SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 201

MOBILNA APLIKACIJA ZA OČITAVANJE RUKOM PISANOG TEKSTA KORIŠTENJEM DUBOKIH NEURONSKIH MREŽA

Stjepan Ruklić

Zagreb, lipanj 2021.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 201

MOBILNA APLIKACIJA ZA OČITAVANJE RUKOM PISANOG TEKSTA KORIŠTENJEM DUBOKIH NEURONSKIH MREŽA

Stjepan Ruklić

Zagreb, lipanj 2021.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 12. ožujka 2021.

ZAVRŠNI ZADATAK br. 201

Pristupnik:	**Stjepan Ruklić (0036516410)**
Studij:	Elektrotehnika i informacijska tehnologija i Računarstvo
Modul:	Računarstvo
Mentor:	izv. prof. dr. sc. Alan Jović

Zadatak: **Mobilna aplikacija za očitavanje rukom pisanog teksta korištenjem dubokih neuronskih mreža**

Opis zadatka:

Cilj završnog rada je izrada mobilne aplikacije za očitavanje rukom pisanog teksta iz slika. Aplikacija treba koristiti izgrađeni model za detekciju znakova abecede dobiven pomoću dubokih neuronskih mreža. Aplikacija treba omogućiti fotografiranje teksta, poboljšavanje kvalitete očitane slike i tekstni prikaz prepoznatog teksta u slici koristeći odgovarajuće programsko sučelju prema jednom ili više izgrađenih modela. Prethodno izgrađeni modeli u okviru ovog završnog rada mogu biti dostupni na strani klijenta ili na poslužitelju, a potrebno je omogućiti izvoz detektiranog teksta. Modeli se mogu učiti na vlastitom prikupljenom skupu podataka ili na slobodno dostupnim podatcima s weba, kao što je skup podataka: https://www.kaggle.com/landlord/handwriting-recognition. Implementaciju je potrebno napraviti u programskom jeziku po vlastitom izboru.

Rok za predaju rada: 11. lipnja 2021.

SADRŽAJ

1. Uvod

Danas se optičko prepoznavanje znakova (engl. *Optical Character Recognition*, kraće: OCR) često koristi kao oblik unosa podataka iz raznih dokumenata. Pretvorbom dokumenata u digitalni format, moguće ih je lakše analizirati, pretraživati i uređivati.

U ovom završnom radu prikazati ću svoje rješenje izrade aplikacije za očitavanje rukom pisanog teksta korištenjem duboke neuronske mreže. Širok spektar primjene ovakve aplikacije zahtijeva i njen dodatan razvoj. Ovakva aplikacija se može primijeniti za digitalizaciju starih dokumenata, knjiga ili pisama.

Ovaj rad je strukturiran u pet poglavlja, u kojem nakon završetka obrade teme slijedi zaključak, literatura, popis slika, popis tablica te naslov, sažetak i ključne riječi.

U drugom poglavlju opisuje se tehnologija OCR i mogućnosti njene rimjene.

U trećem poglavlju, naziva Duboke neuronske mreže proučavaju se duboke neuronske mreže i postupak učenja dubokih neuronskih mreža.

U četvrtom poglavlju, naziva Model duboke neuronske mreže opisan je način kojim sam od baze podataka došao do modela koji je sposoban za provođenje postupka optičkog prepoznavanja znakova.

U petom, ujedno i posljednjem poglavlju naziva Aplikacija je opisan potpuni MVC model sustava. Ujedno su i opisane tehnologije i alati kojima je isti izgrađen.

Baza algoritma za izradu modela duboke neuronske mreže se temelji na [14], funkcionalnost tog algoritma proširena je mogućnosti prepoznavanja literala i nekih posebnih znakova.

2. Optičko prepoznavanje znakova

OCR je disciplina koja spada u područje istraživanja računalnog vida koji zahtijeva pretvorbu rukom pisanog teksta ili isprintanog teksta u format teksta koji računala razumiju. Tako spremljen tekst na računalu je lakše uređivati, doraditi i pretraživati.[1]

2.1. Uvod u OCR

Optičko prepoznavanje znakova pripada u skupinu tehnika automatskog identificiranja.[2]

2.1.1. Automatsko identificiranje

Klasičan način unošenja podataka u računalo je putem tipkovnice. Ponekad to nije najbolji niti najefikasniji način. U mnogim slučajevima automatsko očitavanje podataka je bolja solucija. U nastavku ću prikazati nekoliko različitih tehnologija automatskog identificiranja i njihovu primjenu.[2]

Prepoznavanje govora

U sustavima za prepoznavanje govora, prepoznaje se izgovoreni zvuk iz predefiniranog skupa naredbi. Takvi sustavi se mogu koristiti za identifikaciju ili za izdavanje glasovnih naredbi.[2]

Sustav računalnog vida

Korištenjem kamere moguće je prepoznati i identificirati objekte prema njihovoj veličini i obliku. Takvi modeli sustava se mogu primijeniti na automatiziranim naplatnim postajama koje mogu klasificirati vrstu vozila.[2]

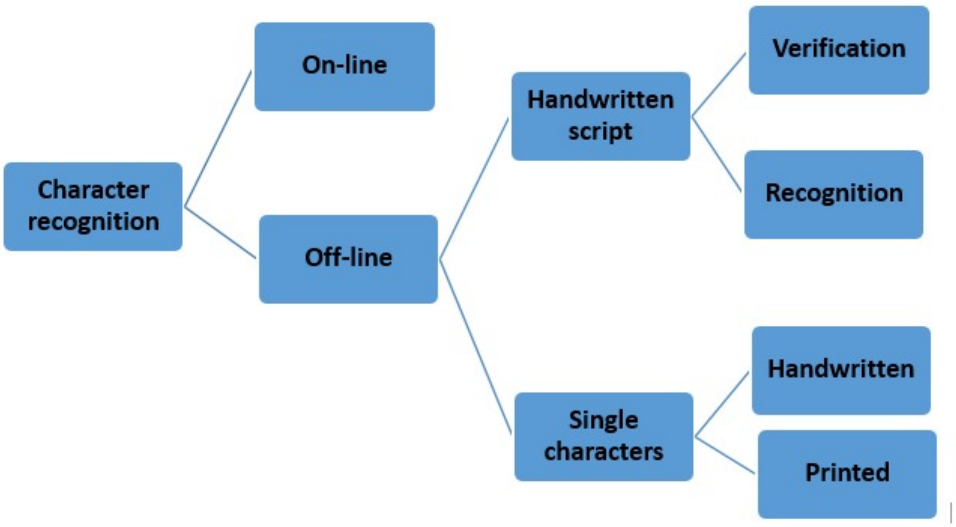
Optičko prepoznavanje znakova

OCR se primjenjuje kada je informacija čitljiva i ljudima i računalima, a do-

kument nije u formatu koji se lagano može uređivati. Za razliku od drugih tehnologija automatskog identificiranja OCR ne zahtijeva ljudski nadzor.[2]

2.1.2. Sustav za optičko prepoznavanje znakova

Optičko prepoznavanje znakova se bavi problemom prepoznavanja optički obrađenim znakovima u nekom obliku digitalnog formata. Takvo prepoznavanje znakova se primjenjuje $off\text{-}line$ (slika 2.1) koje za ulaz prima datoteku teksta kojeg želimo očitati. $Off\text{-}line$ optičko prepoznavanje znakova predstavlja automatsku pretvorbu teksta slike u tekst koji je računalu razumljiv. Moguće je očitati čak i rukom pisani tekst, ali su reultati jako ovisni o kvaliteti ulazne datoteke.[3]



Slika 2.1: Različita područja prepoznavanja znakova

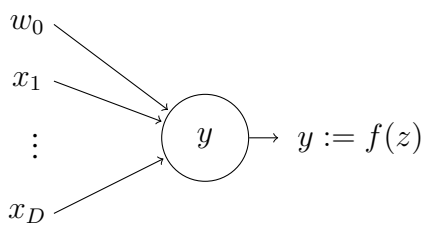
Što je ulaz ograničeniji i bolje strukturiran bolje su performanse OCR-a. No kada se radi o različitim tipovima rukopisa, OCR nije toliko pouzdan.[2]

3. Duboke neuronske mreže

Duboka (umjetna) neuronska mreža je sustav učenja koji koristi mrežu funkcija kako bi mogao razumjeti i prevesti podatke na ulazu u željeni izlaz koji je najčešće u drugom obliku. Koncept duboke neuronske mreže se temelji na ljudskoj biologiji i na načinu kako funkcioniraju neuroni ljudskog mozga da razumiju ulaze ljudskih osjetila.[4]

Umjetni neuron duboke neuronske mreže (engl. $Perceptron$) predstavlja glavnu građevnu jedinicu neuronske mreže. Svaki od neurona duboke neuronske mreže predstavlja matematičku funkciju s jednim ili više ulaznih varijabli i jednom izlaznom (slika 3.1).[5]

Svaki od neurona na svoj ulaz prima signal mnogih drugih neurona te je tako i njegov izlazni signal ulaz mnogih drugih neurona. Takvim mapiranjem ulaza i izlaza postiže se snažna povezanost. Upravo ta visoka povezanost dovodi do složenosti koja se može postići samo jako velikim skupom neurona. Snaga svake od veza između neurona je promjenjiva, čime se može promijeniti utjecaj jednog neurona na drugi. Sposobnost promjene snage pojedine veze je jedan od mehanizama na koji biološki mozak uči tijekom svog životnog vijeka.[6]



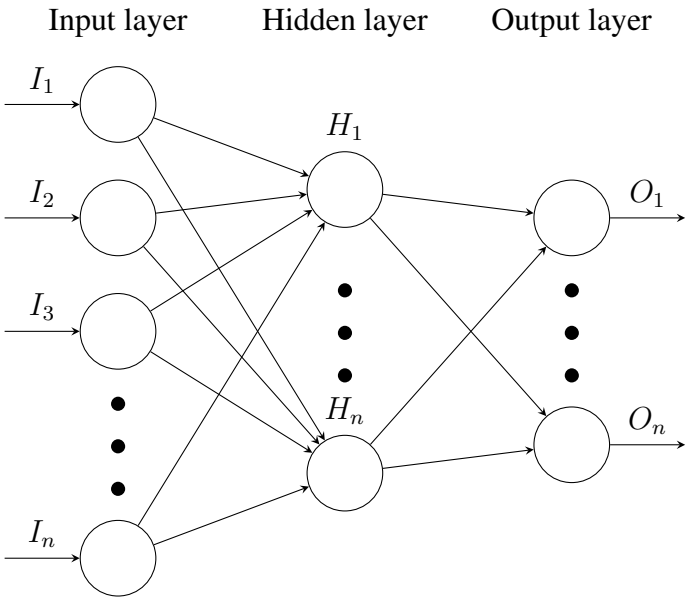
Slika 3.1: Umjetni neuron neuronske mreže

Duboka unaprijedna neuronska mreža (engl. $Feedforward\,neural\,network$) (slika 3.1.) je najjednostavniji oblik duboke neuronske mreže. Ime proizlazi iz činjenice da informacija u toj mreži putuje u samo jednom smjeru, od ulaznog sloja kroz skrivene slojeve prema izlaznom sloju. Bitno svojstvo ovakvih mreža je da tok informacija kroz mrežu ne formira ciklus.[8]

Ulazni sloj unaprijedne neuronske mreže služi za primanje i prijenos ulaznih podataka sljedećem sloju. Format ulaznih podataka je definiran oblikom ulaznog sloja.[9]

Izlazni sloj unaprijedne neuronske mreže služi za prikazivanje rezultata računanja mreže. Svaki od neurona u izlaznom sloju predstavlja jedno od rezultata proračuna mreže.[9]

Svi slojevi koji se nalaze između ulaznog i izlaznog sloja su skriveni slojevi. Skriveni sloj je zadužen za provedbu svih transformacija ulaza u izlaz koji nam može koristiti.[9]



Slika 3.2: Umjetna neuronska mreža

Tako modelirana neuronska mreža ima širok spektar primjene. Neuronska

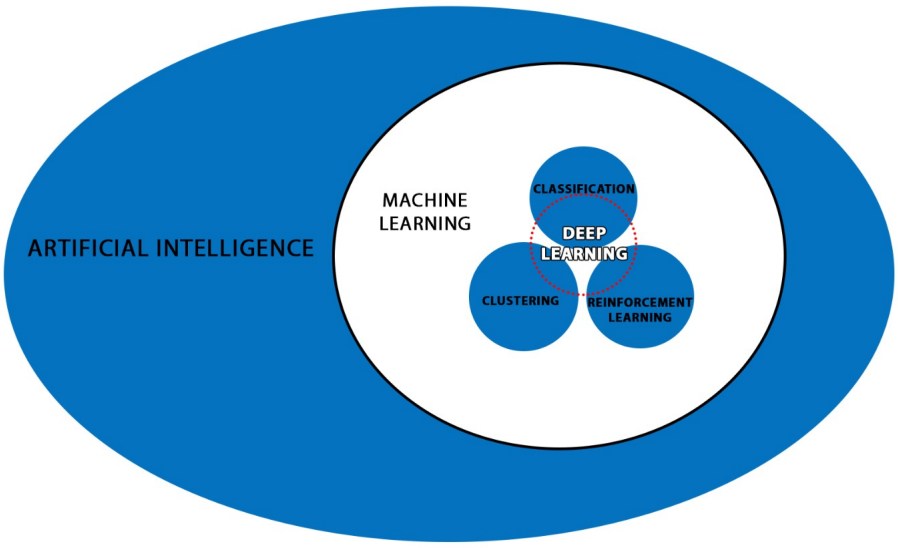
mreža se može oblikovati tako da na ulazu prima sliku, video, datoteku, bazu podataka ili neki drugi oblik digitaliziranog zapisa.[4]

Moguće je gotovo svaki problem strojnog učenja riješiti korištenjem dubokih neuronskih mreža zahvaljujući laganom generaliziranju implementacije (barem teoretski). Neke od najčešćih primjena dubokih neuronskih mreža danas su dijagnoza bolesti, prepoznavanje lica te gotovo svaki problem koji uključuje prepoznavanje uzoraka.[4]

3.1. Duboko učenje

Duboko učenje je podskup općenitijeg područja umjetne inteligencije nazvano strojno učenje, koje je zasnovano na ideji učenja na primjerima.[7] Slika 3.3 prikazuje odnos između umjetne inteligence, strojnog učenja i dubokog učenja.

Duboka neuronska mreža je jedan od mnogo alata i pristupa izvedbe algoritama strojnog učenja.



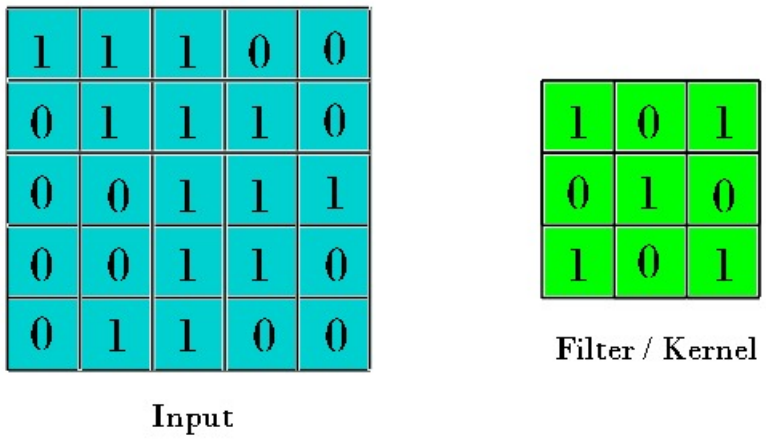
Slika 3.3: Odnos između umjetne inteligencije (AI), strojnog učenja (ML) i dubokog učenja (DL)

3.2. Konvolucijske neuronske mreže

Konvolucijska neuronska mreža je specijalizirana vrsta neuronskih mreža za predobradu nestrukturiranih podataka, posebice vezanih za slike, tekst, zvuk i govor. Konvolucijsku neuronsku mrežu gradi nekoliko različitih slojeva koji su opisani u nastavku.[10]

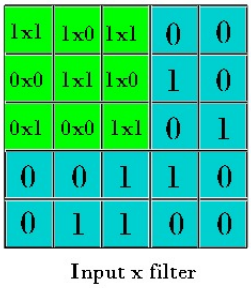
3.2.1. Konvolucijski sloj

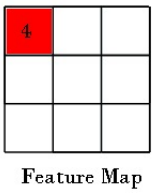
Glavni strukturni element konvolucijske neuronske mreže je konvolucijski sloj. Konvolucija (engl. *convolution*) je matematička operacija spajanja neka dva podatkovna skupa. Kod dubokih neuronskih mreža primjenom konvolucije na ulazni skup podataka i na konvolucijski filtar (slika 3.4) dobivamo značajke (engl. *feature map*).[10]



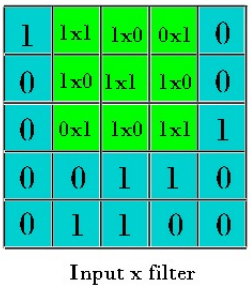
Slika 3.4: Primjer ulaza konvolucijskog sloja i filtra neuronske mreže

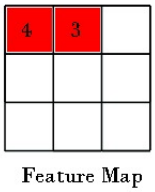
Primjenjujemo konvoluciju na svaku podmatricu ulaza veličine filtra i na filtar konvolucijskog sloja. Svaki element podmatrice ulaza množimo sa odgovarajućim elementom filtra, suma tih umnožaka su elementi matrice značajki. [10]





Slika 3.5: Primjena konvolucije na prvu podmatricu





Slika 3.6: Primjena konvolucije na drugu podmatricu

Na slikama 3.5 i 3.6 prikazana je primjena samo jedne konvolucije na ulaznim podacima, primjenom više konvolucija s različitim filtrima dobivamo različite matrice značajki. Taj skup značajki dobiven primjenom konvolucije na ulazne je izlaz konvolucijskog sloja.[10]

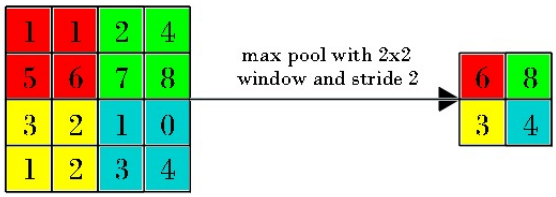
3.2.2. Sažimanje

Za smanjenje dimenzije nakon konvolucijskog sloja slijedi sažimanje (engl. $pooling$). Uporabom sažimanja reduciramo broj parametara, što rezultira kraćim trajanjem treniranja mreže i sprečavanjem prenaučenosti modela neuronske mreže (engl. $overfitting$).[10]

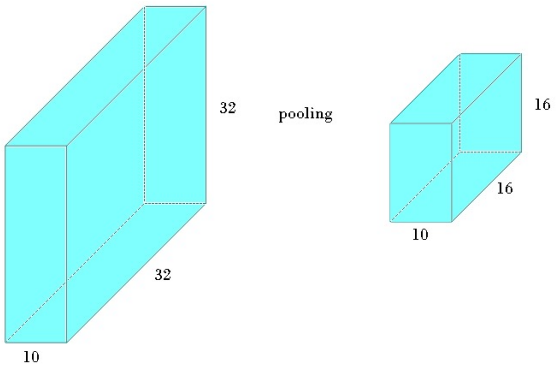
Sažimanje se primjenjuje na svaku od matrica značajki. Primjenom sažimanja smo dobili jednak broj matrica značajki sa smanjenim dimenzijama.[10]

Sažimanje maksimalnom vrijednošću (engl. $max\ pooling$) se pokazalo kao najefikasnija funkcija sažimanja za izvlačenje bitnih značajki (engl. $downsampling$). Sažimanje se još može provesti primjenom funkcija srednje vrijednosti, L2 normom ili funkcijom težinskog usrednjavanja.[10]

Sažimanje maksimalnom vrijednošću dijeli ulazne podatke u matrice sažimanja (engl. $pooling\ window$) te uzima maksimalnu vrijednost te matrice. Pomicanjem tog prozora po ulaznim podacima i računanjem funkcije sažimanja gradimo izlaznu matricu značajki.[10]



Slika 3.7: Primjena sažimanja na jednoj ulaznoj matrici



Slika 3.8: Primjena sažimanja na cijeli ulaz

Sažimanjem smo uspjeli smanjiti dimenzionalnost matrica značajki da smo zadržali samo najznačajnije podatke.[10]Na slici 3.7 prikazano je sažimanje jednog ulaznog sloja, dok je na sici 3.8 prikazana primjena sažimanja na svim ulaznim slojevima.

3.2.3. Potpuno povezani sloj

Kraj konvolucijske mreža nakon konvolucijskog sloja pa onda sažimanja čini jedan ili više potpuno povezanih slojeva (engl. $dense\ layer$).[10]

Znamo da potpuno povezani sloj na ulazu očekuje jednodimenzionalni vektor, a trenutni izlaz naše mreže je višedimenzionalan. Kako bi pripremili ulaz u potpuno povezani sloj između zadnjeg sloja sažimanja i prvog potpuno povezanog sloja dodajemo sloj za poravnanje (engl. $flatten$). Poravnanje višedimenzionalnog ulaza je zapravo preslagivanje svih elemenata u jedan vektor koji odgovara ulazu u potpuno povezani sloj.[10]

3.3. Povratne neuronske mreže

Povratna neuronska mreža (engl. $Recurrent\ Neural\ Network$) spada u klasu neuronskih mreža s internom memorijom. Za razliku od potpuno povezane neuronske mreže, povratne neuronske mreže računaju izlaz na temelju ulaznih informacija i trenutnog stanja (konteksta).[11]

Na arhitekturu neurona povezane neuronske mreže je dodana veza s izlaza

na ulaz. Ta veza "pamti" stanje neurona prethodnog koraka te ima utjecaj na odluku neurona u trenutnom koraku. Takav model se pokazao posebno dobar za probleme kao što su prepoznavanje teksta, prepoznavanje govora ili analiziranje različitih oblika signala.[11]

Kod duljih ulaznih sekvenci dolazi do problema u prijenosu bitnih informacija. Taj problem se naziva nestajući gradijent (engl. $vanishing\ gradient$).[11]

3.3.1. Mreža s dugom kratkoročnom memorijom

Mreža s dugom kratkoročnom memorijom (engl. $Long\ short\ term\ memory$, LSTM) je posebna verzija povratnih neuronskih mreža kojom je razriješen problem nestajućeg gradijenta. Mreža s dugom kratkoročnom memorijom se sastoji od jednog sloja povezanih LSTM ćelija. [11]

LSTM ćelija se sastoji od tri regulatora koji su zaduženi za određivanje trenutnog stanja i izlaznog stanja.[11]

Ulazna vrata (engl. $input\ gate$) filtriraju ulazne podatke i određuju koje će se vrijednosti ažurirati.[11]

Vrata za zaborav (engl. $forget\ gate$) kao ulaz primaju vrijednost trenutnog ulaza i izlaz prošlog koraka (kontekst). Uloga ovih je izdvajanje bitnih podataka od nebitnih.[11]

Izlazna vrata (engl. $output\ gate$) određuju vrijednost konteksta (engl. $hidden\ state$) koji se računa iz trenutnog stanja i konteksta prethodne iteracije.[11]

4. Model duboke neuronske mreže

Problem koji pokušavam riješiti je izvlačenje rukom pisanog teksta iz slike. Izvlačenje teksta je zapravo klasificiranje određivanje znakova sa slike i klasificiranje u određeni znak. Ograničio sam se na skup znakova američke abecede sve brojeve i znakove #, $, &, -, ', @ i ' ' (razmak).

Glavni dio ovog sustava koji radi najveći dio posla je model neuronske mreže. Za problem izvlačenja rukom pisanog teksta sam se odlučio za konvolucijsku povratnu neuronsku mrežu (engl. $Convolutional\ Recurrent\ Neural\ Network$, kraće: CRNN) čija je arhitektura opisana u poglavlju 4.6. Trenutno ta kombinacija konvolucijske i povratne neuronske mreže ostvaruje najbolje rezultate kod primjene u OCR-u.[21]

4.1. Skup podataka za učenje modela neuronske mreže

Kao skup podataka za učenje, validaciju i testiranje modela neuronske mreže koristim baze podataka $Handwriting\ Recognition$[12] i $HandWritten\_Character$[13].

Baza podataka $Handwriting\ Recognition$[12] sastoji se od 400,000 slika rukom pisanih imena i prezimena, od kojih je 206,799 slika imena te 207,024 slika prezimena. Slike su podijeljena u skup za učenje (80%), skup za validaciju (10%) te skup za testiranje (10%).

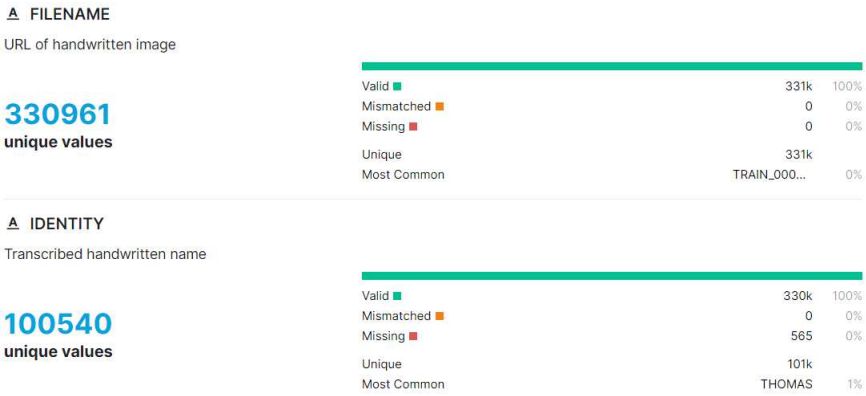
Slika 4.1 prikazuje strukturu baze podataka $Handwriting\ Recognition$ te njen sadržaj.

```
+-handwritten_names_dataset/
  +-test_v2/
  | +-test/
  |   +-TEST_0001.jpg
  |   +-TEST_0002.jpg
  |   +-TEST_0003.jpg
  |   +-TEST_0004.jpg
  |   +-... <additional files>
  +-train_v2/
  | +-train/
  |   +-TRAIN_00001.jpg
  |   +-TRAIN_00002.jpg
  |   +-TRAIN_00003.jpg
  |   +-TRAIN_00004.jpg
  |   +-... <additional files>
  +-validation_v2/
  | +-validation/
  |   +-VALIDATION_0001.jpg
  |   +-VALIDATION_0002.jpg
  |   +-VALIDATION_0003.jpg
  |   +-VALIDATION_0004.jpg
  |   +-... <additional files>
  +-written_name_test_v2.csv
  +-written_name_train_v2.csv
  +-written_name_validation_v2.csv
```

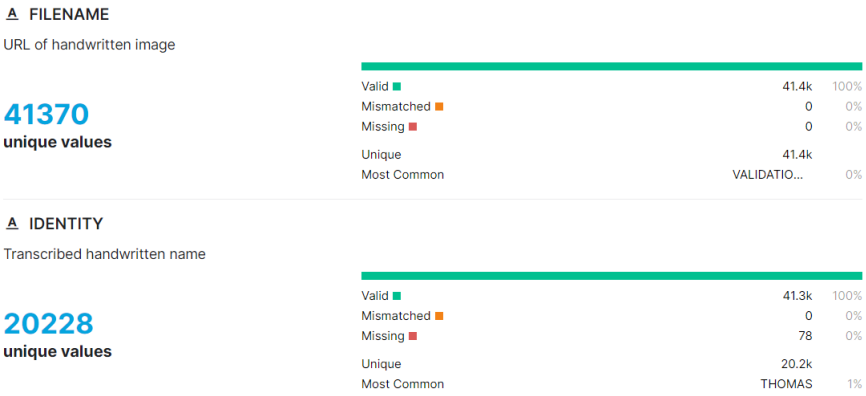
Slika 4.1: Struktura podataka $Handwriting\ Recognition$[12] baze

Za svaki od skupova postoji .csv tablica koja opisuje njihov sadržaj. Tablice se sastoje od dva stupca FILENAME i IDENTITY. Stupac FILENAME sadrži imena datoteka slike, a stupac IDENTITY za svaku pojedinu sliku sadrži ime ili prezime koje je napisano ili UNREADABLE ako je sadržaj slike nerazumljiv.

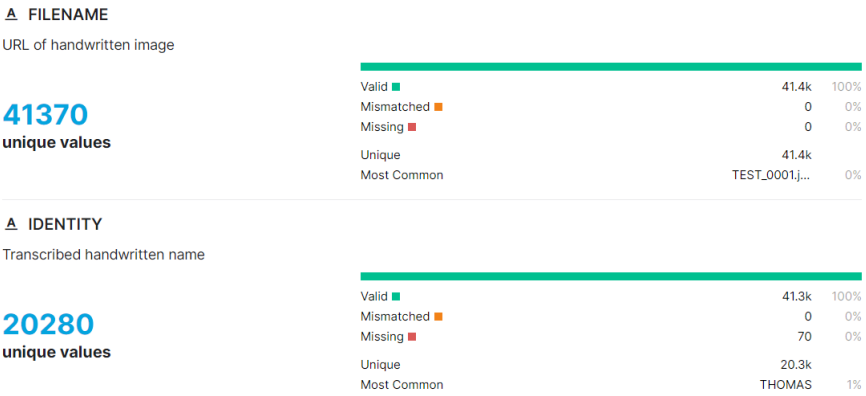
Slike 4.2, 4.3 i 4.4 prikazuju svojstva skupova za učenje, validaciju i testiranje baze podataka $Handwriting\ Recognition$. Bitna informacija za učenje neuronskre mreže je broj različitih vrijednosti polja IDENTITY.



Slika 4.2: Svojstva skupa za učenje[12]



Slika 4.3: Svojstva skupa za validaciju[12]

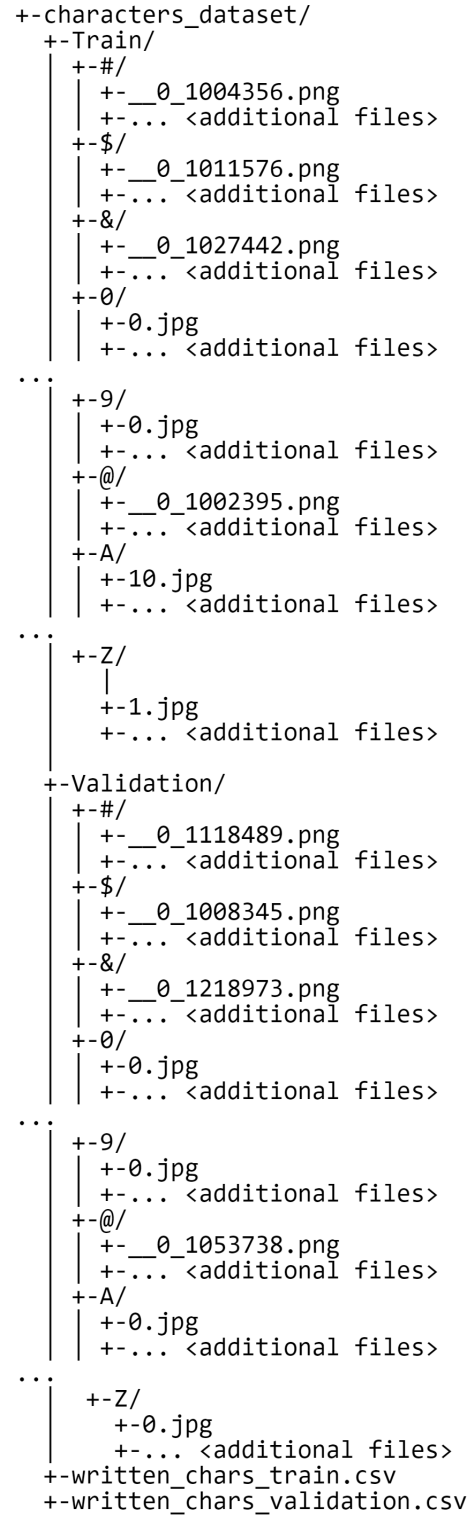


Slika 4.4: Svojstva skupa za testiranje[12]

Baza podataka $HandWritten\_Character$ sastoji se od 857,000 slika rukom pisanih znakova. Sadržaji slika su sve znamenke i slova američke abecede uz znakove #, $, & i @. Baza podataka je raspodijeljena u skup za učenje od 836,499 slika (97%)

i skup za validaciju koji sadrži 20,501 (3%) slika.[13]

Slika 4.4 prikazuje strukturu baze podataka $HandWritten\_Character$ te njen sadržaj.



Slika 4.5: Struktura podataka $HandWritten\_Character$ baze

Svaki od znakova baze se nalazi u mapi čije ime odgovara sadržaju svake od slike koju sadrži. Za potrebe učenja modela neuronske mreže sam kreirao .csv datoteke za oba skupa. Kreirao sam te .csv datoteke da odgovaraju strukturi .csv datoteka

baze podataka *Handwriting Recognition*. Za potrebe treniranja modela neuronske mreže iz ove baze sam koristio samo slike znamenaka te znakova #, $, &, ', -, @ i ' '(razmak).[12]

4.2. Obrada podataka

Kako bih pokrenuo učenje modela neuronske mreže na slikama iz baze podataka potrebno je pripremiti podatke u format za nadzirano učenje.

Iz učitanih slika potrebno je izbaciti slike koje nemaju definiranu jednu od vrijednosti IDENTITY ili FILENAME te slike kojima je IDENTITY vrijednosti UNREADABLE što je prikazano u isječku koda listing 1.

```
train = pd.read_csv('written_name_train_v2.csv')
valid = pd.read_csv('written_name_validation_v2.csv')
train_numbers = pd.read_csv("written_chars_train.csv")
valid_numbers = pd.read_csv("written_chars_validation.csv")

train.dropna(axis=0, inplace=True)
valid.dropna(axis=0, inplace=True)

train = train[train['IDENTITY'] != 'UNREADABLE']
valid = valid[valid['IDENTITY'] != 'UNREADABLE']

train['IDENTITY'] = train['IDENTITY'].str.upper()
valid['IDENTITY'] = valid['IDENTITY'].str.upper()

train.reset_index(inplace = True, drop=True)
valid.reset_index(inplace = True, drop=True)
```

Listing 1: Programski kod za filtriranje valjanih slika[14]

Za učenje modela neuronske mreže radi ograničenja RAM memorije sam bio ograničen na 40,000 slika od kojih sam 35,000 (87.5%) odredio za učenje, a ostalih 5,000 (2.5%) slika za validaciju (listing 2).

```
train_size = 35000
valid_size= 5000

train = train[:train_size - 14000]
valid = valid[:valid_size - 1400]

train = pd.concat([train, train_numbers])
valid = pd.concat([valid, valid_numbers])

train.dropna(axis=0, inplace=True)
valid.dropna(axis=0, inplace=True)

train.reset_index(inplace = True, drop=True)
valid.reset_index(inplace = True, drop=True)
```

Listing 2: Programski kod za određivanje slika s kojima ćemo trenirati model neuronske mreže[14]

Nakon odabira slika potrebno ih je učitati u listu. Učitavanje slika prikazano isječcima koda listing 3 i listing 4 obavljamo pomoću naredbe $imread$ paketa $cv2$ kojoj predajemo putanju do određene slike. Prilikom učitavanja slike, vrijednosti njenih piksela skaliramo kako bi dobili crno-bijelu sliku.

```
train_x = []

for i in range(train_size - 14000):
    img_dir = 'train_v2/train/' + train.loc[i, 'FILENAME']
    image = cv2.imread(img_dir, cv2.IMREAD_GRAYSCALE)
    image = preprocess(image)
    image = image/255.
    train_x.append(image)
```

Listing 3: Programski kod za učitavanje slika baze $HandWritten\_Character$ koje smo odabrali za učenje[14]

```
br = 0
x = train_size - 14000
for name in "#$&0123456789@":
  for i in range(1000):
      img_dir = 'Train/' + name + "/" +
                    train.loc[x + br, 'FILENAME']
      image = cv2.imread(img_dir, cv2.IMREAD_GRAYSCALE)
      image = preprocess(image)
      image = image/255.
      train_x.append(image)
      br += 1
```

Listing 4: Programski kod za učitavanje slika baze *Handwriting Recognition* koje smo odabrali za učenje

Nakon učitavanja slika ih dodatno obrađujem naredbom $process(image)$ (listing 5). Naredba $process$ prima sliku koju preoblikujemo u matricu 256 x 64 piksela. Ako je slika manjih dimenzija slika je popunjena bijelim pikselima. Nakon preoblikovanja slike rotiramo je za 90° u smjeru kazaljke na satu.

Završetkom $process$ piksele slike normaliziramo u skup [0, 1] tako što svaki od piksela slike podijelimo sa 255 što je maksimalna vrijednost piksela.

```
def preprocess(img):
    (h, w) = img.shape

    final_img = np.ones([64, 256])*255 # blank white image

    # crop
    if w > 256:
        img = img[:, :256]

    if h > 64:
        img = img[:64, :]
```

```
    final_img[:h, :w] = img
    return cv2.rotate(final_img, cv2.ROTATE_90_CLOCKWISE)
```

Listing 5: Programski kod za učitavanje slika koje smo odabrali za testiranje[14]

Isječak koda listing 6 prikazuje kreiranje 3 dodatne liste za potrebe učenja modela neuronske mreže. Lista $train\_y$ se sastoji od teksta koji je napisan na slikama. U listi $train\_label\_len$ se nalaze duljine teksta napisanih na slikama, s istom svrhom je napravljena i lista $train\_input\_len$ u koju ćemo spremati duljine teksta koji smo dobili prilikom učenjam modela. $train\_output$ lista služi za spremanje teksta koji je model očitao iz slike tijekom učenja.

```
train_y = np.ones([train_size, max_str_len]) * -1
train_label_len = np.zeros([train_size, 1])
train_input_len = np.ones([train_size, 1]) * (num_of_timestamps-2)
train_output = np.zeros([train_size])

for i in range(train_size):
    train_label_len[i] = len(train.loc[i, 'IDENTITY'])
    train_y[i, 0:len(train.loc[i, 'IDENTITY'])] =
                label_to_num(train.loc[i, 'IDENTITY'])
```

Listing 6: Programski kod pripreme podataka za učenje modela neuronske mreže[14]

Dodatne funkcije $label\_to\_num(lebel)$ i $num\_to\_label(num)$ (listing 7) služe za mapiranje znakova u brojku te obratno. Ovakvo mapiranje slova u brojke nam olakšava računanje funkcije gubitka tijekom učenja modela neuronske mreže.

```
alphabets = u"#$&@0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ-' "
max_str_len = 24
num_of_characters = len(alphabets) + 1 # +1 for ctc pseudo blank
num_of_timestamps = 64 # max length of predicted labels
```

```
def label_to_num(label):
    label_num = []
    for ch in label:
        label_num.append(alphabets.find(ch))

    return np.array(label_num)

def num_to_label(num):
    ret = ""
    for ch in num:
        if ch == -1:  # CTC Blank
            break
        else:
            ret+=alphabets[ch]
    return ret
```

Listing 7: Programski kod pripreme podataka za učenje modela neuronske mreže[14]

4.3. Izrada modela neuronske mreže

Struktura modela neuronske mreže se sastoji od slojeva konvolucijske mreže koji se preko potpuno povezanog sloja spaja s tri sloja povratne neuronske mreže. Konvolucijski dio modela se sastoji od tri sloja konvolucijska sloja što se pokazalo kao standardna praksa za izvlačenje značajki rukom pisanog teksta. U praksi broj LSTM slojeva varira, u mojem slučaju tri LSTM sloja od 256 ćelija se pokazalo kao najbolje rješenje. Isječak koda listing 8 prikazuje kreiranje arhitekture duboke neuronske mreže.

```
input_data = Input(shape = (256, 64, 1), name = 'input')

inner = Conv2D(32, (3, 3), padding = 'same', name =
↪  'conv1', kernel_initializer =
↪  'he_normal')(input_data)
```

```
inner = BatchNormalization()(inner)
inner = Activation('relu')(inner)
inner = MaxPooling2D(pool_size=(2, 2), name =
 ↪  'max1')(inner)

inner = Conv2D(64, (3, 3), padding = 'same', name =
 ↪  'conv2', kernel_initializer = 'he_normal')(inner)
inner = BatchNormalization()(inner)
inner = Activation('relu')(inner)
inner = MaxPooling2D(pool_size = (2, 2), name =
 ↪  'max2')(inner)
inner = Dropout(0.3)(inner)

inner = Conv2D(128, (3, 3), padding = 'same', name =
 ↪  'conv3', kernel_initializer = 'he_normal')(inner)
inner = BatchNormalization()(inner)
inner = Activation('relu')(inner)
inner = MaxPooling2D(pool_size = (1, 2), name =
 ↪  'max3')(inner)
inner = Dropout(0.3)(inner)

# CNN to RNN
inner = Reshape(target_shape=((64, 1024)), name =
 ↪  'reshape')(inner)
inner = Dense(64, activation = 'relu', kernel_initializer
 ↪  = 'he_normal', name = 'dense1')(inner)

## RNN
inner = Bidirectional(LSTM(256, return_sequences=True),
 ↪  name = 'lstm1')(inner)
inner = Bidirectional(LSTM(256, return_sequences=True),
 ↪  name = 'lstm2')(inner)
inner = Bidirectional(LSTM(256, return_sequences=True),
 ↪  name = 'lstm3')(inner)

## OUTPUT
```

```
inner = Dense(num_of_characters, kernel_initializer
 ↪  ='he_normal',name = 'dense4')(inner)
y_pred = Activation('softmax', name = 'softmax')(inner)

model = Model(inputs = input_data, outputs = y_pred)
```

Listing 8: Programski kod za stvaranje konvolucijskog povratnog modela neuronske mreže[14]

Slika 4.6 prikazuje potpunu arhitekturu gotovog modela, koji se sastoji od 3,988,972 parametara od kojih je moguće regulirati (naučiti) 3,988,524 podijeljenih na ukupno 22 sloja. Većina parametara se nalazi u povratnom dijelu mreže (95.45%).

```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input (InputLayer)           [(None, 256, 64, 1)]      0
_________________________________________________________________
conv1 (Conv2D)               (None, 256, 64, 32)       320
_________________________________________________________________
batch_normalization (BatchNo (None, 256, 64, 32)       128
_________________________________________________________________
activation (Activation)      (None, 256, 64, 32)       0
_________________________________________________________________
max1 (MaxPooling2D)          (None, 128, 32, 32)       0
_________________________________________________________________
conv2 (Conv2D)               (None, 128, 32, 64)       18496
_________________________________________________________________
batch_normalization_1 (Batch (None, 128, 32, 64)       256
_________________________________________________________________
activation_1 (Activation)    (None, 128, 32, 64)       0
_________________________________________________________________
max2 (MaxPooling2D)          (None, 64, 16, 64)        0
_________________________________________________________________
dropout (Dropout)            (None, 64, 16, 64)        0
_________________________________________________________________
conv3 (Conv2D)               (None, 64, 16, 128)       73856
_________________________________________________________________
batch_normalization_2 (Batch (None, 64, 16, 128)       512
_________________________________________________________________
activation_2 (Activation)    (None, 64, 16, 128)       0
_________________________________________________________________
max3 (MaxPooling2D)          (None, 64, 8, 128)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 64, 8, 128)        0
_________________________________________________________________
reshape (Reshape)            (None, 64, 1024)          0
_________________________________________________________________
dense1 (Dense)               (None, 64, 64)            65600
_________________________________________________________________
lstm1 (Bidirectional)        (None, 64, 512)           657408
_________________________________________________________________
lstm2 (Bidirectional)        (None, 64, 512)           1574912
_________________________________________________________________
lstm3 (Bidirectional)        (None, 64, 512)           1574912
_________________________________________________________________
dense4 (Dense)               (None, 64, 44)            22572
_________________________________________________________________
softmax (Activation)         (None, 64, 44)            0
=================================================================
Total params: 3,988,972
Trainable params: 3,988,524
Non-trainable params: 448
_________________________________________________________________
```

Slika 4.6: Izgrađeni konvolucijski povratni model neuronske mreže

4.4. Učenje modela neuronske mreže

Nakon što smo definirali arhitekturu duboke neuronske mreže potrebno ju je prevesti u oblik koji možemo naučiti. Za prevođenje modela koristimo funkciju $compile$ kojoj predajemo optimizacijsku funkciju (engl. $optimizer$), mjeru koju pratimo tijekom učenja modela (engl. $metrics$) te funkciju gubitka (engl. $loss\ function$).

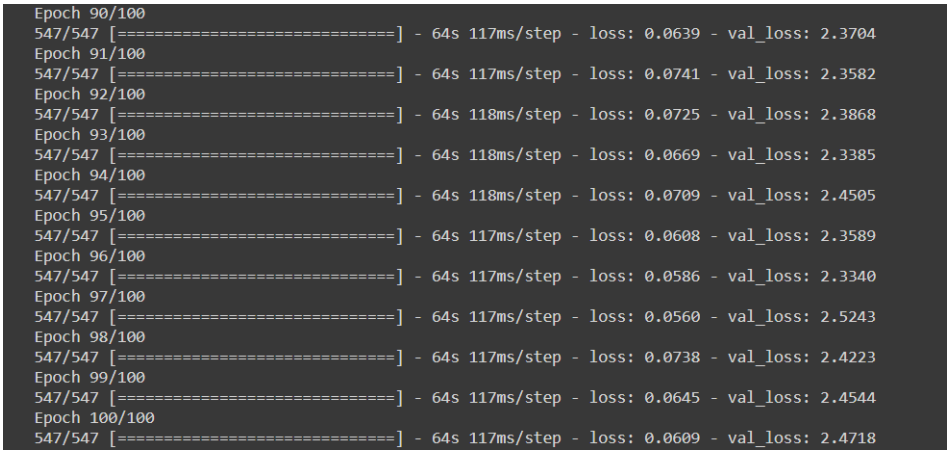
Optimizacijske funkcije su algoritmi kojima mijenjamo parametre neuronske mreže kako bi smanjili funkciju gubitaka. ADAM (engl. $ADAptive\ Moment\ Estimation$) optimizator je kombinacija AdaGrad i RMSprop algoritama. ADAM procjenjuje promjenu parametara na temelju prvog i drugog momenta gradijenta.

Fukcija gubitka služi za usporedbu predikcije modela tijekom nadziranog učenja. Pomoću funkcije gubitka možemo pratiti točnost modela tijekom učenja.

```
model_final.compile(loss={'ctc': lambda y_true, y_pred: y_pred},
        optimizer=Adam(lr = 0.0001))

model_final.fit(x=[train_x, train_y, train_input_len,
        train_label_len], y=train_output,
        validation_data=([valid_x, valid_y, valid_input_len,
        valid_label_len], valid_output),
        epochs=100, batch_size=64)
model.save("nums_and_chars_3_lstm_mo_dense.h5")
```

Listing 9: Programski kod za pokretanje treniranja modela neuronske mreže[14]

Učenje mreže prikazano isječkom koda listing 9 započinje pozivom funkcije $fit()$ kojoj predajemo skupove za učenje, validaciju, veličinu serije (engl. $batch\ size$) koja označava nakon koliko ulaznih podataka ćemo ažurirati parametre te broj epoha koji predstavlja broj puta koji ćemo upotrijebiti sve slike za učenje.



Slika 4.7: Proces učenja modela neuronske mreže

4.5. Vrednovanje modela

Nakon učenja ispitujem točnost modela na skupu za testiranje koji se sastoji od slika iz obiju baza podataka. Obrada slika jednako je obradi slika skupa za učenje i validaciju.

Skup za testiranje se sastoji od 40,000 slika od kojih je 12,000 (30%) iz baze podataka $Handwriting\ Recognition$[12], a ostalih 28,000 (70%) iz baze podataka $HandWritten\_Character$[13]. Iz baze $HandWritten\_Character$[13] sam izabrao po 2,000 slika brojeva od 0 do 9 te posebnih znakova @, #, $ i &.

Model ostvaruje točnost od 85.97% pri očitavanju pojedinih znakova, a nešto manju točnost od 83.86% pri očitavanju kompletnih riječi u slučaju kada sam testirao model na kombinaciji baza (slika 4.8).

```
y_true = test.loc[0:test_size, 'IDENTITY']
correct_char = 0
total_char = 0
correct = 0

for i in range(test_size):
    pr = prediction[i]
    tr = y_true[i]
    total_char += len(tr)

    for j in range(min(len(tr), len(pr))):
        if tr[j] == pr[j]:
            correct_char += 1

    if pr == tr :
        correct += 1

print('Correct characters predicted : %.2f%%' %(correct_char * 100 / total_char))
print('Correct words predicted      : %.2f%%' %(correct * 100 / test_size))
```

```
Correct characters predicted : 85.97%
Correct words predicted      : 83.86%
```

Slika 4.8: Točnost modela na kombinaciji baza

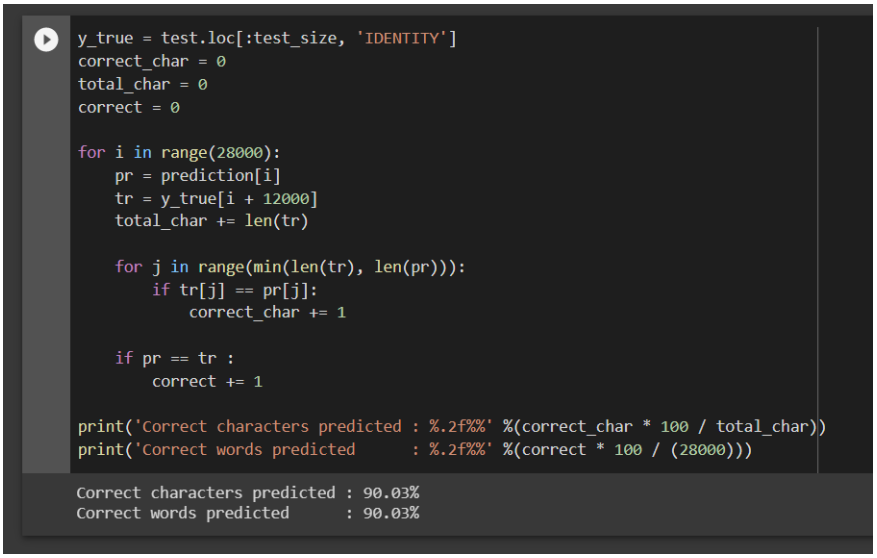
Model ostvaruje točnost od 84.52% na podacima testnog skupa iz baze podataka *Handwriting Recognition* na pojedinom znaku, a od toga je 69.47% ispravno očitanih riječi (slika 4.9).

```
y_true = test.loc[0:test_size, 'IDENTITY']
correct_char = 0
total_char = 0
correct = 0

for i in range(test_size - 28000):
    pr = prediction[i]
    tr = y_true[i]
    total_char += len(tr)

    for j in range(min(len(tr), len(pr))):
        if tr[j] == pr[j]:
            correct_char += 1

    if pr == tr :
        correct += 1

print('Correct characters predicted : %.2f%%' %(correct_char * 100 / total_char))
print('Correct words predicted      : %.2f%%' %(correct * 100 / (test_size - 28000)))
```

```
Correct characters predicted : 84.52%
Correct words predicted      : 69.47%
```

Slika 4.9: Točnost modela na bazi *Handwriting Recognition*

Na podacima *HandWritten_Character*[13] testnog skupa model ostvaruje točnost od 90.03% (slika 4.10).



```
y_true = test.loc[:test_size, 'IDENTITY']
correct_char = 0
total_char = 0
correct = 0

for i in range(28000):
    pr = prediction[i]
    tr = y_true[i + 12000]
    total_char += len(tr)

    for j in range(min(len(tr), len(pr))):
        if tr[j] == pr[j]:
            correct_char += 1

    if pr == tr :
        correct += 1

print('Correct characters predicted : %.2f%%' %(correct_char * 100 / total_char))
print('Correct words predicted      : %.2f%%' %(correct * 100 / (28000)))
```

```
Correct characters predicted : 90.03%
Correct words predicted      : 90.03%
```

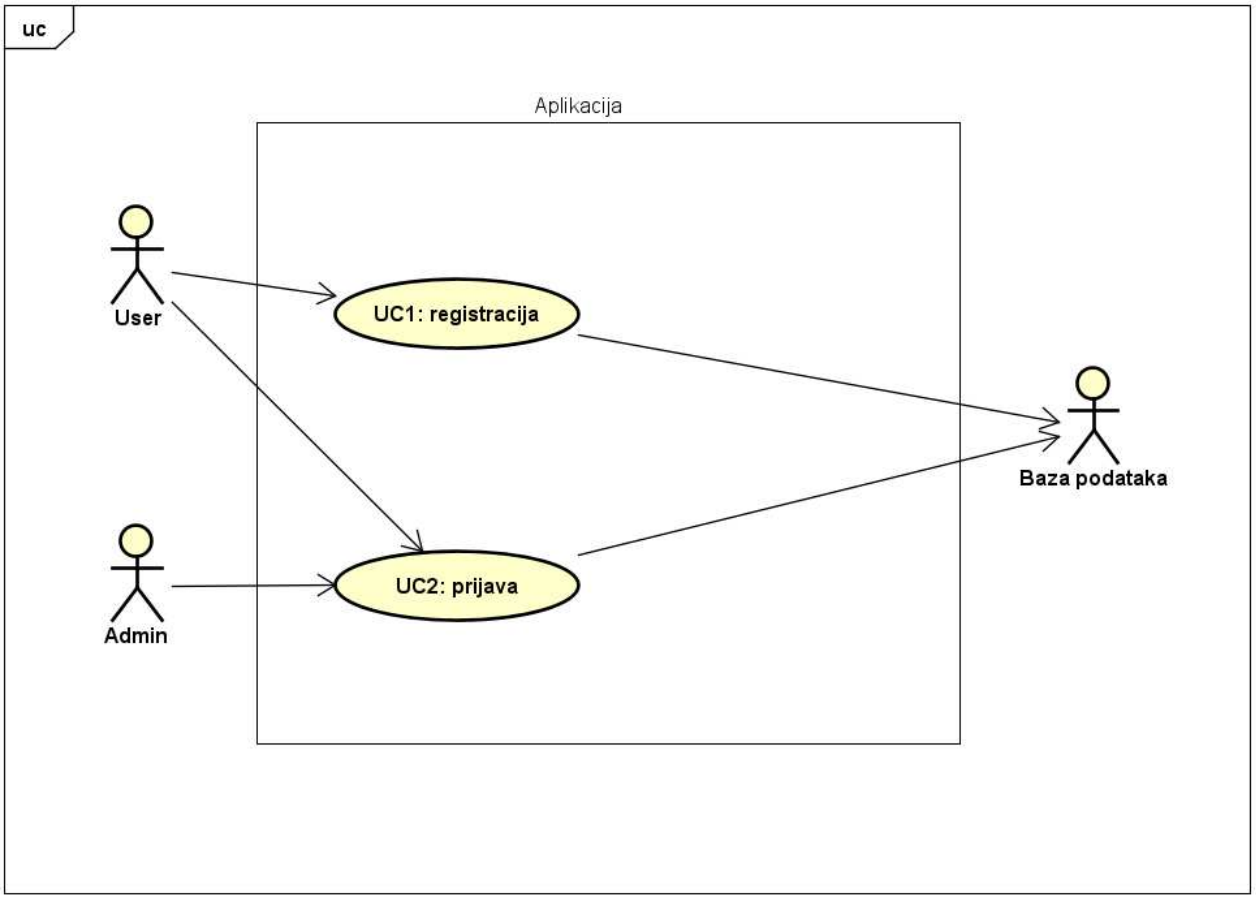
Slika 4.10: Točnost modela na bazi $HandWritten\_Character$

5. Aplikacija

Kako bih olakšao korištenje modela duboke neuronske mreže za prepoznavanje rukom pisanog teksta napravio sam mobilnu aplikaciju za sustav Android. Aplikacija podržava prepoznavanje rukom pisanog teksta sa slika slikanih u aplikaciji ili slika dovezenih iz galerije. U aplikaciji je također implementirana mogućnost izvoza teksta skeniranog sa slike u .txt formatu. Aplikaciji se pristupa preko forme za prijavu korištenjem e-pošte i lozinke.

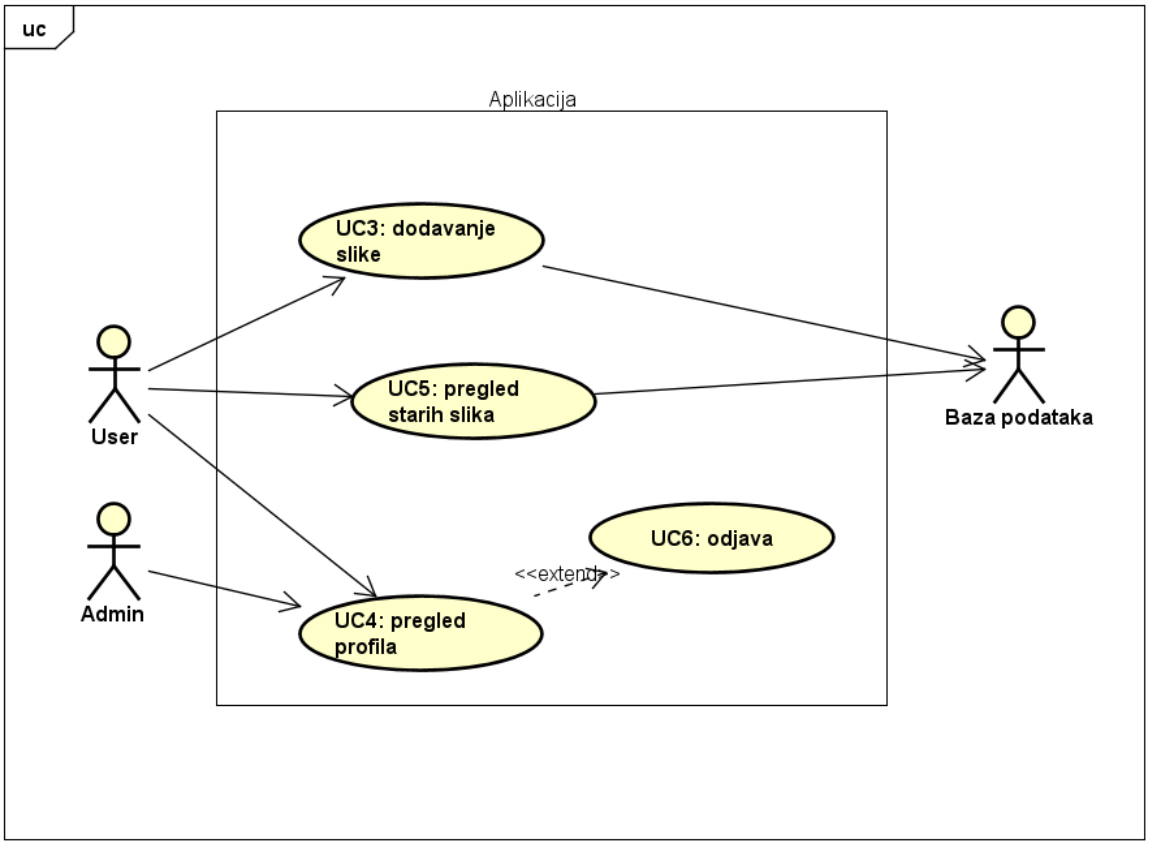
5.1. Funkcionalni zahtjevi sustava

Aktori ovog sustava su svi korisnici i admin. Korisnik prilikom ulaska u aplikaciju ima mogućnost prijave ili registracije u sustav. Za razliku od korisnika, korisnički podaci admina su unaprijed upisani u bazu podataka te za stjecanje ovlasti admina se potrebno prijaviti ispravnim korisničkim podacima. Nakon uspješne registracije u sustav korisnik ima mogućnost prijave u sustav korisničkim podacima unesenim pri registraciji. Korisnik dobiva mogućnost korištenja funkcionalnosti sustava tek nakon uspješne prijave u sustav (slika 5.1).



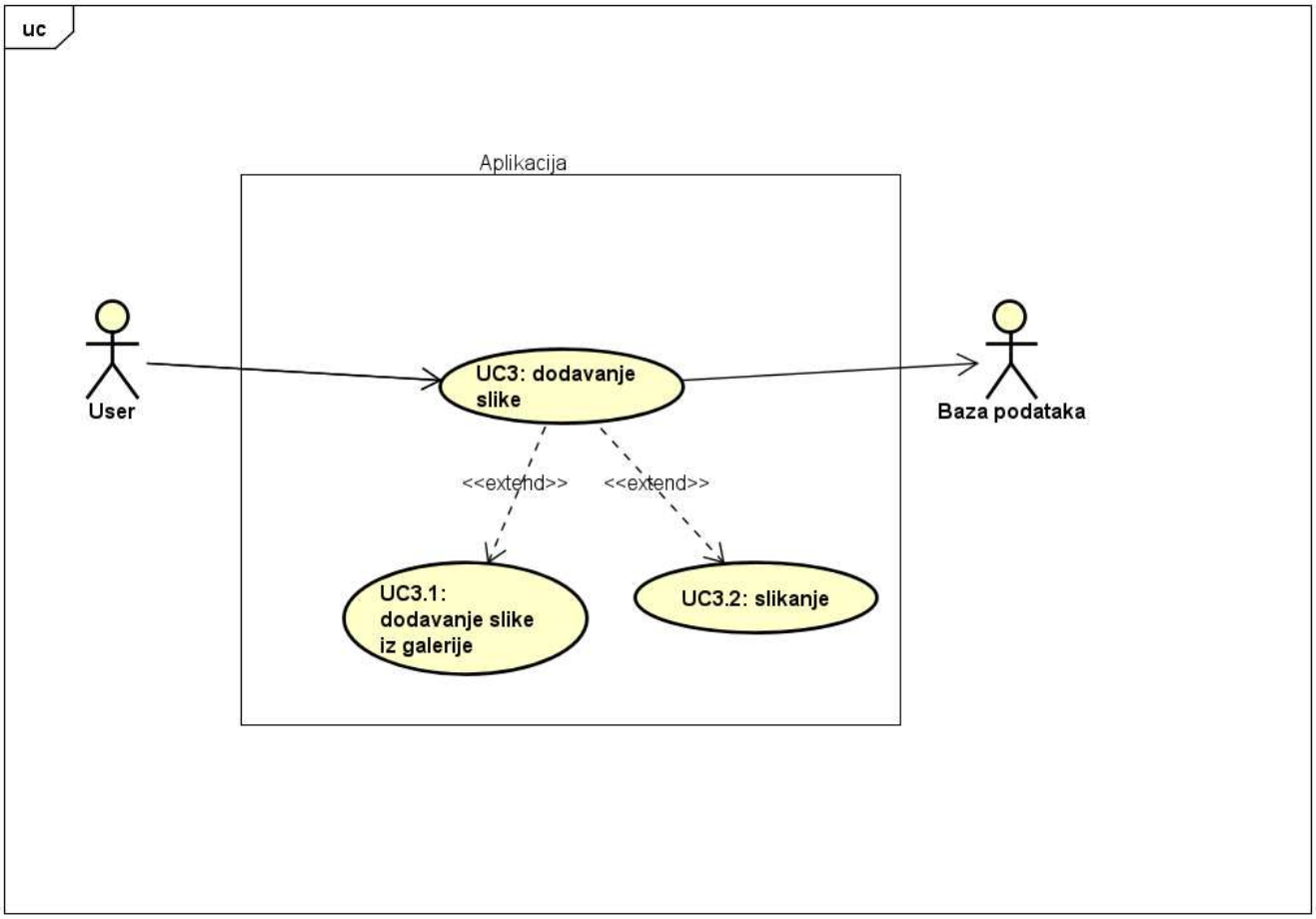
Slika 5.1: Dijagram obrazaca uporabe 1

Nakon uspješne registracije korisnik ima mogućnost pregleda profila, starih slika ili mogućnost skeniranja nove slike. Korisnik, a tako i admin na stranici profila mogu pregledati ime, prezime i mail koji su unijeli prilikom registracije te se ujedno i odjaviti. Pregled starih slika i korištenje opcije za skeniranje slike je omogućen samo korisniku (slika 5.2).



Slika 5.2: Dijagram obrazaca uporabe 2

Kako bi korisnik iskoristio glavnu funkcionalnost očitavanja rukom pisanog teksta iz slike ima mogućnost uvoza slike iz galerije ili slikanjem nove slike, koja se prilikom odabire opcije "UPLOAD" šalju na obradu. Po završetku obrade slike korisnik dobiva rezultat u .txt formatu (slika 5.3).



Slika 5.3: Dijagram obrazaca uporabe 3

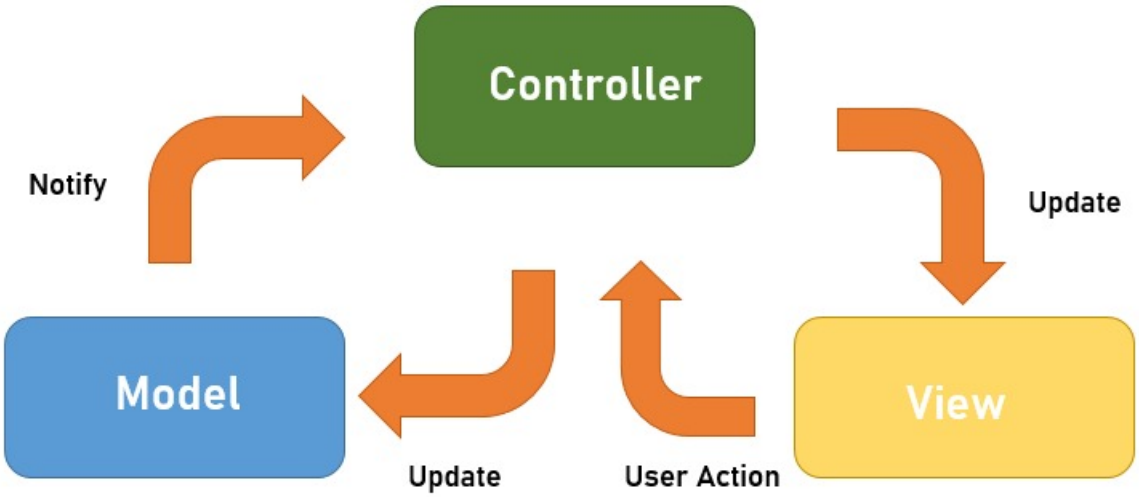
5.2. Arhitektura sustava

Arhitektura aplikacije je podijeljena u tri podsustava: poslužiteljska strana (mobilna aplikacija), klijentska strana i baza podataka. Takva arhitektura sustava zasnovana je na MVC (*Model View Controller*) modelu prikazanom na slici 5.4.

Sloj *Model* predstavlja strukture podataka koje sustav koristi, u našem slučaju to je baza podataka.

View predstavlja prikaz podataka unutar aplikacije, drugim riječima to je sve što korisnik može vidjeti.

Sloj *Controller* upravlja i slojem *Model* i slojem *View*. On služi za razmjenu podataka između slojeva *Model* i *View*, ali i za ažuriranje podataka u istim.



Slika 5.4: MVC model

5.3. Tehnologije i alati

Tehnologije koje su korištene za izradu aplikacije su podijeljene na dva dijela: klijentski i poslužiteljski. Korištene tehnologije na klijentskom dijelu aplikacije su programski jezik Java u radnom okviru *Android Studio* u kombinaciji s alatom *Gradle* alatom za izgradnju aplikacije. Na poslužiteljskom djelu aplikacije isto kao i na klijentskom dijelu koristi se programski jezik Java uz alat *Maven* za automatsku izgradnju, programski okvir *Spring*, programski okvir *Hibernate* , programski okvir *JakartaPersistenceAPI* te baza podataka *PostgreSQL* uz platformu za upravljanje bazom podataka *PgAdmin* .

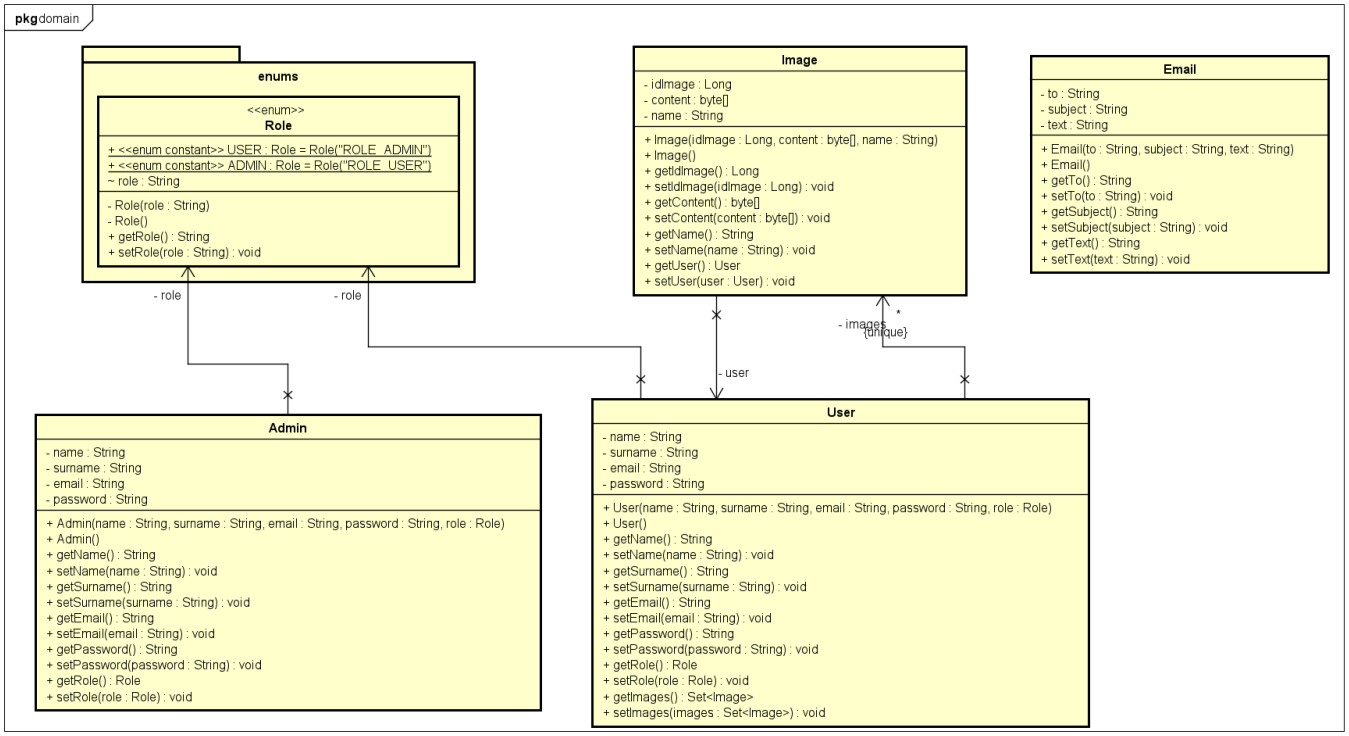
5.4. Poslužiteljska strana

Poslužiteljska strana aplikacije sastoji se od modela (engl. *model*), usluge (engl. *service*), nadzornika (engl. *controller*) i repozitorija (engl. *repository*).

Modeli su Javine klase koje su uz programski alat *Hibernate* mapirane u relacijske tablice baze podataka. Svaki model predstavlja jednu relacijsku tablicu u bazi

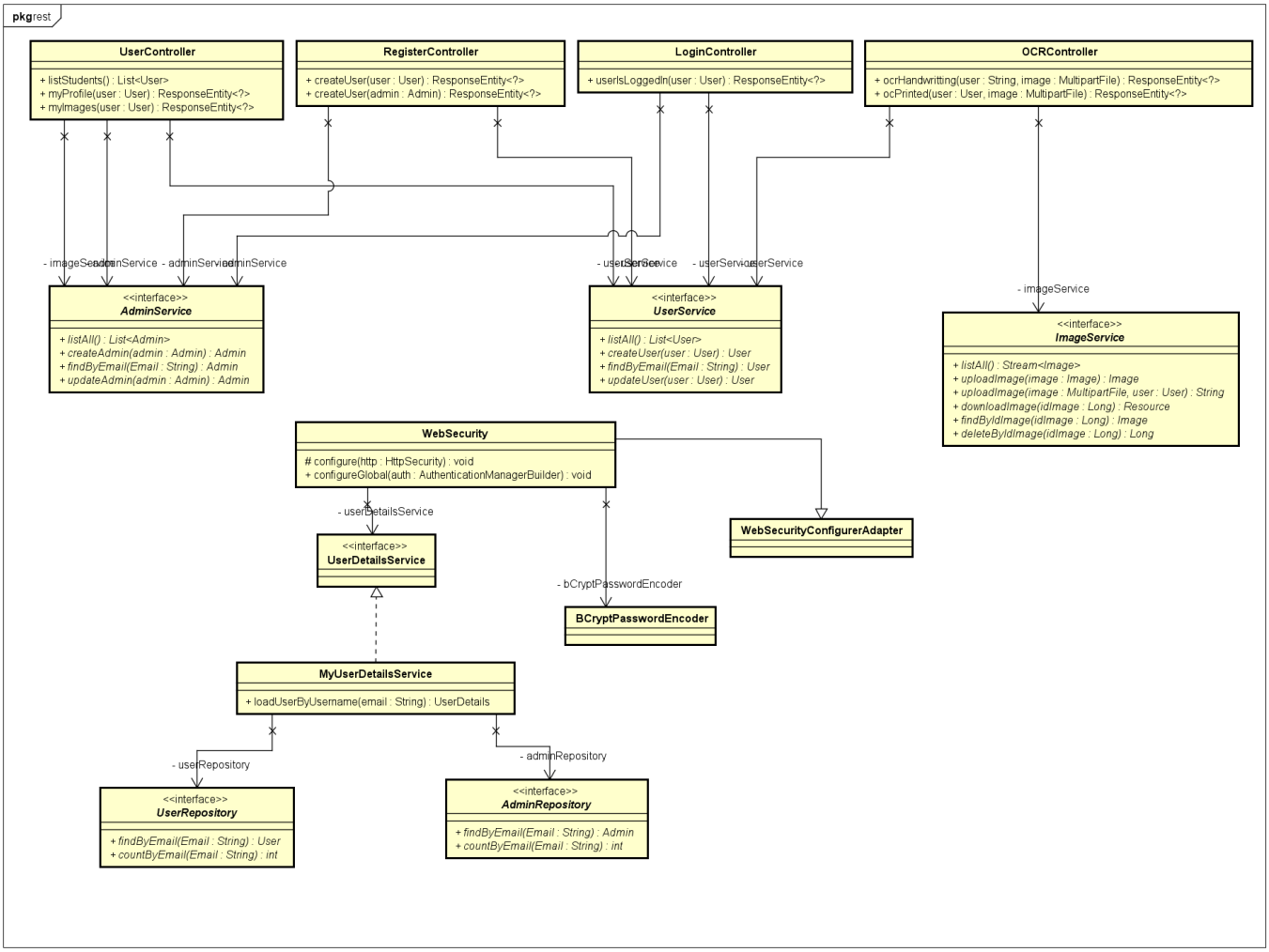
podataka, a njeni članski atributi su stupci u istoj.[19]

Modeli se označavaju s anotacijom $@Entity$, a primarni ključ koji je atribut klase se označava anotacijom $@Id$. Moguće je povezivati entitete vezama koji se označavaju anotacijama: $OneToMany$, $OneToOne$, $ManyToOne$ i $ManyToOne$.[19], [20]Slika 5.5 prikazuje dijagram klasa sloja model aplikacije.



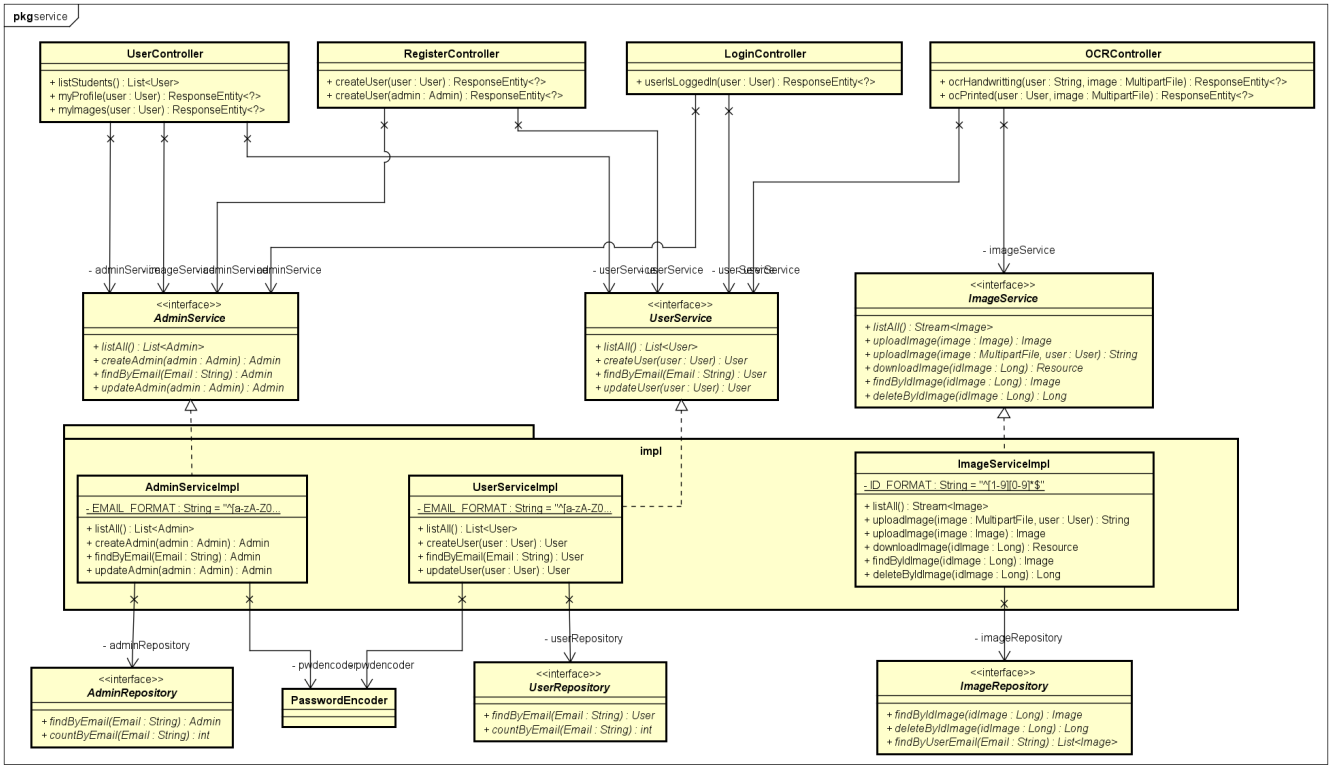
Slika 5.5: Dijagram razreda Model

Nadglednik (engl. $controller$) je Javina klasa koja služi za primanje HTTP zahtjeva. Nadglednici se označavaju anotacijom $@RestController$ u kombinaciji s anotacijom $@RequestMapping("")$ koja označava putanju. Klasa nadgednika sadrži metode koje predstavljaju REST API-je. Te metode označavaju HTTP metode koje se označavaju anotacijama $@GetMapping$, $@PostMapping$, $@PutMapping$ ili $@DeleteMapping$.[19], [20]Arhitektura sloja nadglednika prikazana je slikom 5.6.



Slika 5.6: Dijagram razreda REST

Programska logika aplikacije se označava se označava anotacijom $@Service$. Sloj usluge nadopunjuje metode za pristup bazi generiranih od alata $Hibernate$ ORM. Rješenje programske logike ove aplikacije prikazano je slikom 5.7.



Slika 5.7: Dijagram razreda Service

5.4.1. REST servisi u springu

REST (engl. $REpresentational\ State\ Transfer$) je arhitekturni stil za opisivanje željene web arhitekture. Stil REST se sastoji od klijenta, poslužitelja, resursa te HTTP operacija. REST oblikovni obrazac omogućava i pojednostavljuje komunikaciju klijenta i poslužitelja.[19], [20]

Pomoću REST arhitekture smo postigli neovisnost klijenta i poslužitelja, što omogućuje odvojeni.

5.4.2. JPA i ORM

Jakarta Persistence API (engl. $Application\ Programming\ Interface$, JPA) JPA je rješenje za dosljednost podataka (engl. $data\ persistence$). $Jakarta\ Persistence$ API sastoji se od sljedećih područja:

- Jakarta Persistence API
- Jezik za postavljanje upita JPQL (engl. $query\ language$)
- Metapodaci objektno-orjentiranog mapiranja

Specifikacija JPA je zapravo Javino sučelje koje opisuje upravljanje relacijskim podacima u aplikaciji korištenjem $Java\ Platform$, $Standard\ Edition$ i $Java\ Platform, Enterprise\ Edition/JakartaEE$.[16], [17]

$Hibernate$ ORM (engl. $Object$ - $Relational\ Mapping$) je alat za programski jezik Java. Alat $Hibernate$ je implementacija JPA API specifikacija koja omogucuje mapiranje objektno-orijentiranih modela u relacije baze podataka.[18]

Glavna uloga alata $Hibernate$ je mapiranje Javinih klasa u relacijske tablice SQL baze podataka. Dodatno $Hibernate$ pruža olakšano postavljanje upita nad bazom podataka. Pristupi bazi podataka se ostvaruju pozivom funkcija $find()$, $find()$, $save()$, $delete()$ ili $count()$. Modifikacijom tih naredbi se ostvaruje željeni upit nad bazom podataka koji vraća rezultat ovisno o vrsti upita.[18], [19]Primjer korištenja alata $Hibernate$ je vidljiv u isječku koda 10.

```
public interface ImageRepository extends
 ↪  JpaRepository<Image, Long> {

        Image findByIdImage(Long idImage);

        Long deleteByIdImage(Long idImage);

        List<Image> findByUserEmail(String Email);

}
```

Listing 10: Sučelje za realizaciju upita nad bazu podataka korištenjem $Hibernate$ ORM-a

5.4.3. Radni okvir *Spring security*

$SpringSecurity$ je moćan i prilagodljiv radni okvir za autentifikaciju i kontrolu pristupa. Postao je standard aplikacija koje se temelje na Spring programskom okviru. Ovaj radni okvir pruža autentifikaciju, autorizaciju te neke dodatne opcije

sigurnosti.[17]

5.4.4. Pokretanje Pythonovog koda na poslužitelju

Za očitanje rukom pisanog teksta iz slike potrebno je pokrenuti pythonovu skriptu iz programa koji je pisan u programskom jeziku Java. Taj problem se rješava primjenom Javine klase *Process* kao što je prikazano isječkom koda listing 11. Pozivom metode *getRuntime*() dobivamo izvršno okruženje dodijeljenog procesa. U tom izvršnom okruženju možemo lagano pokrenuti pythonovu skriptu za očitavanje rukom pisanog teksta.

```
@Override
public String uploadImage(MultipartFile image, User user)
 ↪  throws IOException {

        String fileName =
         ↪  StringUtils.cleanPath(image.getOriginalFilename());

        Image newImage = new Image();
        newImage.setContent(image.getBytes());
        newImage.setName(fileName);
        newImage.setUser(user);

        Image savedImage =
         ↪  imageRepository.save(newImage);

        String imagePath = "C:/Users/38591/Desktop/" +
         ↪  savedImage.getIdImage() + "." +
         ↪  fileName.substring(fileName.lastIndexOf('.')
         ↪  + 1);

        // save image on disk
        FileOutputStream fos = new
         ↪  FileOutputStream(imagePath);

        try {
            fos.write(image.getBytes());
```

```
} catch(Exception e ) {
        e.printStackTrace();
}
finally {
    fos.close();
}

StringBuilder sb = new StringBuilder();
// call python script with image path as argument
try {

        String s = null;
        Process process =
         ↪  Runtime.getRuntime().exec("Python
         ↪  C:/Users/38591/Desktop/model.py " +
         ↪  imagePath);
        BufferedReader stdInput = new
         ↪  BufferedReader(new
         ↪  InputStreamReader(process.getInputStream()));
        BufferedReader stdError = new
         ↪  BufferedReader(new
         ↪  InputStreamReader(process.getErrorStream()));

        while((s = stdInput.readLine()) != null)
         ↪  {
                sb.append(s);
        }

        while((s = stdError.readLine()) != null)
         ↪  {
                sb.append(s);
        }

} catch(IOException e) {
        e.printStackTrace();
}
```

```
        return sb.toString();
}
```

Listing 11: Programski kod za pokretanje očitavanje teksta iz slike

5.5. Baza podataka

Za implementaciju baze podataka odabrao sam sustav $PostrgeSQL$ i $PgAdmin$. U ranoj fazi razvoja aplikacije sam koristio H2 "in-memory" SQL bazu podataka. U nastavku ću prikazati korištene relacije podataka.

User		
name	VARCHAR	Ime korisnika aplikacije.
surname	VARCHAR	Prezime korisnika aplikacije.
email	VARCHAR	Račun elektroničke pošte korisnika aplikacije.
password	VARCHAR	Lozinka kojom korisnik pristupa računu.
role	INT	Broj koji označava ulogu (0 - ADMIN, 1 - USER).

Tablica 5.1: User

Relacija $User$ (tablica 5.1) sadrži sve korisnike aplikacije koji su registrirani u sustavi i njihove osobne podatke.

Admin		
name	VARCHAR	Ime admina aplikacije.
surname	VARCHAR	Prezime admina aplikacije..
email	VARCHAR	Račun elektroničke pošte admina aplikacije.
password	VARCHAR	Lozinka kojom zaposlenik pristupa računu.
role	INT	Broj koji označava ulogu (0 - ADMIN, 1 - USER).

Tablica 5.2: Admin

Relacija $Admin$ (tablica 5.2) sadrži sve osobne podatke o adminima aplikacije.

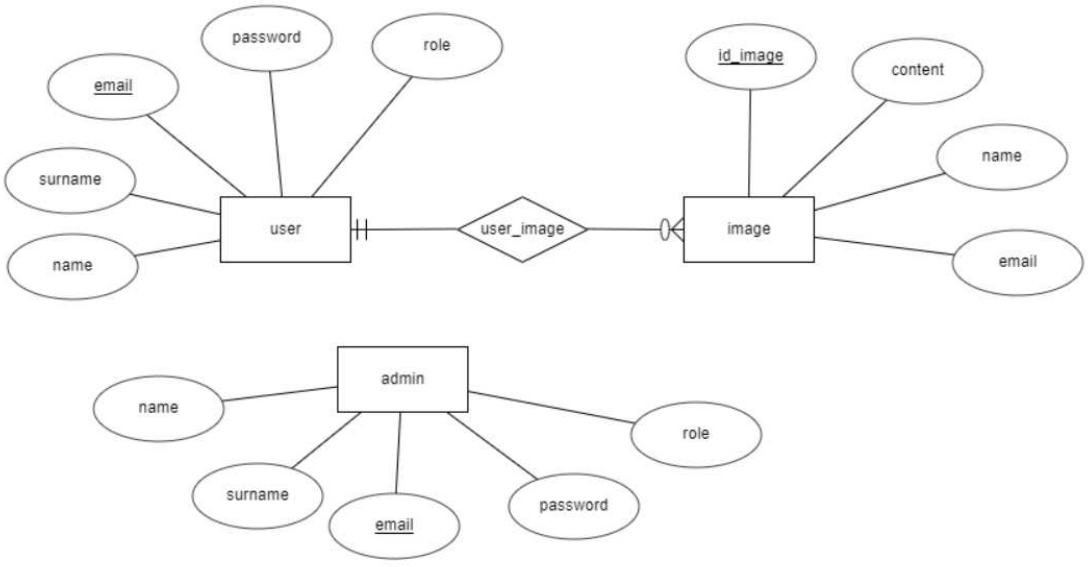
Image		
id_image	LONG	Jedinstveni identifikator slike.
name	VARCHAR	Naziv slike.
content	BYTE[]	Zapis slike u oktetima.
email	VARCHAR	Račun elektroničke pošte korisnika kojemu pripada slika.

Tablica 5.3: Image

Relacija $Image$ (tablica 5.3) sadrži sve slike koji su korisnici poslali aplikaciji na obradu te podatke o korisniku čija je slika.

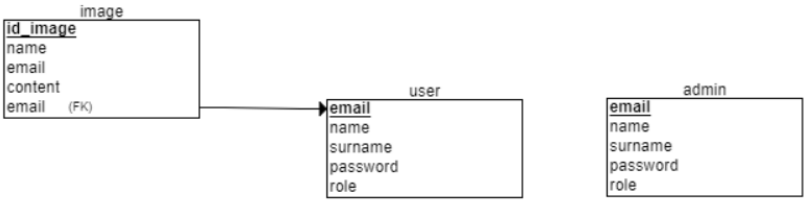
5.5.1. Dijagram baze podataka

ER dijagram baze podataka prikazan slikom 5.8 prikazuje modele (objekte) sustava te njihovu međuovisnost.



Slika 5.8: ER dijagram baze podataka

Korištenjem web servisa $ERDplus$ se iz ER dijagrama lagano generira relacijska shema baze podataka. Slika 5.9 prikazuje relacijski model baze podataka naše aplikacije.



Slika 5.9: Relacijska shema baze podataka

6. Zaključak

U području računalnog vida duboke neuronske mreže daju rezultate koji su među najboljima ikada postignuti. U ovom radu je prikazan jedan od načina primjene te tehnologije.

Cilj rada je očitavanje rukom pisanog teksta iz slike. Kroz ovaj rad smo prošli kroz izradu modela duboke neuronske mreže koja je po arhitekturi konvolucijska povratna neuronska mreža. Model sam zaokružio programskom podrškom koja je lagana za upotrebu. Provedeno testiranje modela daje pogrešku sustava od 10-15%, što je za primjenu u komercijalne svrhe previše. Danas najkorišteniji OCR sustavi kao što su $ABBY\ FineReader$ ili $Adobe\ Acrobat$ garantiraju točnostima očitavanja od 99% do 99.8%.[22]

Model bi se mogao poboljšati promjenom parametara ili brojem neurona u pojedinom sloju. Veliku ulogu u točnosti modela ima kvaliteta slike koju mu predajemo, te bi trebalo nadograditi model s efikasnijim algoritmom obrade slika na ulazu.

U primjeni dubokih neuronskih mreža vidim obećavajuću budućnost.

7. Literatura

[1] Optical character recognition, 27.03.2021., *Optical character recognition*, https://en.wikipedia.org/wiki/Optical_character_recognition, (25.05.2021.)

[2] Eikvil, L., Optical character recognition: Introduction to OCR. 1. izdanje. 01.12.1993. (26.05.2021)

[3] Eikvil, L., Optical character recognition: Optical Character Recognition. 1. izdanje, 1993. (26.05.2021)

[4] What is a Neural Network?, 07.03.2016., *Neural Network*, https://deepai.org/machine-learning-glossary-and-terms/neural-network, (28.05.2021)

[5] Géron, A. Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow. USA: O'Reilly, 2. izdanje, 2019. (20.06.2021)

[6] Andy Turner, Artificial Neural Networks, 05.06.2015., *Artificial Neural Networks*, http://andrewjamesturner.co.uk/ArtificialNeuralNetworks.php (30.06.2021)

[7] Buduma, N. Fundamentals of Deep Learning Designing Next-Generation Machine Intelligence Algorithm. USA: O'Reilly, 1. izdanje, 2017. (30.05.2021)

[8] Feedforward neural network, 07.05.2021., *Feedforward neural network*, https://en.wikipedia.org/wiki/Feedforward_neural_network, (26.05.2021)

[9] Bruno B., Artificial neural networks, 23.05.2019., *Artificial neural networks*, https://cinnamon.agency/blog/post/artificial_neural_networks, (26.05.2021)

[10] Arden Dertat, Applied Deep Learning - Part 4: Convolutional Neural Networks, 02.12.2017., *Applied Deep Learning - Part 4: Convolutional Neural Networks*, https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1 (21.05.2021)

[11] Michael Phi, Illustrated Guide to LSTM's and GRU's: A step by step explanation, 24.09.2018., *Illustrated Guide to LSTM's and GRU's: A step by step explanation*, https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9e (22.05.2021)

[12] Vishwas Saini, Handwriting Recognition, 5.08.2020., *Handwriting Recognition*, https://www.kaggle.com/landlord/handwriting-recognition, (10.05.2021)

[13] HandWritten_Character, 17.09.2018., *HandWritten_Character*, https://www.kaggle.com/vaibhao/handwritten-characters (10.05.2021)

[14] Jebastin Nadar, Handwriting Recognition using CRNN in Keras, 25.08.2020., *Handwriting Recognition using CRNN in Keras*, https://www.kaggle.com/samfc10/handwriting-recognition-using-crnn-in-keras (04.05.2021)

[15] Spring Data JPA, https://spring.io/projects/spring-data-jpa, (03.06.2021)

[16] Jakarta_Persistence, 03.06.2021., *Jakarta_Persistence*, https://handwiki.org/wiki/Jakarta_Persistence, (02.06.2021)

[17] Spring Security, https://spring.io/projects/spring-security, (03.06.2021)

[18] Hibernate ORM, https://hibernate.org/orm/ (01.06.2021)

[19] Hrvoje Šimić, Backend - prvi dio, 30.10.2019., *Backend - prvi dio*, https://drive.google.com/file/d/1T6OH9YAMkwjYOxgIRbm4yRxIPXkWb5H8/view (01.06.2021)

[20] Building a RESTful Web Service, https://spring.io/guides/gs/rest-service/ (03.06.2021.)

[21] Iryna Sydorenko, OCR with Deep Learning: The Curious Machine Learning Case, 05.02.2021., *OCR with Deep Learning: The Curious Machine Learning Case*, https://labelyourdata.com/articles/ocr-with-deep-learning/ (22.05.2021.)

[22] Adrain Try, ABBYY FineReader Review, 24.03.2021., *ABBYY FineReader Review* https://www.softwarehow.com/abbyy-finereader-review/ (10.06.2021)

POPIS SLIKA

POPIS TABLICA

Mobilna aplikacija za očitavanje rukom pisanog teksta korištenjem dubokih neuronskih mreža

Sažetak

Duboke neuronske mreže i njihova primjena su sve zastupljenije u rješavanju problema računalnog vida zahvaljujući svojim visoko rangiranim rezultatima.

U ovom radu najprije opisujemo pojam neuronske mreže te neophodne segmente za učenje modela, a to su: podaci, mjere uspješnosti i arhitektura modela. U nastavku je prikazan programske podrške koja olakšava korištenje modela.

Ključne riječi: Neuron, duboka neuronska mreža, povratna neuronska mreža, LSTM, konvolucijska neuronska mreža, optičko prepoznavanje znakova, OCR, strojno učenje, duboko učenje

Mobile application for reading handwritten text using deep neural networks

Abstract

Deep neural networks and their application are increasingly represented in solving computer vision problems thanks to their highly ranked results.

In this paper, we first describe the notion of a neural network and the necessary ones segments for model learning, namely: data, performance measures, and model architecture. Below is software that makes it easy to use the model.

Keywords: Neuron, deep neural network, recurrent neural network, LSTM, recurrent neural network, convolutional neural network, optical character recognition, OCR, machine learning, deep learning