Generiranje slika sa zakrivljenim tekstom i ispitivanje performansi postojećih metoda za detekciju teksta

Landeka, Leon

Master's thesis / Diplomski rad

2022

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Josip Juraj Strossmayer University of Osijek, Faculty of Electrical Engineering, Computer Science and Information Technology Osijek / Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:200:094485

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni studij

GENERIRANJE SLIKA SA ZAKRIVLJENIM TEKSTOM I ISPITIVANJE PERFORMANSI POSTOJEĆIH METODA ZA DETEKCIJU TEKSTA

Diplomski rad

Leon Landeka

Osijek, 2022.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

Obrazac D1: Obrazac za imenovanje Povjerenstva za diplomski ispit

Osijek, 04.12.2022.

Odboru za završne i diplomske ispite

Imenovanje Povjerenstva za diplomski ispit

Ime i prezime Pristupnika:	Leon Landeka
Studij, smjer:	Diplomski sveučilišni studij Automobilsko računarstvo i komunikacije
Mat. br. Pristupnika, godina upisa:	D-50ARK, 11.10.2020.
OIB studenta:	04482244739
Mentor:	Izv.prof.dr.sc. Ratko Grbić
Sumentor:	,
Sumentor iz tvrtke:	Matteo Brisinello
Predsjednik Povjerenstva:	Prof. dr. sc. Marijan Herceg
Član Povjerenstva 1:	Izv.prof.dr.sc. Ratko Grbić
Član Povjerenstva 2:	Izv. prof. dr. sc. Mario Vranješ
Naslov diplomskog rada:	Generiranje slika sa zakrivljenim tekstom i ispitivanje performansi postojećih metoda za detekciju teksta
Znanstvena grana diplomskog rada:	**Umjetna inteligencija (zn. polje računarstvo)**
Zadatak diplomskog rada:	Detekcija teksta na slikama je vrlo izazovan zadatak iz nekoliko razloga. Tekst na slikama može se pojaviti u raznim veličinama, bojama i oblicima. Također, tekst se može pojaviti i u različitim vrstama pisama (latinica, ćirilica, kinesko pismo, arapsko pismo, itd.). Trenutno, aktualne metode detekcije teksta dotakle su se izazova u kojem se tekst na slici nalazi u zakrivljenom obliku (polukrug, kružni oblik, oblik zmije, itd.). Zadatak ovog rada je umjetno generirati određeni broj slika koje sadrže zakrivljeni tekst. Nakon generiranja skupa podataka potrebno je istrenirati Mask R-CNN mrežu na stvarnim podacima (Total-Text skup podataka), te istu mreže potrebno je trenirati na stvarnim podacima (Total-Text) + umjetno generiranim podacima
Prijedlog ocjene pismenog dijela ispita (diplomskog rada):	Dobar (3)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 1 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 2 bod/boda Jasnoća pismenog izražavanja: 2 bod/boda Razina samostalnosti: 2 razina
Datum prijedloga ocjene od strane mentora:	04.12.2022.

Potvrda mentora o predaji konačne verzije rada:	*Mentor elektronički potpisao predaju konačne verzije.*
	Datum:

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA
I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

IZJAVA O ORIGINALNOSTI RADA

Osijek, 21.12.2022.

Ime i prezime studenta:	Leon Landeka
Studij:	Diplomski sveučilišni studij Automobilsko računarstvo i komunikacije
Mat. br. studenta, godina upisa:	D-50ARK, 11.10.2020.
Turnitin podudaranje [%]:	1

Ovom izjavom izjavljujem da je rad pod nazivom**: Generiranje slika sa zakrivljenim tekstom i ispitivanje performansi postojećih metoda za detekciju teksta**

izrađen pod vodstvom mentora Izv.prof.dr.sc. Ratko Grbić

i sumentora ,

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

SADRŽAJ

1. UVOD

Detekcija teksta na slikama vrlo je izazovan zadatak jer se tekst može pojaviti u različitim oblicima, veličina, bojama, pismima i slično tomu. Jedan od oblika teksta jest zakrivljeni koji se pojavljuje u nekoliko različitih oblika poput polukruga i zakrivljene linije. S obzirom na navedeno, treba napomenuti kako upravo nepravilni oblici dodatno otežavaju postupak detekcije. Može se reći, kako je prepoznavanje tekstualnih znakova, postupak kojim se tekst čita s digitalne slike uz pomoć računalnog programa. Budući da se tekst pojavljuje u svakodnevnoj okolini, potrebna je sve veća primjena algoritama za detekciju teksta te se takva primjena proširila i u automobilskoj industriji. Takvi algoritmi omogućuju čitanje teksta s prometnih znakova, registracijskih oznaka i slično. Treba naglasiti kako za čovjeka, detekcija teksta predstavlja prilično jednostavnu radnju, dok je računalu to složen proces koji zahtijeva različite algoritme za obradu slike.

Moderni algoritmi detekcije teksta temelje se na dubokim neuronskim mrežama za čije su učenje potrebni kvalitetno označeni skupovi podataka. Nedavno je, pojava mnogih tekstualnih skupova podataka pridonijela značajnom napretku za detekciju i prepoznavanje teksta. Spomenuti skupovi podataka sastoje se od slika na kojima se tekst nalazi u prirodnim scenama, gdje za svaku sliku, u zasebnoj datoteci, postoje oznake gdje se nalazi tekst. Najčešći oblik oznaka su granični okviri u obliku pravokutnika. Bitno je napomenuti kako su pokraj metoda koje koriste granične pravokutnike s poravnatom x-osi, nastale metode koje koriste rotirane granične pravokutnike, tzv. više-orijentirane granične pravokutnike.

Također, dodatna motivacija za rad pojavljuje se jer je primijećeno kako većina dostupnih skupova podataka za učenje algoritma za detekciju teksta, sadrže ravne i više-orijentirane instance teksta, dok samo neznatan broj skupova podataka sadrži zakrivljene instance teksta. Cilj rada je ispitati poboljšavaju li se performanse detekcije teksta tako da se pri treniranju neuronskih mreža koriste skupovi podataka, koji za oznake teksta, umjesto graničnih pravokutnika imaju granični okvir u obliku poligona. Preciznije će se oznake teksta postići modifikacijom postojeće skripte za generiranje i označavanje slika.

U drugom poglavlju dan je pregled problema detekcije zakrivljenog teksta i pregled postojećih skupova podataka te rješenja za detekciju zakrivljenog teksta. Trećim poglavljem opisan je proces generiranja zakrivljenog teksta na slikama, kao i način označavanja generiranog skupa podataka te sam proces treniranja i testiranja neuronske mreže. U četvrtom je poglavlju opisana evaluacija dobivenih rezultata za detekciju zakrivljenog teksta, dok je na samom kraju rada dan zaključak.

2. PREGLED POSTOJEĆIH RJEŠENJA

U ovom su poglavlju opisani problemi koji se mogu pojaviti prilikom detekcije zakrivljenog teksta. Također, dan je opis postojećih skupova podataka koji se mogu koristiti za detekciju teksta. Osim toga, u zadnjem je dijelu opisano nekoliko postojećih rješenja za detekciju zakrivljenog teksta.

2.1. Problem detekcije teksta

Detekcija teksta podrazumijeva detekciju svih prisutnih instanci teksta s graničnim okvirima na ulaznoj slici pomoću određenog algoritma. Kao što je već spomenuto, detekcija teksta predstavlja veliki izazov jer se tekst na slikama pojavljuje u raznim veličinama, bojama i oblicima. Također, tekst se može pojaviti i u različitim pismima poput latinice, ćirilice, kineskog pisma, arapskog pisma i mnogim drugim pismima. Ono što treba istaknuti jest to da zadatak detekcije teksta dodatno otežava pojava teksta u različitim orijentacijama. Najtočnije metode detekcije teksta na slikama temelje se na dubokim neuronskim mrežama. Najčešće korištene mreže su mreže za detekciju objekata, primjerice, *Mask-R-CNN* [1] koje će biti korištene za potrebe ovoga rada.

2.2. Pregled postojećih skupova podataka za detekciju teksta

U današnje vrijeme postoji više gotovih skupova podataka za izradu algoritama detektora teksta te će neki od tih skupova biti opisani u nastavku.

1. ***SynthText*** – objavljen 2016. godine i sadrži preko 800 000 umjetno generiranih slika. Uglavnom se koristi za treniranje mreža s podacima iz stvarnoga svijeta. Anotacija se sastoji od osi poravnatog graničnog okvira na bazi riječi i na bazi znakova s transkripcijom. Motivacija za izgradnju ovog skupa podataka jest ta što su postojeći skupovi podataka generalno mali [2].

2. ***ICDAR2003*** **–** sadrži 509 scena s tekstom. Sve instance teksta pojavljuju se u horizontalnoj orijentaciji. Zatim se pojavljuje *ICDAR2011* kojem je broj slika smanjen na 484 kako bi se uklonilo dupliranje u prethodnoj verziji. Nakon toga, u skupu podataka *ICDAR2013* dodatno je smanjen broj slika iz prethodnih skupova na ukupno 462 slike. No, 2015. godine *ICDAR* pokreće novi izazov pod nazivom „*Incidental Scene Text*“, koji se još naziva i *ICDAR2015* i ima 1670 slika. Treba napomenuti, kako je ovaj skup podataka izazovniji od prijašnjih verzija jer uključuje tekst s proizvoljnom orijentacijom. Osim toga, *ICDAR2015*

prvi je poznati skup podataka koji koristi četverokut, tj. poligon kao zapis stvarnog graničnog okvira (engl. *Ground truth*) [2].

3. ***MLT*** – višejezični skup podataka koji je prikupljen za detekciju teksta, prepoznavanje i identifikaciju. Treba naglasiti, kako skupovi za testiranje i treniranje imaju 9000 slika svaki, 9 različitih jezika (Kineski, Japanski, Korejski, Engleski, Francuski, Arapski, Talijanski, Njemački i Indijski) sa 6 različitih pisama. Motivacija je za izradu ovog skupa podataka izazov zajednice za dizajniranjem detektora teksta koji je robustan na različita pisma koje postoje u današnjem svijetu [2].

4. ***COCO-Text*** – objavljen 2016. godine i sadrži 63 686 slika i 173 589 označenih područja teksta. Sastoji se, uglavnom, od horizontalnog i više-orijentiranog teksta te malog broja zakrivljenog teksta. Kao granične okvire, koristi pravokutnike poravnate po osima koji označavaju poziciju teksta na slici [2].

5. ***Total-Text*** - najpopularniji skup podataka koji sadrži zakrivljeni tekst. Navedeni skup podataka sadrži 1555 slika dobivenih iz stvarnog svijeta. Tekst je na slikama ručno označen trima različitim orijentacijama, uključujući horizontalnu, više-orijentacijsku i zakrivljenu orijentaciju. Međutim, navedeni skup podataka često nije dovoljan za kvalitetno treniranje mreža za detekciju zakrivljenog teksta. Ovaj skup podataka sadrži samo latinsko pismo, a anotacija je na razini riječi te prilikom anotacije postoji od 4 do 12 vrhova [2].

6. ***CUTE80*** – Risnumawan objavljuje prvi skup podataka koji sadrži zakrivljeni tekst, ali nažalost ima samo 80 slika i ograničene scenarije [2].

7. ***CTW1500*** – skup podataka koji je najsličniji *Total-Text* skupu. Oba su skupa slična u smislu razmjera, graničnih okvira u obliku poligona i s barem jednom instancom zakrivljenog teksta po slici. Onoliko koliko je bitno napomenuti sličnosti, bitno je navesti i postojeće razlike. *CTW1500* sadrži latinsko i kinesko pismo te anotacija je na razini linije. Osim toga, *CTW1500* je označen samo s prostornom lokacijom, dok se u *Total-Text* označavanje sastoji od prostornih lokacija, transkripcije, orijentacije i maske na razini elemenata slike (engl. *Pixel*) za detekciju teksta te prepoznavanja i segmentacije [2].

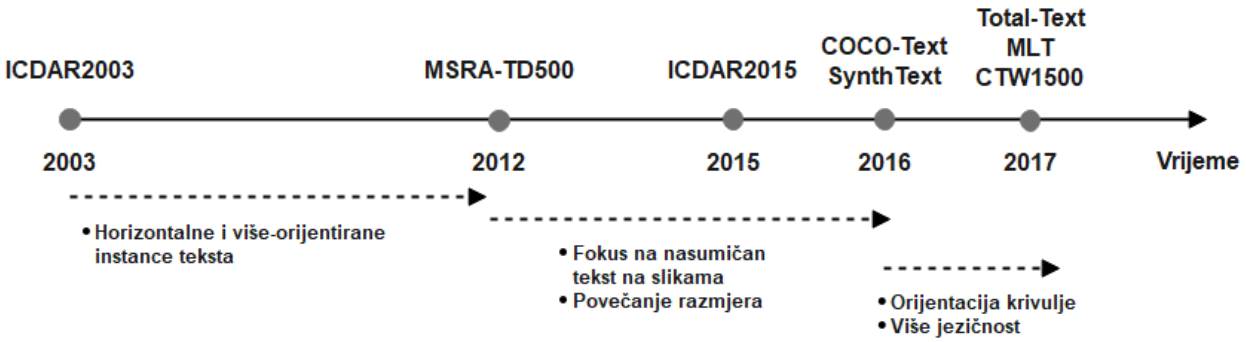
U tablici 2.1. sažeto su prikazane glavne karakteristike opisanih skupova podataka, kao što su broj slika u trening i testnom skupu, jezik, orijentacija i slično. Treba naglasiti, radi lakšeg razumijevanja, kako u stupcu zadatak slovo *D* označava detekciju, *S* označava segmentaciju, dok

R označava prepoznavanje. Nadalje, pod stupcem orijentacija, slovo *H* označava horizontalan tekst, *M* označava više-orijentiran, dok slovo *C* označava zakrivljeni tekst.

Tablica 2.1. Prikaz karakteristika skupova podataka za detekciju teksta [3].

Skup podataka	Broj slika (Trening/Test)	Broj instanci	Jezik	Zadatak	Orijentacija	Godina
SynthText	800000	800000	Engleski	D, R	H, M	2016.
ICDAR-2013	462 (229/233)	1943	Engleski	D, S, R	H	2013.
ICDAR-2015	1670 (1000/670)	11886	Engleski	D, R	H, M	2015.
MLT	18000 (9000/9000)	-	9 jezika	D, I	H, M	2017.
COCO-Text	63686 (43686/20000)	173589	Engleski	D, R	H, M	2016.
Total-Text	1555 (1255/300)	11459	Engleski	D, S, R	H, M, C	2017.
CUTE80	80	-	Engleski	D, S, R	H, M, C	2014.
CTW1500	1500 (1000/500)	10751	Engleski, kineski	D	H, M, C	2017.

Uz tablični prikaz, na slici 2.1. dan je i slikoviti prikaz nastajanja spomenutih skupova podataka kroz vrijeme.



Slika 2.1. Vremenska linija navedenih skupova podataka [3].

2.3. Pregled rješenja za detekciju teksta

Autori su iz literature [4] koristeći *CTW1500* skup podataka, formulirali jednostavni, ali efikasni detektor (*CTD*) za tekst u obliku poligona, koji može izravno detektirati zakrivljeni tekst. Za razliku od tradicionalnih metoda detekcije, *CTD* detektor odvaja grane za predviđanje pomaka širine i visine graničnog pravokutnika. Dalje je u radu spomenuto kako se mrežna arhitektura može neprimjetno integrirati korištenjem nove metode pod nazivom transverzalna i uzdužna pomična veza (engl. *Transverse and longitudinal offset connection* - TLOC) koja koristi povratnu neuronsku mrežu (engl. *Recurrent neural network*) za učenje inherentne vezu između točaka lociranja. Također, predstavljene su dvije postprocesorske (engl. *Post-processing*) metode za daljnje poboljšanje sposobnosti generalizacije *CTD* metode: *Non-polygon suppression (NPS)* i *Polygonal non-max suppression (PNMS)*. Arhitektura *CTD* detektora može se podijeliti u tri sekcije, a to su kostur, *RPN* i regresijski modul. Kostur usvaja popularne, unaprijed istrenirane modele iz *ImageNet-a* i koristi odgovarajući model za dobro podešavanje, poput *VGG16* i *ResNet*. U radu je kao kostur korišten *ResNet-50*. Treba istaknuti kako su upravo *RPN* i regresijski modul povezani pomoću kostura. Nadalje, kako bi se detektirao zakrivljeni tekst s poligon oznakom (labelom), potrebno je modificirati regresijski modul, tako što će se dodati lokacije vrhova krivulje (engl. *Curved locating points*). Kako bi se evaluirale performanse lokalizacije, korišten je *PASCAL VOC* protokol koji koristi 0.5 *IoU* prag kako bi odredio točne ili netočne pozitive. Ono po čemu se razlikuje dobiveni rezultat u radu jest to što je izračunat *IoU* između poligona, umjesto osi-poravnatih pravokutnika.

U radu je [2] dan prijedlog novog detektora teksta pod nazivom *Polygon-Faster-RCNN* koji je prerađena verzija *Faster-RCNN* detektora. Ovaj je detektor sposoban za detekciju teksta u svim orijentacijama jer koristi regresiju poligona umjesto parametara graničnih okvira. Također,

predstavljen je alat pod nazivom *Total-Text-Tool (T3)* koji značajno smanjuje potrebno vrijeme anotiranja, ali ipak ostvaruje kvalitetan stvarni granični okvir. Bitno je naglasiti, kako spomenuti alat smanjuje vrijeme anotiranja za 25 %, a podudara se oko 84 % s anotacijama koje su napravili ljudi. Instance teksta u radu anotirane su na razini riječi i koriste se poligon granični okviri. Izračun presjeka između područja predviđanja i područja stvarnog graničnog okvira, temeljna je jezgra *DetEval* i *PascalVOC* protokola. Originalni algoritam za izračun presjeka područja, u oba spomenuta protokola, dizajniran je za presjek između pravokutnika. Treba napomenuti, kako je u radu taj modul zamijenjen s algoritmom koji je sposoban izračunati presjek područja između poligona. Također, u radu su oba navedena protokola pisana u *MATLAB* i *Python* programskim jezicima. Autori su, također, dali prijedlog nove metode za kodiranje, pod nazivom *Text Line EncodingMethod,* koja pretvara bilo koji stvarni granični okvir regije teksta u parametar linije teksta, gdje parametar predstavlja ravnu crtu u smjeru zakrivljenosti teksta. Proces kodiranja razdvaja šest vrhova graničnog okvira u tri suprotna para. Bitno je napomenuti, kako je metoda kodiranja fleksibilna za sve postojeće granične okvire teksta. Svi su modeli prošli kroz jednak proces učenja, od čega su prvo inicijalizirani s predtreniranim težinama *ImageNet-a*. Raspored treninga započinje sa 100 000 iteracija na *SynthText-u*, zatim slijedi sljedećih 100 000 iteracija na podacima iz stvarnog svijeta iz *COCO-Text-a*. Na kraju se dobro podešavaju (engl. *Fine-tuning*) s ciljanim trening setom za drugih 50 000 iteracija.

U radu [5] umjesto detekcije zakrivljenog teksta kao poligonske regresije ili segmentacije, problem tretiraju kao proces širenja regija. Predstavljen je prijedlog pod nazivom *Conditional Spaital Expansion (CSE)* mehanizma. Ono što treba naglasiti jest da je *CSE* fleksibilna metoda od kako se proizvoljna unutarnja točka, tj. *seed* može specificirati na bilo kojoj lokaciji unutar ciljane regije teksta. *Seeding* proces podrazumijeva izbor lokacije unutar objekta iz koje se ekspanzijom izdvaja odgovarajuća regija objekta. Proširenje se provodi selektivnim spajanjem susjednih pod-regija kako bi se formirala regija ciljanog objekta. Pod-regije su apstrahirane kao značajne točke ili čvorovi koji su uzorkovani iz ulazne slike s diskretnim lokacijama. Također, organizirane su kao mreža i lokalno im je dodijeljen ekspandirajući indikator koji predstavlja smjer spajanja susjednih čvorova. Kako bi se odredile *seed* lokacije te ravnomjerno uzorkovale značajke, korištena je bilinearna interpolacija iz regije označene graničnim okvirom. Primijenjena je *cross-entropy-loss* metoda za svaki čvor kako bi se optimizirao *CSE* model. Arhitektura sustava sastoji se od kostura mreže, *Faster-RCNN* i *CSE* modula. Kostur se koristi za kodiranje slike u prostorne značajke, a sastoji se od *ResNet-34 i Feature Pyramid Network*-a (*FPN*). Također, usvojena je *Faster-RCNN* mreža s *ResNet-34* za inicijalizaciju *seed* lokacija i odgovarajućih mreža u svim

eksperimentima. Prilikom optimizacije korišten je *Adamov* optimizator za treniranje mreže. Skup podataka za početno treniranje sastoji se od 10 000 slika iz cijelog skupa *ICDAR-17 MLT* i skupova za treniranje *MSRA-TD500, Total-Text* i *CTW1500*. Nakon toga, odrađena je evaluacija na *Total-Text* i *CTW1500* skupovima podataka.

3. IZRADA DETEKTORA TEKSTA NA TEMELJU UMJETNO GENERIRANOG ZAKRIVLJENOG TEKSTA

Za izradu određenog detektora teksta, potrebno je imati označen skup podataka. Pri tome, podaci trebaju biti precizno označeni tako da svaka instanca teksta ima granični okvir u obliku poligona kojim je označena pozicija teksta na slici te pripadajuću masku koja označava poziciju teksta. U radu je za generiranje skupa podataka korištena postojeća skripta za generiranje slika [6] koja je preuređena kako bi umjesto pravokutnih graničnih okvira, generirala granične okvire u obliku poligona oko instance teksta. Na taj je način moguće generirati zakrivljeni tekst i ispitati koliko prisutnost takvih instanci u podatkovnom skupu za učenje utječe na konačnu efikasnost izgrađenog detektora teksta.

U potpoglavlju 3.1. opisan je detaljan proces podešavanja i instalacije potrebnih biblioteka i alata za pokretanje skripte za generiranje teksta. U potpoglavlju 3.3. opisan je proces izrade vlastitog skupa podataka sa zakrivljenim tekstom, dok su u potpoglavlju 3.5. opširno opisani procesi treniranja i testiranja odabrane neuronske mreže za detekciju teksta na slikama.

3.1. Podešavanje i instalacija potrebnih alata i biblioteka

Vlastito rješenje za generiranje slika sa zakrivljenim tekstom izrađeno je na *Linux* operacijskom sustavu *Ubuntu 18.04* u programskom jeziku *Python* uz brojne alate i biblioteke poput *Numpy*, *OpenCV*, *Matplotlib*, *Pip*, *h5py* i dr.

Pip je alat koji se koristi za instaliranje i upravljanje softverskim paketima u *Python-u* [7]. Instalacija se alata na *Linux* operacijskom sustavu pokreće u terminalu sljedećom naredbom:

sudo apt install python3-pip

OpenCV (engl. *Open Source Computer Vision*) je biblioteka otvorenog koda koja sadrži funkcije i više od 2500 optimiziranih algoritama za strojno učenje i računalni vid. Spomenuti se algoritmi koriste za detekciju i prepoznavanje lica, identifikaciju objekata, praćenje pokretnih objekata, stvaranje 3D modela objekata i sl. Također, podržane su implementacije za *Python, C++, Java* i *MATLAB* programske jezike te za *Linux, Windows, MacOS i Android* operacijske sustave. Treba napomenuti da se *OpenCV* uglavnom koristi za primjene u stvarnom vremenu [8]. Instalacija biblioteke na *Linux* operacijskom sustavu pokreće se u terminalu sljedećom naredbom:

sudo pip install opencv-python

NumPy je osnovna biblioteka za znanstveno računanje u *Python* programskom jeziku. Biblioteka pruža višedimenzionalne nizove, razne izvedene objekte i razne operacije na nizovima, uključujući matematičke, logičke operacije, manipulacije objektima, osnove linearne algebre, osnovne statističke operacije i slično [9]. Instalacija se biblioteke na *Linux* operacijskom sustavu pokreće u terminalu sljedećom naredbom:

sudo python -m pip install --user numpy

Matplotlib je biblioteka za kreiranje statičnih, animacijskih i interaktivnih vizualizacija u *Python* programskom jeziku uz njegovo numeričko matematičko proširenje *NumPy*. Također, biblioteka pruža objektno orijentirane API-je za iscrtavanje, koristeći alate opće namjene [10]. Pokreće se u terminalu sljedećom naredbom:

sudo python -m pip install -U matplotlib

H5py paket je *Python* sučelje (engl. *Interface*) za *HDF5* binarni format podataka. Paket omogućuje spremanje velikog broja numeričkih podataka i manipuliranje tim podacima pomoću *NumPy*. Osim toga, može se iterirati preko skupa podataka u datoteci i provjeravati atribute, podijeliti višeterabajtne skupove podataka koji su pohranjeni na disku i slično. Također, stvorene su datoteke u binarnom formatu i može ih se razmjenjivati među drugim programima, poput, *IDL* i *MATLAB-a* [11]. Instalacija biblioteke na *Linux* operacijskom sustavu pokreće se u terminalu sljedećom naredbom:

sudo apt-get install libhdf5-dev

Kako bi se pokrenula *SynthText* skripta za generiranje slika [6] potrebno je instalirati *Visual Studio Code* editor i dodati proširenje za programski jezik *Python*. U radu je korištena 3.6.9. verzija *Python* programskog jezika. Uz to, potrebno je instalirati sljedeće pakete:

- *Cached-property* = = 1.5.2
- *Cycler* = = 0.11.0
- *H5py* = = 3.1.0
- *Kiwisolver* = = 1.3.1
- *Matplotlib* = = 3.3.4
- *Numpy* = = 1.19.5
- *Opencv-python* = = 3.3.0.10

- *Pillow* = = 8.4.0
- *Pygame* = = 1.9.6
- *Pyparsing* = = 3.0.6
- *Python-dateutil* = = 2.8.2
- *Scipy* = = 1.5.4
- *Six* = = 1.16.0
- *Wget* = = 3.2

3.2. Opis korištenog skupa podataka za generiranje zakrivljenog teksta

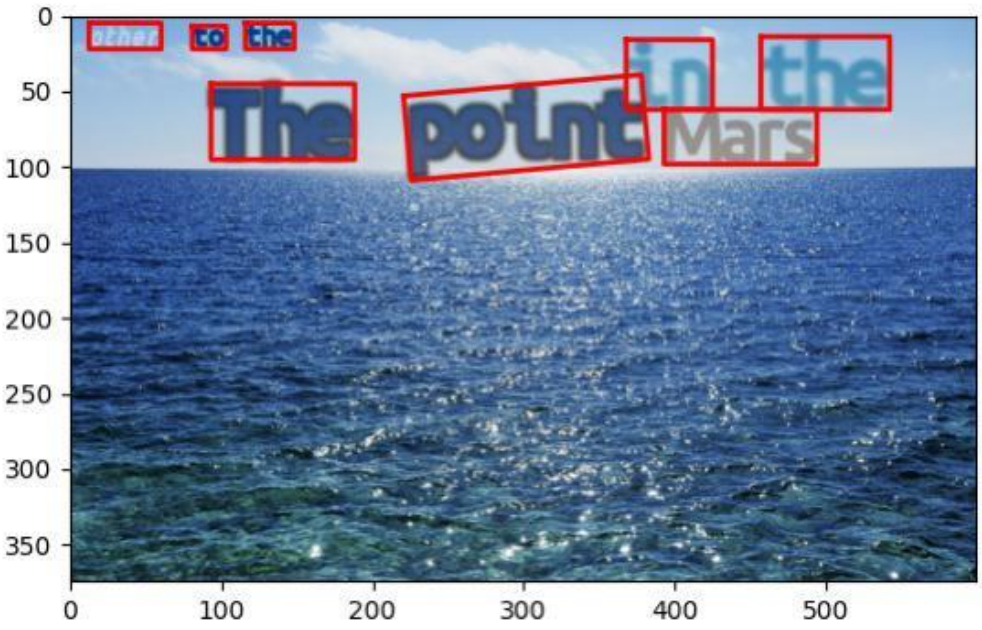
U radu je korišten skup podataka pod nazivom *SynthText in the Wild Dataset*. Korišteni skup podataka dolazi s 11 040 originalnih *RGB* slika bez teksta, dok su autori [6] ukupno generirali 858 750 slika s tekstom pomoću skripte *gen.py*. Prilikom preuzimanja skupa podataka, segmentacijske slike i slike s informacijom o dubini bile su odvojene u dvije *h5* datoteke te je bilo potrebno napraviti još jednu *h5* datoteku s originalnim *RGB* slikama i spojiti ju s preostalim dvjema u novu *h5* datoteku. Prije toga, potrebno je instalirati *HDFView* za učitavanje slika. Tada je napravljena konačna *h5* datoteka u kojoj se nalaze informacija o dubini za svaku sliku, segmentacijske slike i originalne *RGB* slike. Treba napomenuti, kako je od ukupno 11 040 slika, upotrijebljeno 8010 slika jer jedino je za njih dostupna informacija o dubini. Odlučeno je kako će se svaka slika generirati 10 puta, gdje niti jedna neće imati identičan tekst.

Nakon toga, potrebno je pokrenuti skriptu *gen.py* koja iz konačne *h5* datoteke učitava slike te na njih dodaje tekst na pogodna mjesta, sprema u novu *SynthText.h5* datoteku i prikazuje sliku (slika 3.1.) naredbom:

python3 gen.py --viz

Treba istaknuti, ako je opcija *--viz* uključena, tada će generirani izlaz biti vizualiziran. Kako bi se izostavila vizualizacija, potrebno je izostaviti opciju *--viz*. No ako se žele vizualizirati rezultati koji su spremljeni u *SynthText.h5*, kao što su koordinate teksta, broj instanci teksta na slici i slično, tada se pokreće naredba:

python3 visualize_results.py



Slika 3.1. Prikaz generirane slike s oznakama teksta na razini pravokutnog graničnog okvira.

3.3. Izrada vlastitog skupa podataka sa zakrivljenim tekstom

3.3.1. Opis zakrivljenog teksta

Curved text detector - CTD [4] prvi je moderni detektor teksta koji je dizajniran s obzirom na detekciju zakrivljenog teksta. Motivacija za prikupljanje *Total-Text* skupa podataka postigla se zbog nedostatka zakrivljenog teksta u postojećim skupovima podataka. Smatra se, kako je nedostatak javno dostupnog skupa podataka sa zakrivljenim tekstom razlog zašto je došlo do ideje za njegovim nastajanjem. Geometrijski gledano, ravna linija nema varijaciju kuta duž linije i može se opisati kao linearna funkcija:

$$y = mx + c \tag{3-1}$$

Zakrivljena linija nije ravna linija, nema ograničenja na varijaciju kuta duž linije. Gledajući iz perspektive samog teksta, uočava se da je tekst horizontalne orijentacije skup znakova koji mogu biti povezani ravnom linijom, odnosno poravnati su s donjom zamišljenom linijom u većini slučajeva. Više-orijentacijski tekst, također, može biti povezan ravnom linijom, dok znakovi u zakrivljenom tekstu nemaju kutni pomak, već se smatra da odgovaraju polinomnoj liniji na razini teksta [2].

Trenutni skupovi podataka imaju malo zakrivljenog teksta i to nije dovoljno kako bi se tekst označio s poligonima ili graničnim pravokutnicima. Koristeći granične okvire u obliku poligona, postoje tri značajne prednosti [4]:

- Izbjegavanje nepotrebnog preklapanja. S obzirom na to da se tekst može pojaviti u mnogim oblicima, tradicionalna metoda lokalizacije pomoću četiriju točaka neće biti toliko precizna, odnosno, dolazi do pretjeranog preklapanja graničnih okvira. Kako je prikazano na slici 3.2.(a), pravokutni granični okvir ne može izbjeći velik broj suvišnih preklapanja, dok zakrivljeni granični okvir može.
- Manje pozadinskog šuma. Kao što je vidljivo na slici 3.2.(b), ako se tekst pojavi u zakrivljenoj formi, tada se može dogoditi da pozadinski šum izaziva smetnje za pravokutni granični okvir.
- Izbjegavanje više redaka teksta. U zadnje vrijeme, popularne metode za prepoznavanje zahtijevaju jedan red teksta u svakom graničnom okviru. U nekim slučajevima kao na slici 3.2.(c), svi pravokutni granični okviri ne mogu izbjeći sadržanost teksta u više redaka, dok zakrivljeni granični okvir može riješiti taj problem.

a) b) c)

Slika 3.2. Usporedba pravokutnih i poligon graničnih okvira za lokalizaciju teksta [4] a) Prikaz izbjegavanja nepotrebnog preklapanja, b) Prikaz smanjenja pozadinskog šuma, c) Prikaz naslaganog teksta.

3.3.2. Generiranje slika sa zakrivljenim tekstom

Za generiranje zakrivljenog teksta napravljena je funkcija *sin_function()* koja pravi zakrivljeni tekst na osnovu sinusne funkcije pomoću *math.sin()* funkcije (slika 3.3.). *Math* predstavlja matematički modul koji dozvoljava izvršavanje matematičkih operacija na brojevima. Funkcija *math.sin()* vraća vrijednost sinusa ulaznog parametra, u ovom slučaju, funkcija vraća sinus svakog slova u riječi.

```
def sin_function(self,x):
        amp = 6
        freq = 2*math.pi*x
        return amp*math.sin(freq*x)+amp
```

Slika 3.3. Implementacija sinusne funkcije *sin_function()*

Nakon definiranja funkcije *sin_function()*, napravljena je *for* petlja koja prolazi slovo po slovo u riječi te na svakom slovu izvršava se *sin_function()* funkcija, kako je prikazano na slici 3.4.

```
wl = len(word_text)
mid_idx = wl//2
curve = []
for i in range(wl):
    curve.append(self.sin_function(i-mid_idx))
```

Slika 3.4. Prikaz primjene funkcije *sin_function()* za generiranje zakrivljenog teksta.

Također, može se postaviti uvjet ako je duljina slova u riječi veća od nekog broja, da se tek onda tekst iskrivljuje. Kako je vidljivo na slici 3.5. postavljeni su parametri teksta koji kontroliraju generiranje zakrivljenog teksta. Parametar *min_nchar* označava koliki je minimalan broj slova u riječi po pojedinoj slici, *min_font_h* označava minimalnu visinu fonta slova, a *max_font_h* označava maksimalnu visinu slova.

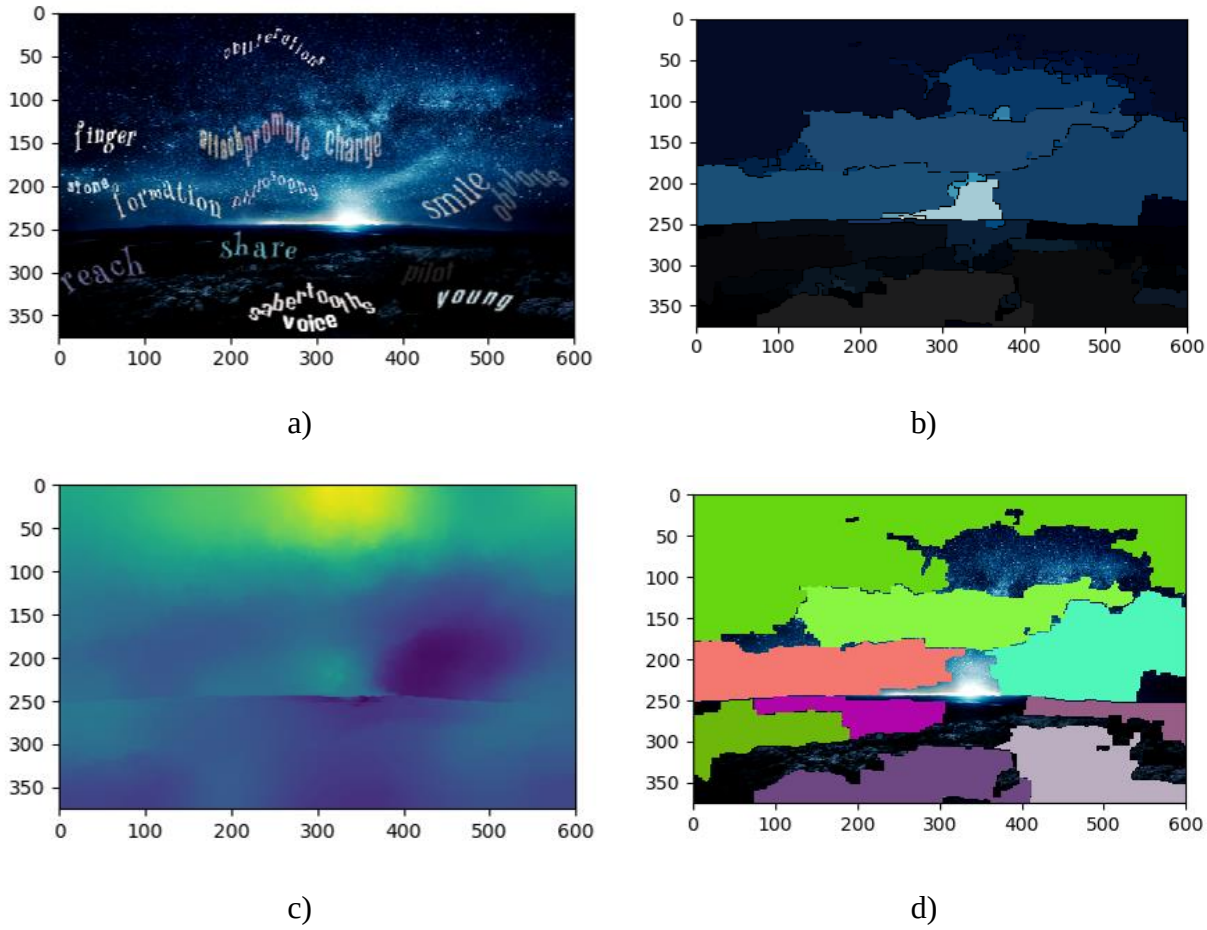
```
self.min_nchar = 2
self.min_font_h = 15
self.max_font_h = 25
```

Slika 3.5. Parametri koji kontroliraju generiranje zakrivljenog teksta.

Prema postavljenim parametrima generiran je skup podataka koja sadrži 62 442 slika s generiranim tekstom. Osim toga, korištena je baza od 4000 engleskih riječi te su se koristila tri različita fonta, a to su:

- *henny-penny.regular.ttf*
- *modern-antiqua.book.ttf*
- *denk-one.regular.ttf*

Nadalje, skripta generira tekst na nasumične lokacije te je broj riječi na slici, također, nasumičan. Na slici 3.6.(a) može se vidjeti prikaz jedne slike iz skupa podataka s generiranim zakrivljenim tekstom koji se dodaje na pogodne lokacije vidljive na slici 3.6.(d) s obzirom na segmentacijsku sliku (slika 3.6.(b)) i informacija o dubini (slika 3.6.(c)).

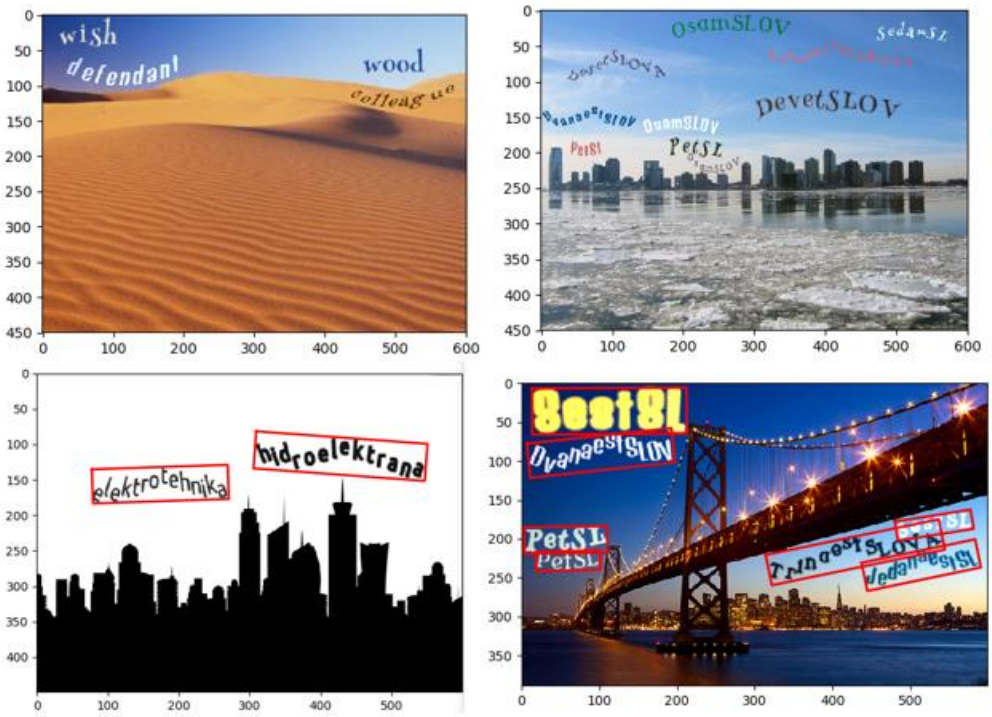


a) b)

c) d)

Slika 3.6. a) Prikaz konačnog generiranog teksta na slici, b) Prikaz segmentacijske slike, c) Prikaz slike s informacijom o dubini, d) Prikaz pogodnih lokacija za generiranje teksta.

Na slici 3.7. prikazano je nekoliko primjera generiranih slika sa zakrivljenim tekstom tijekom modifikacije parametara teksta.



Slika 3.7. Prikaz generiranog zakrivljenog teksta na pojedinim slikama iz skupa podataka.

3.3.3. Pohranjivanje oznaka teksta prema *ICDAR2019* zapisu

Za označavanje zakrivljenog teksta na slikama korišten je zapis stvarnog graničnog okvira (engl. *Ground truth*) prema *ICDAR2019 Robust Reading Challenge on Arbitrary-Shaped Text* zapisu pod *Task 1 and 3* [12].

S obzirom na to da je skripta *visualize_results.py* izvorno korištena za vizualizaciju rezultata i ispis koordinata teksta pravokutnog graničnog okvira, kako se vidi na slici 3.8., ona je dodatno modificirana tako da ispisuje koordinate poligon graničnog okvira oko riječi. Primjer izvornog ispisa koordinata za jednu riječ je:

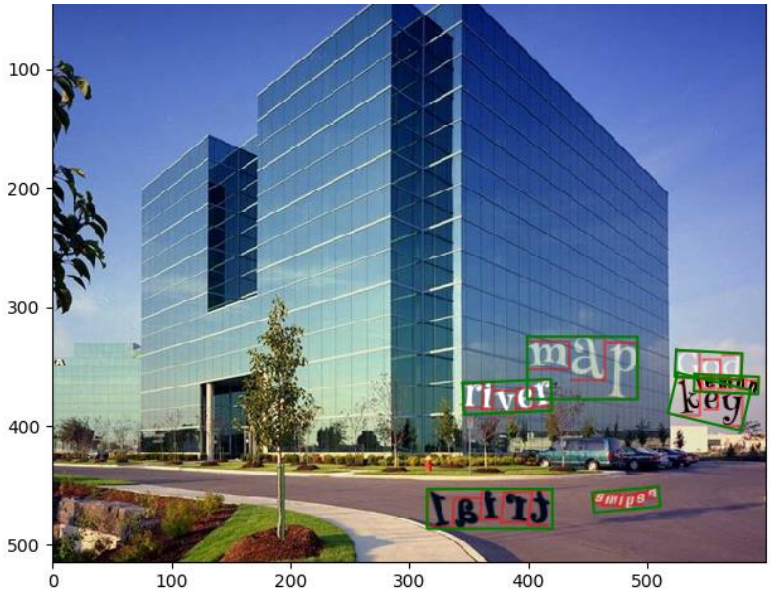
[[590.36923 453.74265 455.30087 510.92746 509.36923]

[451.0535 456.2211 472.99442 467.8268 451.0535]]

gdje prva zagrada označava *x* koordinate graničnog pravokutnog okvira, dok druga zagrada označava *y* koordinate graničnog pravokutnog okvira na razini riječi.

Također, zelenom je bojom označen granični okvir na razini riječi, a crvenom bojom prikazan je granični okvir na razini slova kako je vidljivo na slici 3.8. Oznake u *ICDAR2019* zapisu potrebno

je zapisati u *JSON* datoteku. *JSON* predstavlja format datoteke za razmjenu podataka koji koristi ljudski čitljiv tekst.



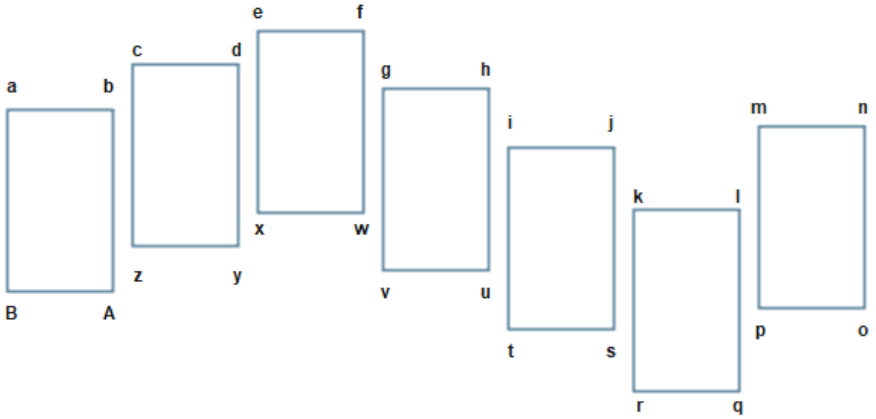
Slika 3.8. Prikaz koordinata pravokutnog graničnog okvira na razini riječi.

Ono što je potrebno napraviti jest *JSON* datoteku, koja će sadržavati sve informacije o tekstu na slikama u skupu podataka kako bi se pohranio stvarni granični okvir. U radu su slike imenovane na sljedeći način:

ime_[IME_SLIKE].png_N

gdje se *ime_[IME_SLIKE]* odnosi na naziv slike u skupu podataka, dok se *N* odnosi na trenutni broj generiranja slike, tako npr. *ime_city.png_6* predstavlja sliku *ime_city.png* koja je generirana šesti put. U *JSON* datoteci, svaki naziv slike poput, *ime_[IME_SLIKE].png_N* odgovara listi koja predstavlja određenu instancu teksta na slici i sadrži informacije transkripcije, jezika, zastavice težine i graničnog okvira.

Na slici 3.9. može se vidjeti prikaz kako su označeni granični okviri pojedinih slova po instanci teksta, počevši od slova „a“ u smjeru kazaljke na satu, gdje zadnja točka završava ponovno u slovu „a“. Vrhovi su simbolički označeni slovima engleske abecede, kako bi se vidio redoslijed označavanja slova.



Slika 3.9. Prikaz označavanja graničnog okvira teksta u *JSON* datoteci.

Kod za generiranje *JSON* datoteke sa zakrivljenim tekstom nalazi se u prilogu P.3.1. dok se kod za generiranje *JSON* datoteke s ravnim tekstom nalazi u prilogu P.3.2. Bitno je napomenuti, kako je koordinate poligon graničnog okvira potrebno spremati kao cjelobrojne brojeve u *JSON* datoteku. Na slici 3.10. može se vidjeti puni prikaz *JSON* datoteke nakon njenog generiranja.


```
{
    "info": {
        "description": "Curved_SynthText dataset",
        "version": "1.0",
        "date": "12\/5\/2022"
    },
    "license": [
        {}
    ],
    "annotations": {
        "ime_ant+hill_1.jpg_0": [
            {
                "transcription": "medicine",
                "language": "Latin",
                "illegibility": false,
                "points": [[x1,y1], [x2,y2], ... , [xn,yn]]
            }
        ],
        "ime_camping_132.jpg_9": [
            {
                "transcription": "university",
                "language": "Latin",
                "illegibility": false,
                "points": [[x1,y1], [x2,y2], ... , [xn,yn]]
            }
        ],
        ...
    }
}
```


Slika 3.10. Prikaz generirane *JSON* datoteke.

Na slici 3.10. $x_1, y_1, x_2, y_2, \dots, x_n, y_n$ predstavljaju koordinate poligon graničnog okvira određene instance teksta na slici, tako da $[x_1, y_1]$ predstavljaju x i y koordinate točke „a“ koja je prikazana

na slici 3.8. i tako dalje do kraja sve dok se ne dođe ponovno u točku „a“. Transkripcija označava instance teksta koje se nalazi na pripadajućoj slici, a jezik označava jezik odgovarajuće instance teksta na slici, dok nečitkost (engl. *Illegibility*) označava “*Do Not Care*“ regije teksta kada je postavljeno na istinu (engl. *True*) što ne utječe na rezultate. Također, postoji i nekoliko listi, a to su:

- Lista *info* - sadrži osnovne informacije o skupu podataka.
- Lista *license* – upotrebljava se ako se koriste podaci čija su prava ograničena.
- Lista *annotations* – sadrži sve potrebne informacije o svakom tekstu koji se generira i dodaje na sliku. Lista se sastoji od imena slika gdje je svako ime slike nova lista koja sadržava objekte unutar sebe.

Konačan prikaz oznaka graničnih okvira u obliku poligona prikazan je na slici 3.11.(a).

a) b)

Slika 3.11. a) Vizualni prikaz oznaka slika kada se koristi granični okvir u obliku poligona, b) Vizualni prikaz oznaka slika kada se koristi granični okvir u obliku pravokutnika.

Nakon izrade vlastitog skupa podataka, potrebno je generirati segmentacijske maske (engl. *Segmentation mask*) koje predstavljaju oznaku graničnog okvira koji se odnosi na poziciju teksta na slici. Svaka maska predstavlja jednu instancu teksta, a maske se međusobno razlikuju drugim bojama. Korišten je *RGB* zapis boja gdje je odabrano 30 različitih boja što prikazuje slika 3.12.

```
colors = [(255, 0, 0),(235, 238, 0),(11, 238, 0),(116, 13, 250),(234, 77, 8),(6,
41, 2),(13, 103, 95), (219, 6, 72), (223, 104, 66),
(211, 106, 95),(211, 106, 45),(188, 243, 245),(73, 120, 168),(149, 120, 168),(2,
134, 168),(2, 0, 42), (51, 0, 2), (104, 102, 14),
(104, 102, 102),(252, 27, 252), (252, 233, 253),(0, 0, 255), (0, 255, 0), (28, 7,
13),(63, 61, 61), (2, 251, 114), (2, 93, 253),
(99, 234, 1), (52, 70, 1), (52, 70, 101)]
```

Slika 3.12. Prikaz definiranih boja za izradu maski.

Nakon toga maske su generirane tako da se koordinate poligon graničnog okvira oko pojedinog teksta učitavaju iz *JSON* datoteke, zatim se te koordinate koriste za generiranje maske pojedine instance teksta. Maske su prvo generirane u boji, a zatim su prebačene u sliku u sivim tonovima. Postupak generiranja maski prikazan je na slici 3.13.

```
root = '/home/Train/Curved_SynthText/'
img_path = os.path.join(root, "Curved_SynthText_Dataset/")
counter = 0

for img in glob.glob(img_path + "*.*"):
    images = os.path.basename(img)

impath = os.path.join(root, "Curved_SynthText_Dataset/", images)
counter += 1
print(counter, images)

img = Image.open(impath).convert("RGB")
width, height = img.size

f = open('CurvedSynth.json')
data = json.load(f)
boxes = []
mask = []
for i in data['annotations']:
    if i == images:
        for j in data['annotations'][i]:
            mask.append(j['points'])

txtbbFinal = []
img2 = Image.new("RGB", (width, height), (0, 0, 0))
count=0
for txtbb in mask:
    xmax, ymax = np.amax(txtbb, axis=0)
    xmin, ymin = np.amin(txtbb, axis=0)
    boxes.append([xmin, ymin, xmax, ymax])
    bb_coordinates = [tuple(x) for x in txtbb]
    draw = ImageDraw.Draw(img2)
    draw.polygon(bb_coordinates, fill=colors[count])
    count+=1
maskimg = img2.convert('L')
maskimg.save("/home/Train/Curved_SynthText/Masks/" + images, "PNG")
```

Slika 3.13. Prikaz koda koji generira maske teksta na temelju danih koordinata poligon graničnih okvira.

Na slici 3.14. može se vidjeti prikaz generiranih maski za pripadajući zakrivljeni tekst.

a) b)

Slika 3.14. a) Prikaz originalne slike sa zakrivljenim tekstom, b) Prikaz pripadajuće maske.

Također, na isti način generirane su maske i za ravan tekst (slika 3.15.) no razlikuju se po tome što su koordinate učitane iz različite *JSON* datoteke te su korištene slike s generiranim ravnim tekstom.

a) b)

Slika 3.15. a) Prikaz originalne slike s ravnim tekstom, b) Prikaz pripadajuće maske.

3.3.4. Podjela izgrađenih skupova podataka

Pomoću modificirane skripte spomenute u potpoglavlju 3.2. izgrađeni su vlastiti skupovi podataka s generiranim ravnim i zakrivljenim tekstom. Točnije, generirano je 31 496 slika s ravnim tekstom, gdje je tekst označen pravokutnim graničnim okvirima. Također, generirano je 62 442

slika sa zakrivljenim tekstom, gdje je tekst označen graničnim okvirima u obliku poligona. U vlastito izgrađenim skupovima podataka nazvanih *ZakrivljeniSkup10k*, *ZakrivljeniSkup5k*, *RavniSkup10k* i *RavniSkup5k*, slike su podijeljene na trening skup, validacijski skup i testni skup, kako je prikazano u tablici 3.1.

Tablica 3.1. Prikaz podjele u vlastito izgrađenim skupovima podataka.

		Broj slika
ZakrivljeniSkup10k	Trening skup podataka	10 000
	Validacijski skup podataka	2000
	Testni skup podataka	1000
RavniSkup10k	Trening skup podataka	10 000
	Validacijski skup podataka	2000
	Testni skup podataka	1000
ZakrivljeniSkup5k	Trening skup podataka	5000
	Validacijski skup podataka	2000
	Testni skup podataka	1000
RavniSkup5k	Trening skup podataka	5000
	Validacijski skup podataka	2000
	Testni skup podataka	1000

3.4. *Mask R-CNN*

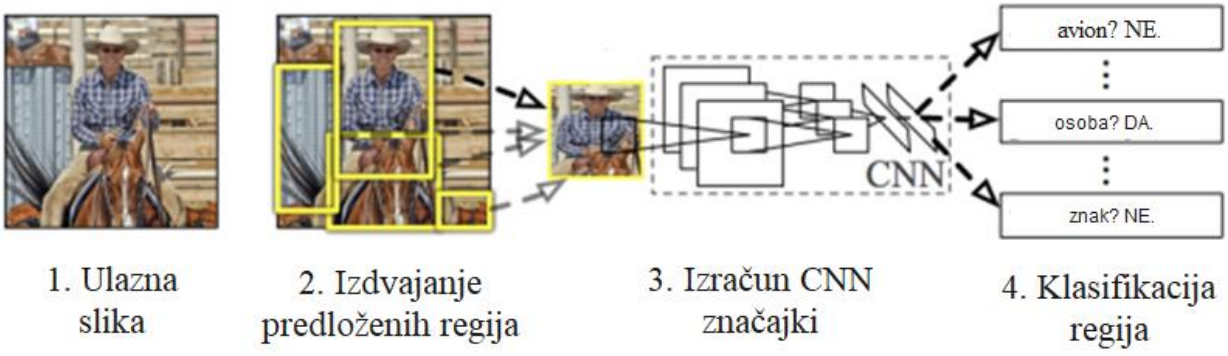
Konvolucijske neuronske mreže (engl. *Convolutional Neural Network* - CNN) su tip neuronske mreže koje se koriste u klasifikaciji slika i detekciji objekata jer efikasno izdvajaju značajke iz

slike. Također, ovaj tip mreže je osnova i za mnoge druge hibridne mreže. Ono što treba naglasiti jest to da se *CNN* arhitektura sastoji od tri glavna dijela:

1. Konvolucijski sloj – sloj koji pomaže sažeti ulaznu sliku kao mapu značajki primjenom filtara.
2. Sloj sažimanja (engl. *Pooling layer*) – sloj koji smanjuje rezoluciju (engl. *Donwsampling*) svake pojedine aktivacijske mape.
3. Potpuno povezani sloj – povezuje svaki neuron jednog sloja sa svakim neuronom prethodnog sloja.

Kombinacija ovih slojeva omogućava dizajniranje neuronske mreže koja je nakon procesa treniranja u mogućnosti detektirati objekte od interesa u slici [13].

Region-Based Convolutional Neural Network (RCNN) predstavlja tip modela strojnog učenja koji se koristi za detekciju objekata. Ovaj pristup koristi granične okvire oko objekata koje tada evaluira konvolucijska mreža na svim regijama od interesa (engl. *Region of interest* - ROI) kako bi klasificirala više regija u ispravne klase [13]. Problem *R-CNN* mreže je veliko vrijeme potrebno za učenje modela i nemogućnost implementacije mreže u stvarnom vremenu. Također, kod *R-CNN* mreže se selektira određen broj regija i onda se koristeći odabrane algoritme rekurzivno pozivaju slične regije na slici te se potom spajaju u veće regije koje se na kraju klasificiraju [14]. Arhitektura *R-CNN* mreže prikazana je na slici 3.16.

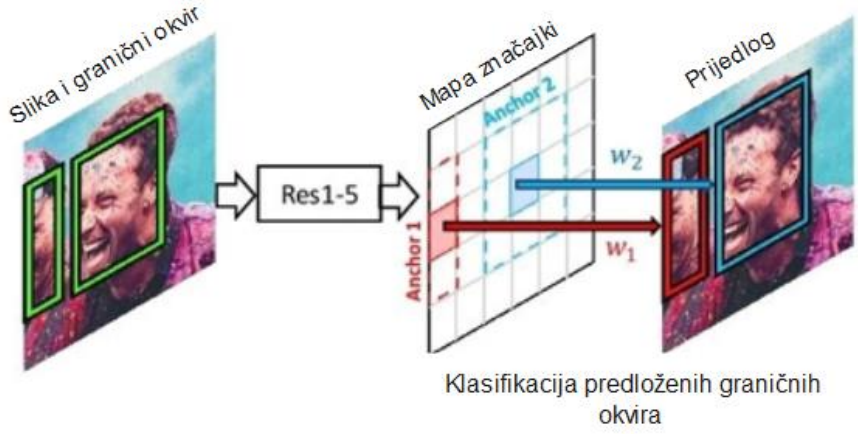


Slika 3.16. Arhitektura *R-CNN* mreže [13].

Fast R-CNN rješava neke od problema *R-CNN* mreže. U procesu izrade mape značajki više nije potrebno svaki put izdvojiti veliki broj regija, već se iz ulazne slike prvo stvara konvolucijska mapa značajki na kojoj se onda izdvajaju regije, što minimizira vrijeme učenja i testiranja. Kako je selekcija regija dugotrajan proces i proces obrade u stvarnom vremenu i dalje velik, razvijen je novi tip *CNN* mreže pod nazivom *Faster R-CNN,* gdje se koristi posebna neuronska mreža (engl.

Region Proposal Network - RPN) samo za učenje selektiranja regija na slici [14]. *Faster R-CNN* sastoji se od dviju faza koje predstavljaju dva izlaza za svakog kandidata objekta [13], a to su:

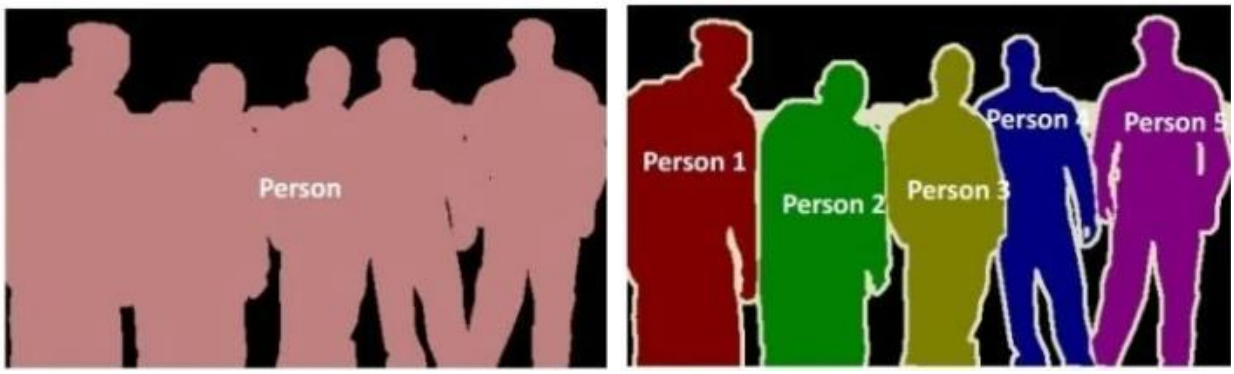
1. *Region Proposal Network* – neuronska mreža koja predlaže više objekata koji su dostupni unutar određene slike što je vidljivo na slici 3.17.
2. *Fast R-CNN* – izdvaja značajke koristeći *RoIPool* iz svakog graničnog okvira kandidata te zatim izvodi klasifikaciju i regresiju graničnih okvira



Slika 3.17. Koncept *Region proposal* mreže [13].

Mask R-CNN predstavlja osnovni okvir (engl. *Framework*) za segmentaciju instanci objekata i produžena je verzija *Faster R-CNN* mreže, tako što je paralelno dodana grana (*eng. Branch*) za predikciju maske objekata s postojećom granom za prepoznavanje graničnih okvira (engl. *Bounding box*). Bitno je spomenuti kako *Faster R-CNN* nije dizajnirana za element slike po element slike (engl. *Pixel-to-pixel*) poravnanje između ulaza i izlaza mreže, a to je ključni element *Mask R-CNN* mreže. *Mask R-CNN* pokraj dvaju izlaza *Faster R-CNN* mreže, dodaje treću granu, koja kao izlaz daje masku za svako područje od interesa (engl. *Region of interest* - *ROI*). Dva glavna tipa segmentacije slika su:

- Semantička segmentacija
- Segmentacija instanci

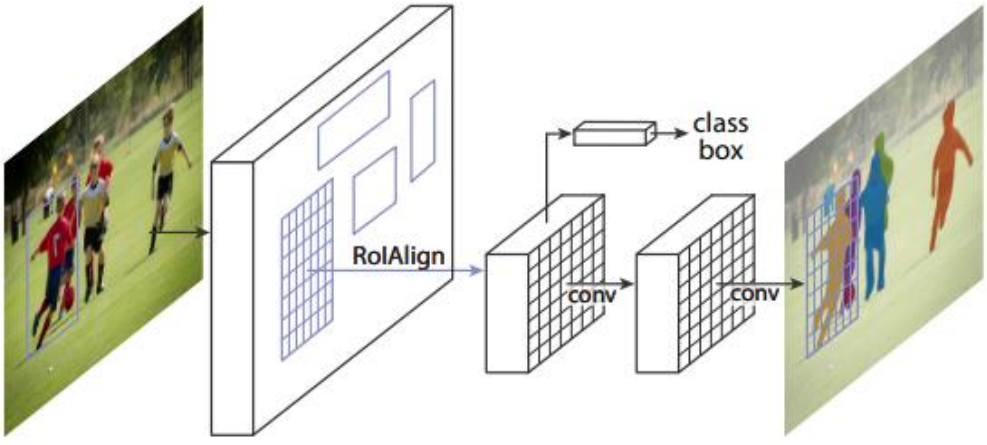


a) b)

Slika 3.18. a) Prikaz semantičke segmentacije, b) Prikaz segmentacije instanci [13].

Semantička segmentacija klasificira svaki element slike u fiksni set kategorija bez razlikovanja instanci objekata, tj. bavi se identifikacijom/klasifikacijom sličnih objekata u jednu klasu s razine elementa slike. Kako je prikazano na slici 3.18.(a) svi objekti su klasificirani kao jedna cjelina, klasa osoba (engl. *Person*) [13].

Segmentacija instanci vrlo je izazovna jer zahtijeva ispravnu detekciju svih objekata na slici te preciznu segmentaciju svih instanci. Kombinira elemente detekcije objekata iz standardnih zadataka računalnog vida, gdje je cilj klasificirati pojedini objekt i lokalizirati svaki objekt koristeći granične okvire i semantičku segmentaciju, tako da se svaki element slike klasificira u fiksni set kategorija. Kako se dodani izlaz maske razlikuje od klase i izlaznog okvira, tako se zahtijeva izdvajanje mnogo boljeg prostornog rasporeda objekata [1]. Kako je prikazano na slici 3.18.(b) svaki objekt je „osoba“, ali proces segmentacije izdvaja svaku instancu osobe kao jednu cjelinu. Također, na slici 3.19. dan je konačni prikaz *Mask R-CNN* mreže.



Slika 3.19. *Mask R-CNN* za segmentaciju instanci [1].

3.5. Upute za pokretanje treniranja i testiranja *Mask R-CNN* mreže

Za programsko okruženje korištena je *Anaconda* koja predstavlja distribucijsku platformu otvorenog koda za upravljanje paketa i njihovu implementaciju. Nakon instalacije kreirano je novo okruženje naredbom u terminalu:

conda create --name Model1

gdje *Model1* predstavlja ime okruženja. Zatim je potrebno aktivirati stvoreno okruženje naredbom:

conda activate Model1

Kako bi se pokrenulo treniranje i testiranje mreže, potrebno je instalirati *PyTorch* biblioteku naredbom:

pip install torch==1.9.0+cu111 torchvision==0.10.0+cu111 torchaudio==0.9.0 -f https://download.pytorch.org/whl/torch_stable.html

U radu su odabrane navedene verzije paketa.

PyTorch je radno okruženje (engl. *Frameowrk*) otvorenog koda za strojno učenje. Biblioteka je optimizirana za duboko učenje pomoću tenzora koristeći procesor (engl. *Central processing unit* - CPU) ili grafičku karticu (engl. *Graphics processing unit* - GPU). Bitno je napomenuti, kako je u radu korišten *TorchVision Object Detection Finetuning Tutorial* [15].

Nadalje, ulaz u mrežu predstavljaju slike s generiranim tekstom i pripadajuće maske generiranog teksta. Treba istaknuti i to kako pojedini skupovi podataka imaju unaprijed dostupne maske za preuzimanje, dok neke ipak nemaju. U radu iz korištenih skupova podataka, spomenutih u potpoglavlju 3.3.4., bilo je potrebno vlastito generirati maske (način generiranja opisan je u potpoglavlju 3.3.3.). Nakon generiranja maski za postojeće slike s generiranim tekstom, odradilo se sve potrebno kako bi se pokrenuo trening. Prije treninga bilo je potrebno definirati skup podataka koji se trebao naslijediti iz klase pod nazivom *torch.utils.data.Dataset*. Nakon definiranja skupa podataka, definiran je model *Mask R-CNN* koji je opisan u poglavlju 3.4.

Na slici 3.21. može se vidjeti postupak podjele skupa podataka na trening, validacijski i testni skup. Također, korišten je *randomperm* atribut kako bi se slike nasumično permutirale.

```
dataset = Dataset_ArT('Train/Curved_SynthText, get_transform(train=True))
indices = torch.randperm(len(dataset)).tolist()
dataset_train = torch.utils.data.Subset(dataset, indices[:-52442])
dataset_val = torch.utils.data.Subset(dataset_val, indices[20000:-22000])
dataset_test = torch.utils.data.Subset(dataset_test, indices[-1000:])
```

Slika 3.21. Primjer podjele *CurvedSynth10k* skupa podataka.

Nakon toga, napravljen je učitavač podataka (engl. *Dataloader*) za svaki skup koji će se koristiti za treniranje i testiranje mreže. Na slici 3.22. može se vidjeti proces treniranja gdje se prolazi kroz sve epohe i nakon svake epohe prati se iznos kriterijske funkcije (engl. *Loss function*) trening skupa (engl. *Train loss*) i validacijskog skupa (engl. *Validation loss*). Spomenuta se kriterijska funkcija koristi kako bi se na kraju iscrtao graf gubitaka trening i validacijskog skupa koristeći *writer.add_scalar()* funkciju. Također, nakon svake epohe modeli se spremaju u putanju koja se unosi pod *PATH* varijablom. Cilj je dobiti najbolji model, a najbolji model je onaj s najmanjim gubitkom na validacijskom skupu te se taj model kasnije koristi za proces testiranja. Za iscrtavanje grafa gubitaka korišten je *TensorBoard* alat za vizualizaciju, koji se instalira naredbom u terminalu:

pip3 install tensorboard

Zatim je potrebno, u terminalu, pozicionirati se u željeni direktorij i pokrenuti naredbu:

tensorboard --logdir=runs

gdje *runs* predstavlja direktorij u kojem će se spremati informacije potrebne za vizualizaciju grafa pomoću *TensorBoard* alata. Također, potrebno je koristiti klasu *SummaryWriter* kojoj se pridružuje putanja do direktorija iz kojega će *TensorBoard* čitati vrijednosti.

```
writer = SummaryWriter("runs/mnistCurved10_2_1")
PATH = '/home/Train/Curved_SynthText/Modeli_10_2_1/'
for epoch in range(num_epochs):
        loss_value = train_one_epoch(model, optimizer, data_loader, device,
epoch, print_freq=10)
        lr_scheduler.step()

        val_loss  = evaluate_loss(model, data_loader_val, device=device)

        torch.save(model, os.path.join(PATH, 'epoch-{}.pt'.format(epoch)))

        writer.add_scalar("Loss/train", loss_value, epoch)
        writer.add_scalar("Val/train", val_loss, epoch)


writer.close()
```

Slika 3.22. Prikaz koda za postupka treniranja.

Nakon što se dobije najbolji model za pojedini skup podataka iz faze treniranja, slijedi faza testiranja. Testiranje je obavljeno na vizualni i matematički način koji će biti detaljnije opisani u četvrtom poglavlju. Za pokretanje matematičkog načina testiranja, koristi se postojeća funkcija za testiranje. Potrebno je, nakon procesa treniranja, učitati model s najmanjim gubitkom na validacijskom skup, iz putanje u koju su se spremali modeli tijekom treniranja. Kako je vidljivo na slici 3.23. odabran je model nakon 30 epoha.

```
PATH = '/home/Train/Curved_SynthText/Modeli_10_2_1/epoch-30.pt'
model = torch.load(PATH)
evaluate(model, data_loader_test, device=device)
```

Slika 3.23. Prikaz koda za postupak testiranja.

4. EVALUACIJA PREDLOŽENOG RJEŠENJA

U ovom poglavlju opisani su rezultati detekcije teksta koji su trenirani na generiranim skupovima podataka koji su opisani u potpoglavlju 3.3.4. Prilikom testiranja korišteno je računalo s *Linux* operacijskim sustavom (*Ubuntu 20.04. LTS*), procesor *Intel Core i9-11900F* s *32GB RAM* i grafička kartica *NVIDIA GeForce RTX3080Ti*. Metrike koje su se koristile za testiranje u radu vezane su uz prosječnu preciznost. Kako bi se došlo do pojma prosječna preciznost, prvo će se objasniti pojmovi preciznost (engl. *Precision)* i odziv (engl. *Recall)*. Također, prvo je potrebno objasniti elemente pomoću kojih se računaju preciznost i odziv, a to su:

- Istinito pozitivan rezultat (engl. *True positive* - TP) – predikcija modela da na određenoj lokaciji postoji tekst, što je ujedno točno, tekst postoji.
- Lažno pozitivan rezultat (engl. *False positive* – FP) – predikcija modela da na određenoj lokaciji postoji tekst, ali tekst ne postoji
- Lažno negativan rezultat (engl. *False negative* – FN) – slika koja nije detektirana kao tekst, a ona to je.

Pri tome se koristi IoU (engl. *Intersect over Union*) koji daje mjeru koliko je točno predviđen granični pravokutnik u odnosu na stvarni granični pravokutnik (engl. *Ground truth*), kako bi se rezultat ocijenio kao točna ili netočna detekcija. Preciznost je omjer između broja istinito pozitivnih detektiranih rezultata i ukupnog broja pozitivnih rezultata.

$$\text{preciznost} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{4-1}$$

Odziv predstavlja omjer broja istinito pozitivnih rezultata i zbroja istinito pozitivnih i lažno negativnih rezultata.

$$\text{odziv} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{4-2}$$

S obzirom na prethodno napisano dolazi se do toga da se prosječna preciznost računa za svaku klasu posebno, a definirana je područjem ispod preciznost-odziv krivulje, gdje *y*-os predstavlja preciznost, dok *x*-os predstavlja odziv. Nakon dobivanja prosječne preciznosti za svaku klasu, računa se ukupna prosječna preciznost koja se računa kao zbroj svih prosječnih preciznosti podijeljenih s brojem klasa. Metrike prosječne preciznosti koje su se koristile prilikom evaluacije su:

- *AP50* – prosječna preciznost kada je presjek preko unije (*engl. Intersect over Union, IoU*) veći od 50%
- *AP75* – prosječna preciznost kada je *IoU* preko 75%
- *APs* – prosječna preciznost za male objekte, a to su objekti koji su površinom manji od 32^2 elemenata slike
- *APm* - prosječna preciznost za objekte srednje veličine, a to su objekti koji su površinom između 32^2 i 96^2 elemenata slike
- *APl* – prosječna preciznost za velike objekte, a to su objekti koji su površinom veći od 96^2 elemenata slike.

Vizualni način pokazuje vizualnu detekciju teksta na slikama, dok matematički način prikazuje razne varijacije metrika prosječne preciznosti. Za vizualno testiranje korištena je postojeća skripta dostupna na [16]. Za pokretanje vizualnog testiranja potrebno je prvo napraviti tri direktorija:

- *input* - u ovom direktoriju nalaze se ulazne slike na kojima će se testirati
- *src* – u ovom direktoriju nalazi se sav potreban kod koji se koristi za vizualizaciju, dostupno na [16]
- *outputs* – u ovom direktoriju spremaju se izlazne slike nakon vizualizacije

U datoteci *mask_rcnn_images.py* koja se nalazi u *src* direktoriju, potrebno je postaviti putanju do željenog modela koji će se koristiti za testiranje slika, nakon toga ga je potrebno učitati pomoću *model=torch.load(PATH)*. Osim toga, kako bi se pokrenula vizualizacija, potrebno je u terminalu postaviti putanju do *src* datoteke i pokrenuti naredbu:

python3 mask_rcnn_images.py --input ../input/image.jpg

gdje *input* predstavlja direktorij s ulaznim slikama, a *image* predstavlja ime željene slike.

Na slici 4.1. vidljiv je prikaz vizualnog testiranja jedne slike iz generiranog skupa podataka sa zakrivljenim tekstom, a koja se ne nalazi u trening skupu, ni u validacijskom, a ni u testnom skupu. Korišten je prag (engl. *Threshold*) 0.965.

Slika 4.1. Prikaz izlaza iz mreže generirane slike.

Uz vizualni prikaz, dan je još i prikaz izlaza matematičkog testiranja što se može vidjeti na slici 4.2.

```
Accumulating evaluation results...
DONE (t=0.12s).
IoU metric: bbox
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.876
 Average Precision  (AP) @[ IoU=0.50       | area=   all | maxDets=100 ] = 0.980
 Average Precision  (AP) @[ IoU=0.75       | area=   all | maxDets=100 ] = 0.960
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.797
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.879
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.904
```

Slika 4.2. Prikaz rezultata matematičkog testiranja.

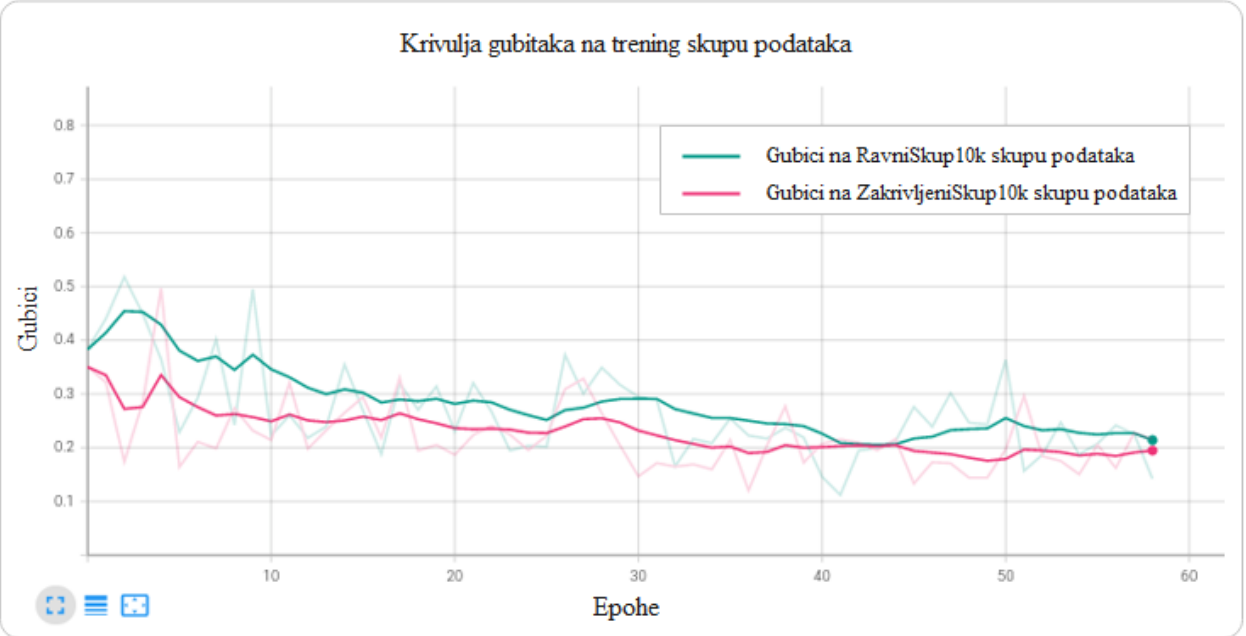
4.1. Postignuti rezultati nakon testiranja

Ranije je u radu, u potpoglavlju 3.3.4., detaljno opisana podjela skupova podataka koji su se koristili prilikom treniranja i testiranja *Mask R-CNN* mreže. Bitno je naglasiti kako svaki spomenuti trening skup podataka sadrži pripadni validacijski i testni skup, koji su dani u tablici 3.1.

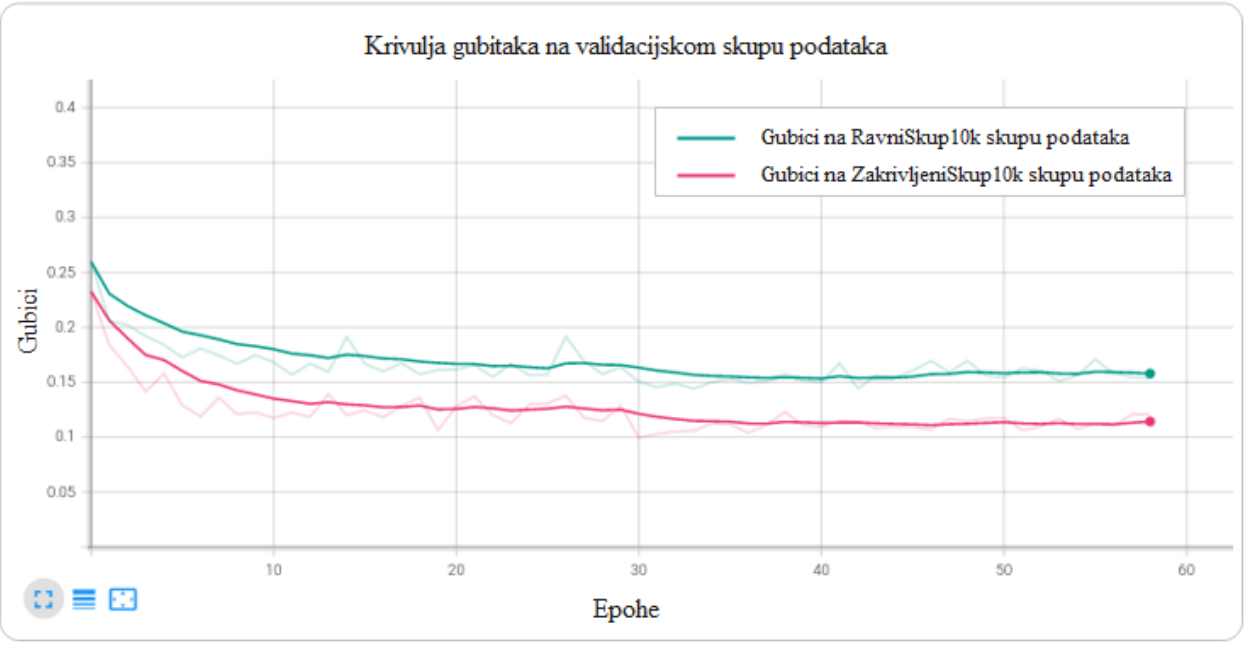
4.1.1. Rezultati testiranja na modelima koji su trenirani na 10 000 slika

Prilikom treniranja *Mask R-CNN* mreže, odlučeno je kako će svaki proces treniranja sadržavati 60 epoha gdje se nakon svake epohe spremaju modeli, odnosno, ukupno 60 modela. Najbolji je model izabran na temelju krivulje gubitaka na izabranom validacijskom skupu. Primjer

krivulje gubitaka na skupovima trening podataka s generiranim zakrivljenim i ravnim tekstom vidljiv je na slici 4.3. dok je na slici 4.4. prikazana krivulja gubitaka na validacijskim skupovima podataka s generiranim zakrivljenim i ravnim tekstom.



Slika 4.3. Prikaz funkcije gubitaka na trening skupu tijekom epoha, gdje je zelenom bojom označen rezultat učenja na *RavniSkup10k* skupu podataka, dok je ružičastom bojom označen rezultat učenja na *ZakrivljeniSkup10k* skupu podataka.



Slika 4.4. Prikaz funkcije gubitaka na validacijskom skupu tijekom epoha, gdje je zelenom bojom označen rezultat učenja na *RavniSkup10k* skupu podataka, dok je ružičastom bojom označen rezultat učenja na *ZakrivljeniSkup10k* skupu podataka.

Za testiranje je izabran model koji je imao najmanje gubitke na validacijskom skupu. U *ZakrivljeniSkup10k* skupu podataka, najbolji model je nakon 30 epoha, dok je najbolji model u *RavniSkup10k* skupu podataka, nakon 42 epohe. Rezultati detekcije teksta testiranih modela na testnom skupu vidljivi su u tablici 4.1. te kako bi se razumjela tablica objasnit će se pojmovi koji se nalaze u njoj:

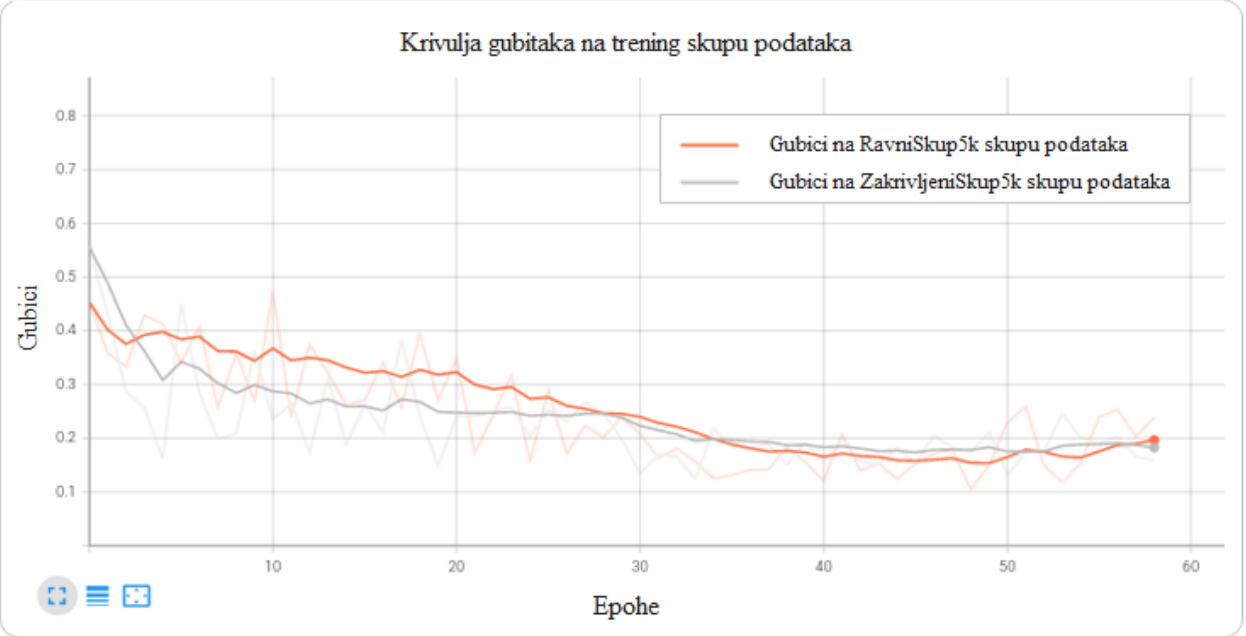
- T_MR_RSP - model koji je treniran i testiran na *RavniSkup10k* skupu podataka.
- T_MR_ZSP - model koji je treniran na *RavniSkup10k* skupu podataka i testiran na *ZakrivljeniSkup10k* skupu podataka.
- T_MZ_ZSP - model koji je treniran i testiran na *ZakrivljeniSkup10k* skupu podataka.
- T_MZ_RSP - model koji je treniran na *ZakrivljeniSkup10k* skupu podataka i testiran na *RavniSkup10k* skupu podataka.

Tablica 4.1. Prikaz usporedbe rezultata modela testiranih na različitim skupovima podataka gdje se proces treniranja sastojao od 10 000 slika.

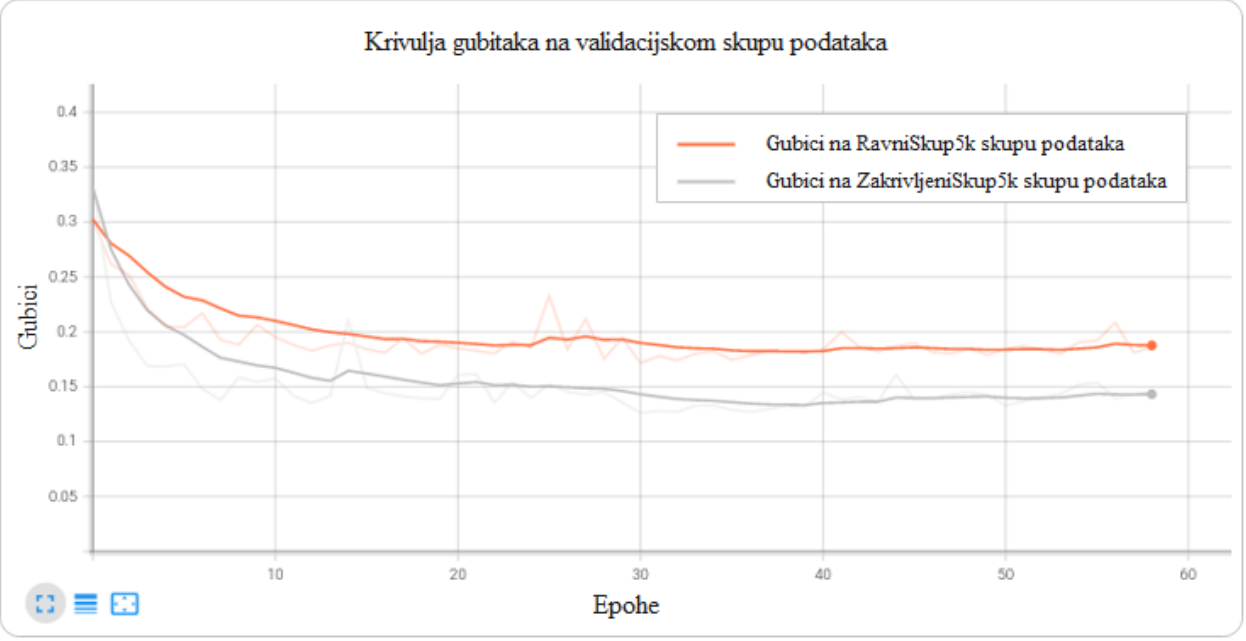
Metrike detekcije objekata						
Modeli	AP	AP50	AP75	APs	APm	APl
T_MR_RSP	82.3	97.9	94.6	74.9	82.6	81.7
T_MZ_ZSP	87.6	98.0	96.0	79.7	87.9	90.4
T_MZ_RSP	61.9	97.5	70.4	66.8	62.6	54.6
T_MR_ZSP	52.1	89.0	58.6	65.9	52.8	36.2

4.1.2. Rezultati testiranja na modelima koji su trenirani na 5000 slika

Prilikom treniranja *Mask R-CNN* mreže, odlučeno je kako će svaki proces treniranja sadržavati 60 epoha gdje se nakon svake epohe spremaju modeli, odnosno, ukupno 60 modela. Najbolji je model izabran na temelju krivulje gubitaka na izabranom skupu. Primjer krivulje gubitaka na skupovima trening podataka s generiranim zakrivljenim i ravnim tekstom vidljiv je na slici 4.5. dok je na slici 4.6. prikazana krivulja gubitaka na validacijskom skupu podataka s generiranim zakrivljenim i ravnim tekstom



Slika 4.5. Prikaz funkcije gubitaka na trening skupu tijekom epoha, gdje je narančastom bojom označen rezultat učenja na *RavniSkup5k* skupu podataka, dok je sivom bojom označen rezultat učenja na *ZakrivljeniSkup5k* skupu podataka.



Slika 4.6. Prikaz funkcije gubitaka na validacijskom skupu tijekom epoha, gdje je narančastom bojom označen rezultat učenja na *RavniSkup5k* skupu podataka, dok je sivom bojom označen rezultat učenja na *ZakrivljeniSkup5k* skupu podataka.

Za testiranje izabran je model koji je imao najmanje gubitke na validacijskom skupu. U *ZakrivljeniSkup10k* skupu podataka, najbolji model je nakon 30 epoha, dok je najbolji model u

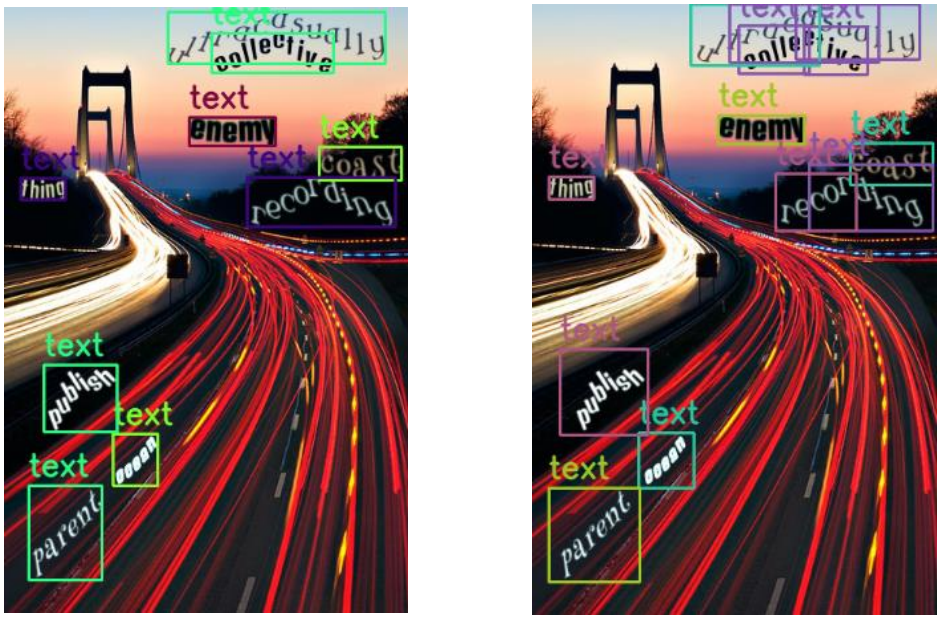
RavniSkup10k skupu podataka, nakon 30 epohe. Rezultati detekcije teksta testiranih modela na testnom skupu vidljivi su u tablici 4.2. Sve skraćenice objašnjene su u potpoglavlju 4.1.1.

Tablica 4.2. Prikaz usporedbe rezultata modela testiranih na različitim skupovima podataka gdje se proces treniranja sastojao od 5000 slika.

Metrike detekcije objekata						
Modeli	AP	AP50	AP75	APs	APm	APl
T_MR_RSP	80.2	97.8	93.4	75.0	80.7	77.8
T_MZ_ZSP	86.7	97.9	95.9	79.8	87.0	87.5
T_MZ_RSP	61.4	96.5	70.7	60.9	62.3	53.2
T_MR_ZSP	55.4	90.4	63.9	65.9	55.8	44.4

4.2. Analiza dobivenih rezultata

Proces treniranja mreže na korištenim skupovima podataka poprilično je dug i računalno zahtjevan. Učenje *Mask R-CNN* mreže na trening skupu koji sadrži 10 000 slika, trajalo je oko 16 sati, dok je učenje mreže na trening skupu od 5000 slika trajalo oko 10 sati. Kako je vidljivo iz tablice 4.1. prosječna preciznost je dosta velika na modelima koji su trenirani i testirani na skupu podataka s jednakim načinom generiranja teksta (ravne instance ili zakrivljene instance teksta), dok oni modeli koji su trenirani na jednom skupu, a testirani na drugom skupu imaju lošije rezultate. Tako se vidi, kako model, koji je treniran na *ZakrivljeniSkup10k* skupu podataka i testiran na *RavniSkup10k* skupu podataka, postiže bolje rezultate od modela koji je treniran na *RavniSkup10k* skupu podataka i testiran na *ZakrivljeniSkup10k* skupu podataka. No iz tablice 4.2., vidljivo je kako se dolazi do istog zaključka koji je donesen u tablici 4.1. Analizom dobivenih rezultata utvrđeno je kako se prosječna preciznost nije značajno smanjila kada je korišten skup podataka s 5000 slika u odnosu na skup podataka s 10 000 slika. Moguće je koristiti i manji skup podataka, iako se s većim skupom podataka dobije bolja detekcija zakrivljenog teksta. Također, uz dane tablične rezultate, dan je i vizualni prikaz slike iz testnog skupa podataka sa zakrivljenim tekstom. Na slici 4.7.(a) može se vidjeti detekcija teksta pomoću modela koji je istreniran na *ZakrivljeniSkup10k* skupu podataka, dok se na slici 4.7.(b) može vidjeti detekcija teksta pomoću modela koji je istreniran na *RavniSkup10k* skupu podataka.



a) b)

Slika 4.7. a) Vizualni prikaz detekcije teksta modela učenog na *ZakrivljeniSkup10k* skupu podataka, b) Vizualni prikaz detekcije teksta modela učenog na *RavniSkup10k* skupu podataka.

Kako je vidljivo iz slike 4.7. model koji je treniran na *ZakrivljeniSkup10k* skupu podataka, bolje detektira instance teksta od modela koji je treniran na *RavniSkup10k* skupu podataka. Na slici 4.7.(b) može se vidjeti kako model koji je treniran na *RavniSkup10k* skupu podataka neke instance zakrivljenog teksta detektira kao dva odvojena teksta. Također, dan je i vizualni prikaz (slika 4.8.) nasumične slike koja se ne nalazi niti u jednom skupu podataka korištenih u radu gdje se također može vidjeti kako model koji je treniran na *ZakrivljeniSkup10k* skupu podataka, bolje detektira instance teksta od modela koji je treniran na *RavniSkup10k* skupu podataka.

a) b)

Slika 4.8. a) Vizualni prikaz detekcije teksta modela učenog na *ZakrivljeniSkup10k* skupu podataka, b) Vizualni prikaz detekcije teksta modela učenog na *RavniSkup10k* skupu podataka.

5. ZAKLJUČAK

U ovome je diplomskome radu ispitana efikasnost detektora teksta temeljenog na dubokim neuronskim mrežama. Način na koji je ispitana efikasnost jest povećanje preciznosti označavanja slika na sintetički generiranom skupu podataka na kojem je tekst smješten u slike prirodnih scena i sadrži oznake teksta na razini pravokutnog graničnog okvira. Također, kako bi se moglo vidjeti poboljšava li se detekcija teksta prilikom treniranja modela na skupu s preciznijim oznakama, kreirani su vlastiti skupovi podataka gdje skup podataka s generiranim ravnim tekstom ima oznake teksta na razini pravokutnog graničnog okvira, dok skup s generiranim zakrivljenim tekstom ima oznake teksta na razini graničnog okvira u obliku poligona. Osim toga, ispitana je prosječna preciznost na dvama skupovima podataka u četiri varijacije. Na temelju dobivenih računskih podataka, primjećuje se kako prosječna preciznost detekcije teksta raste prilikom povećanja broja slika u skupu podataka, što je bilo očekivano. Može se zaključiti kako se generiranjem graničnih okvira u obliku poligona oko instance teksta poboljšava detekcija teksta u odnosu na granične okvire u obliku pravokutnika. Rezultati detekcije teksta potkrijepljeni su i vizualno, gdje se jasno vidi kako učenje mreže s graničnim okvirima u obliku poligona, poboljšava detekciju teksta. Daljnje povećanje preciznosti detekcije teksta, moglo bi se poboljšati dodatnim povećanjem broja slika u skupu podataka kao i poboljšanjem preciznosti oznaka teksta. Također, broj ulaznih slika pri treniranju te sam proces treniranja i testiranja mogao bi se poboljšati korištenjem veće računalne moći, npr. jače grafičke kartice.

LITERATURA

[1] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN." arXiv, Jan. 24, 2018. Accessed: Nov. 21, 2022. [Online]. Available: http://arxiv.org/abs/1703.06870

[2] C.-K. Ch'ng, C. S. Chan, and C.-L. Liu, "Total-Text: toward orientation robustness in scene text detection," *Int. J. Doc. Anal. Recognit. IJDAR*, vol. 23, no. 1, pp. 31–52, Mar. 2020, doi: 10.1007/s10032-019-00334-z.

[3] C. C. Kheng, "CURVED TEXT DETECTION AND GROUND TRUTH GENERATION FOR NATURAL SCENE IMAGES," p. 130, 2018.

[4] Y. Liu, L. Jin, S. Zhang, C. Luo, and S. Zhang, "Curved scene text detection via transverse and longitudinal sequence connection," *Pattern Recognit.*, vol. 90, pp. 337–345, Jun. 2019, doi: 10.1016/j.patcog.2019.02.002.

[5] Z. Liu, G. Lin, S. Yang, F. Liu, W. Lin, and W. L. Goh, "Towards Robust Curve Text Detection With Conditional Spatial Expansion," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Long Beach, CA, USA, Jun. 2019, pp. 7261–7270. doi: 10.1109/CVPR.2019.00744.

[6] A. Gupta, "SynthText." Nov. 16, 2022. Accessed: Nov. 21, 2022. [Online]. Available: https://github.com/ankush-me/SynthText

[7] T. pip developers, "pip: The PyPA recommended tool for installing Python packages." Accessed: Nov. 28, 2022. [Online]. Available: https://pip.pypa.io/

[8] "About," *OpenCV*. https://opencv.org/about/ (accessed Nov. 21, 2022).

[9] "NumPy." https://numpy.org/ (accessed Nov. 21, 2022).

[10] "Matplotlib — Visualization with Python." https://matplotlib.org/ (accessed Nov. 21, 2022).

[11] "HDF5 for Python." https://www.h5py.org/ (accessed Nov. 21, 2022).

[12] "Tasks - ICDAR2019 Robust Reading Challenge on Arbitrary-Shaped Text - Robust Reading Competition." https://rrc.cvc.uab.es/?ch=14&com=tasks (accessed Nov. 21, 2022).

[13] "Everything about Mask R-CNN: A Beginner's Guide - viso.ai." https://viso.ai/deep-learning/mask-r-cnn/ (accessed Nov. 21, 2022).

[14] F. Mravić, "Automatsko detektiranje rukometnog terena na slici," p. 36.

[15] "TorchVision Object Detection Finetuning Tutorial — PyTorch Tutorials 1.13.0+cu117 documentation." https://pytorch.org/tutorials/intermediate/torchvision_tutorial.html (accessed Nov. 30, 2022).

[16] "Instance Segmentation with PyTorch and Mask R-CNN," *DebuggerCafe*, Nov. 23, 2020. https://debuggercafe.com/instance-segmentation-with-pytorch-and-mask-r-cnn/ (accessed Nov. 28, 2022).

SAŽETAK

Cilj ovog diplomskog rada je ispitati hoće li se generiranjem zakrivljenog teksta na skupu podataka poboljšati preciznost detektora teksta. Uz to, opisana su rješenja i dan je pregled postojećih skupova podataka za detekciju teksta. Također, opisan je i proces izrade vlastitih skupova podataka te sam proces treniranja i testiranja duboke neuronske mreže. Na kraju rada su evaluirani i analizirani dobiveni rezultati koji su predstavljeni tablicama i slikama.

Ključne riječi: detekcija teksta, duboka neuronska mreža, generiranje zakrivljenog teksta.

ABSTRACT

The main goal of this thesis is to examine whether the accuracy of text detector will be improved by generating curved text on the dataset. In addition, solutions are described and an overview of already existing datasets for text detection is given. Also, the process of creating own datasets and the process of training and testing a deep neural network is described. At the end of the thesis the obtained results were evaluated and analyzed which is presented in tables and figures.

Keywords: deep neural network, generating curved text, text detection.

ŽIVOTOPIS

Leon Landeka rođen je 1. 8. 1997. u Vinkovcima. Završio je tehničku školu Ruđera Boškovića u Vinkovcima. Nakon srednje škole upisuje Fakultet elektrotehnike, računarstva i informacijskih tehnologija, tadašnji Elektrotehnički fakultet u Osijeku, smjer Elektrotehnika. Godine 2020. stječe akademski naziv sveučilišni prvostupnik (lat. Baccalaureus) inženjer elektrotehnike i informacijskih tehnologija. Iste godine upisuje diplomski sveučilišni studij automobilsko računarstvo i komunikacije na istom fakultetu. Na drugoj godini diplomskog studija postaje stipendist tvrtke *TTTechAuto*, tadašnji institut RT-RK u Osijeku. Aktivno se služi engleskim jezikom u govoru i pismu.

Potpis:

PRILOZI

- **P.3.1.** Kod za izradu JSON datoteke za skup podataka s ravnim tekstom.

```
with open("synthTextRAVNI.json") as json_file:
    data=ujson.load(json_file)
    temp=data["annotations"]
    #temp.append(finalDict)

def write_json(data, filename='synthTextRAVNI.json'):
    with open(filename,'w') as file:
        ujson.dump(data,file,indent=4)

def viz_textbb(text_im, imageName, charBB_list, wordBB, textToList, alpha=1.0):
    coordinate = []
    name = []
    imageData = { }
    dictList = []
    for i in range(0,len(textToList)):
        for j in range (0,4):
            coordinate.append([wordBB[0][0][j][i], wordBB[0][1][j][i]])
        name.append(coordinate)
        coordinate=[]
        result = []
        for sublist in name[0]:
            float_sublist = []
            for x in sublist:
                float_sublist.append(int(x))
            result.append(float_sublist)

        imageData['transcription']=i
        imageData['language']="Latin"
        imageData['illegibility']=False
        imageData['points']=result
        dictionary_copy = imageData.copy()
        dictList.append(dictionary_copy)

        del(dictionary_copy)
        result=[]
        name=[]

    finalDict = {f'ime_{imageName}':dictList}
    temp.update(finalDict)

    write_json(data)

def main(db_fname):
    db = h5py.File(db_fname, 'r')
    dsets = sorted(db['data'].keys())
```

```
    print ("total number of images : ", colorize(Color.RED, len(dsets),
highlight=True))
    for k in dsets:
        rgb = db['data'][k][...]
        charBB = db['data'][k].attrs['charBB']
        wordBB = db['data'][k].attrs['wordBB']
        txt = db['data'][k].attrs['txt']
        textToList = (db['data'][k].attrs['txt']).tolist()
        wordList = db['data'][k].attrs['wordBB'].tolist()

        viz_textbb(rgb, k,[charBB], [wordList], textToList)
        print ("image name        : ", colorize(Color.RED, k, bold=True))
        print ("  ** no. of chars : ", colorize(Color.YELLOW, charBB.shape[-
1]))
        print ("  ** no. of words : ", colorize(Color.YELLOW, wordBB.shape[-
1]))
        print ("  ** text         : ", colorize(Color.GREEN, txt))

    db.close()
json_file.close()
if __name__=='__main__':
    main('results/synthRAVNI.h5')
```

- **P.3.2.** Kod za izradu JSON datoteke za skup podataka sa zakrivljenim tekstom.

```
with open("CurvedSynth.json") as json_file:
    data=ujson.load(json_file)
    temp=data["annotations"]
    #temp.append(finalDict)

def write_json(data, filename='CurvedSynth.json'):
    with open(filename,'w') as file:
        ujson.dump(data,file,indent=4)

def viz_textbb(text_im, imageName, charBB_list, wordBB, textToList, alpha=1.0):
    start = 0
    coordinate = []
    name = []
    upperList = []
    downList = []
    FinalFinal = []
    imageData = { }
    dictList = []

    for eachWord in textToList:
        length = len(eachWord)
        for i in range(0,4):
            for j in range(start,length+start):
```

```
                coordinate.append([charBB_list[0][0][i][j],
charBB_list[0][1][i][j]])
            name.append(coordinate)
            coordinate = []
        for j in range(0, length):
            for i in range(len(name)) :
                if(i == 0 or i == 1):
                    upperList.append(name[i][j])
                if(i == 2):
                    downList.append(name[i+1][j])
                if(i == 3):
                    downList.append(name[i-1][j])

        down = reversed(downList)
        joinList = [*upperList,*down,upperList[0]]
        for element1 in range(len(joinList)):
            for element2 in range(len(joinList[element1])):
                joinList[element1][element2]= int(joinList[element1][element2])

        FinalFinal.append(joinList)

        imageData['transcription']=eachWord
        imageData['language']="Latin"
        imageData['illegibility']=False
        imageData['points']=joinList
        dictionary_copy = imageData.copy()
        dictList.append(dictionary_copy)

        del(dictionary_copy)
        name=[]
        upperList = []
        downList = []
        start = len(eachWord) + start

    finalDict = {f'ime_{imageName}':dictList}
    temp.update(finalDict)

    write_json(data)

def main(db_fname):
    db = h5py.File(db_fname, 'r')
    dsets = sorted(db['data'].keys())
    print ("total number of images : ", colorize(Color.RED, len(dsets),
highlight=True))
    for k in dsets:
        rgb = db['data'][k][...]
        charBB = db['data'][k].attrs['charBB']
        wordBB = db['data'][k].attrs['wordBB']
        txt = db['data'][k].attrs['txt']
```

```
        textToList = (db['data'][k].attrs['txt']).tolist()

        viz_textbb(rgb, k,[charBB], wordBB, textToList)

        print ("image name        : ", colorize(Color.RED, k, bold=True))
        print ("  ** text         : ", colorize(Color.GREEN, txt))

    db.close()
json_file.close()
if __name__=='__main__':
    main('results/CurvedSynthText.h5')
```