Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek

Raspoznavanje uzoraka i strojno učenje

PROJEKTNI ZADATAK

Klasifikacija spola na temelju otiska prstiju

Andreja Gojić

Osijek, 2022.

Sadržaj

1. Uvod

Neuronske mreže su pronašle široku primjenu u raznim područjima, s obzirom da su još uvijek metoda koja se istražuje tek će se utvrditi za rješavanje kojih problema su najpogodnije. Konvolucijske neuronske mreže su se prilikom obrade slike pokazale izrazito uspješnima stoga će se za rješavanje problema klasifikacije spola na temelju otiska prstiju koristiti konvolucijska neuronska mreža.

Cilj ovoga zadatka jest teorijski opisati neuronske mreže, opisati skup podataka koji će se koristiti, istrenirati nekoliko različitih neuronskih mreža te izraditi korisničku aplikaciju za lakše korištenje istreniranih modela.

2. Pregled područja i problematike

Umjetne neuronske mreže dio su umjetne inteligencije, dok je umjetna inteligencija dio računalne znanosti koja se bavi projektiranjem inteligentnih računalnih sustava koji predočavaju karakteristike koje povezujemo s inteligencijom u ljudskom ponašanju. Umjetne neuronske mreže su nastale kao imitacija rada ljudskog mozga koji se sastoji od velikog broja neurona povezanih u mrežu s ciljem obrade velikog broja različitih informacija.

Umjetna neuronska mreža sastoji se od umjetnih neurona baš kao i biološka neuronska mreža. Po uzoru na biološke neurone i umjetni neuroni su građeni od dendrita, tijela neurona i aksona, a neuroni su međusobno povezani sinapsama. Umjetni neuroni primaju ulazne podatke putem dendrita od prethodnog sloja neurona, zatim se u tijelu neurona odvija računanje te odluka prosljeđuju li se izlazni podatci sljedećem sloju ili ne. Računanje koje se odvija u tijelu neurona jest funkcija, ulazne podatke možemo označiti s **X**, a izlazne s **Y**, tada bi funkcija izgledala **Y** = *f*(**X**; **W**), gdje su **W** parametri funkcije neurona, tzv. težine. Težine su kao i ulazni podatci realni brojevi, a njihova vrijednost se ugađa treniranjem mreže.

Sama neuronska mreža sastoji se od više neurona koji su posloženi u slojeve. Osnovna neuronska mreža se sastoji od tri sloja neurona, to su ulazni, skriveni i izlazni sloj. Ulazni sloj kako mu i samo ime kaže predstavlja prvi sloj u koji ulaze ulazni podatci iz sustava, odnosno podatci koji predstavljaju određeni problem. Izlazni sloj je posljednji sloj neuronske mreže i podatci koji izlaze iz njega predstavljaju rješenje problema. Svi slojevi koji se nalaze između ulaznog i izlaznog sloja nazivaju se skriveni slojevi te se u njima odvija računanje koje dovodi do rješenja.

Kada se govori o umjetnim neuronskim mrežama uglavnom se misli na unaprijedne slojevite mreže, kod koje su izlazi iz jednog sloja neurona spojeni na ulaze sljedećeg sloja neurona, ali postoje i povratne mreže kod kojih se izlazi iz jednog sloja mogu spojiti na ulaze prethodnog sloja. Neuronske mreže mogu biti potpuno povezane i nepotpuno povezane. Potpuno povezane neuronske mreže su one kod kojih je izlaz iz svakog neurona spojen na ulaze svih neurona sljedećeg sloja, dok kod nepotpuno povezanih mreža izlazi neurona ne moraju biti povezani sa svim ulazima sljedećeg sloja neurona.

Postupak učenja umjetnih neuronskih mreža je iterativni postupak podešavanja vrijednosti težinskih faktora. Učenje se odvija prema nekoj od brojnih metoda učenja kao što su metoda gradijentnog spusta ili algoritam propagacije unatrag. Prilikom jednog prolaza informacije kroz

neuronsku mrežu generira se vrijednost koja se nakon toga uspoređuje sa stvarnom vrijednošću izlaza te se na temelju razlike vrijednosti podešavaju težinski faktori.

Uz pomoć umjetnih neuronskih mreža moguće je razvrstati ulazne u podatke u klase (klasifikacija), moguće je odrediti predviđanje razvoja određenih varijabli s obzirom na ulazne podatke (regresija). Prepoznavanje govora, objekata na slikama i sl. primjeri su klasifikacije, dok su procjena potrošnje automobila, cijena stanova i sl. primjeri regresije.

2.1. Korišteni postupci strojnog učenja

Za rješavanje klasifikacije spola na temelju otiska prstiju korištena je konvolucijska neuronska mreža. Konvolucijska neuronska mreža je nadogradnja na standardnu višeslojnu potpuno povezanu neuronsku mrežu, ona se također sastoji od ulaznog, skrivenog i izlaznog sloja, a specifična je po konvolucijskom sloju i sloju sažimanja. Arhitektura ovakve mreže započinje s jednim ili više konvolucijskih slojeva, zatim slijedi sloj sažimanja te se postupak ponavlja ovisno o složenosti ulaza i rezultatima koje postiže mreža. Ovakva arhitektura neuronske mreže se pokazala kao izrazito dobra u radu sa slika te rješavanje problema prepoznavanja objekata na slikama.

Konvolucijski sloj se sastoji od definiranog broja filtara s težinama koje je potrebno naučiti kako bi mreža davala dobre rezultate. Filtri su matrice manjih prostornih dimenzija od ulaza. Rezultat konvolucije je aktivacijska mapa koja sadrži odzive filtera na svakoj prostornoj poziciji. Mreža će naučiti težine unutar filtra kako bi se filtar aktivirao na mjestima gdje prepoznaje određena svojstva slike, poput prepoznavanja rubova.

Nakon konvulacijskog sloja slijedi sloj sažimanja. U sloju sažimanja kako mu i samo ime kaže slika se sažima, tj. smanjuje joj se veličina, ali osim toga povećava se prostorna neosjetljivost neuronske mreže na manje pomake značajki u uzorcima. Najčešće korišteni načini sažimanja su sažimanje usrednjavanjem i sažimanje maksimalnom vrijednošću. Sažimanje usrednjavanjem podrazumijeva se zamjenom grupiranih vrijednosti njihovom srednjem vrijednošću, dok se kod sažimanja maksimalnom vrijednošću grupirani podatci zamijene najvećom vrijednošću tih podataka.

2.2. Korištene konvolucijske neuronske mreže

Prilikom izrade programskog rješenja korištene su tri različite proizvoljne konvolucijske mreže, AlexNet, ResNet50 i VGG16, njihova upotreba bit će opisana u sljedećem poglavlju, a ovdje će se opisati njihova arhitektura.

2.2.1. AlexNet

AlexNet je konvolucijska neuronska mreža koji je građena od osam slojeva, pet konvolucijskih slojeva i tri fully connected sloja. Prvi sloj je konvolucijski sloj s 96 filtera veličine 11x11, zatim slijedi pooling sloj. Nakon pooling sloja nalazi se drugi konvolucijski sloj s 96 filtera veličine 5x5, potom slijedi još jedan pooling sloj, a nakon njega treći, četvrti i peti konvolucijski slojevi koji su uzastopno poredani, a nakon njih se nalazi pooling sloj. Treći i četvrti konvolucijski slojevi su identični, a sastoje se od 384 filtera veličine 3x3, dok se posljednji, peti, konvolucijski sloj sastoji od 256 filtera veličine 3x3. Svi konvolucijski slojevi koriste ReLU aktivacijsku funkciju. Na kraju ahritekture se nalaze tri fully connected sloja, prvi je veličine 9216, dok su druga dva veličine 4096. Tablica 1 prikazuje AlexNet arhitekturu.

Layer	Feature map	Kernel size
Conv 1	96	11x11
Pool 1	96	3x3
Conv 2	256	5x5
Pool 2	256	3x3
Conv 3	384	3x3
Conv 4	384	3x3
Conv 5	256	3x3
Pool 3	256	3x3
FC	-	9216
FC	-	4096
FC	-	4096

Tablica 1, *AlexNet arhitektura*

2.2.2. ResNet50

ResNet arhitektura je prva uvela takozvano preskakanje slojeva koje podrazumijeva da se na izlaz određenog bloka doda njegov ulaz, to jest ulaz nekoga bloka je izravno spojen na izlaz toga bloka. Ovakav pristup ubrzava duboku mrežnu konvergenciju jer će se slojevi s lošijim performansama regularizacijom preskočiti te tako smanjiti probleme s gradijentom poput nestajanja ili eksplodiranja gradijenta. Arhitektura ResNet neuronske mreže, prikazana u tablici 2, može se podijeliti u pet etapa:

- Prva etapa: sadrži jedan konvolucijski sloj od 64 filtera
- Druga etapa: sadrži tri uzastopna konvolucijska bloka, a svaki blok se sastoji od tri konvolucijska sloja s 64, 64 i 128 filtera
- Treća etapa: sadrži 4 konvolucijskih blokova koji se sastoje od tri konvolucijska sloja s 128, 128 i 256 filtera
- Četvrta etapa: sadrži 6 konvolucijskih blokova s tri konvolucijska sloja koji imaju 256, 256 i 1024 filtera
- Peta etapa: sadrži tri konvolucijska bloka koji imaju tri konvolucijska sloja s 512, 512 i 2048 filtera

Layer	Feature map	Kernel size
Conv 1	64	7x7
3 x Conv 2	64 64 256	1x1 3x3 1x1
4 x Conv 3	128 128 512	1x1 3x3 1x1
6 x Conv 4	256 256 1024	1x1 3x3 1x1
3 x Conv 5	512 512 2048	1x1 3x3 1x1
FC	-	1000

Tablica 2, *ResNet50 arhitektura*

2.2.3. VGG16

Arhitektura VGG16 konvolucijske mreže se sastoji od pet konvolucijskih blokova i jednog fully connected bloka, prikazana je u tablici 3. VGG16 arhitektura se može podijeliti u šest etapa:

- Prva etapa: sadrži dva konvolucijska sloja od 64 filtera koji su veličine 3x3 te pooling sloj
- Druga etapa: sadrži dva konvolucijska s 128 filtera koji su veličine 3x3 i pooling sloj

- Treća etapa: sastoji se od tri konvolucijska sloja s 256 filtera koji su veličine 3x3 nakon kojih slijedi pooling sloj
- Četvrta etapa: sadrži tri konvolucijska sloja s 512 filtera koji su veličine 3x3 i pooling sloj
- Peta etapa: isto kao i četvrta etapa sadrži tri konvolucijska sloja s 512 filtera koji su veličine 3x3 i pooling sloj
- Šesta etapa: sadrži tri fully connected sloja, prvi je veličine 25088, dok su posljednja dva veličine 4096

Layer	Feature map	Kernel size
2 x Conv 1	64	3x3
Pool 1	64	3x3
2 x Conv 2	128	3x3
Pool 2	128	3x3
2 x Conv 3	256	3x3
Pool 3	256	3x3
3 x Conv 4	512	3x3
Pool 4	512	3x3
3 x Conv 5	512	3x3
Pool 5	512	3x3
FC	-	25088
FC	-	4096
FC	-	4096

Tablica 3, *VGG16 arhitektura*

3. Opis zadatka

3.1. Korišteni podatci

Podatci koji su korišteni za izradu ovoga zadatka preuzeti su s Kaaggle-a. te sadrže oko 55000 slika ljudskih otisaka prstiju. Podatci su podijeljeni u dva direktorija, Real i Altered, a Altered direktorij je podijeljen na tri direktorija, Altered-Easy, Altered-Medium i Altered-Hard. U direktoriju Real se nalaze stvarne slika otisaka prstiju dok su u Altered direktorijima nalaze sintetički izmijenjene verzije otisaka prstiju, izmjene su izrađene u tri razine, jednostavna izmjena, srednja izmjena i teška izmjena. Na slikama 1, 2, 3 i 4 se nalaze primjeri slika iz korištenog skupa podataka.

Slika 1, *Prikaz slika iz Altered-Easy direktorija*

Slika 2, *Prikaz slika iz Altered-Medium direktorija*

Slika 3, *Prikaz slika iz Altered-Hard direktorija*

Slika 4, *Prikaz slika iz Real direktorija*

Nazivi slika su zapisani u sljedećim formatima, "100__M_Left_thumb_finger_CR.BMP" za slike iz Altered direktorija te "100__M_Left_thumb_finger.BMP" za slike iz Real direktorija. S obzirom da je zadatak klasifikacija prema spolu iz naziva slika je potrebna samo oznaka za spol te je izrađena funkcija `extract_label()` koja vraća oznaku o spolovima, programski kod 1 prikazuje ovo funckiju.

```
def extract_label(img_path,train = True):
  filename, _ = os.path.splitext(os.path.basename(img_path))

  subject_id, etc = filename.split('__')
#For Altered folder
  if train:
      gender, lr, finger, _, _ = etc.split('_')
#For Real folder
  else:
      gender, lr, finger, _ = etc.split('_')

  gender = 0 if gender == 'M' else 1

  return np.array([gender], dtype=np.uint16)
```

Programski kod 1, *Funkcija extract_label*

Ova funkcija prima put do slike te boolean vrijednost koja govori koriste li se podatci za treniranje ili testiranje. U funkciji se prvo iz naziva slike makne ekstenzija (pr. 100__M_Left_thumb_finger), zatim se izdvoji dio naziva u kojem se nalazi spol i naziv prsta (pr. M_Left_thumb_finger), a zatim se izdvoji samo spol (pr. M). Kada se odvoji oznaka za spol iz naziva slike spol se označava s 0 za muški ili 1 za ženski te tu vrijednost funkcija vraća.

Zatim je izrađena funkcija `load_data()` koja će učitavati slike iz direktorija, a njena definicija je prikazana u programskom kodu 2.

```
img_size = 96

def loading_data(path,train):
    print("loading data from: ",path)
    data = []
```

```
    for img in os.listdir(path):
        try:
            img_array = cv2.imread(os.path.join(path, img),
cv2.IMREAD_GRAYSCALE)
            img_resize = cv2.resize(img_array, (img_size, img_size))
            label = extract_label(os.path.join(path, img),train)
            data.append([label[0], img_resize ])
        except Exception as e:
            pass

    return data
```

Programski kod 2, *Funkcija loading_data*

Slično kao i prethodnoj funkciji, ovoj se funkciji predaje put do direktor te boolean vrijednost koriste li se slike za treniranje ili za testiranje. Potom se učitava jednokanalna slika u grayscale obliku kojoj se promijeni veličina na 96x96 piksela te se poziva prethodna funkcija `extract_label` iz koje dobivamo vrijednost koja nam govori spol osobe čiji se otisak prsti nalazi na slici, zatim se ta vrijednost sprema u niz `data` i nastavlja se iterativno učitavanje slike iz predanog direktorija sve dok se ne učitaju sve slike.

Nakon učitavanja bilo je potrebno prilagoditi slike za samo treniranje, to se postiglo tako što su se vrijednosti piksela svih učitanih slika podijelili s 255.0 kako bi se ostvario raspon piksela između 0 i 1. Za treniranje neuronske mreže korišteni su podatci iz Altered-Easy, Altered-Medium i Altered-Hard direktorija kako bi se model naučio na teže podatke što će u konačnici rezultirati uspješnijim prepoznavanje stvarnih uzoraka, dok su za testiranje korišteni podatci iz Real direktorija.

3.2. Izrada modela

Kao što je već ranije spomenuto, korištene su tri proizvoljne konvolucijske neuronske mreže, AlexNet, ResNet50 i VGG16. Za razliku od ResNet50 i VGG16, AlexNet nije ugrađena u biblioteku Keras te je bilo potrebno izraditi njen model.

3.2.1. AlexNet

Model AlexNet je izrađen na način da se slijedila ranije opisana arhitektura uz manje izmjene koje su sadržavale dodavanje `BatchNormalization()` sloja koji služi za normalizaciju skrivenih slojeva te na taj način mreža može koristiti veću stopu učenja bez nestajanja ili eksplodiranja gradijenta. Osim navedene izmjene dodan je ulazni sloj koji definira veličinu

ulaznih slika te izlazni sloj koji definira veličinu izlaza. Veličina ulaza je prilagođena dimenzijama slika koje su (96, 96, 1), dok je veličina izlaza vektor od dva elementa od kojih svaki element predstavlja jedan spol, također na izlaznom sloju je dodana aktivacijska funkcija sigmoid koja je najpogodnija za rješavanje ovoga problema jer će vrijednost jednog elementa izlaza uvijek biti približno 1, dok će vrijednost drugog elementa uvijek biti približno 0. Programski kod 3 prikazuje izrađeni model.

```
alex = Sequential()


alex.add(Conv2D(filters=96,kernel_size=(11,11),input_shape=(96,96,1)
))
alex.add(BatchNormalization())
alex.add(Activation('relu'))
alex.add(MaxPooling2D(pool_size=(2,2),strides=(2,2)))


alex.add(Conv2D(filters=256, kernel_size=(5,5), strides=(1,1)))
alex.add(BatchNormalization())
alex.add(Activation('relu'))
alex.add(MaxPooling2D(pool_size=(2,2),strides=(2,2)))


alex.add(Conv2D(filters=384, kernel_size=(5,5), strides=(1,1)))
alex.add(BatchNormalization())
alex.add(Activation('relu'))
alex.add(Conv2D(filters=384, kernel_size=(5,5), strides=(1,1)))
alex.add(BatchNormalization())
alex.add(Activation('relu'))
alex.add(Conv2D(filters=256, kernel_size=(5,5), strides=(1,1)))
alex.add(BatchNormalization())
alex.add(Activation('relu'))
```

```
alex.add(MaxPooling2D(pool_size=(2,2),strides=(2,2)))


alex.add(Flatten())


alex.add(Dense(9216, activation = 'relu'))
alex.add(Dense(4096, activation = 'relu'))
alex.add(Dense(4096, activation = 'relu'))
alex.add(Dense(2, activation = 'sigmoid'))
```

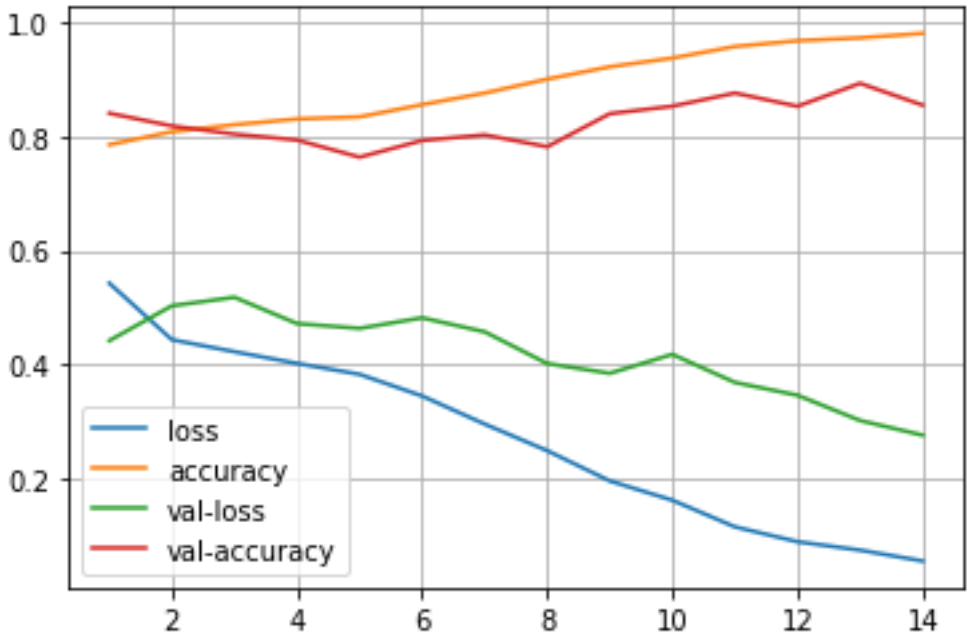
Programski kod 3, *Izrada AlexNet modela*

Kada je model definiran, može se početi treniranje modela, treniranje je izvršeno pozivom funckije `fit()` kojoj su kao ulazni parametre predani skup podataka za treniranje, skup očekivanih izlaza, batch size, broj epoha te postotak koji će se koristiti za validaciju. Batch size je vrijednost koja određuje koji će broj slika propagirati kroz mrežu u jednoj iteraciji epohe. Uz manji batch size bit će precizniji rezultati treniranja, ali će treniranje duže trajati, dok uz veći batch size će se skratit treniranje, ali će rezultati treniranja biti lošiji, prilikom ovoga treniranja odabran je batch size od 128. Broj epoha predstavlja broj prolazaka kroj cijeli skup podataka za treniranje, a u ovome slučaju je odabrano 15 prolazaka. Poziv funkcije fit prikazan je programskim kodom 4.

```
history = alex.fit(train_data, train_labels, batch_size = 128,
epochs = 15, validation_split = 0.2)
```

Programski kod 4, *Poziv funckije fit()*

Slika 5 prikazuje rezultate treniranja.

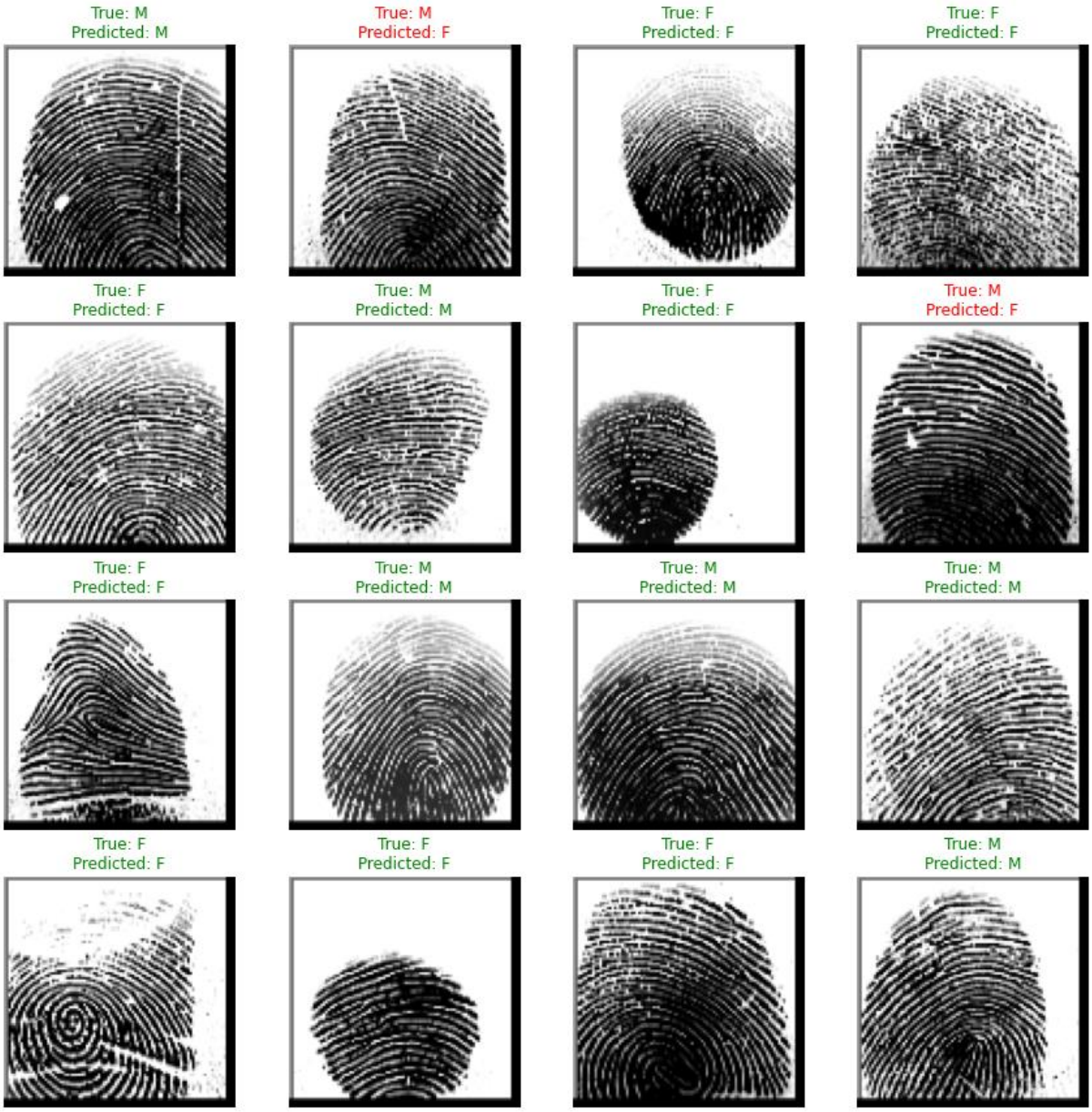


Slika 5, *Rezultati treniranja AlexNet modela*

Nakon treniranja model je spremljen te je izvršeno testiranje na skupu podataka koje model još nije vidio, kako je ranije spomenuto, radi se o podatci iz Real direktorija, rezultati testiranja su prikazani na slici 6. Na istim podatci je napravljena predikcija, a manji broj uzoraka predikcije je prikazan grafički prikazan (Slika 7).

```
Test Loss: 0.2573
Test Accuracy: 92.45%
```

Slika 6, *Rezultati testiranja AlexNet modela*



Slika 7, *Rezultati predikcije AlexNet modela*

3.2.2. ResNet50

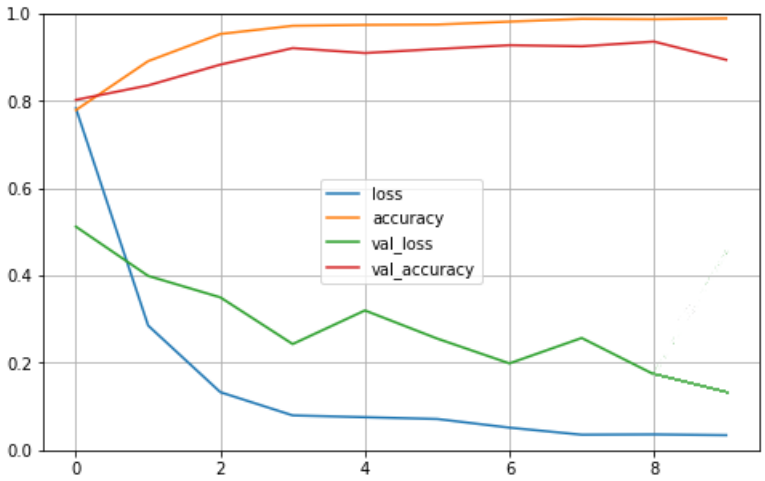
Za izradu ResNet50 modela korištena je gotova arhitektura modela koja se nalazi u Keras biblioteci. Arhitektura je učitana pozivom funckije `ResNet50()` kojoj su predani parametri da se ne koristi predefinirani ulazni sloj, zatim da se radi klasifikacije s dvije klase, da se ne koristi već istreniran model te veličina ulaznih podataka. Nakon pozivanja funkcije `ResNet50()` modelu je dodan sloj koji će matrica pretvoriti u vektor te izlazni sloj kao i kod prošlog modela što je prikazano programskim kodom 5.

```
resnet = Sequential()
```

```
pretrained = ResNet50(include_top = False, classes = 2, weights =
None, input_shape=[96, 96, 1])


resnet.add(pretrained)


resnet.add(Flatten())

resnet.add(Dense(2, activation = 'sigmoid'))
```

Programski kod 5, *Izrada ResNet50 modela*

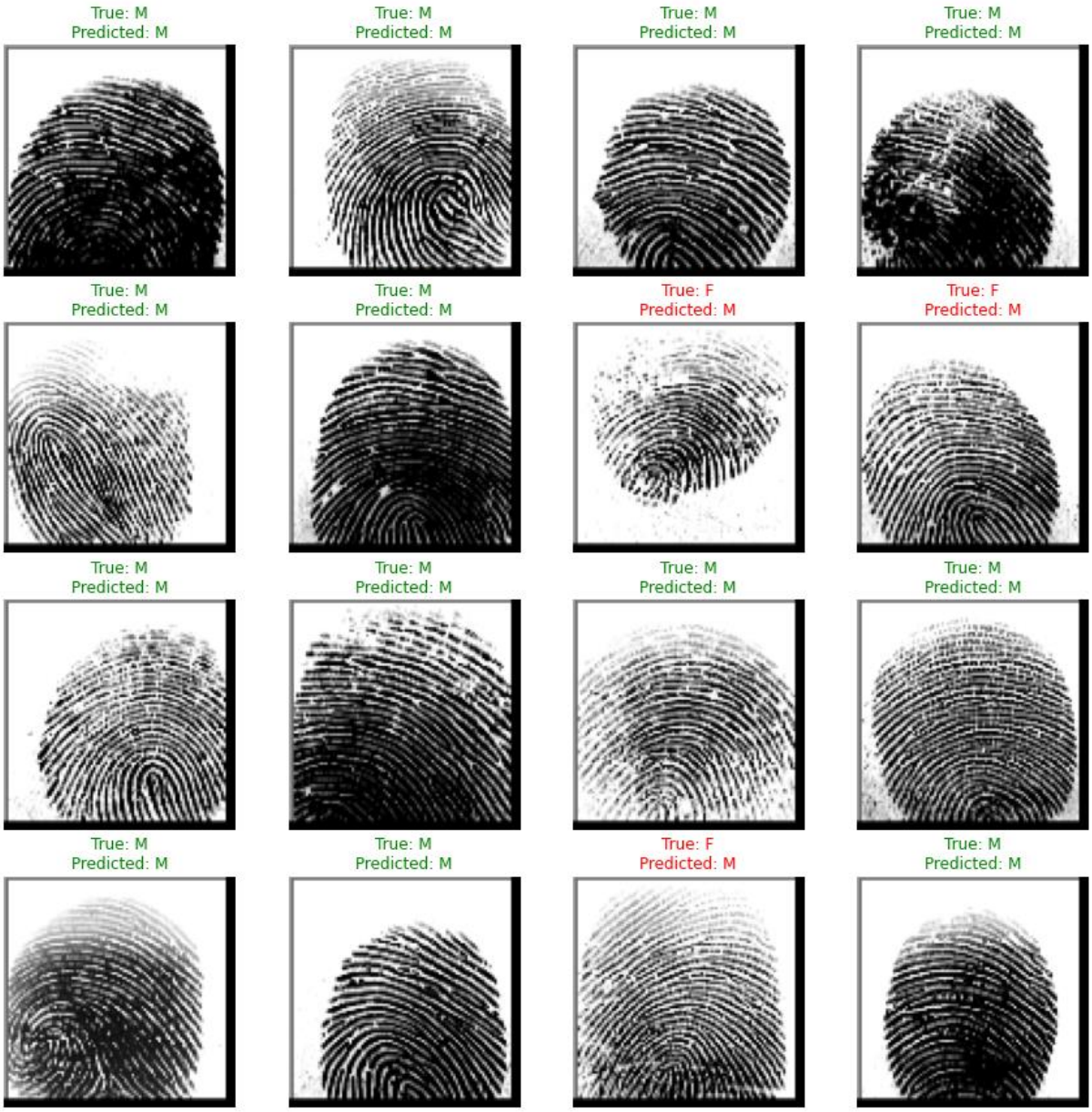
Treniranje, testiranje i predikcija su izvršeni na isti način kao i kod AlexNet modela samo što je prilikom treniranja odabran manji broj epoha zbog veće složenosti modela i duljeg trajanja. Nakon treniranja model je spremljen, a rezultati treniranja, testiranja i predikcije su prikazani na slikama 8, 9 i 10.



Slika 8, *Rezultatit treniranja ResNet50 modela*

```
    Test Loss: 0.43063
Test Accuracy: 89.85%
```

Slika 9, *Rezultati testiranja ResNet50 modela*



Slika 10, *Rezultati predikcije ResNet50 modela*

3.2.3. VGG16

VGG16 model je, kao i ResNet50, izrađen pomoću gotove arhitekture iz Keras biblioteke koja se poziva funkcijom VGG16. Na ovome modelu je odlučeno koristiti ranije istreniranu mrežu, s obzirom na to da se nije pronašla istrenirana mreže na ovome skupu podataka niti na grayscale slikama, korištena je istrenirana mreža na ImageNet skupu podataka. ImageNet je veliki skup podataka koji sadrži preko 14 miliona različitih slika u više od 20 tisuća različitih kategorija. Slike koje se nalaze u ImageNet-u su RGB slike, što znači da i slike koje se predaju mreži treniranoj na ImageNet skupu podataka također moraju biti RGB. Prema tome su sve slike za potrebe ovoga modela učitane kao trokanalne slike, a zbog većeg zauzimanja memorije

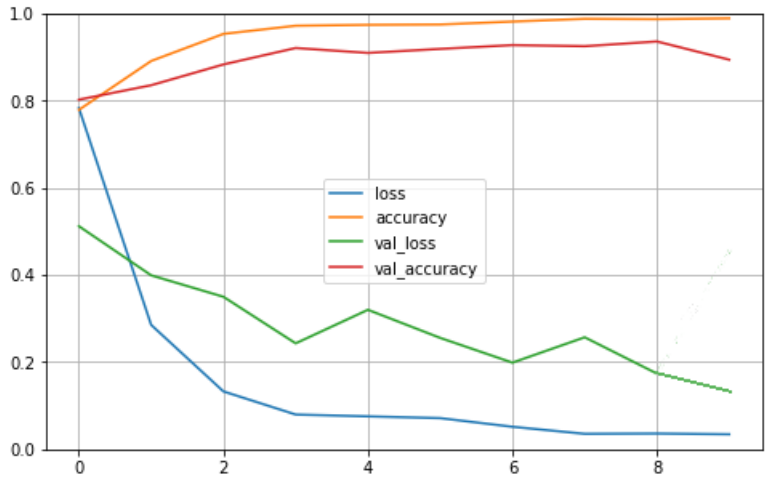
prilikom treniranja nisu korištene slike iz Altered-Medium direktorija. Nakon učitavanja odrađena je ista predobrada kao kod prijašnjih modela, samo što se ovoga puta veličina slika promijenila na dimenzija (96, 96, 3), a ne (96, 96, 1) kao što je bio slučaj kod AlexNet i ResNet50 modela.

Pozivom VGG16 funkcije predani su parametri za korištenje težina za istreniranu mrežu, za ne korištenje ulaznog sloja te dimenzije ulaznog sloja koji će se koristiti. Potom je na istrenirani model dodan sloj za pretvaranje matrice u vektor te izlazni sloj koji je isti kao i kod prethodnih modela. Izrada modela prikaza na je programskim kodom 6.

```
base_model = VGG16(weights='imagenet', include_top=False,
input_shape = [96, 96, 3])


vgg = Sequential()

vgg.add(base_model)

vgg.add(Flatten())

vgg.add(Dense(2, activation = 'sigmoid'))
```

Programski kod 6, *Izrada VGG16 modela*

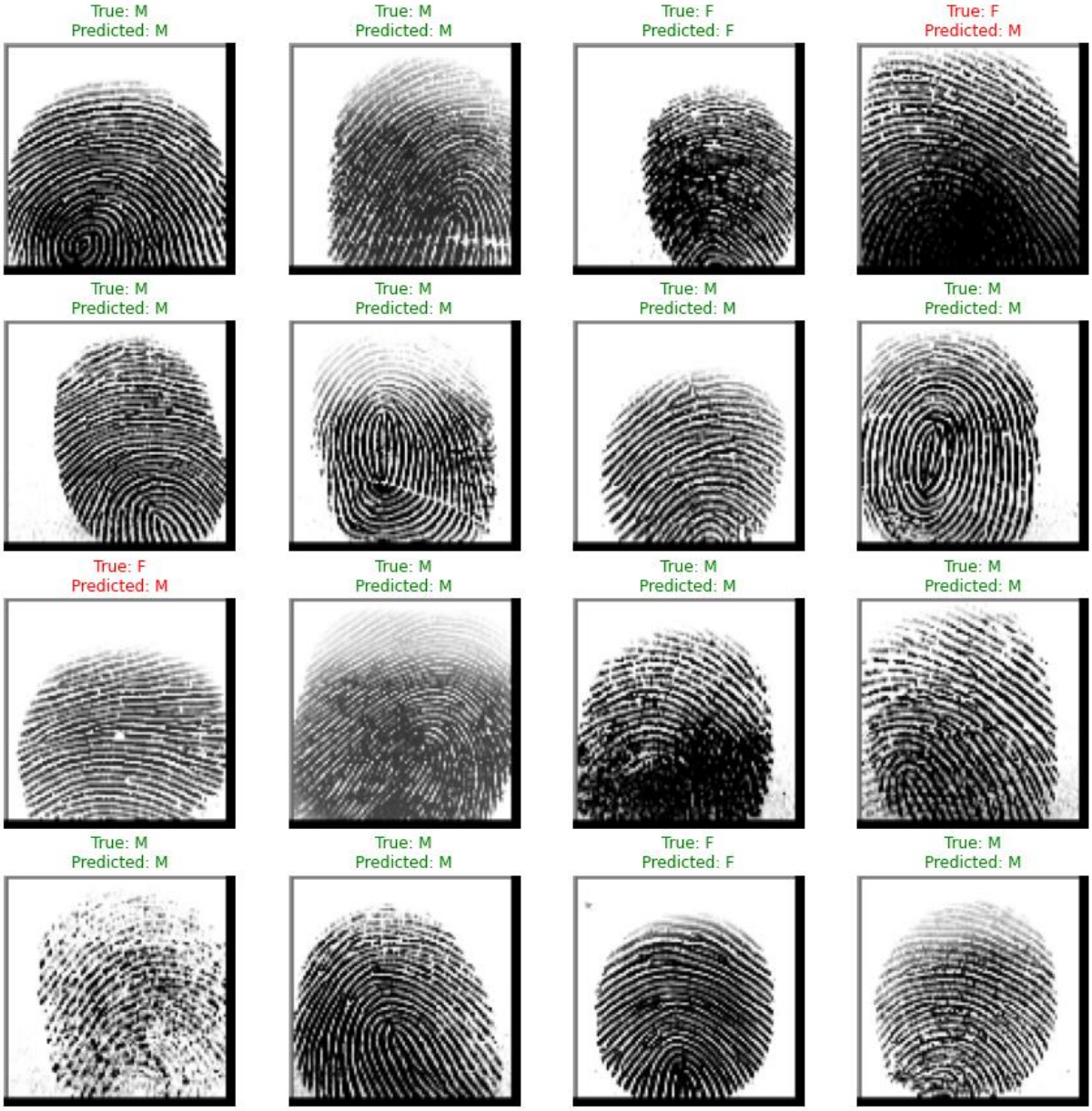
Treniranje, testiranje i predikcija su izvršeni na isti način kao i kod AlexNet modela. Nakon treniranja model je spremljen, a rezultati treniranja, testiranja i predikcije su prikazani na slikama 11, 12 i 13.



Slika 11, *Rezultati treniranja VGG16 modela*

```
    Test Loss: 0.50765
Test Accuracy: 79.50%
```

Slika 12, *Rezultati testiranja VGG16 modela*



Slika 13, *Rezultati predikcije VGG16 modela*

3.3. Izrada aplikacije

Ovako istrenirani modeli nisu pristupačni za korištenje široj populaciji stoga je izrađena web aplikacija koja će to olakšati. Web aplikacija je izrađena u skripti predict.py pomoću Python-ove Streamlit biblioteke koja omogućuje jednostavnu izradu web aplikacija. U aplikaciji je moguće učitati sliku otiska prsta, odabrati željeni model za predikciju te klikom gumba „Predikcija“ izvršiti predikciju nakon čega će se ispisati rezultat predikcije.

Odabir modela za predikciju je moguće izvršiti pomoću tri radio gumba, zatim se lokalno spremljena slika učitava u okvir predviđen za to, a klikom gumba „Predikcija“, ukoliko je učitana slika, poziva se funkcija `importAndPredict()` koja za ulazne parametre prima učitanu sliku i odabrani model. Učitane slike su u RGBA formatu pa ih ova funkcija mijenja u grayscale, ukoliko se radi o AlexNet ili ResNet50 modelima, te u RGB ukoliko se radi o VGG16 modelu, zatim mijenja veličinu slike te vrijednosti svih piksela pretvara iz raspona 0-255 u raspon 0-1, potom učitava odabrani model, vrši predikciju te vraća vektor koji je rezultat predikcije. Programski kod funkcije `importAndPredict()` prikazan je programski kodom 7.

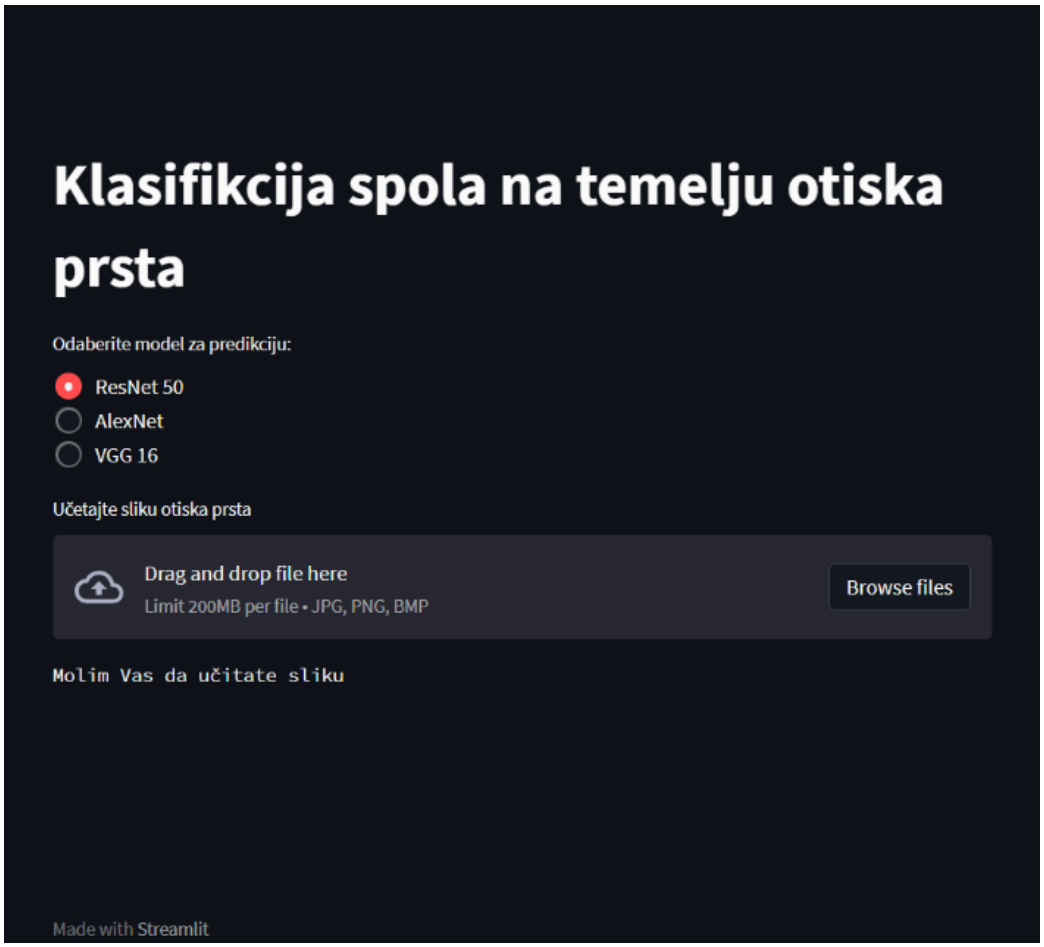
```
def importAndPredict(image_data, choosed_model):

    if choosed_model == 'ResNet 50':

        model = load_model('resnet50_model/')

        img = image_data.resize((96, 96))

        img = img.convert('RGB')

        img = ImageOps.grayscale(img)

        img = np.array(img).reshape(-1, 96, 96, 1)

        img= img/255.0

        pred = model.predict(img)

    elif choosed_model == 'VGG 16':

        model = load_model('vgg_model/')

        img = image_data.resize((96, 96))

        img = img.convert('RGB')

        img = np.array(img).reshape(-1, 96, 96, 3)

        img= img/255.0

        pred = model.predict(img)

    elif choosed_model=='AlexNet':

        model = load_model('alexnet_model/')

        img = image_data.resize((96, 96))
```

```
        img = img.convert('RGB')
        img = ImageOps.grayscale(img)
        img = np.array(img).reshape(-1, 96, 96, 1)
        img= img/255.0
        pred = model.predict(img)
    return pred
```

Programski kod 7, *Funkcija importAndPredict()*

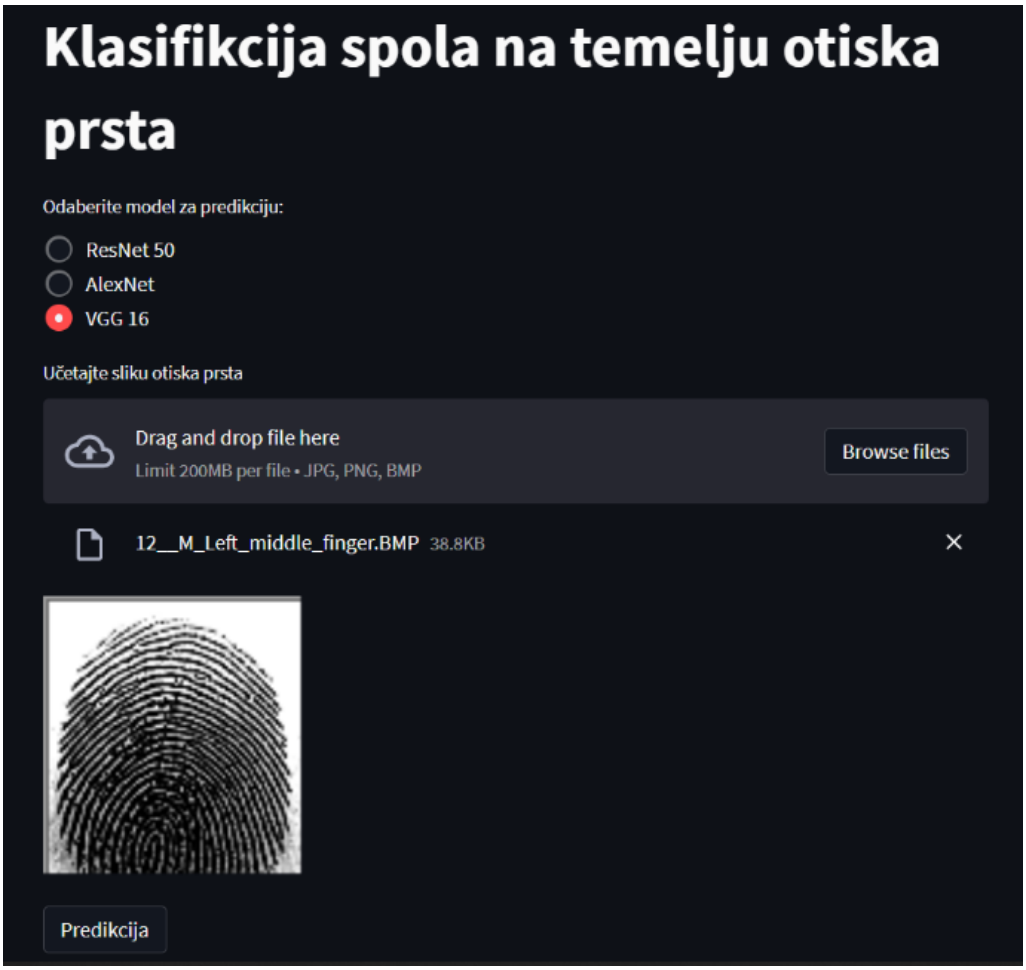
4. Dobiveni rezultati

Izgled početnog zaslona izrađene aplikacija nalazi se na slici 14.



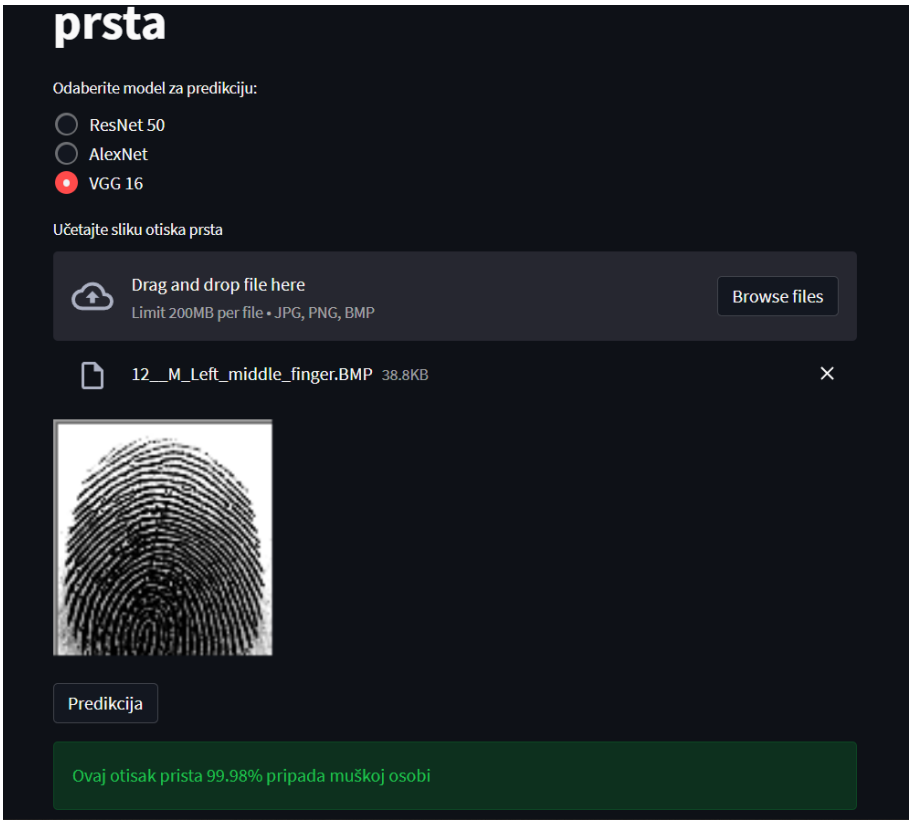
Slika 14, *Prikaz početnog zaslona izrađene aplikacije*

Nakon učitavanja željene slike, prikazuje se učitana slika te se pojavljuje gumb predikcija, to je prikazano na slici 15.

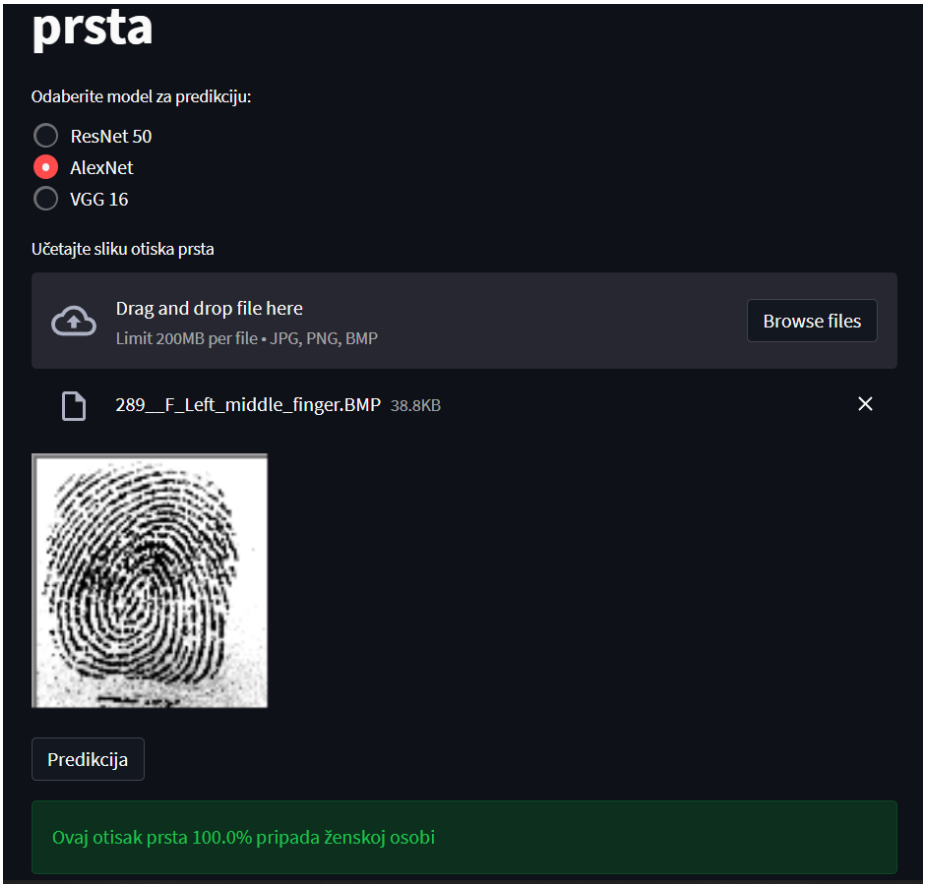


Slika 15, *Izgled aplikacija nakon učitavanja slike*

Klikom na gumb „Predikcija“ na dnu zaslona se ispisuje rezultat predikcije. Na slici 16 se nalazi prikaz rezultata ukoliko se radi o muškoj osobi, dok se na slici 17 nalazi prikaz rezultata za žensku osobu.



Slika 16, *Prikaz rezultata predikcije za mušku osobu*



Slika 17, *Prikaz rezultata predikcije za žensku osobu*

5. Zaključak

Za rješavanje ovog projektnog zadatka korištene su tri različite konvolucijske mreže te je izrađena web aplikacija za jednostavniju uporabu istreniranih mreža. Modeli su trenirani, testira te spremljeni kako bi se kasnije mogli učitati u izrađenoj web aplikaciji. Iz rezultata testiranje mreža je vidljivo kako ResNet50 i VGG16 imaju prilično velike gubitke, razlog tomu je možda što su modeli previše naučili na izmijenjene podatke koji su korišteni za trening pa kada su se susreli s realističnim testnim podacima nisu uspješno izvršili klasifikaciju. Taj problem bi se mogao riješiti na način da se realistični otisci prstiju podijele na trening i test podatke, odnosno da modeli uče na realističnim podatci.

6. Literatura

[1] https://www.analyticsvidhya.com/blog/2021/03/introduction-to-the-architecture-of-alexnet/

[2] https://www.mygreatlearning.com/blog/introduction-to-vgg16/#VGG%2016%20Architecture