Univerzitet u Sarajevu
Elektrotehnički fakultet
Odsjek za računarstvo i informatiku

Projekat

Handwriting text recognition

Metode i primjena vještačke inteligencije

Profesor:

V. prof. dr. Amila Akagić dipl.ing.el.

Studenti:

Zehra Javdan
Nejra Rovčanin
Ema Rudalija

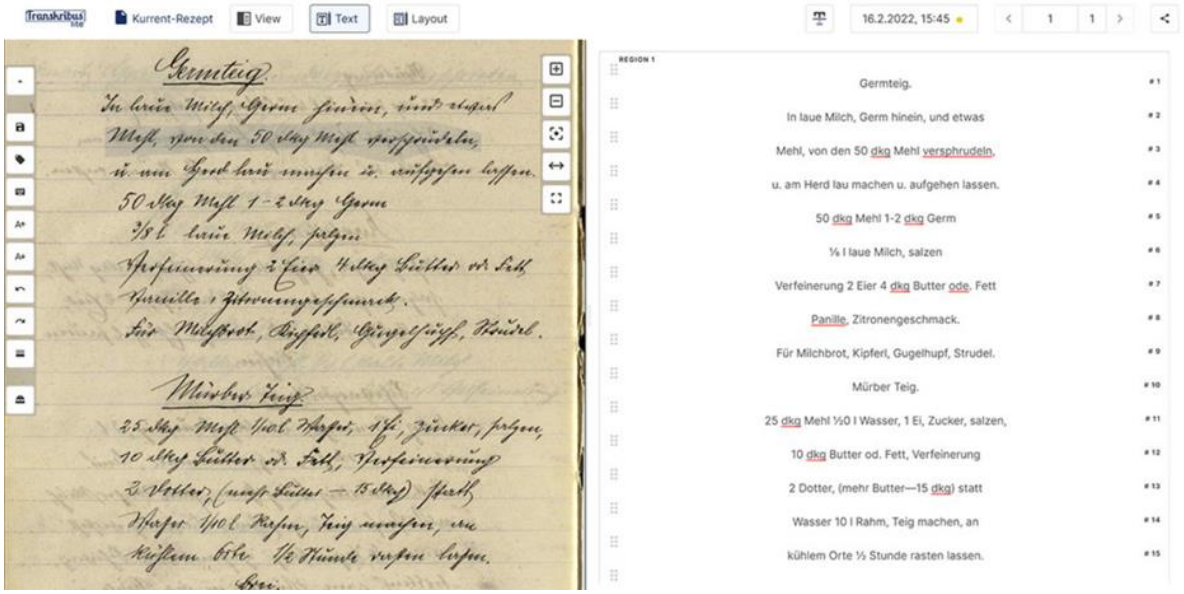
Sarajevo, oktobar 2022.

Sadržaj

1. Opis problema

Handwriting recognition (HWR) ili prepoznavanje rukopisa je sposobnost računara da primi i protumači razumljiv unos nekog rukopisa kao ulaz. Primjeri takvih ulaza mogu biti različiti papirni dokumenti, fotografije i drugi. HWR se može realizovati optičkim skeniranjem, tj. optičkim prepoznavanjem znakova ili inteligentnim prepoznavanjem riječi. Sistem koji je uspostavljen za rad sa pisanim tekstom upravlja formatiranjem, vrši ispravnu segmentaciju u znakove i pronalazi riječi sa velikim procentom vjerovatnoće. Također, sistem bi trebao prepoznavati i druge ručno pisane simbole.

HWR je tehnologija čiji su osnovni koncepti prisutni u svijetu tehnologije još od 1980-ih, ali je njena stvarna primjena i način realizacije došao do izražaja u posljednjih nekoliko godina. Cilj tehnologije je omogućiti digitalizaciju pisanog teksta prepoznavanjem rukopisa. Kao napredni nivo ovog koncepta, također su zamišljeni i uređaji koji mogu predvidjeti šta osoba želi da napiše.



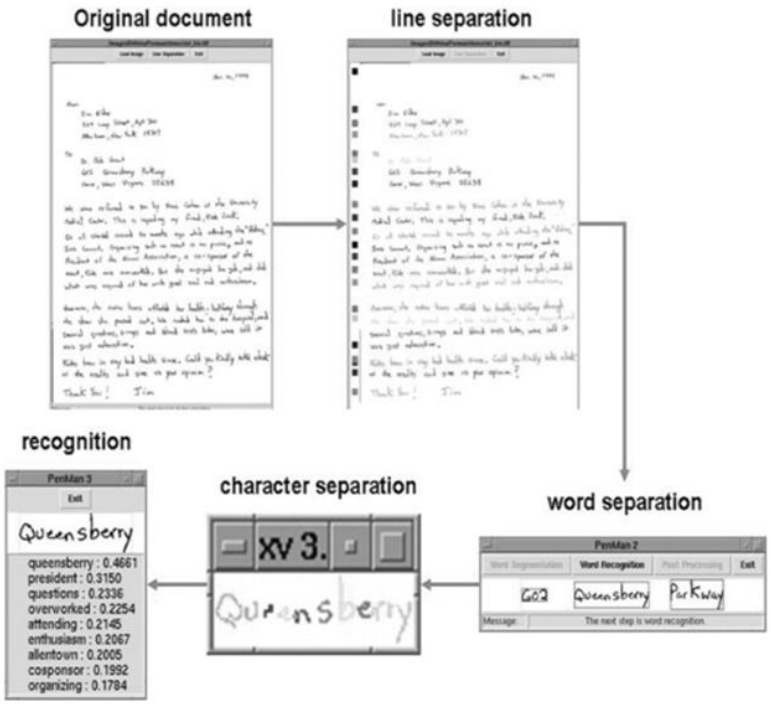
Ilustracija 1: Primjer digitalizacije ručno pisanog teksta

Postoje dvije vrste tehnika za prepoznavanje rukopisa:

- Optičko prepoznavanje znakova (*Optical Character Recognition – OCR*)
- Online prepoznavanje

Optičko ili offline prepoznavanje znakova u osnovi uključuje automatsko pretvaranje teksta sa slike u slova. Kasnije se ti zapisi mogu koristiti u aplikacijama za obradu teksta. Ovakav način prepoznavanja pisanog teksta često zna biti jako izazovan, budući da ljudi imaju različite stilove pisanja. Jedna od mogućih primjena OCR-a je digitalizacija ručno pisanih historijskih dokumenata, što bi omogućilo njihovu lakšu pretragu i sortiranje. Također, digitalizacija dokumenata i njihovo pohranjivanje u računar je danas mnogo sigurnija opcija kod koje su šanse za gubitak smanjene.

Koraci optičkog prepoznavanja su prikazani na slici ispod.



Ilustracija 2: Koraci OCR-a

Koraci sa slike predstavljaju tradicionalne tehnike HWR-a koje se fokusiraju na segmentiranje pojedinačnih znakova. Međutim, modernim tehnikama je moguće prepoznati sve znakove u nekoj segmentiranoj liniji teksta. Ovdje se misli na tehnike mašinskog učenja uz pomoću kojeg računar uči vizuelne karakteristike grafije, izbjegavajući analiziranje znaka po znak. Najsavremenije metode koriste konvolucijske mreže za izdvajanje vizuelnih karakteristika koje kasnije koriste neuronske mreže za izračunavanje vjerovatnoće pojavljivanja određenog znaka. U suštini, ovo je metod koji će se koristiti na našem projektnom zadatku.

Online prepoznavanje rukopisa predstavlja automatsko pretvaranje teksta koji je napisan na posebnom digitalizatoru (npr. tablet ili mobilni uređaj) gdje senzor bilježi pokrete vrha olovke i na taj način vrši konverziju. Dobijeni signal se pretvara u slova, što se kasnije može koristiti u aplikacijama za obradu teksta. Obavezni elementi svakog interfejsa za online prepoznavanje teksta su: olovka kojom korisnik može pisati, uređaj osjetljiv na dodir i softverska aplikacija koja interpretira pokrete olovke po površini za pisanje, prevodeći ih u digitalni tekst.

Od mogućih primjena softvera za prepoznavanje rukopisa važno je izdvojiti sljedeće:

- **Skladištenje podataka**

  Osnovna primjena HWR-a je skladištenje podataka. Kako je ranije naglašeno, pisani dokumenti se digitalizuju i pohranjuju u računar. Prednost ovakvog elektronskog

pohranjivanja je u tome što je potrebno mnogo manje fizičkog prostora za skladištenje. Također, potrebno je i manje zaposlenih koji bi bili zaduženi za sortiranje dokumenata, organiziranje i čuvanje podataka. Ovakav pristup je jako značajan kada su u pitanju npr. zdravstvene institucije i način čuvanja medicinske dokumentacije pacijenata.

- **Historijski dokumenti**

  Ranije spomenuti historijski dokumenti su još jedan razlog za primjenu i razvoj HWR-a. Historijski dokumenti u većini slučajeva postoje isključivo u fizičkom obliku. Radovi poput pisanih rukopisa, starih porodičnih zapisa, ličnih dnevnika ili državnih dokumenata često znaju biti oštećeni, a softver za prepoznavanje rukopisa je u tom slučaju od velike pomoći. Na taj način se historijske činjenice i zanimljivosti mogu veoma lako pohraniti, pregledati i podijeliti sa više ljudi.

- **Tekstologija**

  Tekstologija je jedna od oblasti književnosti u kojoj se izučava originalni tekst nekog djela u cilju utvrđivanja njegove historije i autentičnosti. U suštini, da bi se ostvarili zadani ciljevi ove nauke, potrebno je koristiti izvorno napisani tekst nekog djela (ukoliko isti postoji), a zatim izvršiti njegovu usporedbu sa kasnijim prepisima ili izdanjima. Zbog toga se javlja potreba da se ovakva djela dobro čuvaju u svrhe sadašnjih i budućih istraživanja. HWR pomaže u očuvanju ovih rukopisa u elektronskom formatu i otvara mogućnost pregleda za širu masu ljudi bez rizika oštećenja originalnog djela.

Najveći izazov kod realizacije HWR-a je, kao što je ranije navedeno, činjenica da svačiji rukopis nije isti. Ovo pokreće problem u procesu prepoznavanja znakova u slučajevima kada je potrebno pretvoriti nečiji rukopis u digitalni oblik. Također, potencijalni problem historijskih dokumenata je njihova oštećenost i nečitkost.

2. Analiza trenutnog stanja u oblasti

Rad 1: A Review of Various Handwriting Recognition Methods

Naziv rada: *A Review of Various Handwriting Recognition Methods*

Autori: *Salma Shofia Rosyda, Tito Waluyo Purboyo*

Datum: *Januar, 2018*

U ovom radu opisane su neke od metoda koje se aktivno primjenjuju u prepoznavanju rukopisa. Obrađeno je ukupno osam metoda: konvolucijska neuronska mreža, inkrementalni metod, poluinkrementalni metod, segmentacija linija i riječi, segmentacija zasnovana na dijelu, segmentacija nagiba, ansambl i dijeljenje slike u zone. U nastavku je dat sažetak četiri metode, kao i njihove prednosti i nedostaci.

1. **Konvolucijska neuronska mreža (CNN – Convolutional Neural Network)**

CNN je opisana kao jedna od najkorištenijih metoda za prepoznavanje rukopisa. Detaljno su opisani slojevi jedne ovakve neuronske mreže (convolutional, pooling, fully-connected i softmax) kao i njihov značaj u treniranju modela. Kao faze CNN-metode za klasifikaciju i prepoznavanje navedeni su: preprocesiranje slika, pronalazak ili izrada dataset-a, treniranje, klasifikacija i testiranje.

Prednosti: Prepoznavanje slike ima visoku tačnost kada je CNN istrenirana; također je pokazano da je uspješna u problemima prepoznavanja, između ostalog i HWR.

Nedostaci: U procesu treniranja je potrebno koristiti veliku količinu podataka da bi se postigla željena tačnost, što dovodi do problema predugog treniranja.

2. **Segmentacija linija i riječi (Line&Word Segmentation)**

Ova metoda objašnjava kako se izdvajaju linije i riječi iz ručno napisanog teksta. Prije samog izdvajanja, za ulaznu sliku je prvo potrebno izvršiti procese uklanjanja šuma, binarizacije, detekciju i korekciju iskrivljenosti i stanjivanje. U fazi segmentacije na nivou linije radi se horizontalna i vertikalna analiza dokumenta, kao i detekcija vrhova i udubljenja. Na kraju, da bi se izdvojila riječ, koriste se procesi dilatacije i optimizacije prethodno segmentirane linije.

Prednosti: Pristup je dobar prilikom rada sa štampanim dokumentima.

Nedostaci: Nije najbolji u otkrivanju obrazaca.

3. Ansambl (Ensemble)

Osnovni koncept metode je generisanje više klasifikatora iz jedne bazne klase, a klasifikator u svrhu prepoznavanja bira jednu karakteristiku. Svrha ovakvog pristupa je unapređenje performansi predviđanja statičkih tehnika učenja.

Prednosti: Metoda ima visoku tačnost.

Nedostaci: Komponente segmentacije linija utiču na tačnost.

4. Dijeljenje slike u zone (Zoning)

Ideja ove metode je da se ulazna slika podijeli u zone koje će analizirati dijelove slike pojedinačno. Podjela slike u zone može biti statička (zone su jednake veličine, pri čemu više zona znači i veću tačnost) i dinamička (zone mogu biti različite veličine). Koraci ove metode su kreiranje baze podataka, preprocesiranje, ekstrakcija značajki kroz dijeljenje slike u zone i klasifikacija.

Prednosti: Metoda ima visoku tačnost.

Nedostaci: Da bi se postigla visoka tačnost, potrebno je podijeliti sliku u puno manjih zona.

Na kraju rada, kroz zaključak je objašnjeno da se CNN metoda pokazala najuspješnijom za rad sa rukopisom. Iako se treniranje mreže nekad izvršava dugo, ova metoda je pokazala najbolju tačnost, odnosno najbolje prepoznavanje rukopisa i pretvaranje istog u digitalni oblik.

Rad 2: Handwriting Recognition of Historical Documents with few labeled data

Naziv rada: *Handwriting Recognition of Historical Documents with few labeled data*

Autori: *Edgard Chammas, Chafic Mokbel, Laurence Likforman-Sulem*

Datum: *April, 2018*

U ovom radu obrađena je problematika prepoznavanja rukopisa historijskih dokumenata. Slike takvih dokumenata su često okarakterisane slabom rezolucijom sa narušenim kvalitetom. Redovi su nerijetko preklopljeni i prepisani znakovima iz prethodnog reda. Zbog toga, značajan broj slika nije označen na nivou linije. Izazov je segmentirati linije, a zatim izvršiti proces treniranja HWR sistema.

U svrhu treniranja podataka korišten je READ2017 dataset koji sadrži 130 000 riječi i 10 000 dostupnih stranica teksta. Korištena je metoda rekurentne konvolucijske neuronske mreže (*CRNN – Convolutional Recurrent Neural Network*).

Za CRNN sistem korišteno je sljedeće:

- 13 konvolucijskih slojeva nakon kojih slijede 3 dvosmjerna LSTM sloja sa 256 jedinica po sloju
- Spacial pooling
- ReLU aktivacijska funkcija
- Klizni prozor (window) kvadratnog oblika za potrebe skeniranja linija teksta u smjeru pisanja
- Back propagacija
- Težinska matrica inicijalizirana uniformnom distribucijom $W_{i,j} \sim U(-\sqrt{6}/n\,, \sqrt{6}/n)$, gdje je n ukupan broj ulaznih i izlaznih neurona na sloju
- Adam optimizator sa stopom učenja 0.001
- Batch normalizacija
- CTC funkcija gubitka

Za ekstrakciju slika linija teksta korišten je automatski segmentacijski algoritam. Prvo treniranje je izvršeno nad 10% podataka, a LER (*Laber Error Rate*) iznosio je 9.2%.

U narednom treniranju automatski segmentacijski algoritam je poboljšan Seam Carving tehnikom i dinamičkim programiranjem budući da su neke slike linija iz prve iteracije bile rezultat pogrešne segmentacije jer je početni algoritam bio osjetljiv na ugao pisanja. Nakon toga, uzeto je 90% podataka i ponovno izvršeno treniranje. LER je ovaj put iznosio 7.4%.

U sljedećem koraku je realizovana multi-scale integracija podataka na način da su slike linija razvrstane u tri klase – large, medium i small. S obzirom da dijeljenjem trening skupa u klase količina podataka po klasi biva manja, trening skup svake klase je proširen dodavanjem sintetičkih podataka koji su rezultat skaliranja podataka ostalih klasa. Rezultati ponovnog treniranja su dali LER od 6.5%.

Na kraju, upotrebom model-bazirane normalizacije postignuti su još bolji rezultati, gdje je LER pao na 5.5%.

Na slici ispod se mogu vidjeti rezultati testiranja sva tri modela kroz prikaz procenta grešaka za WER (*Word Error Rate*) i CER (*Character Error Rate*).

System	CER	WER
CRNN (1)	9.18%	25.07%
CRNN retrained with multi-scale data (2)	7.95%	23.09%
(2) + model-based normalization scheme	7.74%	21.58%

Rad ovakvog sistema, odnosno njegov učinak, zavisi od dva faktora: segmentacije i samog procesa prepoznavanja. Vrijedi napomenuti da bi se postignuti rezultati mogli još poboljšati korištenjem učinkovitije segmentacije koja bi omogućila upotrebu više podataka za treniranje. Uprkos složenoj arhitekturi neuralne mreže, postoji veliki uticaj varijabilnosti u rukopisima, što se može negativno odraziti na rezultate.

Rad 3: Arabic handwriting recognition system using convolutional neural network

Naziv rada: *Arabic handwriting recognition system using convolutional neural network*

Autori: *Najwa Altwaijry, Isra Al-Turaiki*

Datum: *Oktobar, 2019*

U ovom radu je predstavljen Hidža1 skup podataka rukopisnih arapskih slova. Hidža je besplatam javno dostupan dataset pojedinačnih slova, prikupljenih sa arapskog govornog područja. U kreiranju skupa je učestvovao 591 učenik između 7 i 12 godina i ukupno su predstavljena 47.434 karaktera. Dataset je analiziran u svrhu razvoja obrazovnih aplikacija za djecu, kao i u svrhu naglašavanja različitosti u analizi rukopisa odraslih osoba i mlađe djece. Opisan je model za vanmrežno prepoznavanje arapskog rukopisa koristeći duboke neuronske mreže (CNN).

Rad je koncipiran u nekoliko dijelova u kojima se govori o CNN-u, srodnim radovima, prijedlogu rješenja (objašnjenje modela) i performansama modela autora u usporedbi s drugim modelima spomenutim u literaturi.

Uspoređuju se performanse modela CNN-for-AHCD na oba skupa podataka, Hidža dataset i AHCD dataset. Nakon dobivenih rezultata donosi se zaključak da je modelu bilo teže obraditi Hidža dataset, ali i da su rezultati za Hidža dataset obećavajući.

Rad 4: Offline Persian Handwriting Recognition with CNN and RNN-CTC

Naziv rada: *Offline Persian Handwriting Recognition with CNN and RNN-CTC*

Autori: *Vahid Mohammadi Safarzadeh, Pourya Jafarzadeh*

Datum: *Januar, 2020*

U ovom radu je predstavljena metoda prepoznavanja rukopisnih perzijskih riječi koja se sastoji od CNN-a za generiranje niza karakteristika iz slike ulazne riječi, nakon čega slijede BLSTM slojevi sa CTC funkcijom. Rad prezentuje model prepoznavanja zasnovan na metodi označavanja sekvenci s dubokim konvolucijskim neuronskim mrežama (CNN) i rekurentnim neuronskim mrežama (RNN).

U radu je naglašeno da perzijsko i arapsko pismo imaju kurzivnu prirodu, što znači da su pojedini znakovi u riječi međusobno povezani, za razliku od engleskog pisma u kojem su znakovi u riječi nepovezani. Konvencione metode dijele svoje algoritme u dvije faze: segmentaciju i prepoznavanje. U ostatku rada su opisana dva navedena postupka. Dolazi se do zaključka da metode imaju prihvatljive performanse u prepoznavanju riječi, ali jedan od nedostataka je taj što je segmentacija dugotrajan postupak.

Neki od navedenih problema koji usporuju proces segmentacije su sljedeći:

- Nekada se znakovi preklapaju okomito
- Mnoga slova imaju različite oblike u različitim dijelovima riječi
- Segmentirani dijelovi slova mogu imati slične oblike drugim znakovima
- Zbog kurzivne prirode i neograničenog stila pisanja, algoritam može proizvesti dodatne segmente u znaku

Nakon segmentacije slijedi faza prepoznavanja u kojoj se identifikuju segmentni dijelovi.

U ostatku rada su obješnjene korištene metode kao i korišteni dataset-ovi, detaljnije je prezentirano rješenje kao i implementacioni dio.

U drugom dijelu rada su detaljnije opisani sljedeći pojmovi:

- *Convolutional Neural Networks (CNN)*
- *Bidirectional Long-Short Term Memory (BLTSM)*
- *Connectionist Temporal Classification (CTC)*
- *Network Architecture*

Dataset-ovi koji su opisani:

- *AHDBase*
- *Hoda*
- *IFN/ENIT*

U zaključku je navedeno da je u radu predložen model koji koristi prednosti i CNN-a i RNN-a u svrhu prepoznavanja riječi. Metoda također koristi prednosti CTC-a za eliminaciju postupka segmentacije. Naglašeno je da je metoda pokazala zadovoljavajuće performanse u poređenju sa drugim najsavremenijim metodama.

Rad 5: Evolutionary convolutional neural networks: An application to handwriting recognition

Naziv rada: *Evolutionary convolutional neural networks: An application to handwriting recognition*

Autori: *Alejandro Baldominos ∗ , Yago Saez, Pedro Isasi*

Datum: *2017*

U radu je razrađena nova metoda mašinskog učenja - **automatizirana neuroevolucija konvolucijskih neuralnih mreža**. Primijenjena, trenirana i evaluirana na MINSTovom datasetu za prepoznavanje ljudskog rukopisa. MINSTov dataset se sačinjavao iz 60 000 trening i 10 000 testnih slika koje su sadržavale ljudski rukopis, u sivim tonovima, pri čemu su podaci prikupljeni iz popisa stanovništva i učenika iz srednje škole. Izvorne slike su bile formata 20 x 20 piksela.

Značenje neuroevolucije se može opisati kao primjena Darwinovskog pogleda na evoluciju čovjeka primjenjen na treniranje konvolucionih neuralnih mreža. Postupak u suštini teži tome da se ne dobije gotova neuronska mreža, već da se izgradi procesom evolucije i opstankom najoptimalnijih jedinica. Konkretni koraci 'algoritma' su:

- nasumično generiramo početnu populaciju
- izvadimo strukturu podataka koja se naziva 'hromosom'
- a sastoji se od osobina ili atributa koje zovemo 'genotipom'.
- Zatim genotip prevodimo u fenotip treniranjem CNN-a,
- odredimo težinske ocjene svakom fenotipu po tome kako zadovoljava kriterije
- mrežu treniramo kroz 5 epoha koristeći 50% podataka
- računamo grešku svakog od fenotipa i evaluiramo ih
- slijed evolucijskih operacija (npr. selekcija, križanje, mutacija) koje se oslanjaju na prikladnost pojedinaca za opstanak najjačih, odnosno dobivamo novu generaciju.

U ovom radu su upotrjebljene dvije različite tehnike evoucionih konvolucionih neuralnih mreža: genetske algoritme i gramatičku evoluciju. Treniranjem i evaluiranjem mreže, došli smo do zaključka da je bolja meotda GE, metoda gramatičke evolucije, jer je dopuštena veća fleksibilnost fenotipova i smanjena redundantnost kodiranja.

Kako smo rekli da je naša CNN evoluirala, kroz treniranje MINST data seta, došla je do dimenzija od četiri konvolucijska sloja s maksimalnim udruživanjem i malom rekurentnom mrežom s tri skrivene jedinice.

Rad 6: Handwriting Recognition using Artificial Intelligence Neural Network and Image Processing

Naziv rada: *Handwriting Recognition using Artificial Intelligence Neural Network and Image Processing*

Autori: *Sara Aqab , Muhammad Usman Tariq*

Datum: *2020*

U ovom radu, razrađene su različite metode koje se koriste za prepoznavanje rukopisa: neuronske mreže, skriveni Markovljev model (HMM), mašinsko učenje, i Support Vector Machine (SVM), kao i razrada OCR modela, i način na koji ga testiramo. Faze OCR modela vidimo ispod.

- **Prikupljanje slika i digitalizacija -** Uključuje dobivanje slikau određenim formatima kao što su PNG i JPEG dobivene putem digitalne kamere, skenera i sl.Korak digitalizacije, uključuje pretvaranje ulaznog papira u elektronički format putem skeniranja
- **Predprocesiranje -** Digitalizirana slika se obrađuje kako bi se uklonile smetnje i nepravilnosti. Neki od tehnika poboljšanja slike su
    - **Uklanjanje buke**: Zvukovi rayličitih vrsta mogu da kontaminiraju slike, radi čeg ih uklanjamo i postižemo bolju kvalitetu slike
    - **Binarizacija:** Transformiše sliku iz sivih tonova u crno-bijele tonove, znatno smanjujući kvantitet i raznovrsnosti informacija sadržanih unutar slie sa sivim tonovima, i pretvaraući je u binarnu sliku.
    - **Normalizacija**: Proces u obradi slike koji mijenja raspon vrijednosti intenziteta piksela u standardnu veličinu.
    - **Ispravljanje iskošenja, stanjivanje**: Transformiše sliku u širinu 1 piksela za lakše prepoznavanje karakter
- **Segmentacija -** najzahtjevniji proces, izvodi se samo u fazi testiranja, cilj je pojednostaviti prikaz slike. Postupak uključuje odvajanje pojedinačnih znakova iz slike, što rezultira u višestrukim segmentima slika poznata kao super pikseli.
- **Ekstrakcija značajki -** U ovoj fazi izdvajaju se karakteristike slike: visina, znak, broj horizontalnih linija, širina znaka, broj krugova, piksela, položaj različitih znakova i broj okomito orijentiranih lukova.
- **Prepoznavanje -** U ovoj fazi za klasifikaciju se koristi neuronska mreža i prepoznavanje likova sa slike.

Testiranje OCR modela se može provesti na tri načina:

- **Jedinično testiranje -** pri čemu pojedinačno testiramo kod svake od faza OCR modela

- **Integracijsko testiranje -** kombinirane su i testiranje pojedine jedinice sustava kao grupa ili jedinica. Glavni cilj je bio razotkriti probleme u međusobnoj komunikaciji.
- **Testiranje valjanosti -** testiranje valjanosti kako bi se utvrdilo je li razvojni proces ispunjava specificirane zahtjeve.
- **GUI testiranje -** testiranje GUI-ja kako bi se osiguralo da sustav grafičko korisničko sučelje zadovoljava navedene specifikacije i je

3. Pregled postojećih datasetova

U ovom dijelu bit će opisani dataset-ovi koji su relevantni za problem koji se rješava, odnosno problem prepoznavanja rukopisa. Potrebno je napomenuti da, pošto se radi o problemu prepoznavanja, a ne klasifikacije, niti jedan od dolje navedenih skupova podataka nema podjelu slika na klase.

Referencirajući se na naučne radove, kao i na online resurse, pronašli smo ukupno 32 dataseta kreiranih za ovaj domen problema. U nastavku je predstavljeno sedam najinteresantnijih i najčešće korištenih.

1. Handwriting recognition set

   Set se sastoji od više od 400 000 ručno napisanih imena ljudi. Slike su podijeljene na trening skup, skup za validaciju i skup za testiranje. Jedan je od rijetkih kod kojih je pronađena specifikacija skupa za validaciju.

2. Cyrillic Handwriting Dataset

   U ovom skupu podataka nalazi se skoro 74 000 segmenata rukopisa na ćiriličnom pismu (sadržaj se odnosi na ruski jezik). Slike su prikupljene preko online izvora i materijala pojedinih članova tima koji su radili na kreiranju istog. Svaka slika predstavlja izraz na ruskom jeziku koji sadrži maksimalno 40 znakova.

3. Aida Calculus Math Handwriting Recognition

   Ovaj set podataka sastoji se od 10 batcheva od po 10 000 slika ručno napisanih matematičkih izraza. Fokus ovog dataseta je na nauci o podacima i principima mašinskog učenja za rješavanje izazovnog zadatka kompjuterske vizije gdje je primarni cilj izdvajanje i prepoznavanje matematičkih izraza sa slike.

4. Arabic Handwritten Characters Dataset

   Kreatori ovog dataseta su za cilj imali iskoristiti znanje iz više radova i studija kako bi poboljšali performanse softvera za prepoznavanje arapskih ručno napisanih znakova. Ovakav pristup je izazovan budući da je i arapsko pismo, kao i svako drugo, podložno različitim stilovima rukopisa, zbog čega je bilo potrebno kreirati veliki skup podataka.

5. Chinese Handwriting Recognition

   Sastoji se od 130 000 slika karaktera koji predstavljaju znakove i znamenke kineskog pisma. Radi se o doradi i modifikaciji CASIA Online&Offline Chinese HWR skupa podataka.

6. **Ottoman Turkish Characters**

   Ovaj dataset se sastoji od 3894 znaka osmanskog turskog pisma. Znakovi su razvrstani po kategorijama formata korištenog u Osmanskom Carstvu (Talik, Matbu, Rika i miješani karakteri). Kreiran je s ciljem da pomogne istraživačima u prevođenju starih otomanskih dokumenata.

7. **Hindi Character Recognition**

   Riječ je o Devanagari, indijskom pismu koje se koristi u Indiji i Nepalu. Pismo se sastoji od 36 znakova i 10 znamenki, a od drugih jezika se razlikuje po odsutnosti velikih slova. Dataset se sastoji od ručno pisanih simbola.

Skup podataka koji će se koristiti u svrhe realizacije našeg projekta je *Handwriting recognition set* koji je prvi naveden u gornjoj listi. Razlog ovog odabira je činjenica da pomenuti dataset ima najviše primjeraka rukopisa od svih razmatranih skupova. Također, jedini je koji ima odvojen skup slika za validaciju zbog čega smatramo da će finalni rezultati imati manju grešku.

Što se tiče potencijalnih rizika, potrebno je izdvojiti standardni problem prilikom treniranja modela za prepoznavanje rukopisa, a to su različiti stilovi i varijacije rukopisa kod ljudi koji mogu dovesti do poteškoća prilikom prepoznavanja i za čovjeka, a kamoli računar. Ovaj rizik se odnosi kako na sve datasetove, tako i na onaj koji smo odabrali.

U tabeli ispod prikazani su osnovni podaci o svakom od gore opisanih datasetova, pri čemu je žutom bojom označen skup koji će se koristiti u daljim fazama treniranja i testiranja.

URL	Br. slika	Količina podataka	Br. slika za treniranje	Br. slika za validaciju	Br. slika za testiranje
Kaggle Dataset	413 823	1 GB	331 059	41 382	41 382
Cyrillic Dataset	73 830	2 GB	70 138	0	3 692
Aida Calculus Math HWR	100 000	11 GB	85 000	0	15 000
Arabic Recognition	16 800	25 MB	13 440	0	3360

Arabic Recognition 2	70 000	53 MB	60 000	0	10 000
Chinese Recognition	130 000	235 MB	105 000	0	25 000
Ottoman Turkish Recogn.	3894	2 MB	3428	0	466
Hindi Character Recogn.	92 000	80 MB	78 200	0	13 800

4. Metode preprocesiranja podataka

U nastavku rada će biti opisan postupak preprocesiranja podataka izabranog *dataset*-a. Preprocesiranje je odrađeno u *Google Colab* alatu.

Prvenstveno su biblioteke i podaci učitane, a nakon toga su podaci pripremljeni za sami proces preprocesiranja. Izbačeni su redovi koji su u koloni IDENTITY imali NaN ili UNDEFINED vrijednost.

```
train_csv = pd.read_csv('written_name_train_v2.csv')
val_csv = pd.read_csv('written_name_validation_v2.csv')
```

```
train_csv = train_csv.dropna()
val_csv = val_csv.dropna()
```

```
train_csv = train_csv[train_csv['IDENTITY'] != 'UNREADABLE']
val_csv = val_csv[val_csv['IDENTITY'] != 'UNREADABLE']
```

Kako se u dataset-u nalaze i mala i velika slova, radi smanjenja kompleksnosti prilikom treniranja neuronske mreže sva slova su konvertovana u mala.

```
train_csv['IDENTITY'] = train_csv['IDENTITY'].str.lower()
train_csv.reset_index(inplace=True, drop=True)

val_csv['IDENTITY'] = val_csv['IDENTITY'].str.lower()
val_csv.reset_index(inplace=True, drop=True)
```

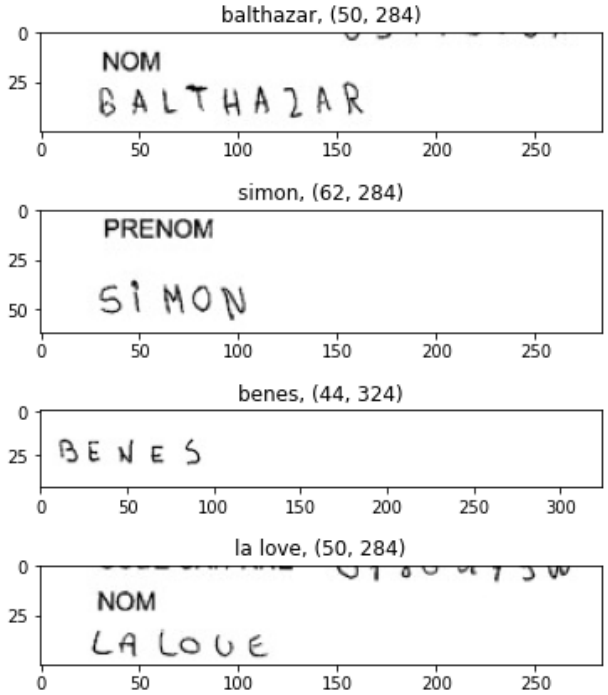
Faza preprocesiranja podataka sastoji se od dva dijela - preprocesiranje labela i preprocesiranje slika. Budući da su u labelama karakteri, tj. nenumeričke vrijednosti, iste je bilo potrebno enkodirati. Za potrebe enkodiranja korišteno je kodiranje na osnovu rječnika uz pomoć Kerasove gotove funkcije Tokenizer.

```
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences

max_length = max([len(label) for label in train_csv['IDENTITY'].values])
tokenizer = Tokenizer(num_words = max_length, char_level = True)
tokenizer.fit_on_texts(train_csv['IDENTITY'].values)

def preprocess_label(label):
  label_sequence = tokenizer.texts_to_sequences([label])
  label = pad_sequences(label_sequence, maxlen=max_length, padding='post')[0]
  return label
```

Na slici ispod jasno se vidi da su slike unutar našeg dataset-a različitih dimenzija.



U procesu preprocesiranja prvo su dimenzije fotografija stavljene na fiksnu veličinu (64x256), budući da neuronske mreže primaju ulaze iste veličine. Ovo je postignuto uz pomoć funkcije *reshape*, koja poslanoj fotografiji mijenja dimenzije i vraća je sa takvim novim dimenzijama.

```
def reshape(image):
  (h, w) = image.shape

  if h > 64:
    image = image[:64, :]

  if w > 256:
    image = image[:, :256]

  res = np.ones([64, 256])*255
  res[:h, :w] = image
  return res
```

Nakon toga je korištena funkcija za normalizaciju koja postavlja vrijednosti piksela na interval [0,1]. Ona je neophodna zbog povećanja dinamičkog raspona, dovodi raspon vrijednosti intenziteta slike u normalnu distribuciju.

```
def normalize(image):
  image = image/255.
```

```
  return image
```

Nakon definisanja svih funkcija, potrebno ih je primijeniti na labele i slike. Naš dataset se sastoji od preko 300 000 slika za treniranje, a kako je u Google Colabu ograničen prostor RAM-memorije, za potrebe treniranja je korišteno 30 000 slika. Za potrebe validacije, a kasnije i testiranja koristit ćemo po 3000 slika.

```
train_len = 30000
val_len = 3000
```

U kodu ispod je prikazana primjena funkcija za preprocesiranje na podatke za treniranje.

```
train_X = [] #Ovdje ce biti slike za treniranje
train_Y = [] #Ovdje ce biti labele za treniranje tj. IDENTITY kolona iz
train_csv
train_label_len = [] #Ovdje ce biti duzine labela
```

```
for i in range(train_len):
    img_dir = 'train_v2/train/'+train_csv.loc[i, 'FILENAME']
    image = cv2.imread(img_dir, cv2.IMREAD_GRAYSCALE)
    image = reshape(image)
    image = normalize(image)
    train_X.append(image)

    current_label = train_csv.loc[i, 'IDENTITY']
    train_label_len.append(len(current_label))
    current_label = preprocess_label(current_label)
    train_Y.append(current_label)
```

U kodu ispod je prikazana primjena funkcija za preprocesiranja na podatke za validaciju.

```
val_X = [] #Ovdje ce biti slike za validaciju
val_Y = [] #Ovdje ce biti labele za validaciju tj. IDENTITY kolona iz val_csv
val_label_len = [] #Ovdje ce biti duzine labela
```

```
for i in range(val_len):
    img_dir = 'validation_v2/validation/'+val_csv.loc[i, 'FILENAME']
    image = cv2.imread(img_dir, cv2.IMREAD_GRAYSCALE)
    image = reshape(image)
    image = rotate(image)
    image = normalize(image)
    val_X.append(image)

    current_label = val_csv.loc[i, 'IDENTITY']
```

```
    val_label_len.append(len(current_label))
    current_label = preprocess_label(current_label)
    val_Y.append(current_label)
```

5. Treniranje modela

Izgradnja našeg modela za treniranje sastoji se od upotrebe konvolucijskih i rekurentnih neuronskih mreža. Za konvolucijsku mrežu je zamišljeno da se sastoji od dva konvolucijska sloja, dva pooling sloja, jednog *dense*-sloja, kao i regularizacije koristeći funkciju *Dropout*.

Prije izgradnje neuronske mreže implementiran je i tuning hiperparametara na način kako je to objašnjeno u sklopu vježbi. U kodu ispod se jasno može vidjeti koji su hiperparametri stavljeni na ispitivanje kroz upotrebu funkcije *Choice*.

```
from kerastuner.tuners import Hyperband
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.layers import Flatten, Input, Conv2D, MaxPooling2D,
Reshape, Bidirectional, LSTM, Dense, Lambda, Activation, BatchNormalization,
Dropout

def HyperModel(hp):
  pokusaj = keras.Sequential()

  pokusaj.add(Conv2D(
      filters=hp.Choice('num_filters_1', values=[16, 32, 64], default = 32),
      kernel_size=hp.Choice('kernel_1', values=[2,3,4,5], default=3),
      strides=hp.Choice('stride1', values=[1,2,3], default=1),
      activation = 'relu',
      input_shape = (256, 64, 1)))

  pokusaj.add(MaxPooling2D(pool_size=2))

  pokusaj.add(Conv2D(
      filters=hp.Choice('num_filters_2', values=[16, 32, 64], default = 32),
      kernel_size=hp.Choice('kernel_2', values=[2,3,4,5], default=3),
      strides=hp.Choice('stride2', values=[1,2,3], default=1),
      activation = 'relu'))

  pokusaj.add(MaxPooling2D(pool_size=2))

  pokusaj.add(Dropout(rate=hp.Float('dropout_1', min_value=0.0, max_value=0.5,
default=0.3, step=0.1)))

  pokusaj.add(Dense(
      units=hp.Int('dense_units', min_value=32, max_value=512,
```

```
step=32,default=128),
      activation=hp.Choice('dense_activation', values=['relu', 'tanh',
'sigmoid'], default='relu')))

  pokusaj.add(Dropout(rate=hp.Float('dropout_2', min_value=0.0, max_value=0.5,
default=0.5, step=0.1)))

  pokusaj.add(Dense(34, activation='softmax'))

  pokusaj.compile(
      optimizer = Adam(
          hp.Float('learning_rate', min_value=1e-4, max_value=1e-2,
sampling='LOG', default=1e-3)),
      loss='categorical_crossentropy',
      metrics=['accuracy'])

  return pokusaj
```

U posljednjoj iteraciji tuninga dobijeni su sljedeći parametri:

Value	Best Value So Far	Hyperparameter
32	32	num_filters_1
2	2	kernel_1
1	1	stride1
64	64	num_filters_2
4	2	kernel_2
2	1	stride2
0.4	0.2	dropout_1
256	256	dense_units
sigmoid	relu	dense_activation
0	0.3	dropout_2
0.00087701	0.0029995	learning_rate
12	2	tuner/epochs
0	0	tuner/initial_epoch
0	2	tuner/bracket
0	0	tuner/round

Konačno, konvolucijska neuronska mreža sastoji se od sljedećih komponenti:

1. **Dvodimenzionalni konvolucijski sloj (Cov2D)**
   - Broj filtera: 32
   - Veličina kernela: 2
   - Aktivacijska funkcija: *relu*
   - Strides: 1
2. **Dvodimenzionalni pooling sloj (MaxPooling2D)**

3. **Dvodimenzionalni konvolucijski sloj (Conv2D)**
    - Broj filtera: 64
    - Veličina kernela: 2
    - Aktivacijska funkcija: *relu*
    - Strides: 1
4. **Dvodimenzionalni pooling sloj (MaxPooling2D)**
5. **Regularizacija (Dropout)**
    - Dropout: 0.2
6. **Dense-sloj (Dense)**
    - Dense units: 256
    - Aktivacijska funkcija: *relu*
7. **Regularizacija (Dropout)**
    - Dropout: 0.3

Budući da je glavna osobina rekurentnih neuronskih mreža da izlaze ranijih iteracija dovode na ulaze narednih, a pošto je problem zasnovan na prepoznavanju znakova, jedna takva mreža je upotrijebljena i u našem modelu. U svrhe implementacije korištena je Kerasova funkcija *Bidirectional* zajedno sa modelom LSTM (*Long Short Term Memory*).

Na kraju je dodan još jedan Dense-sloj sa 34 izlaza (s obzirom da najduži tekst unutar dataset-a sadrži 34 karaktera) i aktivacijskom funkcijom *softmax*.
Kod za izgradnju modela prikazan je u isječku ispod:

```
x = layers.Conv2D(32, kernel_size=2, activation='relu', strides=1,
padding='same', name='Conv1')(input_image)
x = layers.MaxPooling2D((2,2), name='Pool1')(x)

x = layers.Conv2D(64, kernel_size=2, activation='relu', strides=1,
padding='same', name='Conv2')(x)
x = layers.MaxPooling2D((2,2), name='Pool2')(x)
x = layers.Dropout(0.2)(x)

x = layers.Reshape(target_shape=(64,1024), name="Reshape")(x)
x = layers.Dense(256, activation="relu", name="Dense1")(x)
x = layers.Dropout(0.3)(x)

x = layers.Bidirectional(layers.LSTM(256, return_sequences=True,
dropout=0.25))(x)
x = layers.Bidirectional(layers.LSTM(256, return_sequences=True,
dropout=0.25))(x)

y_pred = layers.Dense(34, activation="softmax", name="Output")(x)
```

Što se tiče funkcije gubitka, riječ je o CTC-loss funkciji, a iskorištena je Kerasova gotova funkcija *ctc_batch_loss*.

```
from keras import backend as K
def ctc_loss(args):
    labels, y_pred, input_length, label_length = args
    return K.ctc_batch_cost(labels, y_pred, input_length, label_length)
```

Nakon izgradnje modela i funkcije gubitka, definisani su ulazi i izlaz neuronske mreže.

```
input_image = layers.Input(shape=(256, 64, 1), name='image')
labels = layers.Input(name='label', shape=(34))
input_length = layers.Input(name='input_length', shape=(1,))
label_length = layers.Input(name='label_length', shape=(1,))
```

```
loss_out = layers.Lambda(ctc_loss, output_shape=(1,), name='ctc')([labels,
y_pred, input_length, label_length])
```

```
model = keras.models.Model(inputs=[input_image, labels, input_length,
label_length], outputs=loss_out)
```

Primijenjen je optimizator *Adam* sa stopom učenja 0.003. Stopa učenja je izabrana na osnovu ranije urađenog tuninga hiperparametara.

```
opt=keras.optimizers.Adam(learning_rate=0.003)
model.compile(loss={'ctc': lambda y_true, y_pred: y_pred}, optimizer=opt)
```

Konačno, pokrenuto je treniranje nad slikama za treniranje, a ujedno su kao parametar *validation_data* poslani podaci za validaciju. Ukupno je izvršeno 40 epoha treniranja.

```
history = model.fit(
    x = (train_X, train_Y, np.ones([train_len, 1]) * 48, train_label_len),
    y = np.zeros([train_len]),
    validation_data = ([val_X, val_Y, np.ones([val_len, 1]) * 48,
val_label_len], np.zeros([val_len]) ),
    epochs=40,
    batch_size = 128
)
```

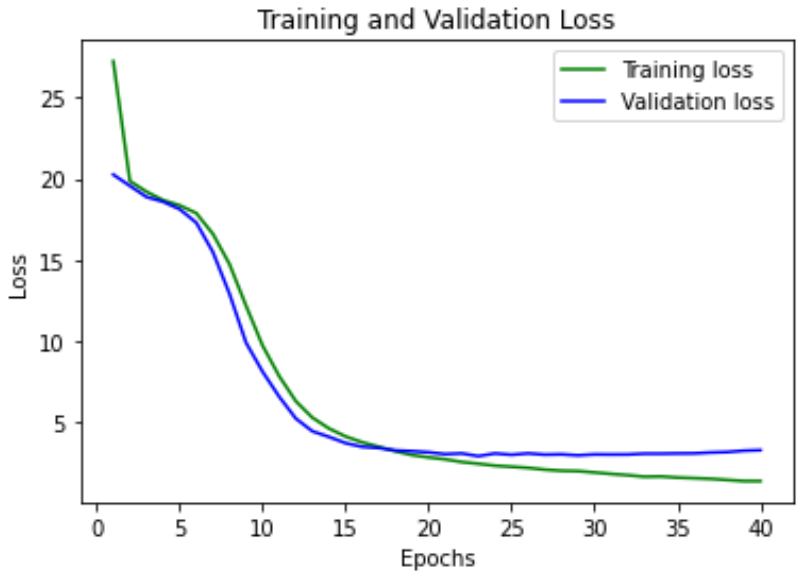
Na slici ispod je prikazan proces treniranja i iznosi funkcije gubitka i za trening i za validacijske podatke.

Procenat uspješnosti predviđanja ovakve neuronske mreže nad validacijskim skupom podataka je:

- 71.7 % tačno predviđenih riječi
- 89.9% tačno predviđenih karaktera

Također, naredni kod generiše grafik na kojem se može vidjeti odnos trening i validacijskih gubitaka i izvršenih epoha.

```
loss_train = history.history['loss']
loss_val = history.history['val_loss']
epochs = range(1,41)
plt.plot(epochs, loss_train, 'g', label='Training loss')
plt.plot(epochs, loss_val, 'b', label='validation loss')
plt.title('Training and Validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
```



Sa grafika jasno možemo vidjeti da je treniranje zaustavljeno u trenutku razdvajanja krivulja gubitka, što znači da smo izbjegli pojavu *underfittinga*, a kako krivulje nisu drastično razmaknute, također nije došlo ni do *overfittinga*.

6. Testiranje modela

Za potrebe testiranja modela koristili smo testni skup podataka izdvojenog unutar folders *test_v2*. U tom folderu se nalazi preko 30 000 slika, a mi smo koristili 3000. Nad istreniranim modelom je pozvana funkcija *predict*, a kao parametar funkcije je zadan niz slika testnog skupa smješten u varijablu *test_X*. Napominjemo da su testni podaci učitani, očišćeni i preprocesirani na isti način kako je to opisano za trening i validacijske podatke.

S obzirom da su dobijene predikcije zapravo nizovi u kojima se nalaze pojedinačne vjerovatnoće pojavljivanja nekog karaktera, takve rezultate je bilo potrebno dekodirati, odnosno pretvoriti ih u nenumeričke vrijednosti. Kako smo ranije koristili funkciju *preprocess_label* unutar koje je upotrijebljena gotova funkcija *texts_to_sequences*, u ovom dijelu dekodiranja ćemo ponovno upotrijebiti *Tokenizer* iz *tensorflow*-biblioteke i pomoću funkcije *sequences_to_texts* dobiti znakove kao rezultat predikcije.

```
def decode_predictions(pred):
    input_len = np.ones(pred.shape[0]) * pred.shape[1]
    results = keras.backend.ctc_decode(pred, input_length=input_len,
greedy=True)[0][0][
        :, :max_length
    ]
    output_text = []
    for res in results:
        decoded = tokenizer.sequences_to_texts([res.numpy()])
        output_text.append(decoded)
    return output_text
```

Sada se u varijabli *pred_texts* nalaze rezultati predikcija modela.

```
preds = model.predict(test_X)
pred_texts = decode_predictions(preds)
```

Funkcija *sequences_to_texts* vraća originalni tekst, međutim između svakog karaktera dodaje razmak. Na sljedećoj slici možemo vidjeti šta se desi kada tokenizeru pošaljemo tekst 'Vještačka inteligencija'. Karakteri su prvo kodirani što se vidi u prvom redu ispisa. Međutim, kada primijenimo inverznu funkciju, vidimo da smo dobili ispis 'V j e š t a č k a i n t e l i g e n c i j a'.

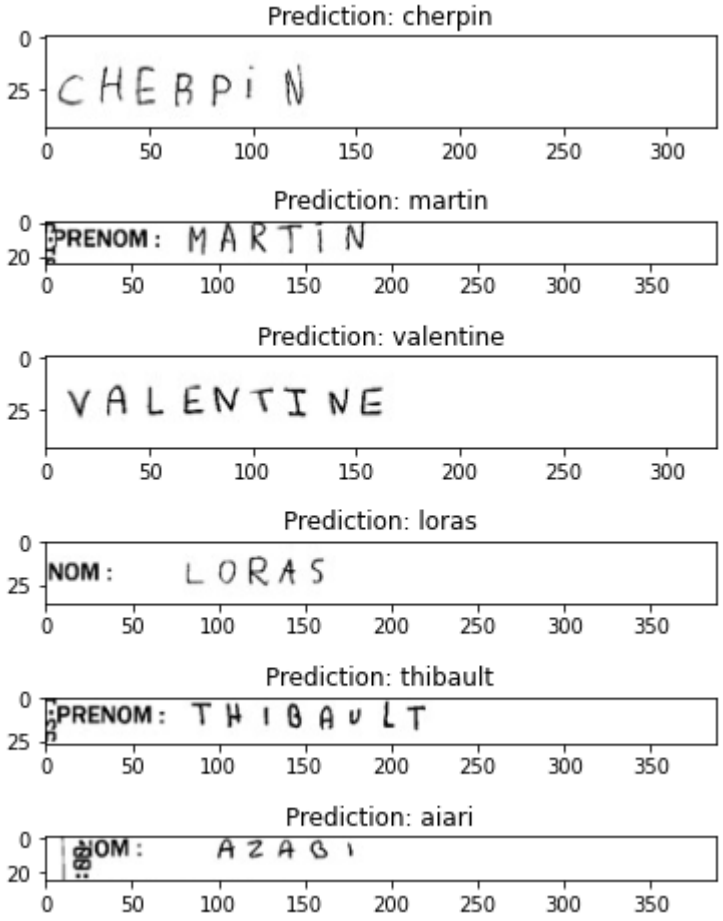
```
Sample sequence:  [19, 21, 1, 10, 8, 2, 12, 23, 2, 25, 3, 4, 8, 1, 5, 3, 16, 1, 4, 12, 3, 21, 2]
Sample text:  v j e s t a c k a  i n t e l i g e n c i j a
```

Da bismo ovo popravili napisali smo kod koji će niz *pred_modified* napuniti tesktovima bez razmaka između znakova. Taj kod je dat u isječku ispod.

```
pred_modified = []

for i in range(val_len):
  current = pred_texts[i][0]
  pom = ''
  for j in range(len(current)):
    if j % 2 == 0:
      pom += current[j]
  pred_modified.append(pom)
```

Na slici ispod možemo vidjeti nekoliko slika iz skupa za testiranje, kao i predikcije koje je model napravio, a koje smo prethodno unijeli u niz *pred_modified*.



Na kraju, ostalo je još da izračunamo procenat uspješnosti modela nad testnim podacima. Kao što je objašnjeno ranije, procenat uspješnosti smo pratili na osnovu dva kriterija: broj uspješno predviđenih imena i broj uspješno predviđenih karaktera.

U nastavku su tri funkcije koje će izračunati ovaj procenat. Prva funkcija, *review_by_word* vraća *True* ili *False* vrijednost, u zavisnosti da li su dvije riječi proslijeđene kao parametri iste.

```
def review_by_word(real, predicted):
  if real == predicted:
    return True
  return False
```

Naredna funkcija vraća procenat tačno predviđenih imena, a kao parametre prima niz u kojem su labele testnog skupa, *test_Y*, i skup predikcija modela.

```
def correct_words(val_true, pred_modified):
  num_of_correct = 0
  for i in range(val_len):
    true = val_true[i]
    predicted = pred_modified[i]
    if(review_by_word(true, predicted)):
      num_of_correct += 1

  percentage = num_of_correct/val_len
  return percentage
```

Konačno, funkcija *correct_characters* vraća informaciju o procentu tačno previđenih karaktera.

```
def correct_characters(val_true, pred_modified):
  num_of_correct = 0
  num_of_total = 0
  for i in range(val_len):
    true_word = val_true[i]
    predicted_word = pred_modified[i]
    num_of_total += len(true_word)

    for j in range(min(len(true_word), len(predicted_word))):
      if true_word[j] == predicted_word[j]:
        num_of_correct += 1

  return num_of_correct/num_of_total
```

Pozivom posljednje dvije funkcije, dobili smo konačne rezultate koji su prikazani u nastavku.

```
p = correct_words(test_Y, pred_modified) * 100
c = correct_characters(test_Y, pred_modified) * 100

print('Percentage of correct predicted words: ', p, '%')
print('Percentage of correct predicted characters: ', c, '%')
```

Procenat uspješnosti predviđanja ovakve neuronske mreže nad testnim skupom podataka je:

- 71.3 % tačno predviđenih riječi
- 89.3% tačno predviđenih karaktera

```
Percentage of correct predicted words:  71.33333333333334 %
Percentage of correct predicted characters:  89.32733592312411 %
```

Također, testiranje je urađeno i nad slikama koje smo smjestili u folder *nove_slike*. Taj folder se nalazi u našem projektnom direktoriju. Folder se sastoji od 50 slika, od kojih smo neke mi kreirali, a neke su preuzete iz dijela neiskorištenih slika za testiranje. Osim toga, u dokumentu *nove_slike.csv* nalaze se i labele za opisani skup slika.

Tačnost modela je mjerena na isti način kao i u prethodno opisanim koracima. Nakon izvršenih predikcija, model je nad ovakvim slikama imao sljedeći procenat uspješnosti:

- 82% tačno predviđenih riječi
- 87.3% tačno predviđenih karaktera

Iako su dobijeni rezultati poprilično zadovoljavajući, treba istaći da je model pokazao neke svoje nedostatke na slikama koje smo mi kreirali. Ovo je posljedica ranije opisanog rizika u vezi sa različitim rukopisima. Osim varijacija u rukopisu, važan faktor koji utiče na tačnost predikcija je i razmak između pojedinih karaktera.

7. Osvrt na dobijeno rješenje

Rezultati uspješnosti predviđanja naše neuronske mreže nad Kaggle testnim skupom podataka iznosili su:

- 71.3 % tačno predviđenih riječi
- 89.3% tačno predviđenih karaktera,

Dok su nad našim kreiranim testnim skupom iznosili:

- 82% tačno predviđenih riječi
- 87.3% tačno predviđenih karaktera

Ovi rezultati su zadovoljavajući, i za procenat tačno predviđenih karaktera i za procenat tačno predviđenih riječi. Naravno, u oba rezultata, procenat tačno predviđenih karaktera je veći, što je i očekivano, radi lakšeg identificiranja posebnih karaktera, nego riječi, koji se u mnogim pismima *(perzijsko i arapsko pismo)* pišu kurzivno, što znači da su pojedini znakovi u riječi međusobno povezani, za razliku od engleskog pisma u kojem su znakovi u riječi nepovezani. Također vidimo da smo uspješno zaobišli i probleme underfittinga i overfittinga.

Rezultati se mogu poboljšti, ali ne drastično, jer je i u uvodu našeg rada naglašeno da će biti teže postići visoku preciznost prepoznavanja, jer se svi rukopisi razlikuju. Čak i ljudima to predstavlja težak posao, a ne računarima. Način na koji bismo poboljšali jeste da uvećamo i trening i testni skup. Naime, mi smo za trening skup koristili 30 000 od 300 000 slika, a za testni 3 000 od 30 000. S tim da bi svako povećanje broja trening slika, drastično uvećavalo i vrijeme treniranja modela.

Kao i što smo ranije naveli, treniranje našeg modela izvršili smo pomoću upotrebe konvolucijskih i rekurentnih neuronskih mreža (u svrhe implementacije korištena je Kerasova funkcija *Bidirectional* zajedno sa modelom LSTM (*Long Short Term Memory*)), te ćemo poređenje naših rezultata vršiti sa rezultatima iz sljedeća dva rada: *A Review of Various Handwriting Recognition Methods i Offline Persian Handwriting Recognition with CNN and RNN-CTC,* upravo jer prvi navedeni rad vrši treniranje svog modela putem konvolucione neuronske mreže, a drugi putem rekurentne neuronske mreže.

Naime, zbog nedostatka izloženih krajnjih rezultata u ova dva rada, uporedit ćemo konstrukcije neuronskih mreža.

U prvom radu neuronska mreža je CNN sa slojevima convolutional, pooling, fully-connected i soft-max, dok su našu konvolucijsku mrežu činili sojevi: dva konvolucijska sloja, dva pooling sloja, jednog *dense*-sloja, kao i funkcija *Dropout* korištena za regularizacije. Osim sličnosti u izgradnji neuronske mreže, primjetili smo slične prednosti i mane konvolucijske mreže: i naša kao i njihova mreža je imala veliku tačnost, ali i vrlo sporo izvršavanje, radi velike količine podataka. U drugom radu je treniranje modela izvršeno konvolucijskim neuronskim mrežama (CNN) i rekurentnim

neuronskim mrežama (RNN). U našem radu je također izvršeno treniranje RNNom, pri čemu je korištena Kerasova funkcija *Bidirectional* zajedno sa modelom LSTM (*Long Short Term Memory*), isto kao i u radu.