Univerzitet u Sarajevu
Elektrotehnički fakultet Sarajevo
Odsjek za računarstvo i informatiku

Prepoznavanje ručno pisanog teksta za bosanski jezik s miješanim stilovima pisanja (štampana i pisana slova)

Vještačka inteligencija

Studenti:
Lamija Borovina, 19356
Kanita Kadušić, 19327
Esma Kurtović, 19358

Profesor:
prof. dr. Amila Akagić

Sarajevo, akademska 2024/25. godina

Sadržaj

Popis tabela

Popis slika

Izbor teme i opis problema

Odabrana tema projekta

Tema ovog projektnog zadatka je *Prepoznavanje ručno pisanog teksta za bosanski jezik s miješanim stilovima pisanja (štampana i pisana slova)*.

Ova tema je odabrana zbog svoje izuzetne relevantnosti i izazovnosti u kontekstu digitalizacije i obrade podataka. Iako je HTR polje koje se intenzivno razvija, specifični zahtjevi bosanskog jezika, koji se aktivno služi s dva stila pisanja, predstavljaju svojevrstan izazov. Razvijanjem rješenja koje efikasno transkribira **ručno pisani tekst na bosanskom jeziku sa štampanim i pisanim stilom** se doprinosi praktičnim primjenama u arhivistici, obrazovanju, administraciji i mnogim drugim sektorima.

Opis problema i izazovi

Definicija problema

Problem koji se rješava kroz ovaj projektni zadatak je razvoj sistema za prepoznavanje ručno pisanog teksta koji će biti sposoban da precizno transkribira rukom pisan tekst napisan na bosanskom jeziku, s ključnim naglaskom na efikasno rukovanje istovremenom prisutnošću štampanih i pisanih slova unutar istog dokumenta.

Tradicionalni HTR sistemi često su trenirani primarno na jednom stilu pisanja - bilo potpuno štampanom ili potpuno pisanom. Međutim, u svakodnevnoj praksi, ljudi često kombinuju ova dva stila unutar istih dokumenata, rečenica, pa čak i pojedinih riječi (npr. kada se koriste velika štampana slova za isticanje ili kada se neki dijelovi pišu štampano, a drugi pisano radi brzine). Ova fleksibilnost ljudskog rukopisa postavlja značajan izazov za HTR modele, koji moraju biti u stanju da mapiraju ovakav “miješani” stil u standardni digitalni tekst.

Ključni izazovi i kompleksnost

Razvoj HTR sistema za bosanski jezik s miješanim stilovima pisanja suočava se s nekoliko značajnih izazova:

- **Miješani stilovi pisanja:** Osnovni izazov je sposobnost modela da precizno prepozna i razlikuje karaktere napisane u štampanom i pisanom stilu, te da se efikasno “prebacuje” između stilova unutar teksta. Vizualne razlike između štampanih i pisanih varijanti istog slova (npr. ‘a’ i ‘*a*’) su značajne, što zahtijeva od modela da nauči širi spektar oblika slova.

- ➔ **Kontinuitet i segmentacija:** Pisana (kurzivna) slova su po pravilu spojena, dok su štampana odvojena. Model mora uspješno rukovati i spojenim i odvojenim karakterima, što komplikuje proces segmentacije slova unutar riječi.
- ➔ **Varijacije u rukopisu:** Ljudskom rukopisu je svojstvena raznolikost. Različiti pojedinci pišu različitim stilovima, veličinom slova, nagibom, te stepenom spojenosti ili odvojenosti slova. Model ne bi smio biti osjetljiv na ove varijacije.
- ➔ **Kvalitet ulaznih podataka:** Skenirani ili fotografisani ručno pisani dokumenti često mogu imati probleme s kvalitetom, poput niske rezolucije, slabog kontrasta, mrlja, sjena, iskrivljenosti ili šuma. Ovi nedostaci mogu značajno otežati ekstrakciju karakteristika i prepoznavanje. *(Prevazilaženje ovog izazova nije dio projektnog zadatka.)*
- ➔ **Nedostatak *dataset*-ova:** Jedan od većih izazova je ograničen broj, ili čak potpuni nedostatak, javno dostupnih, velikih i kvalitetnih *dataset*-ova ručno pisanog teksta na bosanskom jeziku, posebno onih koji obuhvataju miješane stilove pisanja.

Osnovni pojmovi

U kontekstu ovog projekta, sljedeći pojmovi su ključni:

- ❖ Prepoznavanje ručno pisanog teksta (eng. HTR - *Handwritten Text Recognition*): Područje vještačke inteligencije čiji je cilj automatska konverzija ručno pisanog teksta sa fizičkog medija (papir) u digitalni, mašinski čitljiv format. Za razliku od OCR-a (eng. *Optical Character Recognition*) koji se primarno bavi štampanim tekstom, HTR se fokusira na neizbježnu varijabilnost ljudskog rukopisa, uključujući različite stilove pisanja.
- ❖ Skup podataka (eng. *dataset*): Organizovana kolekcija podataka koja se koristi za treniranje, validaciju i testiranje modela.
- ❖ Transkripcija: Proces pretvaranja ulazne slike ručno pisanog teksta u njegov digitalni tekstualni ekvivalent.

Korist i primjena rješenja problema

Rješavanje problema efikasnog HTR-a za bosanski jezik s miješanim stilovima pisanja donosi višestruke koristi i ima širok spektar primjena u različitim sektorima:

- ➔ **Digitalizacija i arhiviranje kulturne baštine:** Omogućava brzu i automatizovanu digitalizaciju ogromnih količina historijskih dokumenata,

rukopisa, starih bilježnica i arhivske građe na bosanskom jeziku, uključujući one pisane miješanim stilovima. Ovo čuva kulturno naslijeđe i čini ga pretraživim i dostupnijim istraživačima i široj javnosti.

- ➔ **Efikasnost u administraciji i birokratiji:** Automatizacija obrade ručno popunjenih formulara, zahtjeva, molbi i drugih dokumenata u javnim i privatnim institucijama, gdje se često kombinuju štampani i pisani unosi. Ovo značajno smanjuje potrebu za ručnim unosom podataka, smanjuje troškove i ubrzava administrativne procese.
- ➔ **Poboljšanje pristupačnosti informacija:** U medicinskom sektoru, omogućava brzu konverziju ručno pisanih medicinskih kartona, recepata i uputnica u digitalni format, čak i ako sadrže miješane stilove. U pravosudnom sistemu, olakšava digitalizaciju sudskih zapisa, policijskih izvještaja i drugih pravnih dokumenata.
- ➔ **Podrška obrazovanju i istraživanju:** Studentima i nastavnicima olakšava prebacivanje ručno pisanih bilješki, eseja i zadataka u digitalni format, što omogućava lakše pretraživanje, uređivanje i dijeljenje, čak i ako su kombinovali stilove pisanja. Istraživačima u humanističkim naukama otvara mogućnost primjene naprednih alata za analizu teksta na ručno pisanim historijskim izvorima.
- ➔ **Razvoj softverskih rješenja:** Otvara vrata za razvoj novih aplikacija i alata koji koriste HTR tehnologiju, kao što su digitalni asistenti za pisanje bilješki, pametne pretrage rukopisa ili sistemi za automatsku indeksaciju dokumenata, koji mogu prepoznati i obraditi raznovrsne stilove ljudskog rukopisa.

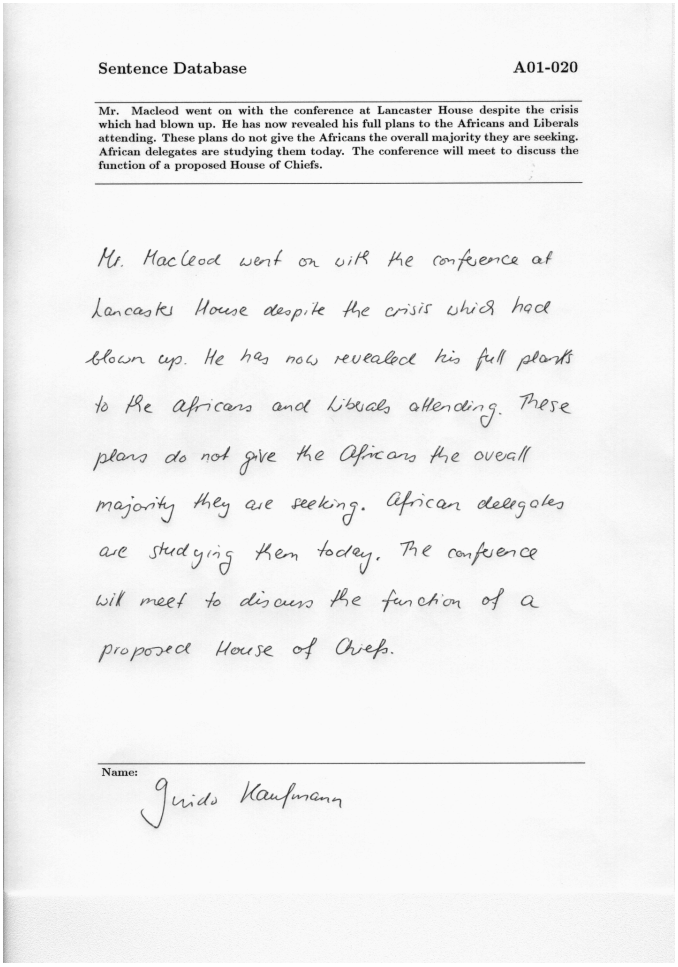
Pregled postojećih *dataset*-ova

Razvoj efikasnog HTR sistema značajno ovisi o dostupnosti i kvaliteti *dataset*-ova za treniranje. Dok globalno postoji niz renomiranih *dataset*-ova za HTR, situacija je znatno drugačija kada je riječ o specifičnim jezicima, a posebno za bosanski jezik i izazov miješanih stilova pisanja. U nastavku su izdvojena tri *dataset*-a:

1. ***IAM Handwriting Database*** [8]

   IAM Handwriting Database sadrži rukom pisani tekst na engleskom jeziku koji se koristi za treniranje i evaluaciju HTR modela te za zadatke identifikacije i verifikacije autora.

   - 657 autora (različitih rukopisa)
   - 1.539 stranica, 5.685 rečenica, 13.353 linija, 115.320 riječi
   - 300 dpi, PNG format, 256 sivih tonova
   - svaka slika ima pridružen XML fajl sa metapodacima



Sentence Database A01-020

Mr. Macleod went on with the conference at Lancaster House despite the crisis which had blown up. He has now revealed his full plans to the Africans and Liberals attending. These plans do not give the Africans the overall majority they are seeking. African delegates are studying them today. The conference will meet to discuss the function of a proposed House of Chiefs.

Mr. Macleod went on with the conference at Lancaster House despite the crisis which had blown up. He has now revealed his full plans to the Africans and Liberals attending. These plans do not give the Africans the overall majority they are seeking. African delegates are studying them today. The conference will meet to discuss the function of a proposed House of Chiefs.

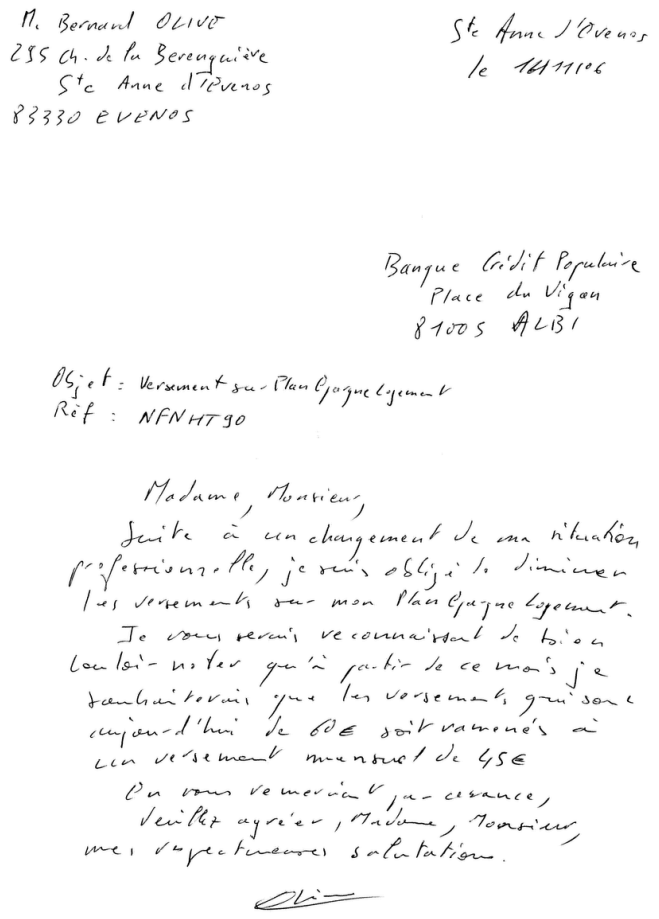
Name: Guido Kaufmann

Slika 1. Primjer podatka iz *IAM Handwriting Database dataset*-a

2. ***RIMES*** [9]

   RIMES je francuski *dataset* s rukom pisanim fiktivnim pismima, pisanim u realnim kontekstima (npr. žalbe, zahtjevi, promjene podataka). Koristi se za HTR, identifikaciju autora, strukturu dokumenata i ekstrakciju informacija.

   - 5.605 komunikacija (9 scenarija dopisa, npr. žalba)
   - 12.610 stranica
   - 8 tipova segmenata (npr. adresa pošiljaoca, datum, tijelo teksta)
   - PNG format, *grayscale*
   - XML anotacije sa transkripcijama i strukturom


M. Bernard OLIVE
285 ch. de la Bérenguière
Ste Anne d'Evenos
83330 EVENOS

Ste Anne d'Evenos
le 16/11/06

Banque Crédit Populaire
Place du Vigan
81005 ALBI

Objet : Versement sur Plan Epargne Logement
Réf : NFNHT90

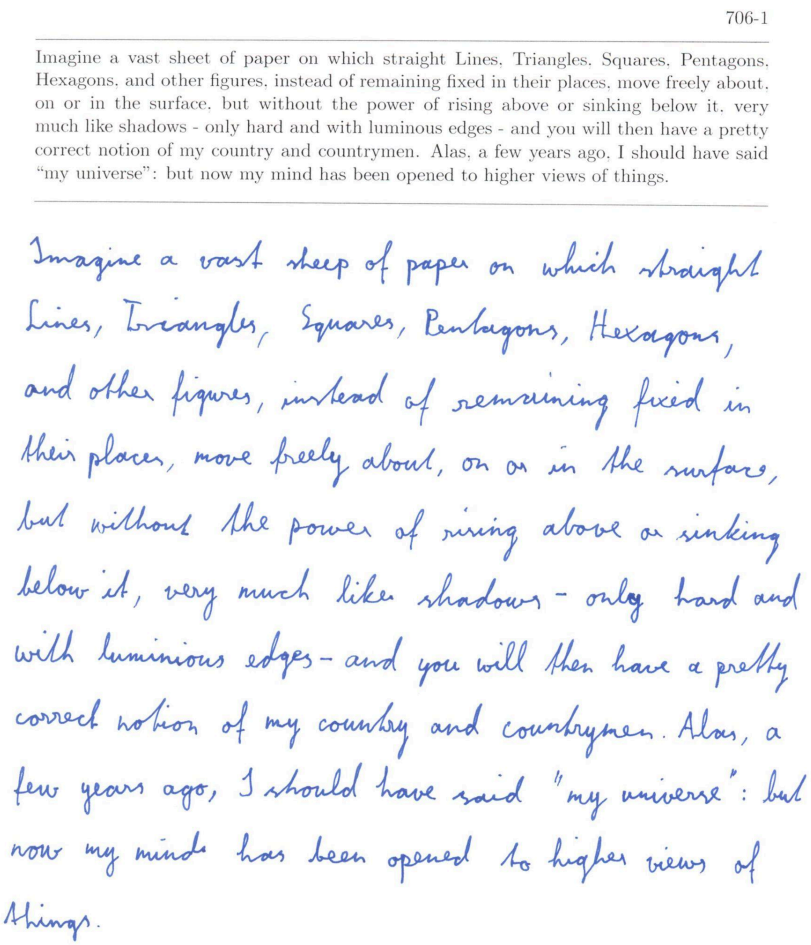
Madame, Monsieur,
Suite à un changement de ma situation professionnelle, je serais obligé de diminuer les versements sur mon Plan Epargne Logement.
Je vous serais reconnaissant de bien vouloir me faire savoir jusqu'à quand je dois maintenir mes versements mensuels de 60€ soit ramenés à un versement mensuel de 45€
Dans votre réponse, je vous remercie,
Veuillez agréer, Madame, Monsieur, mes respectueuses salutations.

Slika 2. Primjer podatka iz *RIMES dataset*-a

3. ***CVL Database*** [10]

   CVL Database je savremeni *dataset* s ručno pisanim tekstovima, namijenjen za zadatke kao što su HTR, identifikacija autora i pretraga riječi u slikama. Sadrži rukom pisane tekstove na njemačkom i engleskom jeziku, prepisane iz poznatih književnih djela.

   - 7 tekstova
   - 310 pisaca (27 pisaca po 7 tekstova, 283 pisca po 5 tekstova)
   - 1.604 stranice
   - 300 dpi, RGB
   - XML anotacije koje sadrže *bounding box* za svaku riječ



706-1

Imagine a vast sheet of paper on which straight Lines, Triangles, Squares, Pentagons, Hexagons, and other figures, instead of remaining fixed in their places, move freely about, on or in the surface, but without the power of rising above or sinking below it, very much like shadows - only hard and with luminous edges - and you will then have a pretty correct notion of my country and countrymen. Alas, a few years ago, I should have said "my universe": but now my mind has been opened to higher views of things.

Imagine a vast sheep of paper on which straight Lines, Triangles, Squares, Pentagons, Hexagons, and other figures, instead of remaining fixed in their places, move freely about, on or in the surface, but without the power of rising above or sinking below it, very much like shadows - only hard and with luminous edges - and you will then have a pretty correct notion of my country and countrymen. Alas, a few years ago, I should have said "my universe": but now my mind has been opened to higher views of things.

Slika 3. Primjer podatka iz *CVL Database dataset*-a

Pregled stanja u oblasti

U nastavku slijedi pregled stanja u oblasti HTR-a, s fokusom na metode i postignute rezultate. Cilj je identificirati ključne tehnike koje se koriste u modernom HTR-u, razumjeti njihove prednosti i ograničenja, te pronaći potencijalne pravce za poboljšanje, relevantne za specifičnost problema HTR-a za bosanski jezik s miješanim stilovima pisanja (štampana i pisana slova).

Evolucija HTR sistema

Pristupi rješavanju problema

Istraživanja u oblasti HTR-a su se razvijala od heurističkih pristupa do kompleksnih neuronskih mreža. Rani pokušaji prepoznavanja rukopisa uglavnom su se oslanjali na statističke metode i ekstrakciju *hand-crafted features*. To je uključivalo analizu geometrije slova, kao što su broj petlji, uglovi, tačke presjeka i slično. Primjeri takvih pristupa uključuju **skrivene Markovljeve modele (HMM)**, koji su bili popularni za modeliranje sekvenci i varijabilnosti rukopisa, te **podržavajuće vektorske mašine (SVM)** za klasifikaciju pojedinačnih karaktera [16]. Iako su ovi sistemi imali određenu primjenu, bili su ograničeni na unaprijed definisane stilove pisanja i zahtijevali su znatno preprocesiranje i *feature engineering*.

Značajni rezultati u oblasti HTR-a su se pojavili s razvojem dubokog učenja. Duboke neuronske mreže, sa svojom sposobnošću samostalnog učenja karakteristika direktno iz podataka (slika), eliminisale su potrebu za *feature engineering*-om. Posebno su se istakle dvije vrste neuronskih mreža:

1. **Konvolucijske neuronske mreže** (eng. CNN - *Convolutional Neural Network*): Koriste se za ekstrakciju vizualnih karakteristika iz slika rukopisa. CNN-ovi mogu prepoznati prostorne obrasce i teksture na različitim nivoima apstrakcije, od osnovnih krivih do složenijih oblika slova, bez obzira na njihovu poziciju na slici [17].
2. **Rekurentne neuronske mreže** (eng. RNN - *Recurrent Neural Network*), posebno LSTM (eng. *Long Short-Term Memory*) mreže: Ove mreže su dizajnirane za obradu sekvencijalnih podataka i izuzetno su efikasne za modeliranje kontekstualnih zavisnosti u tekstu. U HTR-u, LSTM-ovi primaju karakteristike koje je izvukao CNN i sekvencijalno predviđaju karaktere, uzimajući u obzir prethodno prepoznate znakove. Ključni element koji je omogućio treniranje ovih sistema je CTC (eng. *Connectionist Temporal Classification*) sloj, koji rješava problem poravnanja između izlaza neuronske mreže i ciljne transkripcije, bez potrebe za eksplicitnom segmentacijom slike na pojedinačna slova [18].

Noviji napredak donijeli su **transformer** bazirani modeli, originalno razvijeni za obradu prirodnog jezika. Zahvaljujući arhitekturi baziranoj na mehanizmu pažnje (eng. *attention mechanism*), pokazali su izuzetnu sposobnost da "uhvate" dugoročne zavisnosti unutar sekvenci, što ih čini pogodnim za HTR zadatke prepoznavanja cijelih linija ili stranica teksta [19].

Opseg rješavanog problema

- ➔ **Jezici s jednim pismom:** Većina istraživanja i *dataset*-ova (IAM-a za engleski [8], RIMES za francuski [9]) fokusirana je na jezike koji koriste jedno dominantno pismo, uglavnom latinicu. Postoje i napori za jezike s kompleksnijim pismima poput arapskog (KHATT *dataset* [20]) ili indijskih jezika (MODI *dataset* [21]), ali se i oni obično fokusiraju na prepoznavanje unutar jednog specifičnog pisma.
- ➔ **Homogeni stilovi pisanja:** Postojeći HTR sistemi obično postižu visoku tačnost na *dataset*-ovima gdje je stil rukopisa relativno homogen (npr. većina teksta je štampana ili većina je pisana). Rjeđe se direktno adresira problem gdje su štampana i pisana slova često miješana unutar istog dokumenta ili čak riječi.
- ➔ **Historijski dokumenti:** Značajan dio HTR istraživanja posvećen je digitalizaciji i prepoznavanju teksta iz starih, historijskih dokumenata (Bentham *dataset* [22]). Ovi dokumenti često sadrže specifične arhaične stilove, varijacije u mastilu i papiru, što dodaje sloj kompleksnosti.

Korištene metode vještačke inteligencije i postignuti rezultati

Trenutno stanje HTR tehnologije dominantno je oblikovano modelima dubokog učenja, koji su postigli značajno bolje rezultate u odnosu na prethodne metode.

CNN-LSTM-CTC arhitektura

CNN dio efikasno izvlači hijerarhijske vizualne karakteristike iz ulazne slike rukopisa. Izlazi CNN-a se zatim prosljeđuju u RNN slojeve (tipično višeslojne dvosmjerne LSTM mreže) koje su zadužene za modeliranje sekvencijalnih zavisnosti karaktera. Na kraju, CTC sloj omogućava modelu da uči mapiranje između sekvence *feature*-a i sekvence labele (transkripcije) bez potrebe za preciznom segmentacijom svakog pojedinog karaktera. Ovo čini sistem "end-to-end" rješenjem [18].

Na standardizovanim HTR *dataset*-ovima, kao što su *IAM Handwriting Database* [8] i *RIMES* [9], modeli bazirani na CNN-LSTM-CTC arhitekturi postižu značajno niske stope grešaka. Rad [27] navodi sljedeće rezultate:

Tabela 1. Rezultati CNN-LSTM-CTC arhitekture					
		IAM		RIMES	
		CER[1] %	WER[2] %	CER %	WER %
150 dpi	*no language model*	10.1	29.5	3.2	13.6
	with language model	6.5	16.6		
300 dpi	*no language model*	7.9	24.6	2.9	12.6
	with language model	5.5	16.4		

Transformer arhitektura

Transformer arhitektura, prvobitno predstavljena za zadatke obrade prirodnog jezika (eng. NLP - *Natural Language Processing*), revolucionirala je pristup modeliranju sekvencijalnih podataka i brzo je adaptirana za zadatke prepoznavanja teksta, uključujući HTR. Za razliku od rekurentnih mreža koje obrađuju podatke sekvencijalno, transformeri koriste mehanizam pažnje (eng. *attention mechanism*), što im omogućava da obrade cijelu ulaznu sekvencu paralelno i efikasno uhvate dugoročne zavisnosti između elemenata [19].

Na primjer, na standardizovanim *dataset*-ovima poput *IAM Handwriting Database* [8], modeli bazirani na transformerima, poput TrOCR-a (kôd dostupan na repozitoriju[3]), mogu postići izuzetno niske CER-ove. Konkretno, TrOCR je demonstrirao CER od 2.97% na test setu *IAM dataset*-a [19]. Ovi impresivni rezultati postižu se posebno kada se koriste napredne strategije pretreninga na masivnim skupovima podataka i naknadno fino podešavanje (*fine-tuning*) [19].

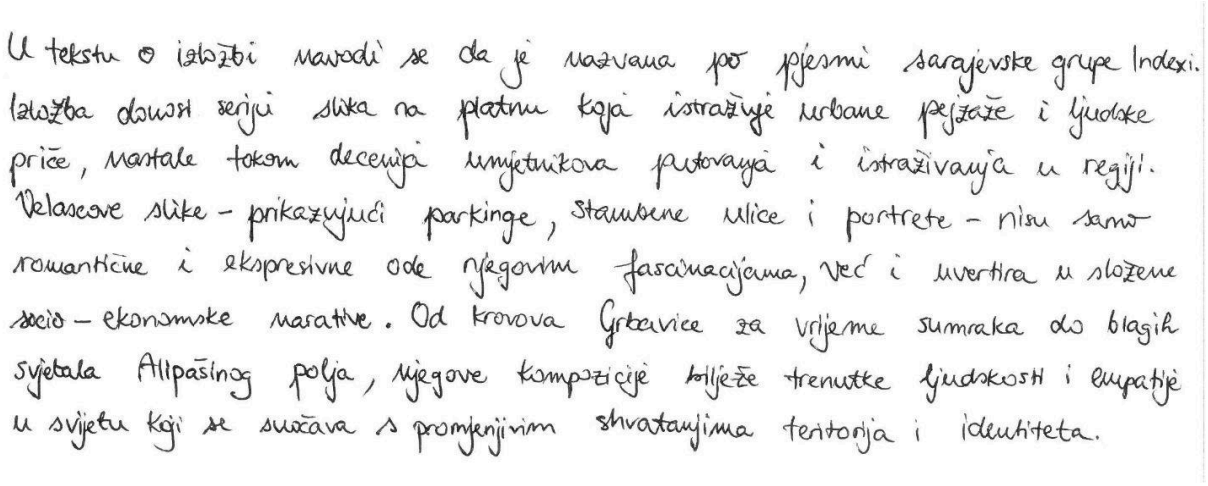
[1] *Character Error Rate*
[2] *Word Error Rate*
[3] https://github.com/microsoft/unilm/tree/master/trocr

Izbor, analiza i preprocesiranje *dataset*-a

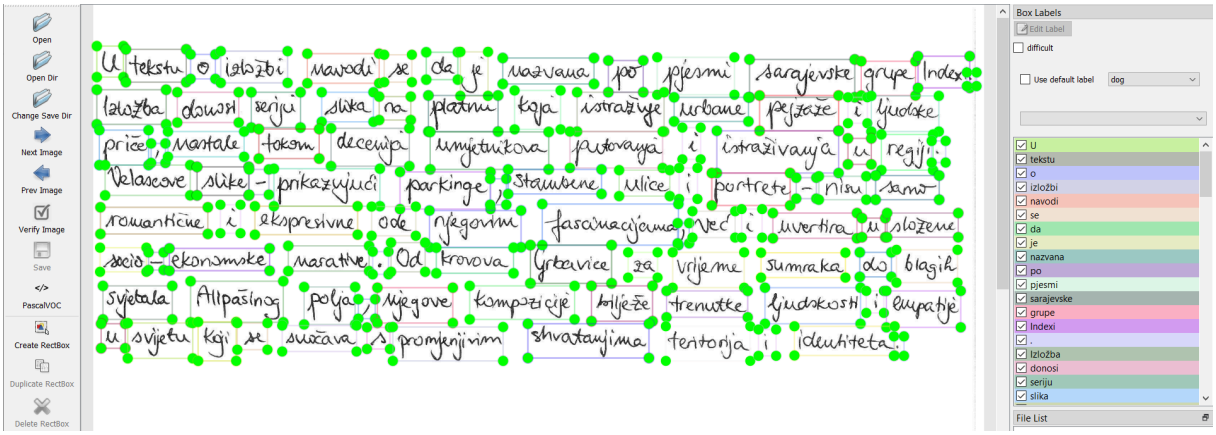
Korišteni dataset je kreiran spajanjem osam individualnih HTR *dataset*-ova, kreiranih u okviru zadatka za predmet Vještačka inteligencija, te izdvajanjem dijela koji se odnosi na *mix* stil rukopisa. Individualni *dataset*-ovi su kreirani sljedećim koracima:

1) Napisano je 20 tekstova sa po 3 stila pisanja (*print* - štampana slova, *cursive* - pisana slova, *mix* - kombinacija štampanih i pisanih). Tekstovi su na bosanskom jeziku (sadrže afrikate č, ć, đ, š, ž) i sastoje se od oko 4 do 6 rečenica, s manjim odstupanjima. Primarni izvori tekstova su internet, knjige, časopisi, zvanični dokumenti i sl. Svaki od 60 tekstova je napisan na zasebnom listu papira. Primjer jednog teksta je dat na Slici 4, pri čemu je odrezan dio slike na kojem nema teksta, radi jednostavnosti prikaza.
2) Listovi papira s tekstovima su skenirani u rezoluciji 300 DPI, s vertikalnom orijentacijom stranice. Slike su imenovane kao [inicijali osobe]_[stil pisanja]_[broj teksta za tu osobu]. Sve slike su u .jpg formatu. Primjer naziva jedne slike je NM_mix_000.jpg.
3) Pomoću alata *labelImg* [1] su označene pojedinačne riječi na svakoj slici (primjer dat na Slici 5). Korišten je *PascalVOC* [2] format labela. Ovime se dobiva XML *file* s istim nazivom kao slika, u kojem su sačuvani podaci o sadržaju i poziciji kreiranih bounding *box*-eva za svaku riječ. Labela za jednu riječ izgleda ovako:

```
<object>
  <name>tekstu</name>                (tekst koji se nalazi na slici)
  <pose>Unspecified</pose>
  <truncated>0</truncated>
  <difficult>0</difficult>
  <bndbox>                           (koordinate bounding box-a)
    <xmin>91</xmin>
    <ymin>124</ymin>
    <xmax>259</xmax>
    <ymax>215</ymax>
  </bndbox>
</object>
```



Slika 4. Primjer rukom pisanog teksta u *mix* stilu



Slika 5. Primjer označenog teksta u labelImg alatu

Pored ovoga, za svaki tekst je kreirana i tekstualna datoteka koja sadrži njegovu transkripciju, kao i jedna *metadata.csv* datoteka na nivou čitavog *dataset*-a, s metapodacima za svaku od 60 slika. Međutim, ovi podaci nisu korišteni u okviru projekta, te stoga neće biti detaljnije obrađeni.

Ovime se dobija sljedeća struktura:

dataset/

├── images/ *(pojedinačne slike)*

│ ├── NM_print_000.jpg

│ ├── NM_cursive_000.jpg

│ ├── NM_mix_000.jpg

| └── ...

├── ground_truth/ *(XML file-ovi s labelama riječi na slici)*

| ├── NM_print_000.xml

| ├── NM_cursive_000.xml

| ├── NM_mix_000.xml

| └── ...

├── transcription/ *(transkripcije tekstova)*

| ├── NM_000.txt

| └── ...

└── metadata.csv *(metapodaci)*

Spajanjem ovih *dataset*-ova, te izvlačenjem dijela za *mix* stil pisanja, dobiva se *mix dataset* sa 156 različitih tekstova. Za svaki se vežu:

- slika rukom pisanog teksta,
- XML *file* s koordinatama i sadržajem *bounding box*-eva za sve pojedinačne riječi u tekstu.

Struktura finalno dobivenog *dataset*-a je sljedeća:

mix/

├── images/ *(pojedinačne slike)*

| ├── AB_mix_000.jpg

| ├── NM_mix_000.jpg

| ├── NM_mix_001.jpg

| └── ...

├── ground_truth/ *(XML file-ovi s labelama riječi na slici)*

| ├── AB_mix_000.xml

| ├── NM_mix_000.xml

| ├── NM_mix_001.xml

| └── ...

Veličina ovakvog *mix dataset*-a je 43.5MB.

Na osnovu ovih podataka su kreirani *dataset*-ovi za pojedinačne riječi i linije teksta. Za smještanje podataka je korišten *Google Drive*, kojem se pristupa iz *Google Colab notebook*-a. Navedeno je priloženo uz projektnu dokumentaciju.

Kreiranje ovih *dataset*-ova prati sljedeće korake, pri čemu se u dokumentaciji primarno referira na prateći *Google Colab* notebook. Koraci 1, 2 i 3 se oslanjaju na kodove iz [23], uz neke izmjene i dodatke vezane za podatke korištene u ovom projektu, što će biti dodatno naglašeno.

1) Kreiranje labela pojedinačnih riječi i linija teksta

I. Definisana je pomoćna funkcija ***read_content***, koja prima naziv XML *file*-a s labelama u *PascalVOC* formatu. Ona parsira dati XML *file*, vodeći se očekivanom strukturom objekata, te vraća listu listi, odnosno matricu u kojoj se svaki red odnosi na labelu jedne riječi i prati sljedeći format

   [*filename*, *w_idx*, *w*, *h*, *xmin*, *ymin*, *xmax*, *ymax*, *name*],

   gdje je:

   - *filename*: naziv *file*-a,
   - *w_idx*: indeks riječi,
   - *w*: širina slike,
   - *h*: visina slike,
   - *xmin*: x-koordinata gornjeg lijevog vrha *bounding box*-a,
   - *ymin*: y-koordinata gornjeg lijevog vrha *bounding box*-a,
   - *xmax*: x-koordinata donjeg desnog vrha *bounding box*-a,
   - *ymax*: y-koordinata donjeg desnog vrha *bounding box*-a,
   - *name*: tekst označene riječi u *uppercase*-u.

II. Kao prva faza u kreiranju labela pojedinačnih riječi, definisana je funkcija ***generate_word_labels***, koja prima putanju do *folder*-a u kojem se nalaze XML *file*-ovi za labelirane slike (u ovom slučaju folder *ground_truth*) i listu naziva tih XML *file*-ova. U okviru ove funkcije se poziva *read_content*, čiji se rezultat upisuje u *word_labels.txt*, pri čemu se podaci u svakom redu razdvajaju znakom '|'. Konkretno, svakoj riječi odgovara zapis sljedećeg formata:

   filename|*w_idx*|*w*|*h*|*xmin*|*ymin*|*xmax*|*ymax*|*name*, kao npr.

   LB_mix_005.jpg|1|3307|4676|263|173|526|363|AKO

   File word_labels.txt se kreira u istom *folder*-u u kojem se nalaze XML *file*-ovi, u ovom slučaju *ground_truth*.

III. Prva funkcija, definisana za kreiranje labela linija teksta, jeste ***generate_line_labels***. Kao i *generate_word_labels*, kao parametre prima putanju do *folder*-a u kojem se nalaze XML *file*-ovi za labelirane slike (u ovom slučaju folder *ground_truth*) i listu naziva tih XML *file*-ova. S obzirom da su na slikama označene pojedinačne riječi, ove funkcija služi za grupisanje tih riječi u linije teksta, na osnovu određenog kriterija.

Za ovu svrhu se koristi klastering algoritam DBSCAN, koji definiše klastere kao guste regije tačaka, razdvojene regijama manje gustine. Ovaj algoritam ima dva hiperparametra: epsilon, koje predstavlja maksimalnu udaljenost između dvije tačke na kojoj one mogu biti smatrane susjedima; te minimalan broj tačaka potreban za formiranje klastera. Konkretno, za jedan klaster se specificira središnja tačka, kao ona koja ima minimalan potreban broj drugih tačaka u svojoj epsilon okolini. Tačke koje su u njenoj epsilon okolini se dodaju u isti klaster. S druge strane, tačke koje ne pripadaju nijednom klasteru se smatraju šumom. [3]

Funkcija *generate_line_labels* kao jednu liniju teksta definiše sve riječi čija je srednja linija približno na istoj visini. Konkretno, koristi funkciju *read_content* za dobivanje podataka o labelama riječi na slici, nakon čega za svaku računa visinu srednje linije kao *(ymin + ymax) / 2*. Potom koristi DBSCAN algoritam za grupisanje riječi u linije

```
cluster = DBSCAN(eps=25, min_samples=1,
metric="precomputed").fit(distances),
```

gdje je *distances* matrica međusobnih udaljenosti srednjih linija *bounding box*-eva (u skladu s *metric="precomputed"*). Parametar *eps* predstavlja epsilon, te će u ovom slučaju u istu liniju biti grupisane sve riječi čije su srednje linije udaljene maksimalno 25px. S obzirom da *min_samples* iznosi 1, jedan klaster/linija se može sastojati i od samo jedne riječi, čime efektivno neće biti tačaka koje se smatraju šumom. [4]

Na kraju se u *line_labels.txt*, koji se kreira u istom *folder*-u u kojem se nalaze XML *file*-ovi (*ground_truth*), za svaku liniju teksta upisuje red u formatu

filename|*l_idx*|*w*|*h*|*l_xmin*|*l_ymin*|*l_xmax*|*l_ymax*|*text*,

gdje je:

- *filename*: naziv *file*-a,
- *l_idx*: indeks linije,
- *w*: širina slike,
- *h*: visina slike,
- *l_xmin*: najmanja x-koordinata u okviru linije, tj. x-koordinata gornjeg lijevog vrha

- *l_ymin*: najmanja y-koordinata u okviru linije, tj. y-koordinata gornjeg lijevog vrha
- *l_xmax*: najveća x-koordinata u okviru linije, tj. x-koordinata donjeg desnog vrha
- *l_ymax*: najveća y-koordinata u okviru linije, tj. y-koordinata donjeg desnog vrha
- *text*: tekst linije u *uppercase*-u, nastao spajanjem pojedinačnih riječi razdvojenih razmakom.

Primjer jednog ovakvog zapisa je:

LB_mix_005.jpg|5|3307|4676|218|630|2603|885|NISAM SE POTRUDIO DA MU ODGOVORIM .

IV. Druga funkcija, definisana za kreiranje labela linija teksta, jeste ***generate_line_labels_m***. (Ova funkcija je naknadno implementirana za potrebe projekta.) Ima iste parametre i svrhu kao *generate_line_labels*, ali grupiše riječi u linije po drugačijem principu. Konkretno, vodi se pretpostavkom da, pošto su riječi u tekstovima labelirane redom, jedna linija teksta traje sve dok je *xmin* nove riječi veće od *xmin* prošle riječi. Ukoliko to nije istina, trenutna riječ označava početak nove linije. Ovakav način određivanja linija je pogodan uzimajući u obzir format tekstova i korišteni redoslijed označavanja riječi. Funkcija *generate_line_labels_m* u *line_labels_m.txt* upisuje sadržaj istog formata kao i *generate_line_labels*. Pritom je značajno napomenuti da, za spajanje pojedinačnih riječi u odgovarajući tekst za dobivenu liniju, koristi i pomoćnu funkciju *assemble_text_with_spaces*, koja kontroliše dodavanje razmaka između riječi (razmak se dodaje iza svake riječi, izuzev ako je iduća riječ jedan od interpunkcijskih znakova “.”, “,”, “!”, “?”, “:”, “;”).

Primjer jednog ovakvog zapisa je:

LB_mix_005.jpg|2|3307|4676|218|630|2603|885|NISAM SE POTRUDIO DA MU ODGOVORIM.

Nakon poziva funkcija *generate_word_labels*, *generate_line_labels* i *generate_line_labels_m* se u *folder*-u *ground_truth* nalaze *word_labels.txt*, *line_labels.txt* i *line_labels_m.txt*, na osnovu kojih je moguće kreirati slike riječi/linija teksta.

2) Kreiranje slika pojedinačnih riječi/linija teksta

I. Definisana je funkcija ***extract_rectangle***, koja kao parametre prima izvornu sliku, koordinate gornjeg lijevog i donjeg desnog vrha pravougaonika, te opcionalno broj piksela za koji treba pomjeriti ova dva vrha, odnosno

efektivno povećati pravougaonik na oba kraja dijagonale. Funkcija vraća dio slike koji obuhvata ovaj pravougaonik.

II. Za kreiranje slika pojedinačnih riječi i njihovih finalnih labela, koristi se funkcija ***generate_word_images***, koja prima putanju do prethodno generisanog *word_labels.txt file*-a, putanju do izvornog *folder*-a sa slikama tekstova i putanju do odredišta, gdje će biti spašene kreirane slike i tekstualni file s labelama. U sklopu ove funkcije se, na osnovu podataka u *word_labels.txt*, učitava slika odgovarajućeg teksta, radi ekstrakcija slika riječi pomoću *extract_rectangle*, te tako dobivene slike spašavaju u *folder word* u proslijeđenom odredištu. Također se u odredišnom *folder*-u kreira *file word.txt*, koji ima skoro istu strukturu kao i *word_labels.txt*, s tim da se na naziv slike sada dodaje i indeks riječi. Odnosno, svaki red teksta će imati sljedeću strukturu:

filename_w_idx|*w_idx*|*w*|*h*|*xmin*|*ymin*|*xmax*|*ymax*|*name*, kao npr.

LB_mix_005_1.jpg|1|3307|4676|263|173|526|363|AKO

Za čitanje inicijalne i spašavanje nove slike, koristi se paket *cv2* iz *open source* biblioteke *OpenCV* za kompjutersku viziju, mašinsko učenje i procesiranje slika. [5] Funkcija *cv2.imread()* vraća pročitanu sliku kao *NumPy* niz oblika (visina, širina, broj_kanala), odnosno 3D matricu. Zahvaljujući ovome, u *extract_rectangle* je nad slikom moguće jednostavno vršiti operacije kao nad *NumPy* nizom. Na kraju, *cv2.imwrite()* spašava *NumPy* niz kao sliku.

III. Za kreiranje slika pojedinačnih linija teksta se koristi funkcija ***generate_line_images***, koja prima iste parametre kao *generate_word_images*, s tim da je za prvi parametar potrebno proslijediti putanju do *line_labels.txt*, odnosno *line_labels_m.txt*. Funkcija zapravo radi isto što i *generate_word_images*, s tim da se kreirane slike linija spašavaju u *folder line*, a labele u *file line.txt* u proslijeđenom odredištu. Svaki red u *line.txt* je u sljedećem formatu:

filename_l_idx|*l_idx*|*w*|*h*|*l_xmin*|*l_ymin*|*l_xmax*|*l_ymax*|*text*, kao npr.

LB_mix_005_5.jpg|5|3307|4676|218|630|2603|885|NISAM SE POTRUDIO DA

MU ODGOVORIM .

Za potrebe ovog projekta, kao odredišni *folder*-i su postavljeni *labels_w*, *labels_l* i *labels_l_m*, u kojima su kreirani *file*-ovi *word.txt* i *line.txt* za labele, odnosno *folder*-i *word* i *line* za slike.

Kada je u pitanju *dataset* za pojedinačne riječi, uzimajući u obzir prirodu rukopisa, očigledan rizik prilikom ovakvog kreiranja jesu poteškoće pri preciznom definisanju granica *bounding box*-a u toku samog označavanja, tako da nema

presijecanja s drugim riječima. Dakle, postoji mogućnost da se na slici neke riječi neće nalaziti samo ta riječ, već i dio neke druge.

Također, u slučaju *dataset*-a za pojedinačne linije, rizik predstavlja tačno određivanje riječi koje pripadaju nekoj liniji, posebno u slučaju neravnih linija teksta. Za označavanje jedne riječi koja nije ravno napisana je često potrebno kreirati i nešto veći *bounding box*, s obzirom da je *box* uvijek pravougaonik (kao na Slici 5) i nije ga moguće rotirati. Ovaj se problem primarno manifestuje prilikom ekstrakcije slika individualnih linija na osnovu koordinata *bounding box*-eva, gdje se na jednoj slici javljaju dijelovi više različitih linija teksta.

3) Podjela *dataset*-a na trening, validacijski i testni set

Za ovu podjelu je implementirana funkcija ***split_train_set***, koja prima putanju do *file*-a s labelama (*labels_w/word.txt*, *labels_l/line.txt* ili *labels_l_m/line.txt*), putanju odredišta, udjele trening, validacijskog i testnog seta (podrazumijevane vrijednosti su 0.7, 0.2 i 0.1), te opcionalni parametar *random_state* za funkciju *train_test_split* iz *sklearn.model_selection*. Ova funkcija vrši podjelu *dataset*-a u navedenom/podrazumijevanom omjeru koristeći i *train_test_split*, te kreira odgovarajuće datoteke *trainset.txt*, *validset.txt* i *testset.txt*, koje sadrže unose po setovima iz izvornog *file*-a s labelama. Za potrebe ovog projekta, odredišta za navedene *file*-ove za pojedinačne riječi i obje varijante linija teksta su *labels_w*, *labels_l* i *labels_l_m*.

Trenutne veličine *folder*-a su:

- *labels_w* - 66.2MB,
- *labels_l* - 66.6MB,
- *labels_l_m* - 67.4MB.

4) Kreiranje HDF5 formata *dataset*-a

Ovaj korak se radi lokalno. Potrebni projekat je moguće dobiti na neki od sljedeća dva načina:

- preuzeti i *unzip*-ovati *VI_HTR-main.zip* iz *folder*-a *repo* pratećeg *Google Drive folder*-a,
- klonirati repozitorij[4].

Nakon toga je potrebno pratiti uputstva data u dijelu *1.4 Kreiranje HDF5 formata dataset-a*, u okviru pratećeg *Google Colab notebook*-a.

Pritom se spomenuti projekat i uputstva oslanjaju na kod i tutorijalski *Google Colab notebook* iz [24], uz određene izmjene i dodatke potrebne za ovaj projekat, što će biti dodatno naglašeno.

[4] https://github.com/ekurtovic4/VI_HTR/tree/main

Kada su u pitanju koraci za kreiranje HDF5 *file*-a, ono što se izvršava pokretanjem

```
python main.py --source = <DATASET_NAME> --transform
```

jeste sljedeće:

I. Kreira se instanca klase *Dataset*, kojoj se kao *source* postavi putanja do *raw folder*-a (u kojem se nalazi folder s podacima), a kao *name* proslijeđeni <DATASET_NAME> argument.

II. Nad tim objektom se poziva metoda *read_partitions*, koja potom poziva metodu s nazivom _<DATASET_NAME> (u slučaju ovog projekta će to biti *_labels_w*, *_labels_l* ili *_labels_l_m*). Metoda pod nazivom *dataset*-a, u okviru klase *Dataset* u *reader.py*, treba da definiše kako pročitati taj dataset u skladu s njegovom strukturom, odnosno organizacijom podataka. Konkretno, svaka od tri navedene metode čita *file*-ove *trainset.txt*, *validset.txt* i *testset.txt*, na osnovu čega kreira rječnik sljedeće strukture:

   dataset[data_type] = {“dt”: [*putanje do slika riječi/linija*], “gt”: [*tekstovi na slikama riječi/linija*]},

   pri čemu *data_type* može biti “train”, “valid” ili “test”. Sve tri metode vraćaju kreirani rječnik *dataset*, koji se pamti u istoimenom atributu.

   Ove tri metode su naknadno implementirane za potrebe projekta, po uzoru na metode za čitanje drugih *dataset*-ova.

III. Nakon toga se nad *Dataset* objektom poziva metoda *save_partitions*, kojoj se prosljeđuje putanja do rezultujućeg <DATASET_NAME>.hdf5 *file*-a, veličina slike i maksimalna dužina teksta koji se veže uz sliku. HDF5 (*The Hierarchical Data Format version 5*) je kompresovan format *file*-ova koji podržava velike količine kompleksnih, heterogenih podataka. Podaci u ovakvom *file*-u se mogu struktuirano organizirati, kao u *file system*-u. [6]

   U svrhu rada s HDF5 *file*-om, koristi se *h5py* biblioteka. Upisivanje u rezultujući *file* se vrši u *batch*-evima, te se za svaku particiju (*train*, *valid*, *test*) u okviru *file*-a kreiraju po dva *dataset*-a:

   - [*particija*]/dt - preprocesirane slike

     Za preprocesiranje slika se koristi funkcija *preprocess* iz *preproc.py*, koja kao parametre prima sliku i željenu veličinu. Ova funkcija koristi implementiranu pomoćnu funkciju *imread*, koja čita sliku kao *grayscale* pomoću *cv2*, te određuje boju pozadine slike na osnovu najčešćeg piksela. Nakon čitanja i izračuna faktora skaliranja, slika se skalira na proslijeđenu željenu veličinu, pri čemu je očuvan njen inicijalni *aspect ratio*. Ukoliko je to potrebno, do tražene veličine se dodaju pikseli boje koja je određena kao pozadina. Funkcija *preprocess* vraća ovako preprocesiranu i transponiranu sliku.

- [*particija*]/gt - *ground truth* tekstovi sa slika, u binarnom formatu

Svi *dataset*-ovi su kompresovani *gzip* kompresijom bez gubitaka, s levelom kompresije 9. Podaci se kompresuju na putu do diska i automatski dekompresuju prilikom čitanja. [7]

Ovime se u *data folder*-u kreira odgovarajući HDF5 file, koji će predstavljati ulaz u model. Veličina ovako dobivenih *file*-ova je:

- *labels_w.hdf5* - 131.3MB,
- *labels_l.hdf5* - 33.5MB,
- *labels_l_m.hdf5* - 30.8MB.

Odabir, formiranje, treniranje i testiranje modela

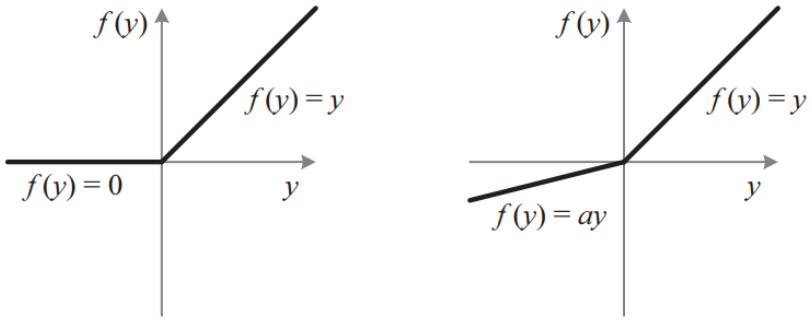
Za potrebe ovog projekta je korišten HTR-flor model implementiran koristeći programski jezik Python i okruženje TensorFlow. Implementacija slijedi korake iz originalnog rada [11] te kodove pripadajućeg GitHub repozitorija [24], uz manje prilagodbe specifične za korišteni skup podataka, eksperimentalne potrebe, te novije verzije korištenih biblioteka.

Arhitektura modela

Model se sastoji od dvije osnovne komponente, konvolucijskog i rekurentnog bloka, te izlaznog sloja.

Konvolucijski blok

Konvolucijski dio modela sadrži niz tzv. mini-blokova, koji kombinuju klasične i *gated* konvolucije, čiji će princip rada biti detaljnije objašnjen u nastavku. Sve konvolucijske operacije koriste *He uniform* inicijalizaciju, a kao aktivacijska funkcija koristi se PReLU (eng. *Parametric* ReLU). PReLU predstavlja unaprijeđenu verziju standardne ReLU funkcije aktivacije i za razliku od nje, adaptivno uči parametar koji određuje nagib u negativnom dijelu funkcije, čime povećava tačnost modela uz minimalan dodatni računski trošak [12]. Razlika između ove dvije funkcije je prikazana na sljedećoj slici.



Slika 6. Grafik ReLU i PReLU aktivacijskih funkcija [12]

Batch normalizacija se primjenjuje na izlaz svakog konvolucijskog sloja, a *Dropout* s vjerovatnoćom od 0.2 se koristi na posljednje tri *gated* konvolucije, s ciljem regularizacije.

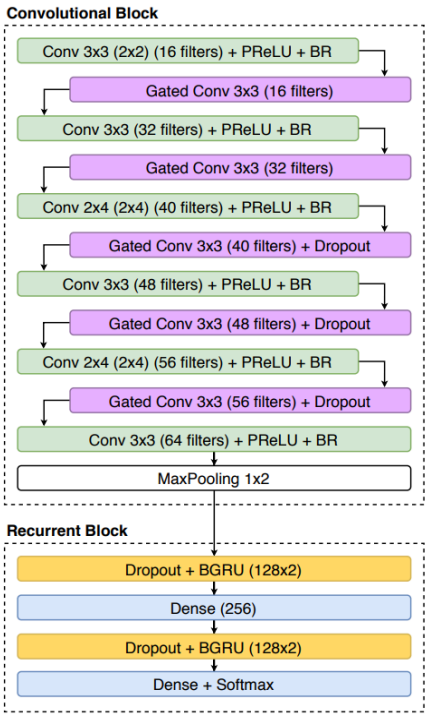
Rekurentni blok

Rekurentni dio modela se sastoji od dva bidirekciona GRU sloja (BGRU). GRU (eng. *Gated Recurrent Unit*) je tip rekurentne neuronske mreže koji koristi mehanizme kontrole protoka informacija kroz tzv. *gate*-ove, čime efikasno pamti važne informacije iz prethodnih koraka. BGRU je njegova proširena verzija koja obrađuje podatke istovremeno u oba smjera, što omogućava bolji kontekstualni uvid u sekvence. To znači da mreža pri obradi ne koristi samo informacije koje dolaze prije trenutnog elementa u nizu, već i one koje dolaze poslije, čime dobija potpuniju sliku [13]. Između ova dva sloja se nalazi jedan *dense* sloj. *Dropout* se također primjenjuje unutar GRU ćelija, s vrijednošću od 0.5.

Izlazni sloj

Na kraju modela se nalazi izlazni *dense* sloj čija je dimenzija jednaka broju znakova u definiranom skupu karaktera, uz dodatni simbol za tzv. CTC *blank token*. Model koristi CTC (eng. *Connectionist Temporal Classification*) pristup [14] za transkripciju, koji omogućava da se ulazna sekvenca (u našem slučaju slika riječi/linija) automatski poravna sa nizom izlaznih znakova (labelom), bez potrebe da se unaprijed zna tačan trenutak pojave svakog znaka. Drugim riječima, CTC omogućava modelu da sam nauči gdje se koji znak pojavljuje u sekvenci, bez ručnog označavanja svake pozicije.

Vizualni prikaz kompletne arhitekture modela je prikazan na slici u nastavku.



Slika 7. Arhitektura HTR-flor modela

Metrike

Za testiranje tačnosti predikcija modela, a time i njegove učinkovitosti, korištene su tri metrike, tipične za HTR modele. Riječ je o CER (eng. *Character Error Rate*), WER (eng. *Word Error Rate*) i SER (eng. *Sequence Error Rate*) metrikama [25]. Za razumijevanje ovih metrika, neophodno je najprije razumjeti *Levenshtein* udaljenost koja daje osnovu za njihovo računanje.

Levenshtein udaljenost (eng. *Levenshtein distance*, LD) predstavlja mjeru sličnosti dva stringa i računa se kao ukupan broj karaktera koje treba obrisati, dodati ili izmijeniti da se izvorni string (*s*) prevede u ciljani string (*t*). Posljedično, što je ova vrijednost veća, to su stringovi više različiti i obrnuto [26].

CER metrika predstavlja ukupnu pogrešku u predikciji pojedinačnih slova. U sklopu projekta je implementirana na način prikazan na slici ispod, gdje *prediction* i *ground_truth* predstavljaju, respektivno, predikciju modela i stvarnu transkripciju teksta sa slike.

```
pd_cer, gt_cer = list(prediction), list(ground_truth)

dist = editdistance.eval(pd_cer, gt_cer)
cer.append(dist / (max(len(pd_cer), len(gt_cer))))
```

Slika 8. Implementacija proračuna CER metrike

Pretvaranjem predikcije i stvarne labele u liste, dobija se niz pojedinačnih karaktera. Ovi karakteri se potom porede koristeći *Levenshtein* udaljenost. Dobivena vrijednost se zatim dijeli s brojem slova duže riječi. Na ovaj način se dobiva mjera pogrešnih karaktera na nivou pojedinačnih slova.

WER metrika se računa na vrlo sličan način. Suštinska razlika se ogleda u tome što *Levenshtein* udaljenost vraća ukupan broj riječi unutar kojih je neophodno napraviti neku od izmjena karaktera, neovisno koliko tih izmjena ima unutar pojedinačne riječi. Normalizacija se također vrši sa ukupnim brojem riječi. Izračun ove metrike je vidljiv na slici ispod.

```
pd_wer, gt_wer = prediction.split(), ground_truth.split()

dist = editdistance.eval(pd_wer, gt_wer)
wer.append(dist / (max(len(pd_wer), len(gt_wer))))
```

Slika 9. Implementacija proračuna WER metrike

SER metrika mjeri postotak sekvenci koje se ne podudaraju tačno s ciljnim sekvencama. Svaka sekvenca se smatra netačnom ako se razlikuje od odgovarajuće ciljane sekvence, bez obzira na broj i vrstu razlika. Implementacija ove metrike je vidljiva na slici ispod.

```
pd_ser, gt_ser = [prediction], [ground_truth]

dist = editdistance.eval(pd_ser, gt_ser)
ser.append(dist / (max(len(pd_ser), len(gt_ser))))
```

Slika 10. Implementacija proračuna SER metrike

Implementacija modela

Implementacija modela je sadržana u dvije osnovne Python datoteke, *model.py* i *layers.py*. Prva od njih sadrži samu definiciju modela, kao i funkcije za treniranje i testiranje, dok druga datoteka sadrži implementaciju jedne od osnovnih komponenti ovog modela, a to je *FullGatedConv2D* sloj. Osnovna funkcija ove klase prikazana je na sljedećoj slici.

```
def call(self, inputs):
        output = super(FullGatedConv2D, self).call(inputs)
        linear = Activation("linear")(output[:, :, :, :self.nb_filters])
        sigmoid = Activation("sigmoid")(output[:, :, :, self.nb_filters:])

        return Multiply()([linear, sigmoid])
```

Slika 11. Implementacija obrade filtera *FullGatedConv2D* sloja

Priložena funkcija se poziva s filterima konvolucijskog sloja kao ulazima i ovaj broj kanala se udvostruči. Na prvu polovinu se primjenjuje linearna aktivacija, dok se na drugu polovinu primjenjuje sigmoidna aktivacija, koja vrijednosti ograničava na opseg [0, 1]. Dobijene dvije polovine se zatim množe element po element.

Rezultat ove operacije je da se linearno aktivirani izlazi moduliraju (tj. pojačavaju ili prigušuju) u zavisnosti od vrijednosti koje je proizvela sigmoidna grana. Ako je neka vrijednost iz sigmoidne grane blizu 1, odgovarajući element iz linearne grane se gotovo u potpunosti propusti, a ako je blizu 0, potisne se. Na taj način, mreža uči filtrirati informacije. Ovo doprinosi boljoj sposobnosti modela da selektivno reaguje na relevantne obrasce u slici, što je posebno korisno kod zadataka kao što je analiza složenih vizuelnih struktura.

Datoteka *model.py* sadrži definiciju *HTRModel* klase, što uključuje definiciju metoda za treniranje i testiranje modela. Funkcija treniranja poziva bibliotečnu *fit()* funkciju, prosljeđujući joj zadane parametre.

```
out = self.model.fit(
      x=x, y=y, batch_size=batch_size, epochs=epochs, verbose=verbose,
      callbacks=callbacks, validation_split=validation_split,
      validation_data=validation_data, shuffle=shuffle,
      class_weight=class_weight, sample_weight=sample_weight,
      initial_epoch=initial_epoch, steps_per_epoch=steps_per_epoch,
      validation_steps=validation_steps, validation_freq=validation_freq
  )
```

Slika 12. Funkcija za treniranje HTR-flor modela

Funkcija za testiranje se oslanja na bibliotečnu *predict()* funkciju, uz dodatno CTC dekodiranje koje omogućava transformaciju izlaza modela u čitljiv tekstualni oblik. Dobijeni rezultati predstavljaju konačnu transkripciju modela, zajedno s pripadajućim vjerovatnoćama. Za treniranje se koristi CTC gubitak.

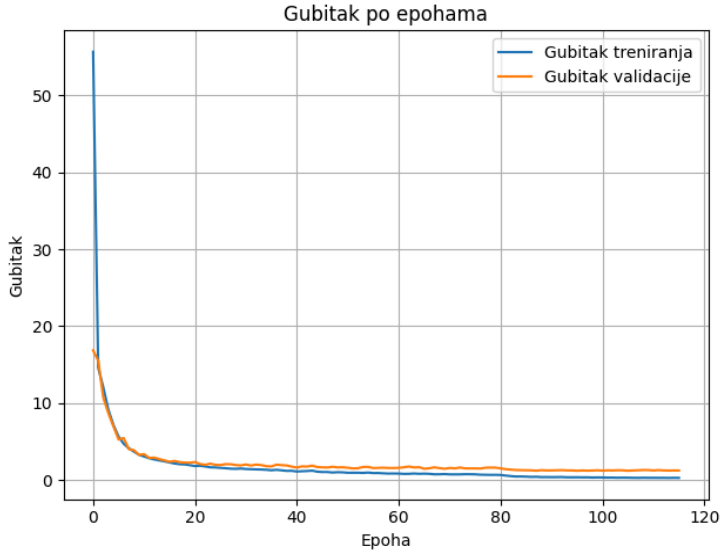
Dodatno, implementirani su mehanizmi ranog zaustavljanja i dinamičkog prilagođavanja parametra stope učenja, pri čemu se obje strategije oslanjaju na praćenje promjena funkcije gubitka na validacijskom skupu podataka. Rano zaustavljanje se aktivira ukoliko ne dođe do poboljšanja validacijskog gubitka tokom 20 uzastopnih epoha, dok se stopa učenja automatski smanjuje faktorom 0.1 nakon 15 epoha bez napretka.

Model koristi AdamW optimizator uz dodatnu normalizaciju gradijenata. AdamW je unaprijeđena verzija adaptivnog optimizatora Adam, koja uvodi tehniku *weight decay* za regularizaciju. Ova tehnika podrazumijeva upravljanje težinskim faktorima nezavisno od same funkcije gubitka [15]. Normalizacija gradijenata se ostvaruje korištenjem prilagođene klase *NormalizedOptimizer*. Ova klasa obavija osnovni optimizator i prije primjene ažuriranja parametara vrši L2 normalizaciju gradijenata, čime se njihova veličina ograničava. Ovime se sprječava da pojedini veliki gradijenti uzrokuju nestabilne promjene u težinskim faktorima modela, što doprinosi stabilnijem procesu treniranja.

Trenirana su ukupno tri HTR-flor modela s ciljem određivanja najbolje reprezentacije korištenih podataka. Sva tri modela su inicijalizirana istim hiperparametrima, izuzev *batch size*-a.

Model treniran na *labels_w* skupu podataka

Prvi od testiranih modela, ujedno i najbolji po metrikama, je treniran na pojedinačnim riječima, odnosno *labels_w* skupu podataka. Zbog većeg obima *dataset*-a, veličina *batch*-a je postavljena na 64 s ciljem bržeg treniranja. Broj epoha je inicijalno postavljen na 300, ali je rano zaustavljanje aktivirano nakon 116 epoha, pri čemu se stopa učenja dinamički umanjila dva puta. Gubitak treniranja i validacije je vidljiv na sljedećoj slici.



Slika 13. Gubitak po epohama za model treniran na *labels_w* skupu podataka

Gubitak treniranja dostiže minimalnu vrijednost 0.385, a gubitak validacije 1.229. Iako je validacijski gubitak nešto veći od gubitka treniranja, ovo su i dalje zadovoljavajući rezultati i nema naznaka *overfitting*-a. Najvjerovatniji razlog ovome je aktivacija mehanizma ranog zaustavljanja, budući da je s grafika uočljivo da se krivulje ova dva gubitka počinju blago udaljavati što se broj epoha povećava.

Nakon treniranja, model je testiran na testnom skupu podataka, gdje je za metrike CER, WER i SER ostvario, respektivno, rezultate 0.0657, 0.1964 i 0.1964.

Budući da je model treniran i evaluiran na nivou pojedinačnih riječi, SER metrika u ovom slučaju odgovara WER metrici, jer se svaka riječ tretira kao zasebna sekvenca. Ove vrijednosti ukazuju na to da se oko 19.64% predikcija modela na nivou riječi u određenoj mjeri ne podudara sa stvarnim transkripcijama. Posmatrano na nivou karaktera, greška iznosi svega 6.57%.

Primjeri predikcija modela na testnom skupu podataka su vidljivi na sljedećim slikama.

Ispravna labela: SUSAM
Predikcija modela: SUSAM

Ispravna labela: TOBOM
Predikcija modela: TOBOM

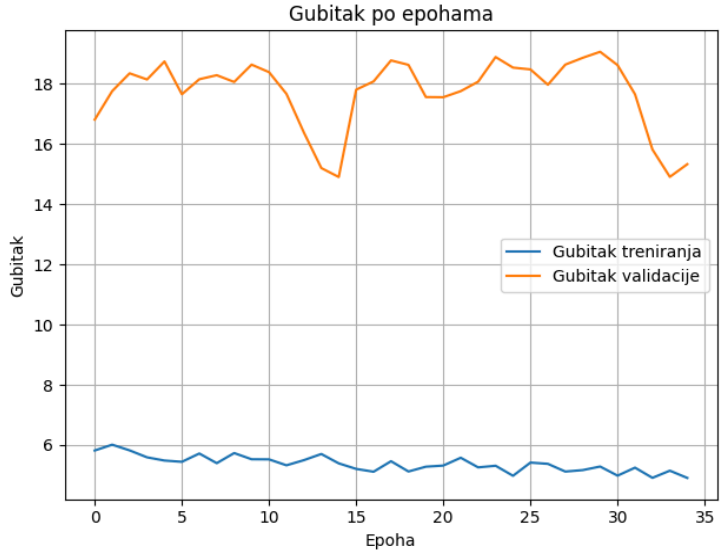
Slika 14. Primjeri rezultata modela na testnom skupu podataka

Model treniran na *labels_l* skupu podataka

Drugi HTR-flor model je treniran i testiran na *labels_l* skupu podataka, odnosno linijama teksta koje nastaju spajanjem slika i labela pojedinačnih riječi iz *labels_w dataset*-a. Kao rezultat toga, dobijeni skup je znatno manjeg obima u poređenju s izvornim *labels_w* podacima, što se negativno odrazilo na performanse modela.

Kada je riječ o postavkama modela, zadržana je ista konfiguracija kao u prethodnom slučaju, izuzev što je veličina *batch*-a smanjena na 16, budući da je vrijeme potrebno za treniranje na ovim podacima bilo znatno kraće. Rano zaustavljanje se aktiviralo već nakon 35 epoha, što se dodatno odrazilo na performanse modela. Konačni gubitak treniranja je iznosio 5.391, a testiranja 14.907. Razlika između ovih vrijednosti bila je primjetna još od početka treniranja. Također, gubitak validacije je kroz epohe oscilirao, bez jasnog trenda ka stabilnoj optimizaciji, dok je gubitak treniranja pokazivao blagi pad, ali bez značajnog napretka.

Glavni razlog za ovakve rezultate leži u kvaliteti samih podataka. Naime, budući da je metoda generisanja linija kojom je formiran *labels_l* skup podataka dovela do brojnih grešaka, slike su često sadržavale znatno veće segmente teksta od jedne linije, a pripadajuće labele nisu uvijek bile precizno usklađene sa sadržajem slika. Posljedica toga je nemogućnost modela da prepozna relevantne obrasce i izvuče korisne značajke iz podataka. Gubitak treniranja i validacije je vidljiv na sljedećoj slici.

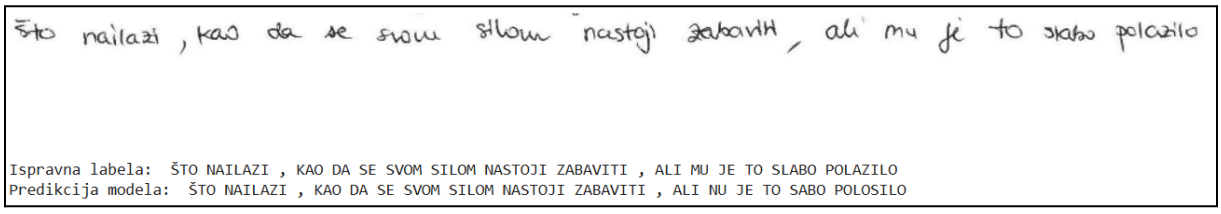


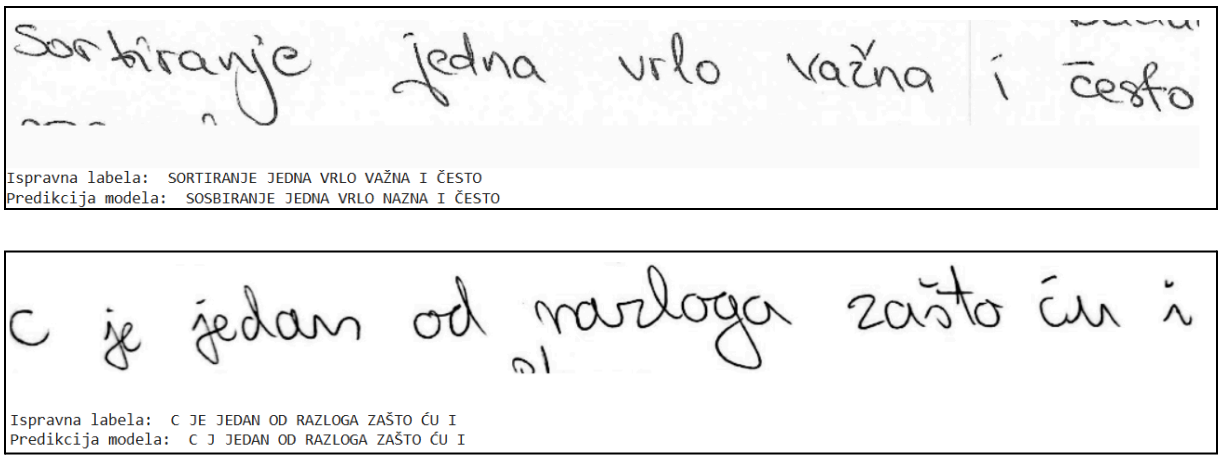
Slika 15. Gubitak po epohama za model treniran na *labels_l* skupu podataka

CER, WER i SER metrike iznose, redom, 0.1164, 0.2796 i 0.7688. Ovi rezultati su znatno lošiji u poređenju s rezultatima modela *labels_w*. Iako je greška na nivou karaktera relativno niska (11.64%), čak 76.9% sekvenci sadrži barem jednu grešku u predikciji. Ovo može ukazivati na to da model često pravi manje greške unutar sekvenci, ali i na postojanje određenih sistematskih (tipskih) grešaka koje se ponavljaju.

Jedan od potencijalnih načina za ublažavanje ovog problema, pored korištenja kvalitetnijeg skupa podataka, jeste popuštanje kriterija za aktivaciju ranog zaustavljanja. Time bi se modelu omogućilo dodatno vrijeme za učenje, čak i uz sporiju optimizaciju. Ipak, u praksi se pokazalo da ova strategija nije dovela do značajnog poboljšanja performansi, dok preveliko povećanje tolerancije može rezultirati *overfitting*-om.

Uzorak rezultata modela na testnom skupu podataka je vidljiv na slikama ispod.



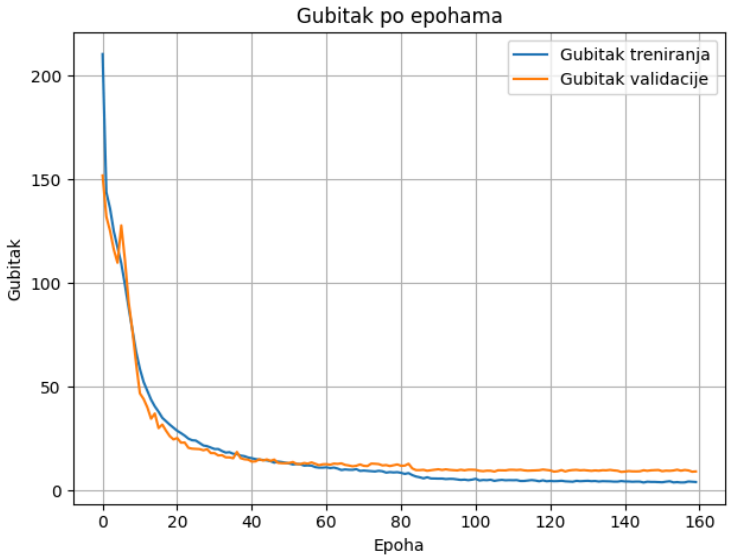


Slika 16. Primjeri rezultata modela na testnom skupu podataka

Model treniran na *labels_l_m* skupu podataka

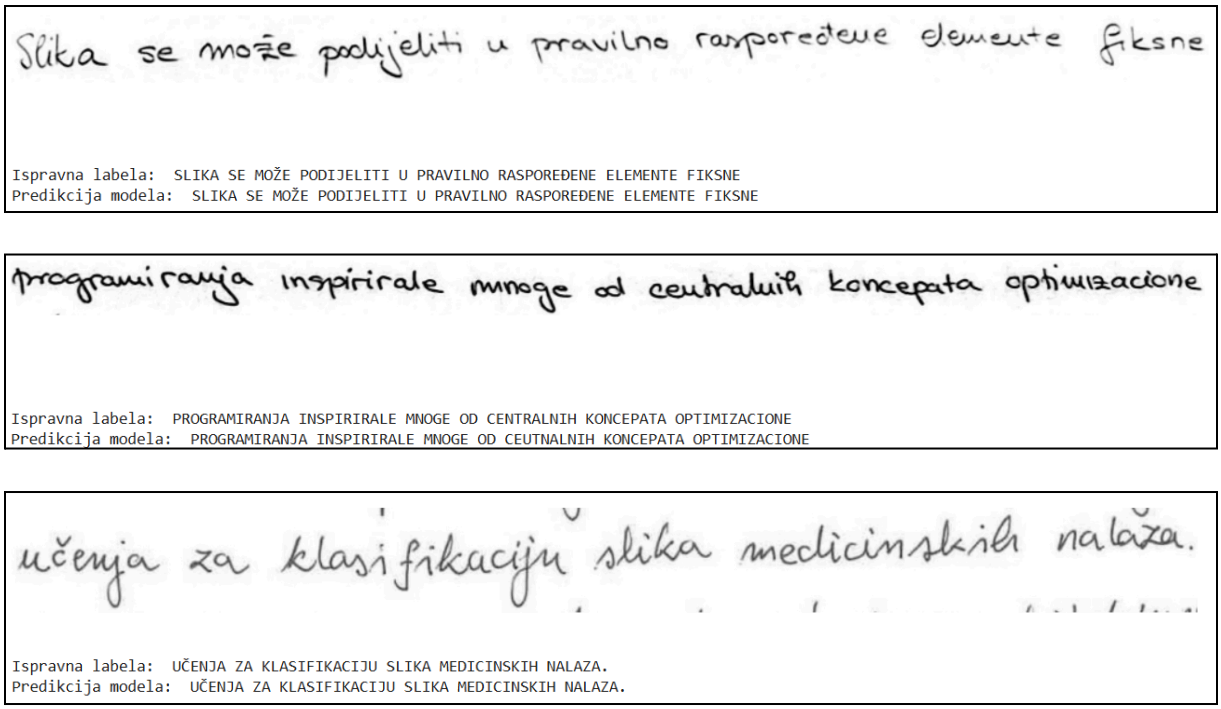
Zbog izrazito loših performansi prethodnog modela, koje su bile direktna posljedica nekvalitetnog skupa podataka korištenog za treniranje, naredni korak je podrazumijevao optimizaciju funkcije za spajanje pojedinačnih riječi u cilju generisanja kvalitetnijih ulaznih podataka.

Na poboljšanom skupu podataka testiran je isti model, pri čemu su postignuti bolji rezultati. Gubitak treniranja je iznosio 4.645, dok je gubitak validacije iznosio 9.037. U ovom slučaju, mehanizam ranog zaustavljanja je aktiviran tek nakon 160 epoha, što je modelu omogućilo dublju i stabilniju prilagodbu podacima. Vizualni prikaz trenda gubitaka tokom treniranja i validacije prikazan je na narednoj slici.



Slika 17. Gubitak po epohama za model treniran na *labels_l_m* skupu podataka

Vrijednosti evaluacijskih metrika CER, WER i SER na testnom skupu podataka su iznosile 0.0812, 0.2674 i 0.7174, respektivno. Iako je model treniran duže i na kvalitetnijem skupu podataka, a funkcija gubitka pokazuje jasne znakove optimizacije, poboljšanja performansi na testnom skupu ostaju ograničena. Broj grešaka na nivou pojedinačnih karaktera jeste smanjen, ali se postotak netačnih sekvenci nije značajno promijenio. Rezultati su vidljivi na sljedećim slikama.



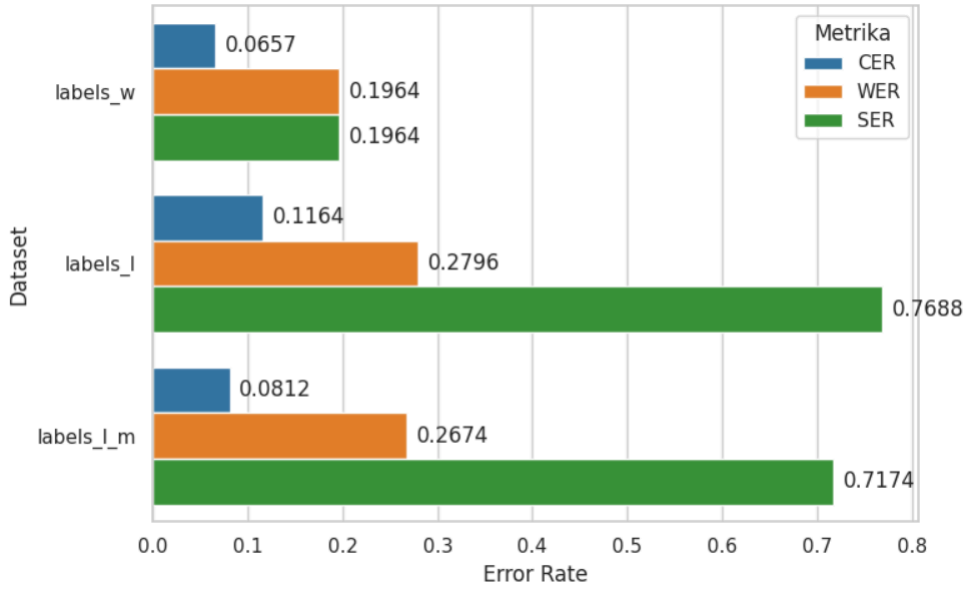
Slika 18. Primjeri rezultata modela na testnom skupu podataka

Ovakvi rezultati mogu ukazivati na više potencijalnih problema. Prvi predstavlja ograničen obim dostupnih podataka. Za efikasno treniranje modela ove složenosti, neophodan je veći i raznovrsniji skup podataka, koji bi omogućio bolje učenje i stabilniju generalizaciju. Drugo, mogući uzrok problema ostaje i nedovoljna usklađenost između slika i pripadajućih labela. Iako je izvršena dodatna obrada, greške u poravnanju teksta i slike i dalje mogu biti prisutne, što negativno utiče na sposobnost modela da nauči relevantne obrasce iz podataka.

Ovi rezultati ukazuju na potrebu za daljim unapređenjem skupa podataka, kroz njegovo proširenje i poboljšanje kvaliteta, kao i na mogućnost dodatnog prilagođavanja arhitekture modela.

Cjelokupni osvrt na problem i dobiveno rješenje

Posmatrajući vrijednosti korištenih metrika CER, WER i SER za različite skupove podataka (*labels_w*, *labels_l* i *labels_l_m*) na Slici 19, zaključuje se da najbolje performanse pokazuje model treniran na pojedinačnim riječima.



Slika 19. Vrijednosti metrika za korištene dataset-ove

Iako je broj grešaka na nivou pojedinačnih karaktera približno isti za sva tri skupa podataka i generalno pokazuje dobre rezultate (CER je u rasponu od 6.57% do 11.64%), to nije slučaj za pojedinačne riječi ili linije teksta. Modeli trenirani na *labels_l* i *labels_l_m* pokazuju lošije performanse na nivou riječi (WER je 27.96% i 26.7%, u odnosu na 19.64% za *labels_w*), kao i prilično loše performanse na nivou sekvenci teksta (SER je 76.9% i 71.7%, za *labels_w* se *de facto* svodi na WER).

Uzimajući u obzir da WER i SER metrike ne prave razliku između broja i vrste razlika u riječi/sekvenci, kao i prikazane rezultate testiranja (Slike 14, 16 i 18), može se pretpostaviti da model često pravi manje greške, ali se iste akumuliranjem odražavaju na konačne vrijednosti metrika.

U poređenju s drugim arhitekturama, HTR-flor model, implementiran kao CNN-GRU-CTC arhitektura, ostvario je značajne rezultate. Najbolje rezultate je postigao na *labels_w* skupu podataka: CER od 6.57% i WER od 19.64%. Referirani radovi bazirani na CNN-LSTM-CTC arhitekturama na *IAM dataset*-u, kada ne koriste jezični model (eng. *language model*), pokazuju WER-ove od 24.6% (za 300 dpi) ili čak 29.5% (za 150 dpi) [27]. S obzirom na to, HTR-flor model demonstrira konkurentne, pa čak i bolje performanse. Ipak, najnapredniji modeli danas, poput

transformer baziranog TrOCR-a, koji koristi napredne strategije pretreninga i ne zahtijeva jezični model zahvaljujući pretreniranim transformer dekoderima, postavljaju znatno više standarde u polju, dostižući CER od 2.97% na *IAM dataset*-u [19].

Važno je istaći i da postoji prostor za dodatna poboljšanja HTR-flor implementacije prikazane u ovom projektu. Kako bi se povećala efikasnost modela, bilo bi korisno detaljnije istražiti uticaj različitih hiperparametara (npr. veličina *batch*-a, *learning rate*, *weight decay* i dr.), kroz sistematsku optimizaciju, ručno ili pomoću automatiziranih alata. Zbog tehničkih ograničenja okruženja Google Colab, prije svega u pogledu vremena izvođenja i dostupne memorije, opsežnija evaluacija nije bila moguća, te su parametri testirani samo u ograničenom broju varijacija. Dodatna izmjena koja bi nesumnjivo pospješila treniranje modela za skupove *labels_l* i *labels_l_m* jeste povećanje obima ulaznih podataka. Kao što je već istaknuto u prethodnim poglavljima, trenutni broj primjera u ovim skupovima nije dovoljan za efikasno učenje modela ove složenosti. Osim toga, i kvalitet anotacija igra važnu ulogu. Pristup korišten u ovom radu, koji podrazumijeva spajanje riječi u linije na osnovu njihove pozicije, iako relativno uspješan (posebno za *labels_l_m*), sadrži greške. Ove greške, pored ograničenog broja primjera, vjerovatno doprinose slabijim rezultatima modela treniranim na tim skupovima.

Literatura

[1] “HumanSignal/labelImg”, GitHub, https://github.com/HumanSignal/labelImg, pristupano: 20.6.2025.

[2] “HumanSignal/labelImg”, Usage - Steps (PascalVOC), GitHub, https://github.com/HumanSignal/labelImg?tab=readme-ov-file#steps-pascalvoc, pristupano: 20.6.2025.

[3] Sachinsoni, “Clustering Like a Pro: A Beginner’s Guide to DBSCAN”, Medium, 26.12.2023. https://medium.com/@sachinsoni600517/clustering-like-a-pro-a-beginners-guide-to-dbscan-6c8274c362c4, pristupano: 20.6.2025.

[4] scikit-learn, “sklearn.cluster.DBSCAN — scikit-learn 0.22 documentation”, Scikit-learn.org, 2017. https://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html, pristupano: 20.6.2025.

[5] GeeksforGeeks, “OpenCV Tutorial in Python”, GeeksforGeeks, 30.1.2020, https://www.geeksforgeeks.org/python/opencv-python-tutorial/, pristupano: 20.6.2025.

[6] “Hierarchical Data Formats - What is HDF5? | NSF NEON | Open Data to Understand our Ecosystems”. www.neonscience.org, https://www.neonscience.org/resources/learning-hub/tutorials/about-hdf5, pristupano: 20.6.2025.

[7] “Datasets - h5py 3.13.0 documentation”. H5py.org, 2025, https://docs.h5py.org/en/stable/high/dataset.html, pristupano: 20.6.2025.

[8] “Research Group on Computer Vision and Artificial Intelligence - Computer Vision and Artificial Intelligence”. *fki.tic.heia-fr.ch*, https://fki.tic.heia-fr.ch/databases/iam-handwriting-database, pristupano: 21.6.2025.

[9] E. Grosicki, M. Carré, E. Geoffrois, E. Augustin, F. Preteux, R. Messina. “RIMES, complete”, *Zenodo*, mart 2024, doi: https://doi.org/10.5281/zenodo.10812725, pristupano: 21.6.2025.

[10] F. Hollaus, “CVL-Database | Computer Vision Lab”. *Tuwien.ac.at*, 17.02.2015, https://cvl.tuwien.ac.at/research/cvl-databases/an-off-line-database-for-writer-retrieval-writer-identification-and-word-spotting/, pristupano: 21.6.2025.

[11] de Sousa Neto, Arthur Flor, et al. “HTR-Flor: A deep learning system for offline handwritten text recognition”. 2020 33rd SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI), IEEE, 2020.

[12] He, Kaiming, et al. “Delving deep into rectifiers: Surpassing human-level performance on imagenet classification”. Proceedings of the IEEE international conference on computer vision, 2015.

[13] Chen, X. Jingxia, D. M. Jiang, Y. N. Zhang. “A hierarchical bidirectional GRU model with attention for EEG-based emotion classification”. Ieee Access 7 (2019): 118530-118540.

[14] Graves, Alex, et al. “Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks”. Proceedings of the 23rd international conference on Machine learning, 2006.

[15] Loshchilov, Ilya, Frank Hutter. “Fixing weight decay regularization in adam”. arXiv.1711.05101 5 (2017): 5.

[16] R. Plamondon, S. N. Srihari. “On-Line and Off-Line Handwriting Recognition: A Comprehensive Survey”. IEEE Transactions on Pattern Analysis and Machine Intelligence, 22(1):63-84, 2000.

[17] Y. LeCun, Y. Bengio, G. Hinton. “Deep learning”. Nature, 521(7553):436-44, 2015.

[18] A. Graves, S. Fernandez, M. Liwicki, R. Bertolami. “A Novel Connectionist System for Unconstrained Handwriting Recognition”. IEEE Transactions on Pattern Analysis and Machine Intelligence, 31(5):855-68, 2009.

[19] M. Li, T. Lv, L. Cui, Y. Lu. “TrOCR: Transformer-based Optical Character Recognition with Pre-trained Models”. arXiv.2109.10282, 2021.

[20] Nizar Charrada. “KHATT-Arabic”, *Kaggle.com*, 2023, https://www.kaggle.com/datasets/nizarcharrada/khattarabic, pristupano: 22.6.2025.

[21] M. S. Deshmukh, “Unconstrained Handwritten Modi Character Dataset”, *Kaggle.com*, 2024, https://www.kaggle.com/datasets/msd6013/unconstrained-handwritten-modi-character-dataset/data, pristupano: 22.6.2025.

[22] J. A. Sánchez, “Bentham Dataset R0”, *Zenodo.org*, 2016, https://zenodo.org/records/44519, pristupano: 25.6.2025.

[23] A. Begovac, “masters-theses-HTR”, GitHub, https://github.com/abegovac2/masters-theses-HTR, pristupano: 17.6.2025.

[24] “handwritten-text-recognition”, GitHub, https://github.com/arthurflor23/handwritten-text-recognition, pristupano: 17.6.2025.

[25] C. S. Anoop, A. G. Ramakrishnan. “CTC-based end-to-end ASR for the low resource Sanskrit language with spectrogram augmentation”. 2021 national conference on communications (NCC), IEEE, 2021.

[26] Haldar, Rishin, Debajyoti Mukhopadhyay. “Levenshtein distance technique in dictionary lookup methods: An improved approach”. arXiv.1101.1232, 2011.

[27] T. Bluche. “Joint Line Segmentation and Transcription for End-to-End Handwritten Paragraph Recognition”. arXiv.1604.08352, 2016.