASIA-B) ili 6 (OUMVLP) znamenki koje određuju identifikacijski broj subjekta, te 3 znamenke za redni broj subjekta (postoji više slika istoga subjekta). Informacije o kutu i stanju nisu bile potrebne za treniranje, te nisu uključene u novo ime datoteke iako su bile prisutne u početnom skupu podataka.

Primjer početne strukture CASIA-B skupa podataka. Prva dva slova označavaju dodatna svojstva hoda: nm - normalno, cl - subjekt nosi kaput, bg - subjekt nosi

torbu. Sljedeće tri znamenke predstavljaju kut iz kojeg je subjekt snimljen. Ostale znamenke predstavljaju redni broj slike.

```
CASIA-B Direktorij
    gallery
        subjektID1
            nm000_000001.png
            nm180_000002.png
            ...
        subjektID2
            bg072_000001.png
            bg090_000002.png
            ...
        ...
    query
        ...
    train
        ...
```

Primjer konačne strukture CASIA-B skupa podataka. Prve četiri znamenke predstavljaju identifikacijski broj subjekta, sljedeće tri znamenke služe za razlikovanje različitih slika istog

```
CASIA-B Direktorij
    gallery
            0000_000.png
            0000_001.png
            0001_000.png
            0001_001.png
            0001_002.png
            ...
        ...
    query
```

```
        ...
    train
        ...
```

Primjer početne strukture OUMVLP skupa podataka. Prve tri znamenke predstavljaju kut iz kojeg je subjekt snimljen. Ostale znamenke predstavljaju redni broj slike.

```
OUMVLP Direktorij
    gallery
        subjectID1
            000_000001.png
            180_000002.png
            ...

        subjectID2
            210_000001.png
            270_000002.png
            ...

        ...
    query
        ...
    train
        ...
```

Konačna struktura kod OUMVLP skupa podataka je slična kao i kod CASIA-B skupa podataka, jedina razlika je upotreba 6 znamenki umjesto 4 za identifikaciju subjekta zbog većeg ukupnog broja subjekata.

```
\begin{verbatim}
OUMVLP Direktorij
    gallery
            000000_000.png
```

```
            000000_001.png
            000001_000.png
            000001_001.png
            000001_002.png
            ...
        ...
    query
        ...
    train
        ...
```

4.1.1 Kod

Isječak koda za pred obradu OUMVLP skupa podataka.

```
source_folder = Path("casiab74").resolve()
destination_folder = "/ProcessedBySID/"

dest_path = source_folder.as_posix() + destination_folder
Path.mkdir(Path(dest_path), exist_ok = True)

def process_file(category, src: Path, subject, i):
    shutil.copy(src, dest_path + "/" + category + "/"  + "_".
    join([str(subject).zfill(6), str(i).zfill(3)]) + ".png")

for category in source_folder.iterdir():
    if not category.name in ["gallery", "query", "train"]:
    continue
    Path.mkdir(Path(dest_path + "/" + category.name), exist_ok
    = True)

    subject_id = 0
    for subject in category.iterdir():
        subject_counter = 0
```

```
17
18          for image in subject.iterdir():
19              process_file(category.name, image, subject_id,
    subject_counter)
20              subject_counter += 1
21          if subject_counter > 0:
22              subject_id += 1
```

4.2 Učitavanje podataka

4.2.1 Definiranje klase skupa podataka

Klasa skupa podataka ima ulogu učitavanja i transformacije pripremljenog skupa podataka iz prethodnog koraka. Definirao sam novu klasu *GaitImageDataset* koja nasljeđuje Pytorch *Dataset* klasu te mora implementirati tri metode: *_init_*, *_len_*, *_getitem_* [13].

*_init_* je konstruktor klase za učitavanje podataka, u ovoj metodi se postavljaju početne vrijednosti i stanja varijabli stvorenoga objekta. Metoda prima listu objekata *GaitImage*. *GaitImage* je jednostavna klasa koja sadrži sliku i broj subjekta prikazanog na toj slici. Konstruktor *GaitImage* klase prima put do slike i broj subjekta, te zatim pokuša otvoriti sliku koja se nalazi na tom putu.

*_len_* je metoda koja vraća dužinu skupa podataka, u ovom slučaju se jednostavno vraća duljina liste objekata *GaitImage* definirane unutar *_init_*.

*_getitem_* prima cijeli broj *index*, a vraća sliku koja se nalazi na toj poziciji u listi *GaitImage* i broj odgovarajućeg subjekta prikazanog na toj slici. Slika se dohvaća iz liste *GaitImage* objekata, te se zatim transformira. Mijenja se veličina slike iz originalne dimenzije od 240 x 240 pixela u 224 x 224 pixela koje očekuju modeli VGGNet i ResNet. Slika se nakon promjene veličine transformira u crno-bijelu sliku (izvorna slika već je crno-bijela, no ovo se radi kako bi bili sigurni da se koristi samo 1 kanal za boju a ne 3 u kasnijim metodama i kako bi dimenzije tensora bile ispravne) i pretvara u tensor. Ovako transformirani tensor slike prebacujemo

na prethodno definirani uređaj. Oznaka (broj odgovarajućeg subjekta) ove slike se također transformira u tensor.

Osim navedenih obaveznih metoda, definirana je jedna dodatna $get_subjects$. Ova metoda vraća listu oznaka jedinstvenih subjekata koja se kasnije koristi za provjeru usklađivanje broja jedinstvenih klasa u skupovima podataka i modelu.

Kod

Isječak koda koji definira *GaitImage* klasu.

```
class GaitImage():
    def __init__(self, path, subject):
        self.image = Image.open(path)
        self.subject = subject
```

Isječak koda koji definira *GaitImageDataset* klasu.

```
class GaitImageDataset(Dataset):
    def __init__(self, images):
        self.images = images

    def __len__(self):
        return len(self.images)

    def get_subjects(self):
        unique_subjects = set()

        for img in self.images:
            unique_subjects.add(img.subject)

        return unique_subjects


    def __getitem__(self, index):
        gait_image = self.images[index]
        image_PIL = gait_image.image
```

```
20
21          transform_to_tensor = transforms.Compose([
22              transforms.Resize((224, 224)),
23              transforms.Grayscale(),
24              transforms.ToTensor()
25          ])
26
27          image = transform_to_tensor(image_PIL).to(device)
28
29          label = torch.tensor(gait_image.subject)
30
31          return image, label
```

4.2.2 Učitavanje podataka

Nakon definiranja potrebnih klasa, potrebno ih je iskoristiti za učitavanje podataka. Radi jednostavnije upotrebe i izbjegavanja nepotrebne duplikacije koda program kao argumente prima skup podataka, model, stopu učenja i broj epoha. Ovisno o danim argumentima, program učitava skup podataka s definirane lokacije.

Veličina skupine Veličina skupine (eng. batch size) određuje koliko slika ulazi u jednu skupinu za treniranje i testiranje. Veće veličina skupine smanjuje vrijeme treniranja i testiranja, ali koriste i više memorije. Eksperimentiranjem sa različitim vrijednostima veličine skupina, odlučio sam koristiti veličinu skupine 32 zbog ograničenosti RAM memorije.

Iz definiranog naziva skupa podataka učitavaju se direktoriji i iteriranjem kroz slike u tim direktorijima stvaraju se objekti klase *GaitImage*, dodaju u listu i zatim se pomoću nje stvara objekt Skupovi podataka se zatim, uz veličinu skupa i opciju miješanja (eng. shuffle) koriste za stvaranje učitavatelja podataka (eng. data loader) za treniranje i testiranje preko postojeće Pytorch *DataLoader* klase [13].

Kod

Ispod je prikazan funkcija koja vrši prethodno opisan postupak stvaranja skupa podataka za treniranje i testiranje.

```
def load_datasets(dataset, batch_size=32, shuffle=True):
    train_folder = Path("/content/" + dataset +"/ProcessedBySID/gallery").resolve()
    query_folder = Path("/content/" + dataset +"/ProcessedBySID/query").resolve()

    train_images = []
    test_images = []

    for img in sorted(train_folder.iterdir()):
        train_images.append(GaitImage(img.as_posix(), img.name[:-8]))

    train_dataset = GaitImageDataset(train_images)

    for img in sorted(query_folder.iterdir()):
        test_images.append(GaitImage(img.as_posix(), img.name[:-8]))

    test_dataset = GaitImageDataset(test_images)

    train_loader = DataLoader(
        train_dataset,
        batch_size,
        shuffle
    )

    test_loader = DataLoader(
        test_dataset,
        batch_size,
        shuffle
```

```
    )

    print("BATCH SIZE:    ", batch_size)

    return train_dataset, test_dataset, train_loader,
test_loader
```

4.3 Priprema modela

Nakon učitavanja podataka potrebno je učitati i postaviti model. Ovisno o danim argumentima, program preuzima jedan od tri predviđena modela, te ih dodatno prilagođava izabranom skupu podataka. Preuzimanje odgovarajućeg modela je jednostavno zahvaljujući Pytorch dokumentaciji [13]. Oba skupa podataka korištena u ovom projektu se sastoje od crno-bijelih slika pa je potrebno izmijeniti prvi konvolucijski sloj modela tako da primaju samo jedan ulazni kanal koji predstavlja intenzitet od crne do bijele umjesto 3 kanala od kojih svaki predstavlja jednu boju (crvenu, zelenu, plavu). Dodatno je potrebno i prilagoditi posljednji potpuno povezani sloj tako da broj izlaza odgovara broju jedinstvenih subjekata u skupu podataka na kojemu se trenira.

Kod

Ispod je prikazan funkcija koja vrši prethodno opisan postupak preuzimanja, učitavanja i prilagodbe izabranoga modela.

```
def get_model(model_arg, num_classes):
    # Preuzmi i ucitaj model
    model = torch.hub.load('pytorch/vision:v0.10.0', model_arg)

    if model_arg == "vgg16":
        # Izmjeni prvi konvolucijski sloj tako da prima crno-
bijele slike
        model.features[0] = nn.Conv2d(1, 64, kernel_size=3,
stride=1, padding=1)
```

```
8
9           # Dohvati broj ulaznih veza iz prethodnog sloja
10          in_features = model.classifier[6].in_features
11
12          # Zamijeni broj izlaznih klasa u posljednjem, potpuno
    povezanom sloju
13          model.classifier[6] = nn.Linear(in_features,
    num_classes)
14      else:
15          # Izmjeni prvi konvolucijski sloj tako da prima crno-
    bijele slike
16          model.conv1 = nn.Conv2d(1, 64, kernel_size=7, stride=2,
     padding=3)
17
18          # Dohvati broj ulaznih veza iz prethodnog sloja
19          in_features = model.fc.in_features
20
21          # Zamijeni broj izlaznih klasa u posljednjem, potpuno
    povezanom sloju
22          model.fc = nn.Linear(in_features, num_classes)
23
24      model.to(device)
25
26      return model
```

4.4 Definiranje funkcije gubitaka i optimizacije

4.4.1 Funkcija gubitaka

Model vraća vektor vrijednosti od 0 do 1, veličina vektora je broj jedinstvenih subjekata ili klasa a vjerojatnosti predstavljaju vjerojatnost da je na ulaznoj slici prikazan odgovarajući subjekt. Funkcija gubitaka izračunava pogrešku ili razliku između stvarne vrijednosti i izlaza modela. Koristi se za procjenu točnosti modela i za us-

mjeravanje modela prema boljim rezultatima. Cilj modela je minimizirati funkciju gubitaka, što je rezultat ove funkcije manji to je model bolji.

U ovom projektu koristi se "Cross Entropy Loss" funkcija gubitaka, poznata i pod nazivom logaritmička funkcija gubitaka (eng. log loss), ovo je često korištena funkcija gubitaka za treniranje modela za klasifikaciju. Zbog primjene logaritma ova vrsta funkcije gubitka vraća eksponencijalno veće rezultate, što je razlika bliža 1 to je vrijednost koju ova funkcija vrača veća. Funkcija gubitaka savršenog modela bi uvijek vraćala 0[14, 15].

Pytorch sadrži knjižicu *torch.nn* unutar koje je već definirana *CrossEntropyLoss* funkcija korištena u ovom projektu[13].

4.4.2 Optimizacijska funkcija

Optimizacijska funkcija ima ulogu ažuriranja parametara modela za vrijeme procesa treniranja. Funkcija određuje koje parametre mora ažurirati na temelju rezultat funkcije gubitaka. U ovom projektu korištena je Adam (eng. Adaptive moment estimation) optimizacijska funkcija. Također sam testirao i rezultati sa SGD (eng. Stochastic gradient descent) optimizacijskom funkcijom, te sam zaključio da Adam puno brže konvergira i postiže veću točnost od SGD[16].

Optimizacijskoj funkciji prosljeđujem stopu učenja iz argumenata pri pokretanju programa. Stopa učenja određuje brzinu promjene parametara modela. Veće vrijednosti rezultiraju bržim konvergiranjem modela i lakšim izbjegavanjem lokalnih minimuma funkcije gubitaka. Manje vrijednosti zahtijevaju više vremena ali smanjuju vjerojatnost prevelike izmjene parametara modela što može dovesti do nestabilnih i neželjenih rezultata.

Optimizacijska funkcija dodatno prima i vrijednost propadanja težina (eng. weight decay). Propadanje težina ili L2 regularizacija pomaže smanjiti preveliko prilagođavanje modela skupu za treniranje (eng. overfitting) tako da smanjuje težinu čvorova.

Overfitting

Do prevelikog prilagođavanja može doći ako je skup podataka nedovoljne veličine, model prekompleksan ili treniranje traje pre dugo. U tim slučajevima model može početi učiti nevažne informacije prisutne u skupu podataka za treniranje te postaje točan jedino na podacima na tim podacima[17].

Pytorch sadrži knjižicu *torch.optim* unutar koje je su već definirane Adam i SGD funkcije korištena u ovom projektu[13].

4.4.3 Kod

Ispod je prikazan kod za definiranje ovih dviju funkcija.

```
# Funckija gubitaka
loss_fn = nn.CrossEntropyLoss()
loss_fn.to(device)

# Optimizacijska funkcija
optimizer = optim.Adam(model.parameters(), lr, weight_decay
    =0.001)
```

Poglavlje 5

Treniranje

5.1 Glavnja petlja treniranja

Treniranje se vrši u određenom broju epoha. Veći broj epoha ostvaruje bolje rezultate, no pritom mu je potrebno više vremena. Postoji granični broj epoha nakon koje dodatno treniranje neće postizati veću točnost.

5.1.1 Ciklus treniranja

Jedan ciklus treniranja se sastoji od iteriranja kroz učitani skup podataka za treniranje. Prvo se resetiraju gradijenti optimizacijske funkcije, zatim se tensori slika i oznaka jedne skupine podataka predaju modelu koji vraća svoja predviđanja. Ova predviđanja zajedno sa stvarnim oznakama subjekata se predaju funkciji gubitaka. Rezultat funkcije gubitaka se koristi za prilagođavanje modela preko optimizacijske funkcije i širenja unatrag, Na kraju se izračunava postotak točnosti modela koji se ispisuje zajedno s rezultatom funkcije gubitaka radi praćenja napretka.

5.1.2 Ciklus testiranja

Nakon ciklusa treniranja vrši se ciklus testiranja. Ciklus testiranja sličan je ciklusu testiranja, glavna razlika je prebacivanje modela u način rada za testiranje. Model

se testira na učitanom skupu podataka za testiranje. Izračunava se funkcija gubitaka radi ispisa i praćenja njene vrijednosti, no ne odrađuje se optimizacija i prilagodba parametara modela. Također se izračunava i ispisuje prosječna točnost ciklusa.

5.1.3 Kod

Ispod su prikazane funkcije za treniranje i testiranje jedne epohe.

```
def train_one_epoch(model, optimizer, loss_fn, train_loader):
    running_loss = 0.0
    avg_acc = 0.0
    avg_top5 = 0.0

    for i, data in enumerate(train_loader):
        # Dohvacanje podataka iz ucitanog skupa podataka
        inputs, labels = data
        inputs = inputs.to(device)
        labels = labels.to(device)

        # Resetiranje gradijenata optimizacijske funkcije
        optimizer.zero_grad()

        # Model odraduje predvidanja
        outputs = model(inputs)

        # Izracun funkcije gubitaka
        loss = loss_fn(outputs, labels)
        loss.backward()

        # Prilagodba parametara
        optimizer.step()

        # Izracun i ispis prosjecne funckije gubitaka i
    tocnosti
        last_loss = loss.detach().item()
```

```
27          running_loss += last_loss
28          top1_accuracy = calculate_topk_accuracy(outputs, labels
    , k=1)
29          top5_accuracy = calculate_topk_accuracy(outputs, labels
    , k=5)
30          avg_acc += top1_accuracy
31          avg_top5 += top5_accuracy
32
33          if i % 1000 == 0:
34              print('     batch {} loss: {} top-1 accuracy: {:.2f
    }% top-5 accuracy: {:.2f}%'.format(i, last_loss,
    top1_accuracy, top5_accuracy))
35
36
37
38      return running_loss / len(train_loader), avg_acc / len(
    train_loader), avg_top5 / len(train_loader)
39
40  def test_one_epoch(model, loss_fn, test_loader):
41      running_loss = 0.0
42      avg_acc = 0.0
43      avg_top5 = 0.0
44
45      for i, data in enumerate(test_loader):
46          # Dohvacanje podataka iz ucitanog skupa podataka
47          inputs, labels = data
48          inputs = inputs.to(device)
49          labels = labels.to(device)
50
51          # Model odraduje predvidanja
52          outputs = model(inputs)
53
54          # Izracun i ispis prosjecne funckije gubitaka i
    tocnosti
55          loss = loss_fn(outputs, labels)
```

```
56
57          last_loss = loss.detach().item()
58          running_loss += last_loss
59          top1_accuracy = calculate_topk_accuracy(outputs, labels
    , k=1)
60          top5_accuracy = calculate_topk_accuracy(outputs, labels
    , k=5)
61          avg_acc += top1_accuracy
62          avg_top5 += top5_accuracy
63
64          if i % 1000 == 0:
65              print('     batch {} loss: {} top-1 accuracy: {:.2f
    }% top-5 accuracy: {:.2f}%'.format(i, last_loss,
    top1_accuracy, top5_accuracy))
66
67      return running_loss / len(test_loader), avg_acc / len(
    test_loader), avg_top5 / len(test_loader)
```

5.2 Hardver i softver korišten za treniranje

Za treniranje je korišten besplatni alat Google Colaboratory zbog nedostatka grafičke kartice dovoljne procesorske snage i virtualne memorije na vlastitom uređaju.[18]. Colaboratory je Googleov servis koji omogućuje besplatno korištenje udaljenih Googleovih uređaja za izvođenje koda u edukacijske svrhe. Uz sve prednosti ovog servisa, postoji i nekoliko nedostataka besplatne verzije: ograničenje na nešto sporiju grafičku karticu Nvidia Tesla T4 i dnevno maksimalno vremensko ograničenje korištenja ovog servisa i grafičke kartice. Uređaji imaju 12 GB RAM-a što ograničava veličinu skupine i usporava treniranje.

Poglavlje 6

Mjere i rezultati

6.1 Mjere

Pri treniranju modela mjerio sam tri glavne vrijednosti kako bih procijenio kvalitetu modela: vrijednost funkcije gubitka, točnost modela i top-5 točnost

6.1.1 Vrijednost funkcije gubitka

Vrijednost funkcije gubitka detaljnije je opisana prethodno u sekciji 4.4.1. Funkcija gubitaka prikazuje koliko se izlaz modela razlikuje od stvarnih točnih vrijednosti. Manji broj je bolji.

6.1.2 Točnost

Točnost se izračunava kao omjer ispravno prepoznatih subjekata u odnosu na ukupan broj subjekata.

$$tocnost = ispravniSubjekti/ukupniSubjekti * 100$$

ispravniSubjekti je broj subjekata koje je model ispravno prepoznao. Ispravno prepoznati subjekt je onaj subjekt čija je istinita oznaka jednaka onoj za koja u izlazu modela ima najveću vjerojatnosti. Točnost se prikazuje postotcima. Veći broj je bolji.

6.1.3 Top-5 točnost

Top 5 predstavlja broj pojavljivanja ispravnog subjekta unutar 5 najvećih vrijednosti za koje model misli da su prepoznati subjekt. Top 5 točnost se računa kao omjer top-5 u odnosu na ukupan broj subjekata. Točnost iz prethodne sekcije je zapravo top-1 točnost. Top-5 točnost može biti korisna za provjeru kreću li se predviđanja modela u ispravnom smjeru, na primjer, ako je top-1 točnost 0 a top-5 točnost visoka, možemo zaključiti da je se predviđanje modela kreće u ispravnom smjeru ali je potreban dodatan rad na detaljima kako bi se usavršio.

$$top5Tocnost = KispravniSubjekti/ukupniSubjekti * 100$$

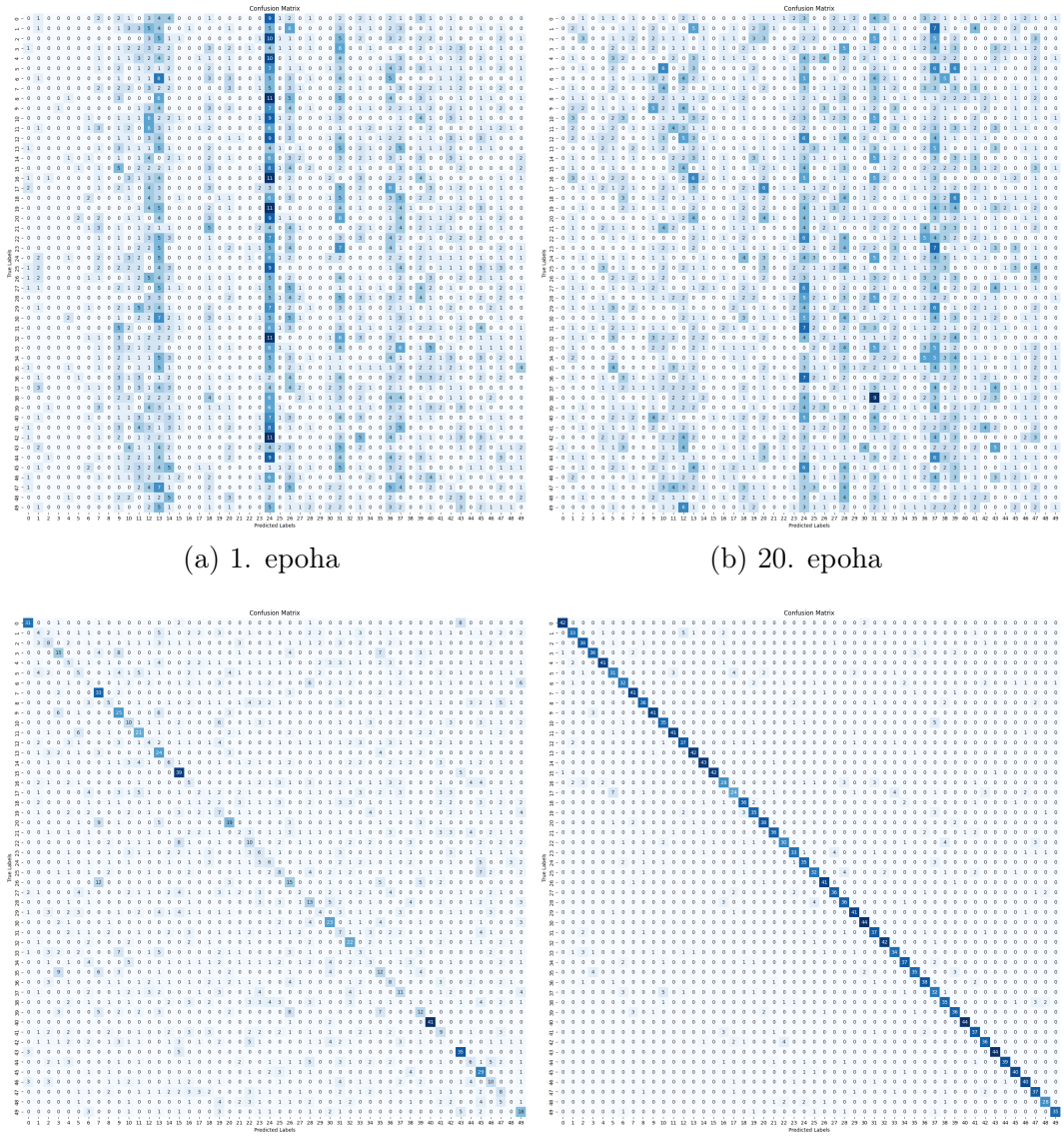
KispravniSubjekti je broj pojavljivanja ispravno prepoznatog subjekta u prvih K subjekata koje je model predvidio sa najvećom vjerojatnosti da se nalaze na slici. Top-5 točnost se prikazuje postotcima. Veći broj je bolji.

6.2 Rezultati

Broj epoha i stopa učenja se razlikuju ovisno o kombinaciji modela i skupa podataka. Testovi su provedeni sa različitim brojevima epoha i stopama učenja a detaljnije su navedeni i prikazani rezultati za one vrijednosti koje su ostvarile najbolji rezultat.

6.2.1 CASIA-B VGG-16

Za skup podataka CASIA-B i model VGG-16 najbolji rezultati su ostvareni uz stopu učenja od 0.000001 nakon 120 epoha. Model ostvaruje 84.2% točnosti pri treniranju i 50.15% točnosti pri testiranju, te 98.87% top-5 točnosti treniranja i 69.68% točnosti testiranja. Za bolje razumijevanje rezultata može biti korisna matrica previđenih subjekata i ispravnih subjekata (eng. confusion matrix) [Slika 6.1].

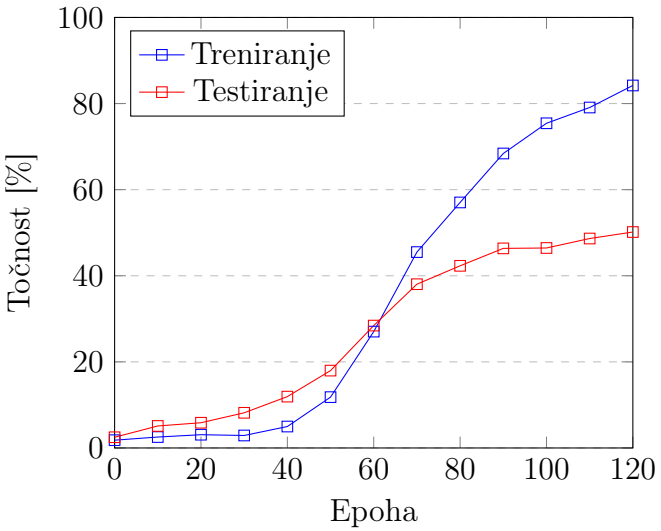


(a) 1. epoha (b) 20. epoha

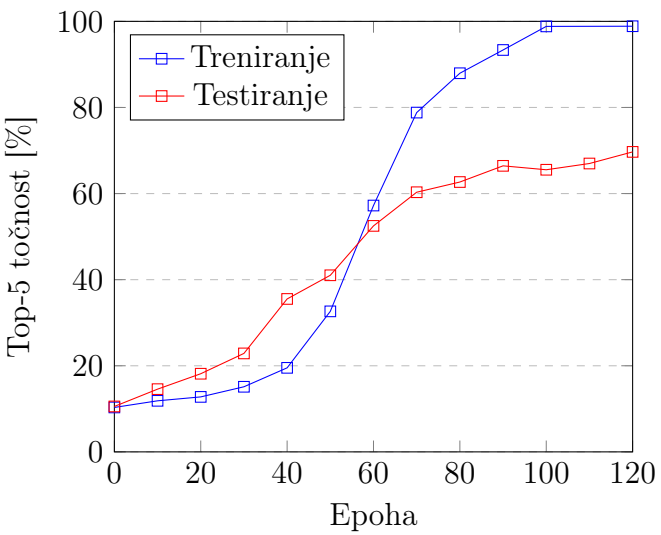
(c) 50. epoha (d) 120.epoha

Slika 6.1 Confusion matica za CASIA-B skup podataka i VGG-16 model. Na vertikalnoy osi su prikazani ispravni subjekti a na horizontalnoj osi predviđanja modela.

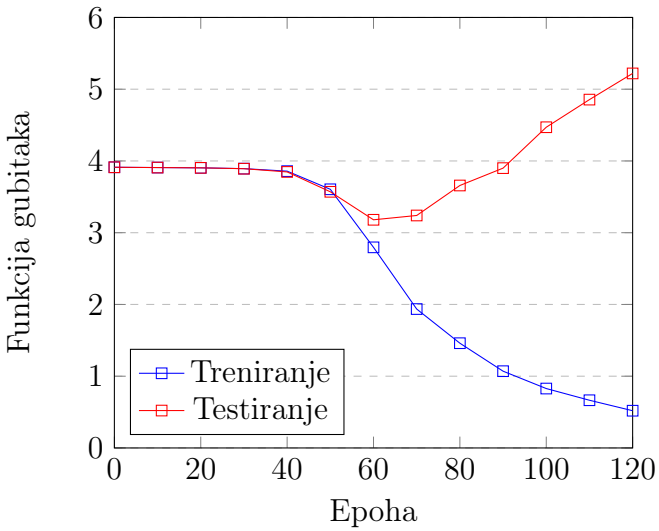
Točnost treniranja i testiranja CASIA-B na modelu VGG-16



Top-5 točnost treniranja i testiranja CASIA-B na modelu VGG-16

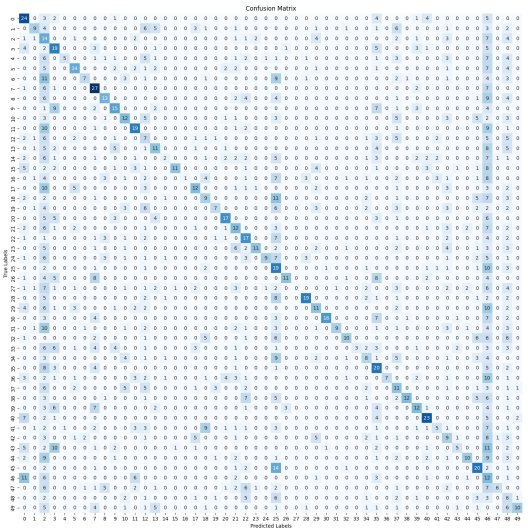


Funkcija gubitaka treniranja i testiranja CASIA-B na modelu VGG-16

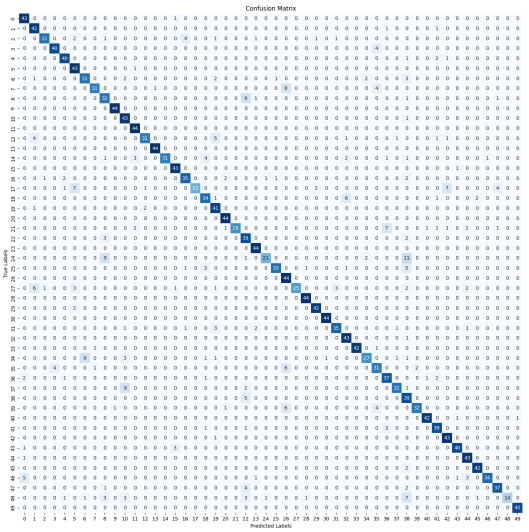


6.2.2 CASIA-B ResNet-18

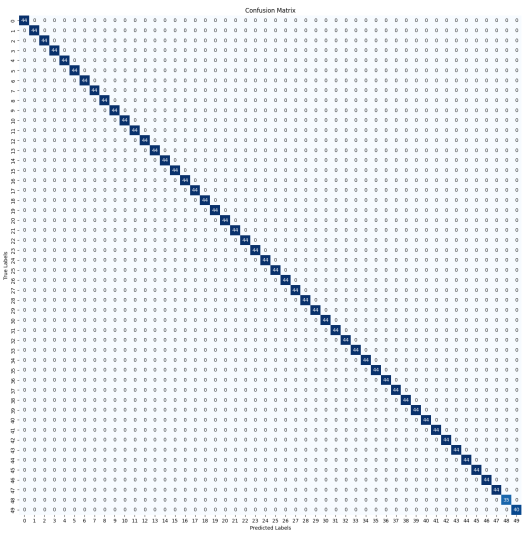
Za skup podataka CASIA-B i model ResNet-18 najbolji rezultati su ostvareni uz stopu učenja od 0.0001 nakon 10 epoha. Model ostvaruje 100% točnosti pri treniranju i 61.45% točnosti pri testiranju, te 100% top-5 točnosti treniranja i 74.37% točnosti testiranja.



(a) 1. epoha



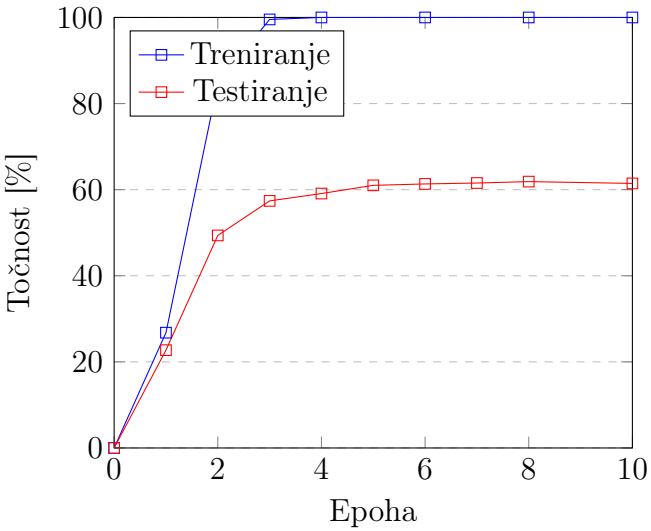
(b) 2. epoha



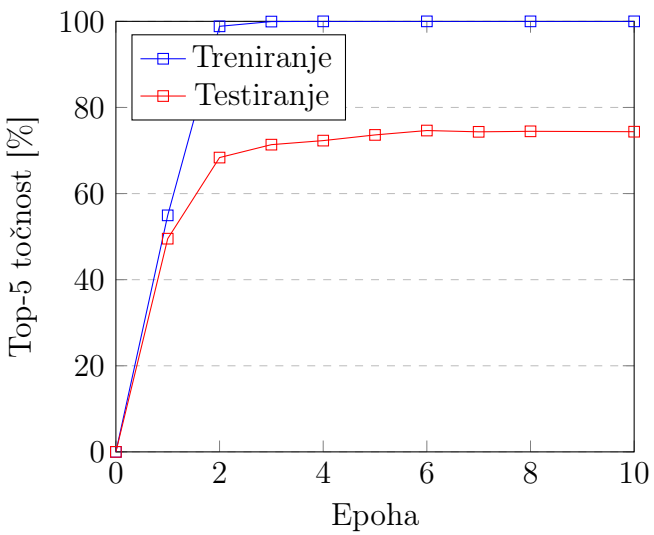
(c) 10. epoha

Slika 6.2 Confusion matica za CASIA-B skup podataka i ResNet-18 model. Na vertikalnoy osi su prikazani ispravni subjekti a na horizontalnoj osi predviđanja modela.

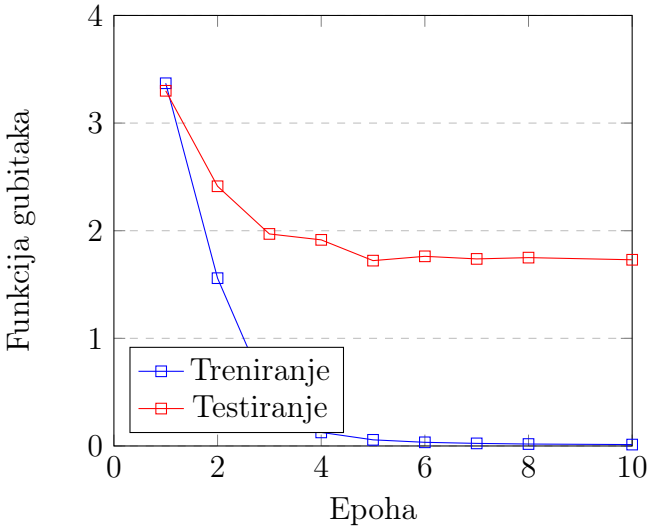
Točnost treniranja i testiranja CASIA-B na modelu ResNet-18



Top-5 točnost treniranja i testiranja CASIA-B na modelu ResNet-18



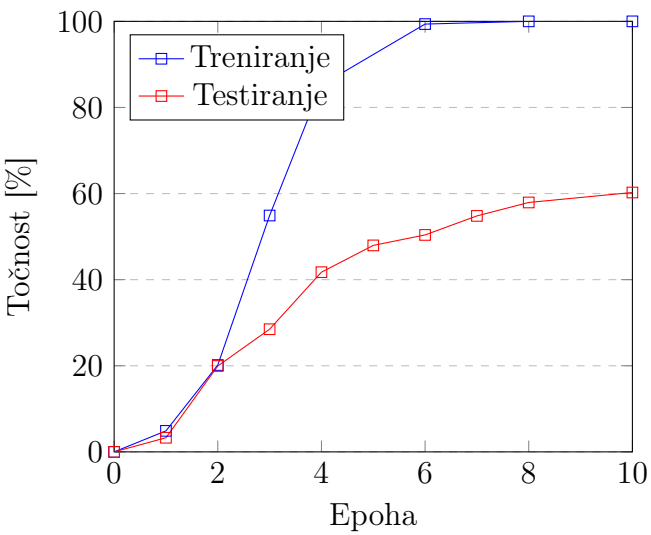
Funkcija gubitaka treniranja i testiranja CASIA-B na modelu ResNet-18



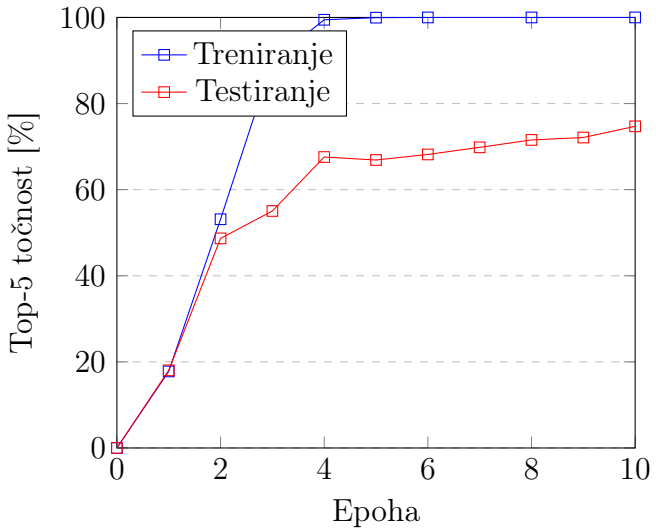
6.2.3 CASIA-B ResNet-50

Za skup podataka CASIA-B i model ResNet-50 najbolji rezultati su ostvareni uz stopu učenja od 0.0001 nakon 10 epoha. Model ostvaruje 100% točnosti pri treniranju i 60.25% točnosti pri testiranju, te 100% top-5 točnosti treniranja i 74.71% točnosti testiranja.

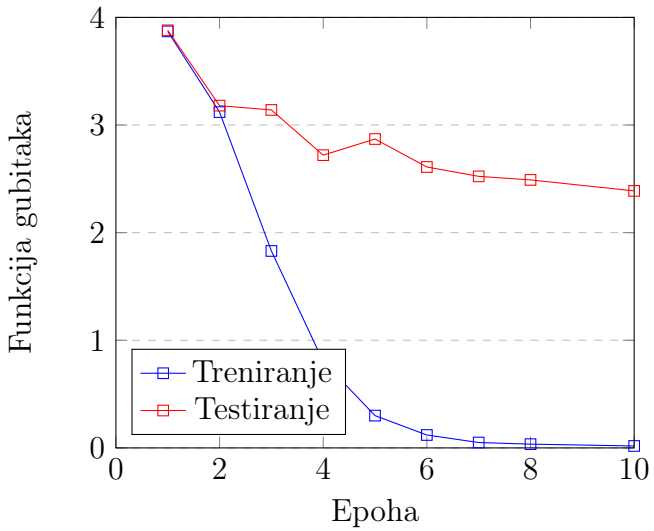
Točnost treniranja i testiranja CASIA-B na modelu ResNet-50



Top-5 točnost treniranja i testiranja CASIA-B na modelu ResNet-50

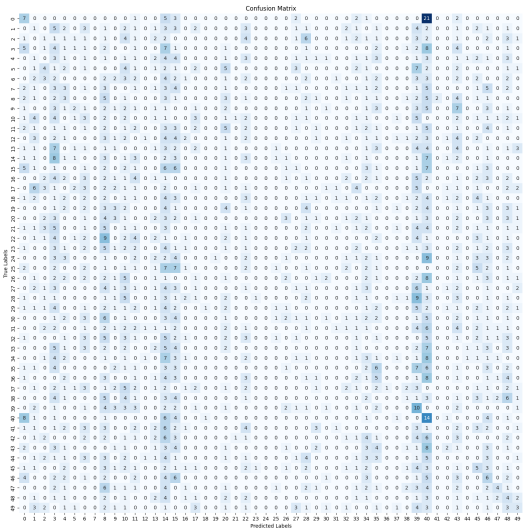


Funkcija gubitaka treniranja i testiranja CASIA-B na modelu ResNet-50

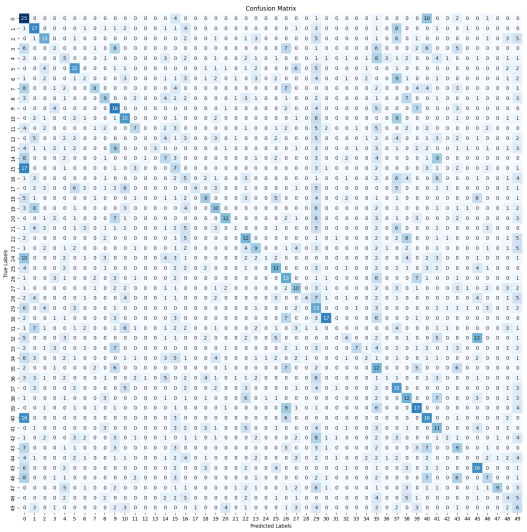


6.2.4 OUMVLP VGG-16

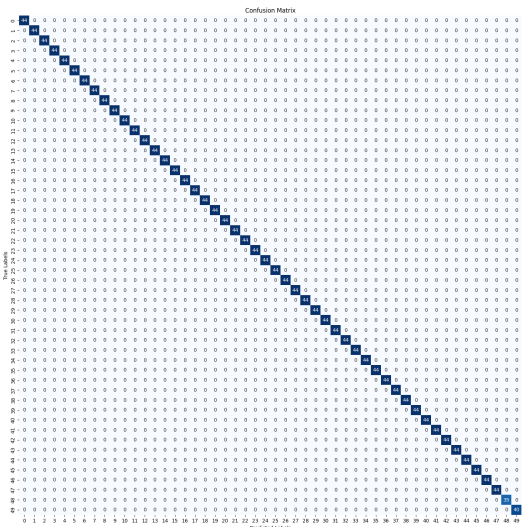
Za skup podataka OUMVLP i model VGG-16 rezultati su vrlo niski, a funkcija gubitaka se povećavala umjesto smanjivala, treniranje je zbog tog razloga prekinuto nakon 5 epoha.



(a) 1. epoha



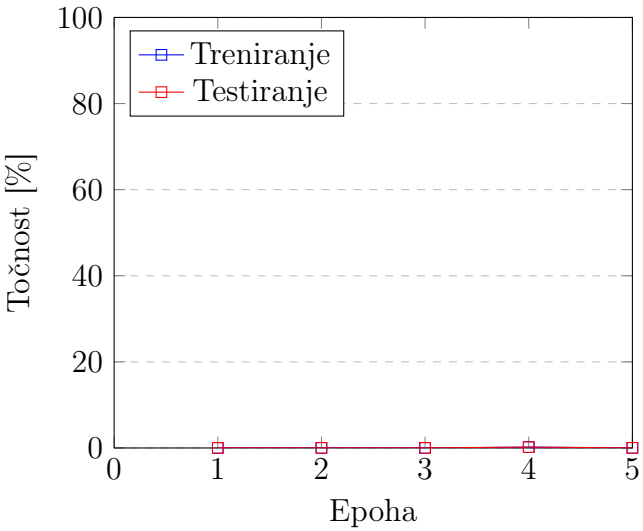
(b) 2. epoha



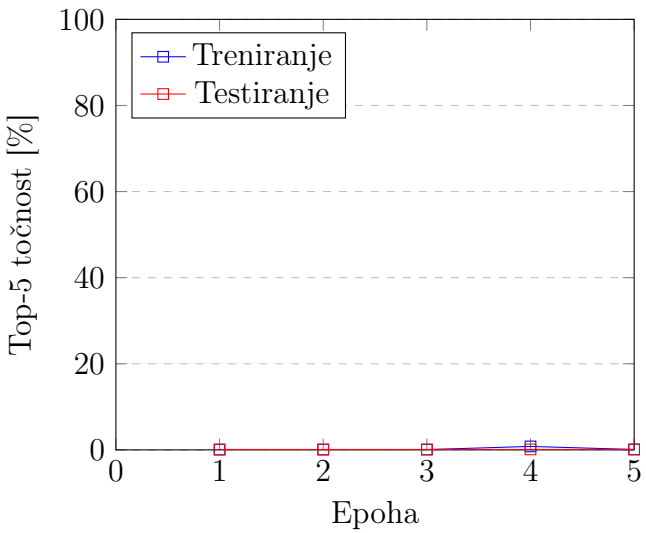
(c) 10. epoha

Slika 6.3 Confusion matica za CASIA-B skup podataka i ResNet-50 model. Na vertikalnoy osi su prikazani ispravni subjekti a na horizontalnoj osi predviđanja modela.

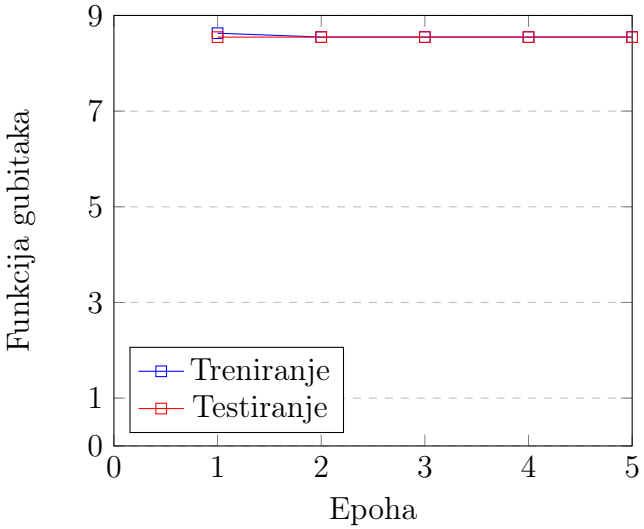
Točnost treniranja i testiranja CASIA-B na modelu ResNet-50



Top-5 točnost treniranja i testiranja CASIA-B na modelu ResNet-50



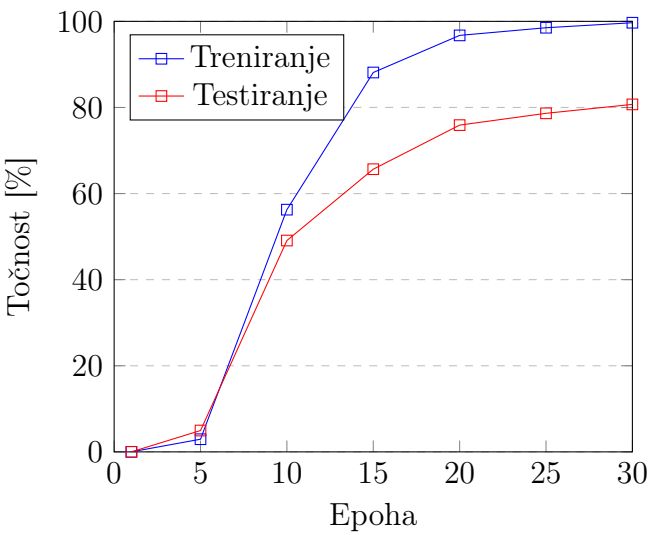
Funkcija gubitaka treniranja i testiranja CASIA-B na modelu ResNet-50



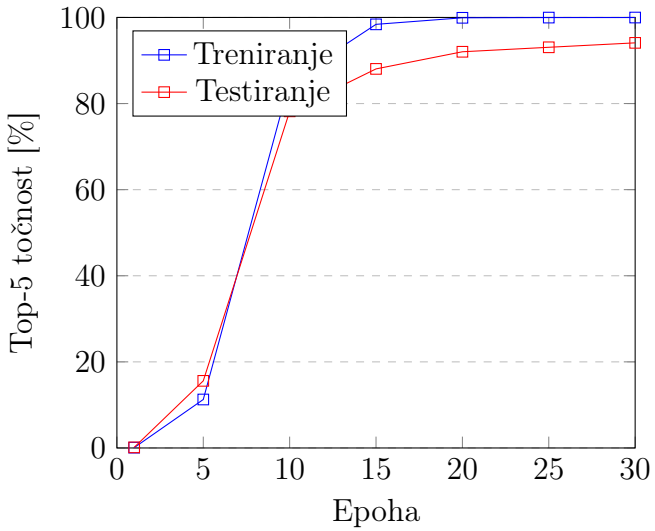
6.2.5 OUMVLP ResNet-18

Za skup podataka OUMVLP i model ResNet-18 najbolji rezultati su ostvareni uz stopu učenja od 0.001 nakon 30 epoha. Model ostvaruje 99.68% točnosti pri treniranju i 80.71% točnosti pri testiranju, te 99.98% top-5 točnosti treniranja i 94.10% točnosti testiranja.

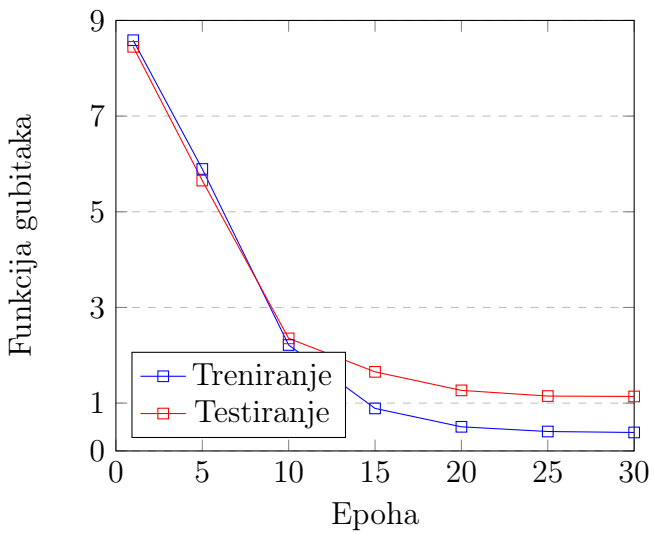
Točnost treniranja i testiranja CASIA-B na modelu ResNet-50



Top-5 točnost treniranja i testiranja CASIA-B na modelu ResNet-50



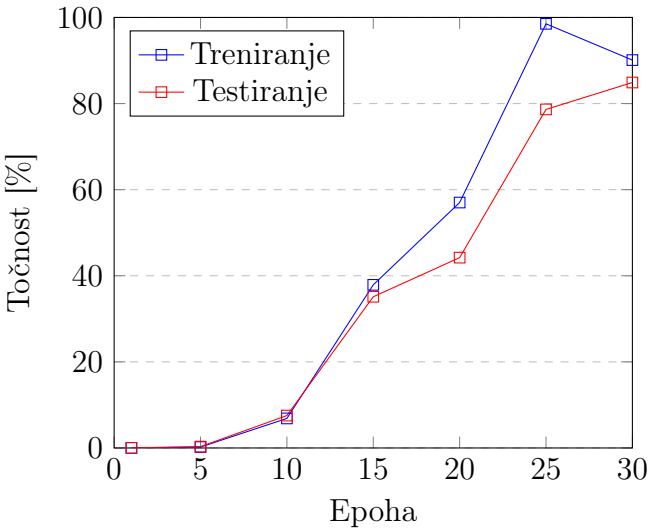
Funkcija gubitaka treniranja i testiranja CASIA-B na modelu ResNet-50



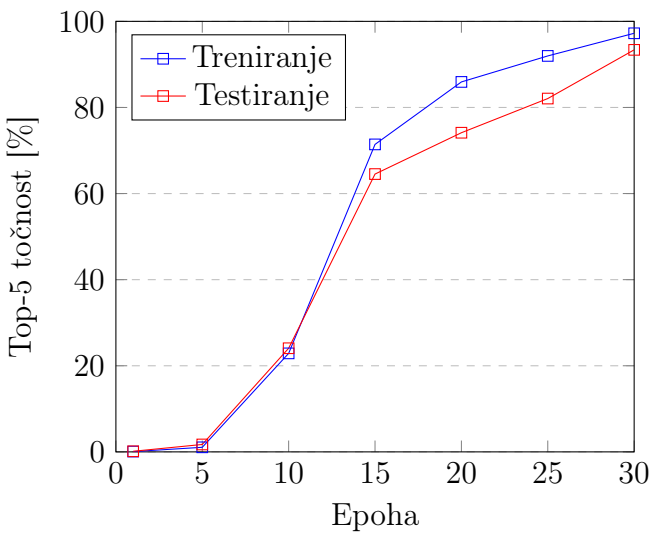
6.2.6 OUMVLP ResNet-50

Za skup podataka OUMVLP i model ResNet-50 najbolji rezultati su ostvareni uz stopu učenja od 0.001 nakon 30 epoha. Model ostvaruje 90.10% točnosti pri treniranju i 84.89% točnosti pri testiranju, te 97.23% top-5 točnosti treniranja i 93.38% točnosti testiranja.

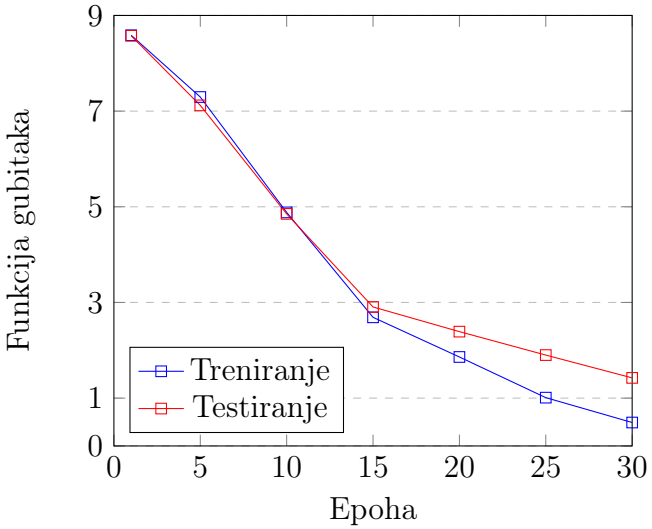
Točnost treniranja i testiranja CASIA-B na modelu ResNet-50



Top-5 točnost treniranja i testiranja CASIA-B na modelu ResNet-50



Funkcija gubitaka treniranja i testiranja CASIA-B na modelu ResNet-50



6.2.7 Rezultati po modelu

Prikaz konačnih rezultata grupiranih po modelu.

Tablica 6.1 Rezultati za VGG-16 model

Skup podataka	Tip	Broj epoha	Funkcija gubitka	Točnost	Top-5 točnost
CASIA-B	Train	120	0.5191	84.2%	98.87%
CASIA-B	Test	120	5.2200	50.16%	69.68%
OUMVLP	Train	5	8.548	0.02%	0.08%
OUMVLP	Test	5	8.548	0.02%	0.10%

Tablica 6.2 Rezultati za ResNet-18 model

Skup podataka	Tip	Broj epoha	Funkcija gubitka	Točnost	Top-5 točnost
CASIA-B	Train	10	0.013	100%	100%
CASIA-B	Test	10	1.730	64.45%	74.37%
OUMVLP	Train	30	0.386	99.68%	99.98%
OUMVLP	Test	30	1.129	80.71%	94.10%

Tablica 6.3 Rezultati za ResNet-50 model

Skup podataka	Tip	Broj epoha	Funkcija gubitka	Točnost	Top-5 točnost
CASIA-B	Train	10	0.519	100%	100%
CASIA-B	Test	10	5.22	60.25%	74.71%
OUMVLP	Train	30	0.49	90.10%	97.23%
OUMVLP	Test	30	1.423	84.89%	93.38%

Poglavlje 7

Zaključak

Skup podataka za treniranje očekivano postiže bolje rezultate od skupa za treniranje te ima nižu funkciju gubitaka. Dublji modeli s rezidualnim vezama postižu bolje rezultate te im je pritom potrebno puno manje epoha. Treniranje i testiranje na CASIA-B skupu podataka je očekivano puno brže od treniranja na većem skupu podataka OUMVLP. Točnost treniranja kod CASIA-B skupa podataka je viša (u nekim slučajevima 100%) od točnosti kod OUMVLP, kod CASIA-B skupa veća je i razlika između točnosti treniranja i testiranja. OUMVLP skup podataka u prosjeku postiže malo nižu točnost, ali razlika između točnosti treniranja i testiranja je manja.

VGG-16 model u svim slučajevima postiže lošije rezultate od ResNet-a. Objašnjenje za lošije rezultate može biti manji broj slojeva, nedostatak rezidualnih veza te postupno nestajanje gradijenata. Rezultati VGG-16 modela za veliki OUMVLP skup podataka su niski zbog nedovoljne kompleksnosti modela. Treniranje je prekinuto nakon nedostatka promjene u rezultatima nakon epoha zbog vremenskog ograničenja Google Colaboratory-a. ResNet-18 i ResNet-50 razlikuju se u broju slojeva i ostvaruju slične rezultate. Veći broj slojeva neuralne mreže općenito pomaže u prepoznavanju više detalja i time pridonosi većoj točnosti, no čak i 18 slojeva postiže vrlo visoku točnost i sposobno je prepoznati i klasificirati subjekte. Zbog većeg broja slojeva treniranje na ResNet-50 modelu zahtijeva više vremena što je posebno izraženo na velikom OUMVLP skupu podataka.

Osim razlika u točnosti, postoji i razlika u broju epoha potrebnih za postizanje

dobrih rezultata. Za VGG-16 potrebno je 12 puta više epoha nego ResNet-u na CASIA-B skupu podataka.

7.1 Moguća poboljšanja

Jedno od mogućih poboljšanja kojime bi se ostvario još bolji rezultat je veći broj epoha kod OUMVLP skupa podataka. Broj epoha je bio ograničen na 30 zbog vremenski zahtjevnom treniranja na velikom broju subjekata, te hardverskih i softverskih ograničenja Google Colaboratory servisa koji sam koristio za pokretanje programa.

Dodatno poboljšanje bi se moglo postići i manjom veličinom skupine, no glavni nedostatak je isti kao i u slučaju većeg broja epoha, uz manju veličinu skupine povećava se vrijeme potrebno za treniranje svake epohe.

Još jedno moguće poboljšanje je dodatna izmjena nekih od parametara funkcije gubitaka i optimizacije. Osim stope učenja i propadanja težina, postoje brojni drugi parametri čijim izmjenama je moguće postići bolje rezultate. Osim "Cross Entropy Loss" postoje i druge funkcije gubitka, na primjer, "Hinge Loss"[15], te je moguće da ni neke od njih ostvarile bolje rezultate. Osim "Adam" postoje i brojne funkcije optimizacije, neke od često korištenih su: "SGD", "RMSProp" i "AdaGrad"[16].

OUMVLP skup podataka sam trenirao i na k-podijeljenoj unakrsnoj validaciji (eng. k-fold cross validation). Ova tehnika dijeli skup podataka na k dijelova otprilike jednakih veličina. Treniranje se provodi k puta, svaki ciklus izabere se jedna od k skupina za testiranje, a ostalih k - 1 skupina se koristi za treniranje. Rezultati se prate i spremaju svaki ciklus te se na kraju izračunava prosječna točnost modela.[19]

Bibliografija

[1] C. for BIometrics and S. Research. (2023, Jan.) Casia gait database. , s Interneta, http://www.cbsr.ia.ac.cn/english/GaitDatabases.asp

[2] T. I. o. S. Department of Intelligent Media and O. U. Industrial Research. (2023, Jan.) Ou-isir biometric database. , s Interneta, http://www.am.sanken.osaka-u.ac.jp/BiometricDB/GaitMVLP.html

[3] IBM. (2023, May) Neural networks. , s Interneta, https://www.ibm.com/topics/neural-networks

[4] J. Gu, Z. Wang, J. Kuen, L. Ma, A. Shahroudy, B. Shuai, T. Liu, X. Wang, G. Wang, J. Cai, and T. Chen, "Recent advances in convolutional neural networks," *Pattern Recognition*, vol. 77, pp. 354–377, 2018. , s Interneta, https://www.sciencedirect.com/science/article/pii/S0031320317304120

[5] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015.

[6] P. Varshney. (2020, Feb.). , s Interneta, https://www.kaggle.com/code/blurredmachine/vggnet-16-architecture-a-complete-guide

[7] C. Fan, J. Liang, C. Shen, S. Hou, Y. Huang, and S. Yu, "Opengait: Revisiting gait recognition toward better practicality," 2023.

[8] C. Shen, S. Yu, J. Wang, G. Q. Huang, and L. Wang, "A comprehensive survey on deep gait recognition: Algorithms, datasets and challenges," 2022.

[9] P. W. Code. (2023, Jan.) Casia-b. , s Interneta, https://paperswithcode.com/dataset/casia-b

[10] IBM. (2023, May) Convolutional neural networks. , s Interneta, https://www.ibm.com/topics/convolutional-neural-networks

[11] C.-F. Wang. (2019, Jan.). , s Interneta, https://towardsdatascience.com/the-vanishing-gradient-problem-69bf08b15484

[12] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2015.

[13] Pytorch. (2023, Jun.). , s Interneta, https://pytorch.org/tutorials/beginner/introyt/trainingyt.html

[14] K. E. Koech. (2020, Oct.) Cross-entropy loss function. , s Interneta, https://towardsdatascience.com/cross-entropy-loss-function-f38c4ec8643e

[15] R. Parmar. (2018, Sep.). , s Interneta, https://medium.com/mlearning-ai/optimizers-in-deep-learning-7bf81fed78a0

[16] Medium. (2021, Mar.). , s Interneta, https://medium.com/mlearning-ai/optimizers-in-deep-learning-7bf81fed78a0

[17] IBM. (2023). , s Interneta, https://www.ibm.com/topics/overfitting Pristupljeno 03.2023.

[18] Google. (2023, Jan.) Google colaboratory. , s Interneta, https://colab.research.google.com/

[19] Machinelearningmastery. (2023, Jul.). , s Interneta, https://machinelearningmastery.com/k-fold-cross-validation/

Sažetak

U ovome završnom radu testirane su različite modele i skupove podataka za treniranje neuralne mreže za prepoznavanje osoba na temelju hoda. Korišteni su modeli ResNet-18, ResNet-50 i VGG-16. Skupovi podataka na kojima su modeli trenirani su OUMVLP i CASIA-B. Testirani su i utjecaji različitih optimizacijskih funkcija i ostalih parametara modela poput veličine skupine i broja epoha na konačnu točnost modela. Zaključeno je da dublje rezidualne neuralne mreže poput ResNet postižu točnije rezultate na oba skupa podataka. Primijećen je i prostor za poboljšanja te je navedeno nekoliko metoda koje bi mogle dovesti do bolje točnosti.

Ključne riječi **– prepoznavanje hoda, neuralne mreže, CASIA-B, OUMVLP, ResNet, VGGNet**

Abstract

In this final work, different models and training datasets were tested for training a neural network for gait recognition. The models used were ResNet-18, ResNet-50, and VGG-16. The training datasets used were OUMVLP and CASIA-B. The impacts of different optimization functions and other model parameters such as batch size and number of epochs on the final accuracy of the models were also tested. It was concluded that deeper residual neural networks like ResNet achieve more accurate results on both datasets. Several methods have been identified that could lead to improved accuracy, indicating potential areas for improvement.

Keywords **– gait recognition, neural network, CASIA-B, OUMVLP, ResNet, VGGNet**