Unaprjeđenje detekcije teksta na slikama preciznim označavanjem skupova za učenje

Dorkić, Ivan

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

Sveučilišni studij

UNAPRJEĐENJE DETEKCIJE TEKSTA NA SLIKAMA PRECIZNIM OZNAČAVANJEM SKUPOVA ZA UČENJE

Diplomski rad

Ivan Dorkić

Osijek, 2022.

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

Obrazac D1: Obrazac za imenovanje Povjerenstva za diplomski ispit

Osijek, 29.11.2022.

Odboru za završne i diplomske ispite

Imenovanje Povjerenstva za diplomski ispit

Ime i prezime Pristupnika:	Ivan Dorkić
Studij, smjer:	Diplomski sveučilišni studij Automobilsko računarstvo i komunikacije
Mat. br. Pristupnika, godina upisa:	D-45ARK, 11.10.2020.
OIB studenta:	21224672198
Mentor:	Izv.prof.dr.sc. Ratko Grbić
Sumentor:	,
Sumentor iz tvrtke:	Matteo Brisinello
Predsjednik Povjerenstva:	Izv. prof. dr. sc. Mario Vranješ
Član Povjerenstva 1:	Izv.prof.dr.sc. Ratko Grbić
Član Povjerenstva 2:	Prof. dr. sc. Marijan Herceg
Naslov diplomskog rada:	Unaprjeđenje detekcije teksta na slikama preciznim označavanjem skupova za učenje
Znanstvena grana diplomskog rada:	**Umjetna inteligencija (zn. polje računarstvo)**
Zadatak diplomskog rada:	Moderni pristupu detekciji teksta u slikama temelje se na dubokim neuronskim mrežama. Za izgradnju ovakih mreža potrebno je imati adekvatne skupove podataka za učenje mreže što znači da je svaki pristuni tekst u slici označen s graničnim pravokutnikom i preciznom maskom na razini piksela. U ovom radu, potrebno je istražiti postojeće skupove podataka za detekciju teksta na slikama. Potrebno je provjeriti postoji li način generiranja precizne maske iz neprecizno označenih podataka. Nakon odabira / generiranja skupa podataka s preciznom maskom potrebno je odabrati jednu duboku neuronsku mrežu koja je pogodna za treniranje nad takvim podacima. Nakon provedenog učenja mreže potrebno je mrežu evaluirati te donijeti odgovarajuće zaključke
Prijedlog ocjene pismenog dijela ispita (diplomskog rada):	Vrlo dobar (4)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 2 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:	29.11.2022.

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:	Ivan Dorkić
Studij:	Diplomski sveučilišni studij Automobilsko računarstvo i komunikacije
Mat. br. studenta, godina upisa:	D-45ARK, 11.10.2020.
Turnitin podudaranje [%]:	4

Ovom izjavom izjavljujem da je rad pod nazivom**: Unaprjeđenje detekcije teksta na slikama preciznim označavanjem skupova za učenje**

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

Detekciju teksta (engl. *text detection*) možemo definirati kao proces pronalaska teksta prisutnog na slici, nakon čega se on okružuje graničnim okvirom (engl. *bounding box*). Granični okvir je najčešće predstavljen u obliku pravokutnika ili poligona s četiri točke. Detekcija teksta za čovjeka predstavlja iznimno jednostavnu radnju dok za računalo je to vrlo složen postupak koji uključuje odgovarajuće algoritme za obradu slike.

Detekcija teksta na slikama je grana računalnog vida koja je u konstantnom usponu. Tekstove s kojima se susrećemo u našim svakodnevnim životima možemo podijeliti u dvije kategorije: strukturirani tekst i nestrukturirani tekst. Strukturirani tekst je tekst s kojim se susrećemo u knjigama, službenim dokumentima itd. To je tekst koji se nalazi na čitkoj pozadini, standardnog fonta i proreda. Danas postoje algoritmi koji vrlo efikasno prepoznaju i detektiraju takav tekst. Nestrukturiran tekst je onaj kojeg nalazimo nasumičnim mjestima u prirodnom okruženju. On se često nalazi na kompleksnim mjestima, nema standardizirani font, prored je nestandardiziran, različitih je boja, često je i zakrivljen. Takav tekst najčešće se nalazi na raznim plakatima, reklamama, nazivima kafića i sl.

Primjeri nestrukturiranog teksta mogu biti izrazito teški računalu za detekciju teksta. Detekcija teksta danas se rješava pomoću umjetnih neuronskih mreža. Za učenje jedne takve mreže potreban je odgovarajuće označeni podatkovni skup. Odgovarajuće označenim podatkovnim skupom smatra se skup koji sadrži oznake gdje se nalazi tekst s pripadajućim graničnim okvirom te što preciznije oznake segmentacijske maske tog teksta. Segmentacijska maska (engl. *segmentation mask*) predstavlja oznaku svakog elementa slike unutar graničnog okvira koji se odnosi na tekst. Granični okvir i segmentacijska maska su potrebni jer se danas problem detekcije teksta najčešće rješava pomoću metode segmentacije instanci (engl. *instance segmentation*). Segmentacija instanci je tehnika detekcije, segmentacije i klasifikacije svakog objekta na slici. Današnje metode još uvijek imaju problem prilikom detekcije teksta u prirodnim slikama. Najveći problem čine tekstovi male veličine i koji su slične boje kao pozadina. Stoga je cilj u ovom radu unaprijediti detekciju teksta na slikama na način da pri treniranju neuronskih mreža koristimo skup slika sa što preciznijim oznakama. U postojećim radovima takve mreže su trenirane sa skupovima gdje je cijeli granični okvir ujedno i segmentacijska maska, a u ovom radu će masku predstavljati samo elementi slike koji pripadaju tekstu. Što preciznije označavanje slika će se postići na način da se modificira postojeća skripta za generiranje i označavanje slika, kako bi ona označavala tekst na razini

elemenata slike. Cilj rada je pokazati hoće li skupovi s preciznim oznakama dati bolje rezultate detekcije teksta od skupova sa nepreciznim oznakama. Precizne oznake će označavati tekst na razini elemenata slike, a neprecizne oznake će označavati cjelokupan prostor unutar graničnog okvira kao tekst.

Rad je strukturiran na sljedeći način. U drugom poglavlju dan je opis problema prilikom detekcije i označavanja teksta, zatim pregled postojećih podatkovnih skupova za detekciju teksta i pregled rješenja za preciznu detekciju teksta. U trećem poglavlju je opisan proces izrade i označavanja šest novih skupova podataka. Tri skupa sadržavaju precizne oznake, a tri neprecizne oznake. Opisan je i sam proces treniranja neuronske mreže i upute za podešavanje i instalaciju potrebnih paketa za generiranje slika sa tekstom i njihovih oznaka. Dane su i upute za treniranje i testiranje neuronske mreže za detekciju teksta koja se temelji na ulaznim podacima u vidu prethodno generiranih slika. U četvrtom poglavlju je detaljno opisana evaluacija dobivenog rješenja za detekciju teksta, a na samom kraju rada dan je zaključak.

2. DETEKCIJA TEKSTA U SLIKAMA I PREGLED POSTOJEĆIH RJEŠENJA

U ovom poglavlju dan je opis problema koji nastaju pri pokušaju detekcije teksta u prirodnim slikama. Dan je pregled postojećih podatkovnih skupova koji mogu služiti za izgradnju i evaluaciju detektora teksta. U zadnjem poglavlju je napravljen pregled nekoliko modernih rješenja za detekciju teksta.

2.1. Opis problema

Većina metoda za detekciju teksta, barem one najefikasnije, temelje se na dubokim neuronskim mrežama. Najčešće se koriste arhitekture neuronskih mreža koje su namijenjene za segmentaciju instanci s odgovarajućim izmjenama, pomoću kojih se dobije detekcija teksta u prirodnim slikama. Izlaz iz takve mreže su instance teksta označene pravokutnim graničnim okvirima te se unutar tih okvira nalazi maska koja prikazuje područje teksta na razini elemenata slike. Da bi se mogle istrenirati neuronske mreže za detekciju teksta potrebno je koristiti nadzirano učenje. Za svaku instancu teksta na svakoj slici u trening skupu podataka treba biti označen granični okvir, te ukoliko to zahtjeva određena neuronska mreža, oznake segmentacijskih maski unutar graničnog okvira. Većina skupova podataka za detekciju teksta ne uključuje oznaku maske, stoga je takvu masku potrebno ručno definirati prilikom pripreme trening podataka ako je ona potrebna. Primjer slike na kojoj je tekst označen samo graničnim okvirom vidljiv je na slici 2.1. Može se koristiti nekoliko načina definiranja maski. Najjednostavniji način označavanja maske je označavanje svakog elementa slike unutar graničnog okvira kao što je vidljivo na slici 2.2. Još jedan način označavanja maske koji se često koristi u radovima je označavanje „suženog“ pravokutnika unutar graničnog okvira, odnosno označuje se manji pravokutnik unutar graničnog okvira.

Slika 2.1. Primjer označene slike samo sa graničnim okvirima

Slika 2.2. Primjer segmentacijske maske označene zelenom bojom

U teoriji, najprecizniji mogući način označavanja maske rezultirao bi maskom koju čine elementi slike unutar graničnog okvira koji upravo pripadaju tekstu na slici. Primjer takve maske prikazan je na slici 2.3.

Slika 2.3. Primjer precizne segmentacijske maske

2.2. Pregled postojećih podatkovnih skupova za detekciju teksta

Skup podataka *ICDAR 2013* sastoji se od 229 trening slika i 233 slike za testiranje. Sve slike su različite rezolucije. To je skup u kojemu su sve slike fotografirane kamerom te je tekst na njima prirodan. Tekst je označen na razini elemenata slike. To je standardni referentni skup podataka za detekciju horizontalnog teksta [1].

Synthtext in the wild je sintetički generiran skup slika u kojem su riječi smještene u slike prirodnih scena. Odabir gdje će se smjestiti tekst na sliku ovisi o izgledu scene i dubinskoj mapi scene. Na taj način je tekst dodan na dovoljno velike površine. Sastoji se od 800 tisuća slika sa otprilike 8 milijuna riječi umjetno dodanih na slike. Slike su različite rezolucije. Svaka riječ je označena sa pravokutnim graničnim okvirom na razini riječi i na razini slova. Svi elementi unutar graničnog okvira predstavljaju segmentacijsku masku [2].

COCO_Text skup podataka sadrži 63686 slika, različite rezolucije, gdje su riječi označene na razini elemenata slike. To je najveći skup slika sa prikazom teksta u prirodnom obliku, odnosno tekst na slikama nije sintetički generiran. Sastoji se većinom od horizontalnog i više-orijentiranog (engl. *multi-oriented*) teksta. Više-orijentiran tekst je tekst koji je okrenut pod određenim kutom, odnosno nije horizontalan. Manji dio slika sadrži i zakrivljen tekst [3].

MLT skup podataka sadrži 18000 slika označenih na razini elemenata slike. Sve slike su fotografirane kamerom te je tekst na njima prirodan. Slike su različite rezolucije. Tekst na slikama je prikazan u prirodnom obliku. Sadrži tekst na 10 različitih jezika i 6 različitih pisama [4].

Incidental Scene Text skup podataka, poznat još i pod imenom *ICDAR 2015*, sadrži 1475 slika označenih na razini elemenata slike. U ovom skupu podataka prikupljene su slike niže kvalitete, odnosno slike u kojima tekst nije u glavnom fokusu. Većina teksta na slikama je izvan fokusa i mutna [5].

Total-Text je skup podataka za detekciju teksta koji se sastoji od 1555 slika. Sadrži primjere horizontalnog, više-orijentiranog i zakrivljenog teksta, odnosno tekstove proizvoljnog oblika. *CTW1500* skup podataka koji je dosta sličan kao *Total-Text*. *CTW1500* sadrži 1500 slika. Oba skupa koriste poligon granične okvire i imaju barem jedan zakrivljeni tekst po slici. Također oba skupa koriste prirodne slike i sintetički generirane slike različitih rezolucija. *Total-Text* sadrži samo tekst na latinici, a *CTW1500* sadrži i kineski tekst. Glavna razlike je da *Total-Text* označava tekst na razini riječi te sadrži segmentacijsku masku na razini elemenata slike, a *CTW1500* označava tekst na razini linija (što znači da ako ima više povezanih riječi na slici, označene su kao jedan objekt) i segmentacijska maska je označena kao svi elementi slike unutar graničnog okvira [6].

2.3. Pregled rješenja za preciznu detekciju teksta

Detekcija teksta proizvoljnog oblika u prirodnim scenama iznimno je zahtjevan zadatak. Većina postojećih rješenja za detekciju teksta na slikama su zasnovani na neuronskim mrežama primarno namijenjenim za segmentaciju instanci koja dovodi do detekcije objekata. U [7] predložena je nova metoda naziva *TextFuseNet*, koja je temeljena uglavnom na arhitekturi *Mask R-CNN-a* i *TextSpottera*, koja koristi bogatije spojene značajke kako bi detektirali tekst. *TextFuseNet* prikuplja i spaja značajke tekstova s različitih razina koristeći višestaznu (engl. *multi-path*) fuzijsku arhitekturu koja može učinkovito uskladiti i spojiti različite prikaze. *TextFuseNet* detektira tekst na razini riječi i slova. Iako *TextFuseNet* za treniranje ne koristi oznake segmentacijske maske na razini elemenata slike, koristi preciznije poligon oznake segmentacijske maske. Te oznake su „sužene“ u odnosu na oznake većine drugih metoda gdje se sve unutar graničnog okvira smatra segmentacijskom maskom. Uspješnost detekcije na ICDAR2013 skupu slika iznosi 94,3%, na ICDAR2015 skupu iznosi 92,1%, na *Total-Text* skupu iznosi 87,1% i na CTW-1500 skupu iznosi 86,6%.

U [8] koristeći *wavelet* značajke (engl. *multiscale wavelet features*) autori predlažu novi „od grubog do finog“ (engl. *coarse-to-fine*) algoritam koji detektira tekst čak i ispod složene pozadine. U gruboj detekciji, nakon što su izračunate značajke valne energije za lociranje svih mogućih

elemenata slike teksta, razvijena je rastuća metoda područja temeljena na gustoći za povezivanje tih elemenata slike u područja koja su dalje odvojena u moguće redove teksta prema strukturnim informacijama. U finoj detekciji primjenjuje se algoritam koji pretražuje najučinkovitije značajke. Najučinkovitije značajke pretražuju se pomoću četiri vrste teksturnih značajki koje predstavljaju uzorak teksture retka teksta. Na kraju koristi se SVM klasifikator za binarnu klasifikaciju, koji prepoznaje objekte na slici te ih klasificira na temelju najučinkovitijih značajki. Detektirani objekt se može klasificirati kao tekst ili kao nešto što nije tekst.

Najveći problem u izgradnji efikasnih detektora objekata je nedostatak velikih skupova podataka koji imaju označen tekst na razini elemenata slike. U [9] je predložena metoda za označavanje teksta na slikama na razini elemenata slike. Korišten je ranije spomenuti *COCO_Text* skup podataka. Iz slika koje se nalaze u tom skupu, izrezano je 1000000 slika teksta po obrubu od pravokutnog graničnog okvira. Zatim su se te slike koristile za treniranje duboke neuronske mreže koja služi za segmentaciju teksta, odnosnu pomoću te mreže se razdvaja pozadina od samoga teksta. Poslije trening faze određena su dva praga. Ako je vjerojatnost manja od 0.3 to znači da je taj dio slike pozadina, ako je vjerojatnost veća od 0.7 to znači da je na slici tekst i ako je vjerojatnost između ta dva praga mreža je nesigurna u odluku. Pomoću te metode generirano je 14690 slika gdje su riječi označene na razini elemenata slike.

3. IZRADA DETEKTORA TEKSTA NA TEMELJU UMJETNO GENERIRANIH PRECIZNO OZNAČENIH SLIKA

Za izradu detektora teksta na prirodnim slikama potrebno je imati odgovarajući označeni skup podataka tj. slika. Pri tome, podaci trebaju biti precizno označeni. To znači da svaka instanca teksta na slici ima pravokutni granični okvir kojim se označava pozicija teksta na slici te masku koja označava slova teksta na razini elementa slike. Za potrebne generiranja podatkovnog skupa korištena je postojeća skripta za generiranje slika dostupna na [10] koja je preuređena kako bi osim pravokutnih graničnih okvira spremala i masku na razini elemenata slike za svaku pojedinu instancu teksta na svakoj slici. Skripta generira tekst koji može sadržavati slova i brojeve. Na taj način je moguće iskoristiti mreže koje tijekom učenja koriste i oznake teksta na razini znakova, odnosno segmentacijske maske.

U potpoglavlju 3.1. je detaljno opisan proces podešavanja i instalacije potrebnih alata i biblioteka kako bi skripta za generiranje podataka ispravno radila. U potpoglavlju 3.2. detaljno je opisan proces izgradnje vlastitog skupa podataka s precizno označenim tekstom na svakoj slici. U potpoglavlju 3.3. opisan je proces treniranja duboke neuronske mreže za detekciju teksta, dok su u potpoglavlju 3.4. dane upute za pokretanje testiranje mreže.

3.1. Podešavanje i instalacija potrebnih alata i biblioteka

Skripta za generiranje slika s tekstom je namijenjena za rad na *linux* operacijskom sustavu, stoga je potrebno, ako se koristi *windows* operacijski sustav instalirati alat za pokretanje više operacijskih sustava na jednom operacijskom sustavu. Alat koji je korišten u ovom radu je *VirtualBox*, koji je besplatan. Zatim je pomoću *VirtualBox-a* kreirano novo okruženje na kojem je instaliran *Ubuntu* operacijski sustav, verzija 18.0.4. *Ubuntu* je besplatan operacijski sustav koji je nastao kao izvedenica sustava Linux i koristi ga kao jezgru svog operacijskog sustava [11]. *Python* verzija koja je instalirana na *Ubuntu-u* je 3.6.9. Zatim je potrebno instalirati *pip*. *Pip* je alat za upravljanje *Python* paketima. Instalira se na način da se unese naredba *sudo apt install python3-pip* u terminal [12]. Nakon toga je potrebno instalirati pakete iz tablice 3.1.

Skripta se pokreće na način da se u terminalu pozicioniramo u direktorij gdje se nalazi skripta te unesemo naredbu *python3 gen.py –viz.* S tom naredbom se vizualizira i slika koju smo generirali zajedno s oznakama, kao što je vidljivo na slici 3.1. Ako se žele generirati slike bez prikaza,

naredba se pokreće bez argumenta --*viz*. U ovom radu slike se spremaju u formatu *.png* i spremaju se bez vidljivog pravokutnog graničnog okvira na njima. Oznake se spremaju kao tekstualni zapis u JSON datoteci, što je detaljno objašnjeno u nastavku rada.

Tab. 3.1. Prikaz *Python* paketa potrebnih za pokretanje skripte za generiranje slika sa tekstom

Ime paketa	**Verzija paketa**	**Naredba za instalaciju**
cached-property	1.5.2	pip3 install cached-property
cycler	0.11.0	pip3 install cycler
h5py	3.1.0	pip3 install h5py
kiwisolver	1.3.1	pip3 install kiwisolver
matplotlib	3.3.4	sudo apt install libjpeg-dev zlib1g-dev pip3 install matplotlib
numpy	1.19.5	pip3 install numpy
opencv	3.3.0.10	pip3 install opencv-python
pillow	8.4.0	pip3 install pillow
pygame	2.1.0	pip3 install pygame
pyparsing	3.0.6	pip3 install pyparsing
dateutil	2.8.2	pip3 install python-dateutil
scipy	1.5.4	pip3 install scipy
six	1.16.0	pip3 install six
wget	3.2	pip3 install wget

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

3.2. Izrada vlastitih skupova podataka

Za uspješno generiranje podatkovnog skupa potrebno je generirati slike s dvjema popratnim JSON datotekama koje sadrže određene informacije o slikama. U jednoj JSON datoteci maska teksta treba biti označena što preciznije, na razini elemenata slike. U drugoj datoteci maska treba biti cijelo područje unutar pravokutnog graničnog okvira. U skripti koja se koristi za umjetno generiranje slika s tekstom dostupne su informacije o koordinatama graničnog okvira i koordinatama gdje se nalazi tekst. Međutim, u skripti nije dostupna informacija koji točno elementi slike pripadaju tekstu koji se dodaje na sliku u procesu generiranja slike. Za kreiranje precizno označenog podatkovnog skupa potrebna je informacija koji elementi slike predstavljaju tekst, stoga je potrebno označiti tekst na slici na razini elemenata slike. Napravljena je funkcija, vidljiva na slici 3.2., koja boja svaki element slike po lokaciji koju nam pruža skripta. Funkcija prima generiranu sliku na kojoj se nalazi tekst i listu koordinata elemenata slike gdje se nalazi tekst te nakon toga vraća sliku gdje su elementi slike koji pripadaju tekstu označeni crvenom bojom. Na slici 3.3. prikazan je uvećani dio slike gdje se nalazi tekst i gdje se uočava navedeni problem -

određeni elementi slike koji su pozadina označeni su kao tekst. Na primjer, pozadina između dva znaka, pozadina unutar slova 'e' i sl.

```
def visulaizeTextLocation(self,img, loc):
    i=0
    copy_image=img.copy()
    copy_image=cv.cvtColor(copy_image,cv.COLOR_BGR2RGB)
    while i<len(loc[0]):
        locx=loc[0][i]
        locy=loc[1][i]
        cv.circle(img, (locy,locx), 0, (255,0,0),1)
        i+=1
    return img
```

Slika 3.2. Funkcija za označavanje elemenata slike koji pripadaju tekstu

Slika 3.3. Nedovoljno precizno označeni elementi slike teksta

3.2.1. Segmentacija slike teksta korištenjem *k* srednjih vrijednosti

Kako bi se preciznije označio tekst na razini elemenata slike, korištena je metoda *k*-srednjih vrijednosti (engl. *K-means clustering*) kako bi se segmentirala slika unutar pravokutnog graničnog okvira koji označava tekst. *K* srednjih vrijednosti jednostavan je algoritam nenadziranog učenja koji se koristi za rješavanje problema *klasteriranja* (grupiranja). Slijedi jednostavnu proceduru particioniranja zadanog skupa podataka u nekoliko *klastera* (grupa) definiranih brojem "k". Centri grupa se postavljaju kao točke, a svi ostali podaci povezuju se s najbližim centrom grupe, centri se preračunavaju i zatim proces počinje ispočetka korištenjem novih prilagodbi dok se ne postigne željeni rezultat [13]. Kada se algoritam *k* srednjih vrijednosti primijeni na sliku u boji, tada dobiveni centri predstavljaju dominantne boje na slici. Budući da je tekst uvijek iste boje, pretpostavka je da će se elementi slike koji čine tekst nalaziti u jednoj grupi, a svi ostali elementi slike (npr. pozadina) u ostalim grupama. U ovom radu je eksperimentalno utvrđen optimalni broj *klastera* i iznosi 4. Ulaz u funkciju *k* srednjih vrijednosti je slika izrezana po rubu pravokutnog graničnog okvira i broj koji predstavlja željeni broj grupa, a izlaz su slike koje predstavljaju svaku

grupu te slika na kojoj su vidljive sve grupe spojene na jednoj slici. Rezultat primjene algoritma *k* srednjih vrijednosti za sliku 3.3. dan je na slici 3.4. Od izlaznih slika jedna će uvijek predstavljati tekst (slika 3.4. a), jedna će predstavljati pozadinu (slika 3.4. b), jedna će predstavljati obrub teksta (3.4. d), a zadnja slika može biti ili još jedan prikaz pozadine ili još jedan obrub (3.4. c).

a) b)

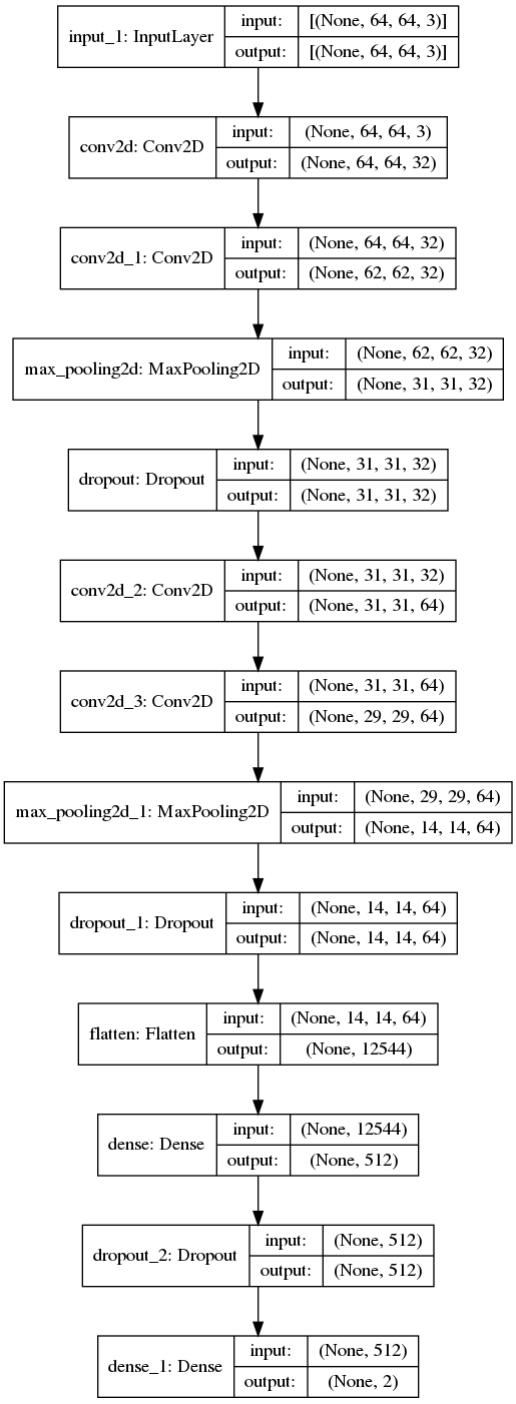
c) d)

Slika 3.4. Segmentacija slike teksta primjenom algoritma *k* srednjih vrijednosti

3.2.2. Izgradnja mreže za klasifikaciju slika dobivenih segmentacijom

Budući da rezultat algoritma *k* srednjih vrijednosti ovisi o početnoj inicijalizaciji centara i samom izgledu slike, ne može se unaprijed znati u kojoj se grupi nalaze elementi slike koji pripadaju tekstu. Stoga je određivanje na kojoj se slici nalazi tekst riješeno pomoću konvolucijske neuronske mreže koja provodi binarnu klasifikaciju. Pri tome slike koje sadrže tekst pripadaju klasi 1, dok sve ostale slike predstavljaju klasu 0. Na taj način je moguće izraditi mrežu koja za određenu sliku daje vjerojatnost da je na njoj tekst. Na taj način se onda može izvršiti identifikacija grupe u kojoj se nalazi tekst. Algoritam *k* srednjih vrijednosti primijenjen je na 812 različitih slika koje sadrže tekst (slično slici 3.3), kako bi se dobile segmentacije svake slike. Dobivene slike su zatim pregledane, svakoj je dodijeljena klasa kojoj pripada i raspoređene su po skupovima. Ukupno je označeno 3248 slika. Na 812 slika se nalazi tekst, a na 2436 slika ne. Skup je podijeljen na skup za učenje, skup za validaciju i skup za testiranje. Svaka slika skalirana je na veličinu 64x64 elemenata slike. Skup za učenje sadrži 2048 slika. Na 512 slika se nalazi tekst, a na 1536 slika ne. Validacijski skup sadrži 200 slika, 50 na kojima se nalazi tekst, odnosno 150 na kojima se ne nalazi. Testni skup sadrži 1000 slika, 250 slika je sa tekstom, a 750 bez teksta. Korištena je i online

augmentacija koja služi za kreiranje različitih varijacija postojećeg skupa podataka, što znači da se umjetno povećava broj slika korištenih za treniranje. Broj epoha pri treniranju je 40, što znači da je algoritam 40 puta iskoristio kompletni skup podataka za učenje. Veličina hrpe (engl. *batch size*) prilikom učenja mreže je 32, što znači da se na temelju 32 slike ažuriraju parametri mreže. Kompletna arhitektura predložene mreže vidljiva je na slici 3.5. Kod predložene mreže je prikazan u prilogu P.3.1.



Slika 3.5. Arhitektura konvolucijske neuronske mreže za klasifikaciju

Evaluacija mreže učinjena je pomoću matrice zabune. Opći oblik matrice zabune dan je tablicom 3.2. Elementi matrice zabune su:

- Istinito pozitivan (engl. *True positive, TP*) – slika koja je klasificirana kao tekst, a to i je.
- Lažno pozitivan (engl. *False positive, FP*) – slika koja je klasificirana kao tekst, a ona to nije.
- Istinito negativan (engl. *True negative, TN*) – slika koja nije klasificirana kao tekst, a ona to i nije.
- Lažno negativan (engl. *False negative, FN*) – slika koja nije klasificirana kao tekst, a ona to je.

Prag za klasifikaciju slika postavljen je na 0.5. Matrica zabune za trening skup podataka prikazana je u tablici 3.3. Točnost klasifikacije na trening skupu podataka iznosi 98.83%. Matrica zabune za testni skup podataka prikazana je u tablici 3.4. Točnost klasifikacije na testnom skupu podataka je 99%. Na slici 3.6. dan je primjer klasifikacije 32 slike. Iznad svake slike je prikazana vjerojatnost koja je dobivena pomoću izgrađene mreže. Vjerojatnost je označena crvenom bojom ako je mreža pogriješila klasu kojoj stvarno pripada dana slika.

Tab. 3.2. Opći oblik matrice zabune

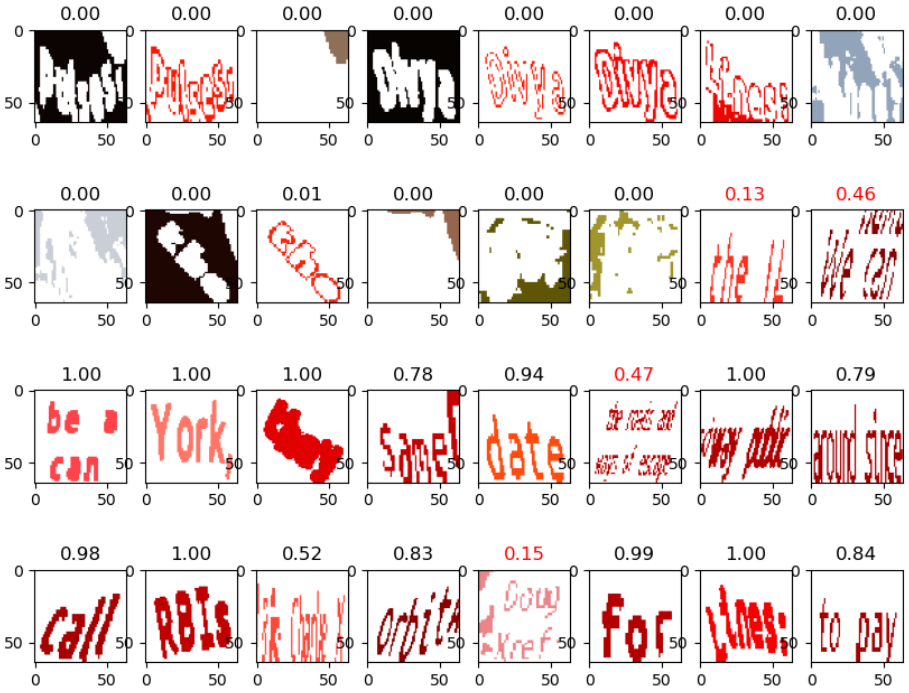
		Predviđeno stanje	
	Ukupno primjera	Ukupno pozitivnih (TP+FP)	Ukupno negativnih (FN + TN)
Stvarno stanje	Ukupno pozitivnih (TP+FN)	TP	FN
	Ukupno negativnih (FP+TN)	FP	TN

Tab. 3.3. Matrica zabune za trening skup podataka

		Predviđeno stanje	
	2048	502	1546
Stvarno stanje	512	495	17
	1536	7	1529

Tab. 3.4. Matrica zabune za testni skup podataka

		Predviđeno stanje	
	1000	248	752
Stvarno stanje	250	244	6
	750	4	746



Slika 3.6. Primjer klasifikacije 32 slike pomoću izgrađene konvolucijske neuronske mreže

Izgrađena neuronska mreža je implementirana u postojeću skriptu za generiranje novog skupa podataka u obliku funkcije koja učitava četiri slike koje predstavljaju segmentiranu sliku pravokutnog graničnog okvira u kojem se nalazi tekst. Funkcija vraća vjerojatnost da se na slici nalazi tekst. Funkcija je prikazana u prilogu P.3.2. Za sliku koja ima najveću vjerojatnost smatra se da sadrži elemente slike koji predstavljaju tekst te se slika predstavlja u obliku binarne slike tj. binarne maske. Binarna maska je matrica veličine širina puta visina slike, a elementi matrice su jednaki 1 ako je na tom mjestu tekst, a jednaki su 0 ako na tom mjestu nije tekst. Budući da je slika koja ulazi u mrežu izdvojeni dio ulazne slike na temelju pravokutnog graničnog okvira koji označava tekst, za dobivanje ispravnih koordinata na razini slike radi se pomak na temelju mjesta gdje je slika izdvojena. Nakon dobivene binarne maske za određeni tekst potrebno je dobiti i binarne maske za svaki znak. Korištena skripta sadrži informacije o koordinatama graničnog

okvira za svaki znak. Elementi binarne maske koji se nalaze unutar graničnog okvira određenog znaka pripadaju tom znaku.

3.2.3. Pohranjivanje oznaka teksta u COCO zapisu

Na temelju dobivene maske određuju se *polygon* ili RLE zapis koji se koristi za označavanje elemenata slike na kojima se nalazi tekst u COCO zapisu. RLE (*engl. run-length encoding*) je oblik kompresije podataka bez gubitka, to je lista koja sadrži broj elemenata slike maske koji se nalaze jedni za drugim, zatim slijedi broj elemenata slike koji nije maska pa zatim ponovno broj elemenata slike koji jesu maska itd. *Polygon* je oblik preciznog označavanja elemenata slike na način da računa niz x i y koordinata uzduž ruba prepoznatog objekta. Iako je RLE zapis nešto precizniji, u ovom radu će se koristiti *polygon* način zapisivanja oznaka iz razloga što odabrana mreža za izgradnju detektora teksta koristi taj zapis. *Polygon* zapis oznaka moguće je dobiti iz binarne maske na način da se pronađu konture na slici, a zatim se kontura zapiše u obliku *polygon* zapisa.

Oznake u COCO zapisu potrebno je zapisati u JSON datoteku. JSON je format datoteke i format za razmjenu podataka koji koristi ljudski čitljiv tekst. Sastoji se od parova atribut-vrijednost i nizova. JSON je datoteka koja u ovom slučaju sadrži određene oznake objekata na slici. JSON datoteke generirana u ovom radu je napravljena po uzoru na JSON datoteku koja je korištena u COCO skupu podataka. Sastoji se od pet listi objekata: *info*, *licences*, *categories*, *images* i *annotations*. Lista *info* sadrži osnovne informacije o skupu podataka poput imena skupa, verzije, godine i datuma kada je skup kreiran. Izgled *info* liste vidljiv je na slici 3.7.

```
"info": [
        {
            "description": "Moj Dataset",
            "version": "1.0",
            "year": 2022,
            "date_created": "2022/05/02"
        }
    ]
```

Slika 3.7. Izgled *info* liste u JSON datoteci

Lista *licences* se dodaje ako se koriste slike ili podaci za koje nemamo prava korištenja bez odgovarajuće dozvole. Ona se sastoji od imena dozvole, broja dozvola i poveznice s koje se mogu zatražiti prava za korištenje određenog skupa podataka. Lista *categories* sadrži 63 kategorije koje

predstavljaju svako slovo engleske abecede (veliko i malo) i brojeve. Svaka kategorija sadrži jedinstveni identifikacijski broj kategorije, ime kategorije te ime nadkategorije koja u ovom radu može biti samo tekst (jer jedini objekt koji se pronalazi na slikama u ovom radu je tekst). Izgled *categories* liste vidljiv je na slici 3.8.

```
"categories": [
    {
        "id": 1,
        "name": "text",
        "supercategory": "text"
    },
    {
        "id" : 2,
        "name" : "0",
        "supercategory" : "text"
    },
    {
        "id" : 3,
        "name" : "1",
        "supercategory" : "text"
    },
    ...
]
```

Slika 3.8. Izgled *categories* liste u JSON datoteci

Lista *images* sadrži informacije o svakoj slici koja se generirala. Sastoji se od jedinstvenog identifikacijskog broja (koji se dobije pomoću brojača, kada se generira slika brojač se poveća za jedan), od jedinstvenog identifikacijskog broja licence (ako je ona potrebna), visine i širine slike te imena slike koji je također povezan s brojačem kako bi ime slike i identifikacijski broj bili implicitno povezani. Prikaz koda koji je dodan u skriptu za generiranje *images* liste JSON datoteci vidljiv je na slici 3.9., dok je izgled *images* liste vidljiv na slici 3.10.

```
        h,w,c=img.shape

        y = {"id": br,
        "licence_id": 1,
        "width": w,
        "height": h,
        "file_name": f"img{br}.png"
        }

        with open('sample_new.json', 'a') as f:
            f.write(json.dumps(y)+",\n")
```

Slika 3.9. Prikaz koda za generiranje *images* liste JSON datoteci

```
"images": [
    {
        "id": 1,
        "licence_id": 1,
        "width": 600,
        "height": 450,
        "file_name": "img1.png"
    },
    {
        "id": 2,
        "licence_id": 1,
        "width": 600,
        "height": 450,
        "file_name": "img2.png"
    },
    ...
]
```

Slika 3.10. Izgled liste *images* u JSON datoteci

Lista *annotations* sadrži informacije o svakoj riječi i o svakom znaku koji se generiraju i dodaju na sliku. Iz tog razloga u skripti koja se koristi u ovom diplomskom radu na dva mjesta se nalazi kod za generiranje JSON datoteke. Prvo se generiraju oznake za svaku riječ. Te oznake sadržavaju jedinstveni identifikacijski broj koji se dodjeljuje pomoću brojača koji se povećava svaki put kad se generira jedna riječ na slici. Drugi objekt u listi je *iscrowd* koji može biti 0 ili 1, 0 se koristi kada se koristi *polygon* zapis, 1 kada se koristi ranije opisani RLE zapis. Kao što je ranije navedeno ovom radu koristi se *polygon* zapis. Objekt *category_id* je uvijek 1 kod oznake za riječ, jer ta kategorija predstavlja tekst. Sljedeći objekt je *area* koji predstavlja površinu označenog područja

na slici, a računa se pomoću predefiniranih funkcija odnosno u slučaju nepreciznih oznaka računa se kao visina puta širina pravokutnog graničnog okvira. Objekt *segmentation* se popunjava *polygon* zapisom elemenata slike na kojima se nalazi tekst. U slučaju generiranja JSON datoteke sa nepreciznim oznakama pod objekt *segmentation* dodaju se koordinate vrhova pravokutnog graničnog okvira određenog teksta na slici. Objekt *image_id* sadrži jedinstveni identifikacijski broj od slike na kojoj se tekst nalazi. Posljednji objekt je *bbox* on sadrži koordinate gornjeg lijevog kuta od pravokutnog graničnog okvira i njegovu širinu i visinu. Kod za generiranje *annotations* liste je vidljiv na slici 3.11.

```
y2 = {
    "id": brojcanik,
    "iscrowd": 0,
    "area": arealist[0],
    "category_id": 1,
    "segmentation": [polygon_list],
    "image_id": br,
    "bbox": [xk, yk, sir, vis]
    }
with open('sample_new2.json', 'a') as f:
    f.write(json.dumps(y2)+",\n")
```

Slika 3.11 Prikaz koda za generiranje *annotations* liste i njeno zapisivanje u JSON datoteku

Nakon generiranja oznaka za svaki tekst, generira se oznaka za svaki znak. Objekti liste su identični kao i za generiranje riječi samo se dobivaju na drugačiji način. Objekt *id* nastavlja se na brojač koji se koristi za tekst i povećava se za svaki znak. Objekt *iscrowd* je uvijek 0 kao i kod teksta. Objekt *category_id* mora sadržavati jedinstveni identifikacijski broj od kategorije ovisno o znaku koji predstavlja. U skriptu je dodana lista svih znakova istim redoslijedom kojim su definirani u kategorijama. Napravljena je funkcija *search* koja kao argumente prima listu znakova i znak koji se traži, te vraća broj na kojem je mjestu pronađen znak (taj broj predstavlja identifikacijski broj kategorije). Prikaz te funkcije je vidljiv na slici 3.12.

```
kategorije = ['\\','text','0','1','2','3','4','5','6',
              '7','8','9','A','B','C','D','E','F','G',
              'H','I','J','K','L','M','N','O','P','Q',
              'R','S','T','U','V','W','X','Y','Z','a',
              'b','c','d','e','f','g','h','i','j','k',
              'l','m','n','o','p','q','r','s','t','u',
              'v','w','x','y','z']

def search(list, platform):
    for i in range(len(list)):
        if list[i] == platform:
            return i, True
    return 0,False

cid,check = search(kategorije,listofChar[nmb])
```

Slika 3.12. Prikaz funkcije za pronalazak identifikacijskog broja kategorije

Objekt *segmentation* koristi iste oznake kao i kod teksta, ali se koristi binarna maska za određeni znak. U slučaju oznaka nepreciznog skupa u *segmentation* objekt dodaju se koordinate vrhova graničnog okvira određenog znaka. U skripti koja se koristi u ovom radu granični okvir za znakove nije pravokutan, a u JSON datoteci granični okvir mora biti zapisan kao pravokutnik. Stoga su napravljene modifikacije da se od četiri koordinate odabere najmanja x koordinata i najmanja y koordinata te koordinate predstavljaju gornji lijevi vrh pravokutnog graničnog okvira. Širina je izračunata kao razlika između najveće i najmanje x koordinate, a visina je razlika između najveće i najmanje y koordinate. Prikaz koda za generiranje *annotations* liste vidljiv je na slici 3.13.

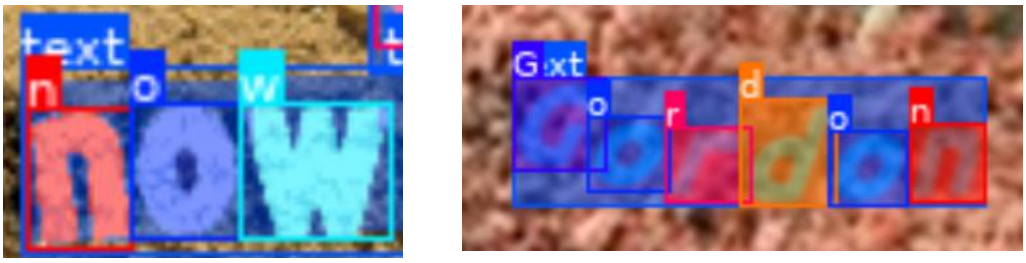
```
y1 = {
    "id": self.broj,
    "iscrowd": 0,
    "area": arealist[0],
    "category_id": cid,
    "segmentation": [polygon_list],
    "image_id": br,
    "bbox": [xmin,ymin,xmax-xmin,ymax-ymin]
    }
with open('sample_new2.json', 'a') as f:
    f.write(json.dumps(y1)+",\n")
```

Slika 3.13. Prikaz koda za generiranje *annotations* liste i njeno zapisivanje u JSON datoteku

Nakon modificiranja skripte, naredbom *python3 gen.py* u terminalu, pokreće se skripta koja generira slike s tekstom i popratnu JSON datoteku. Skripta koristi bazu od 8000 slika, što znači da se generira nešto više od 7000 slika s tekstom jer na određene slike se ne doda tekst zbog malo prigodnog mjesta za dodavanje teksta. Iste slike se mogu koristiti više puta te će se na njima lijepiti različit tekst. Prikaz dijela generirane slika sa preciznim oznakama vidljiv je na slici 3.14. a, a slika sa nepreciznim oznakama vidljiva je na slici 3.14. b Generirano je ukupno 11463 slike koje su raspoređene u osam skupova podataka kako je vidljivo u tablici 3.5. Skupovi imena Precizni sadržavaju precizne oznake segmentacijske maske, a skupovi imena Neprecizni sadržavaju neprecizne oznake maske. Validacijski skup sadrži obje vrste oznaka, a koristimo oznake ovisno koji trening skup podataka koristimo.

Tab. 3.5. Generirani skupovi podataka

Naziv skupa podataka	Broj slika u skupu podataka
Precizni1000	1029
Precizni3000	3265
Precizni5000	5099
Neprecizni1000	1029
Neprecizni2000	3265
Neprecizni3000	5099
Validacijski skup	1038
Testni skup	1032



a) b)

Slika 3.14. Prikaz dijela generirane slike sa preciznim oznakama

3.3. Proces treniranja duboke neuronske mreže za detekciju teksta

Za sam proces treniranja novih modela koristi se već postojeća *TextFuseNet* mreža, koja je temeljena uglavnom na arhitekturi *Mask R-CNN-a* i *TextSpottera.* Ova mreža se koristi u radu jer je trenutno jedina mreža koja može koristiti preciznije oznake teksta i pokazuje najbolje rezultate u području detekcija teksta. Za uspješno treniranje mreže prvo je potrebno stvoriti prigodno programsko okruženje. Za stvaranje programskog okruženja korištena je *Anaconda. Anaconda* je *python* platforma čiji je cilj pojednostaviti upravljanje paketima i njihovim implementacijama. Kako bi se kreiralo okruženje otvara se terminal na mjestu gdje se nalazi datoteka *textfusnet_requirements.yml* koja je prikazana u prilogu P.3.3., u njoj se nalaze paketi potrebni za instalaciju. Zatim se u terminal unosi naredba:

conda env create -f textfusenet_requirements.yml

Zatim je potrebno stvoreno programsko okruženje aktivirati naredbom:

conda activate textfusenet

Zatim je potrebno klonirati git repozitorij naredbom:

git clone https://github.com/ying09/TextFuseNet.git

Nakon toga je potrebno ući u klonirani direktorij naredbom:

cd TextFusenet/

Zatim je potrebna instalacija dodatnih paketa naredbama:

pip install fvcore-master.zip

pip setup.py build develop

Ako se koristi nova verzija CUDA platforme potrebno je zamijeniti sve AT_CHECK(…) pozive s TORCH_CHECK(…) pozivima u datotekama: *TextFuseNet/detectron2/layers/csrc/deformable/deform_conv.h* i *TextFuseNet/detectron2/layers/csrc/deformable/deform_conv_cuda.cu.* Zatim je potrebno registrirati prethodno generirane skupove podataka u datoteku *TextFuseNet/detectron2/data/datasets/builtin.py*. U direktorij *TextFuseNet/configs/ocr/* je moguće definirati konfiguracijsku datoteku za treniranje, jedna takva datoteka je prikazana u prilogu P.3.4.

Navedenu datoteku treba konfigurirati ovisno o tome koji će se skup podataka koristiti. Prikaz kako izgleda registracija skupova vidljiv je na slici 3.15.

```
# ocr datasets register
    "totaltext":("totaltext/train_images", "totaltext/train.json"),
    "ctw1500":("ctw1500/train_images", "ctw1500/train.json"),
    "icdar2013":("icdar2013/train_images", "icdar2013/train.json"),
    "icdar2015":("icdar2015/train_images", "icdar2015/train.json"),
    "synthtext_precise1000":("/home/idorkic/backup/1000/images","/home/idorkic/ba
ckup/1000/sampl4.json"),
    "synthtext_imprecise1000":("/home/idorkic/backup/1000/images","/home/idorkic/
backup/1000/sampl4ncz.json"),
    "synthtext_precise3000":("/home/idorkic/backup/3000/images","/home/idorkic/ba
ckup/3000/sampl4.json"),
    "synthtext_imprecise3000":("/home/idorkic/backup/3000/images","/home/idorkic/
backup/3000/sampl4ncz.json"),
    "synthtext_precise5000":("/home/idorkic/backup/5000/images","/home/idorkic/ba
ckup/5000/sampl4.json"),
    "synthtext_imprecise5000":("/home/idorkic/backup/5000/images","/home/idorkic/
backup/5000/sampl4ncz.json"),
    "synthtext_valid_precise":("/home/idorkic/backup/Valid/testni","/home/idorkic
/backup/Valid/testniprec.json"),
    "synthtext_valid_imprecise":("/home/idorkic/backup/Valid/testni","/home/idork
ic/backup/Valid/testninpc.json"),
    "synthtext_test_precise":("/home/idorkic/backup/Test/images","/home/idorkic/b
ackup/Test/sampl4.json"),
    "synthtext_test_imprecise":("/home/idorkic/backup/Test/images","/home/idorkic
/backup/Test/sampl4ncz.json")
```

Slika 3.15. Prikaz registriranih skupova podataka

Trening se pokreće sljedećom naredbom:

python tools/train_net.py --num-gpus 1 --config-file
configs/ocr/synthtext_precise_101_FPN.yaml

Odlučeno je da će treniranje mreže sadržavati 60 epoha, te da će se nakon svake epohe spremati model. Sam proces treniranja kod skupa podataka koji sadrži 1029 slika trajao je 4 sata i 54 minute, dok je treniranje kod skupa podataka sa 3265 slika trajao 15 sati i 29 minuta, a kod skupa podataka sa 5099 podataka proces treninga trajao je 24 sata i 13 minuta.

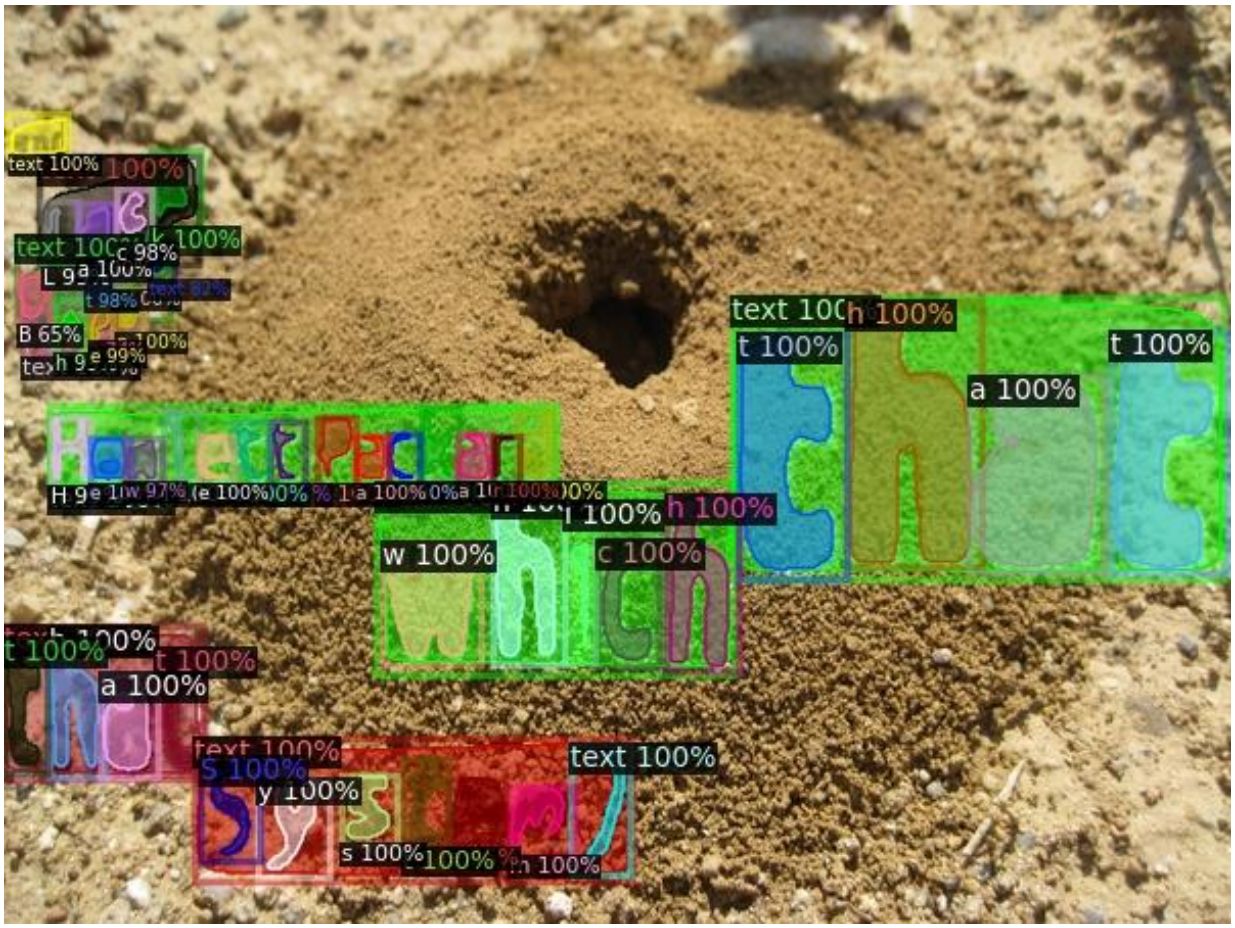
3.4. Upute za pokretanje testiranja

Testiranje dobivenih modela provedeno je na dva načina: vizualno i matematički pomoću odgovarajućih metrika. Vizualni način sadržava vizualizaciju detektiranog teksta i detektiranih znakova na slikama i sigurnost mreže u svoju odluku. Matematički način sadržava razne varijacije

metrika prosječne preciznosti (engl. *average precision*). Kako bi pokrenuli vizualno testiranje potrebno je datoteci *TextFuseNet/demo/icdar2015_detection.py* postaviti putanju modela za testiranje na željeni model koji je odabran. Zatim u direktorij *TextFuseNet/input_images/* postavimo testne slike. Zatim otvorimo terminalu u direktoriju *TextFuseNet* te pokrenemo testiranje sa naredbom:

python demo/icdar2015_detection.py

Testirane slike te popratna tekstualna datoteka, u koju su zapisana mjesta na slikama gdje se nalazi tekst, spremaju se u direktorij *TextFuseNet/test_icdar2015*. Na slici 3.16. vizualno je prikazan izlaz mreže za jednu testnu sliku.



Slika 3.16. Prikaz izlaza mreže za jednu testnu sliku

Za pokretanje računskog testiranja koriste se već postojeće funkcije za testiranje. Potrebno je u konfiguracijskoj datoteci *TextFusNet/configs/ocr/synthtext_precise_101_FPN.yaml* promijeniti putanju do modela koji se testira i do skupa podataka koji se testira. Zatim u datoteci *TextFuseNet/tools/train_net.py* u funkciji *main()* potrebno je dodati željeni model iz

konfiguracijske datoteke te pozvati funkciju *test()*. Prikaz dodanog koda da bi se modeli mogli testirati vidljiv je na slici 3.17., a primjer ispisa rezultata vidljiv je na slici 3.18.

```
def main(args):
    model = Trainer.build_model(cfg)
    DetectionCheckpointer(model,
     save_dir=cfg.OUTPUT_DIR).resume_or_load(cfg.MODEL.WEIGHTS,
     resume=args.resume)
    rez = Trainer.test(cfg, model)
```

Slika 3.17. Prikaz dodanog koda za testiranje modela

```
[10/27 15:11:04 d2.engine.defaults]: Evaluation results for synthtext_test_precise in csv format:
[10/27 15:11:04 d2.evaluation.testing]: copypaste: Task: bbox
[10/27 15:11:04 d2.evaluation.testing]: copypaste: AP,AP50,AP75,APs,APm,APl
[10/27 15:11:04 d2.evaluation.testing]: copypaste: 56.8244,70.2710,67.5213,55.8231,84.3771,89.3404
[10/27 15:11:04 d2.evaluation.testing]: copypaste: Task: segm
[10/27 15:11:04 d2.evaluation.testing]: copypaste: AP,AP50,AP75,APs,APm,APl
[10/27 15:11:04 d2.evaluation.testing]: copypaste: 24.3826,49.1280,21.7627,23.0502,63.1902,71.4293
```

Slika 3.18. Primjer rezultata matematičkog testiranja dobivenog modela

4. EVALUACIJA PREDLOŽENOG RJEŠENJA ZA PRECIZNU DETEKCIJU TEKSTA

U ovom poglavlju predstavljeni su rezultati detektora teksta izgrađenih na generiranim skupovima podataka. Za testiranje je korišteno računalo sa Linux operacijskim sustavom (Ubuntu 20.04. LTS), grafičkom karticom NVIDIA GeForce RTX 3080Ti i procesorom Intel Core i9-11900F uz 32 GB RAM. Prikazani su rezultati tri modela mreže s najvećom preciznošću detekcije teksta trenirane na određenom skupu. Radi lakšeg razumijevanja predstavljenih rezultata modeli će biti imenovani poput skupa na kojem su trenirani uz dodatak broja kako bi imali različita imena (Precizni1000(1), Precizni1000(2), itd.). Svi izgrađeni modeli testirani su na istom testnom skupu podataka. Korištena mreža radi detekciju teksta. Sve metrike dobivene u testiranju su vezane uz prosječnu preciznost (engl. *Average Precision, AP*). Kako bi se moglo razumjeti što je prosječna preciznost, moraju se definirati pojmovi preciznost (engl. *precision*) i odziv (engl. *recall*). Preciznost je omjer između istinito pozitivnih primjera koji su detektirani i svih pozitivnih primjera:

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

Odziv je omjer detektiranih istinito pozitivnih primjera i zbroja detektiranih istinito pozitivnih i detektiranih lažno negativnih primjera:

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

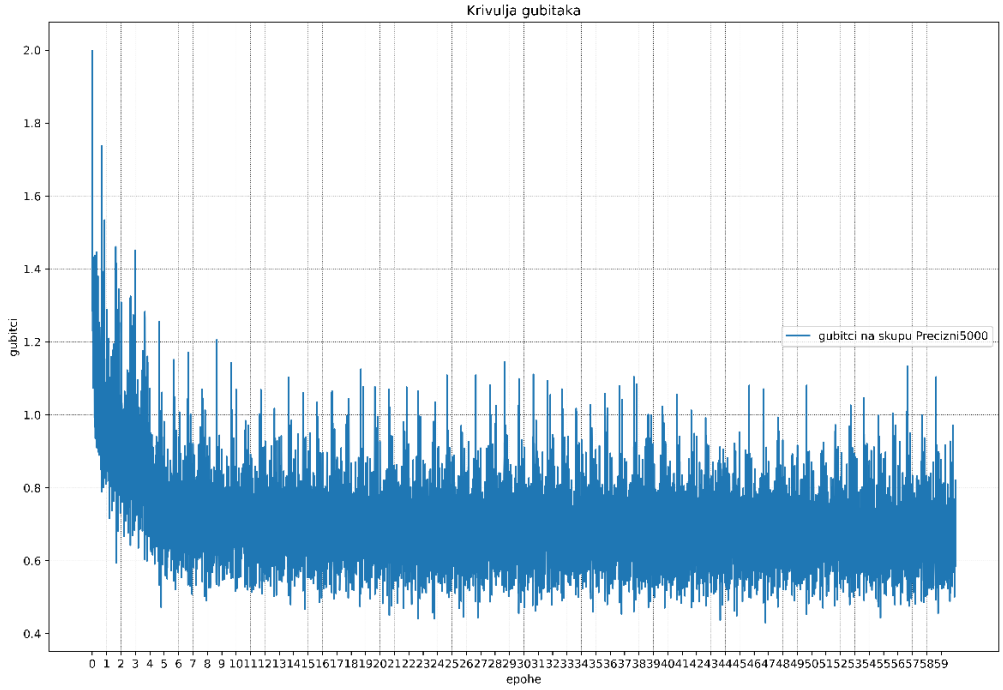
Prosječna preciznost se računa za svaku klasu pojedinačno, a definira se kao područje ispod preciznost-odziv krivulje, gdje x os predstavlja odziv, a y os predstavlja preciznost. Nakon dobivanja prosječne preciznosti za svaku klasu računa se ukupna prosječna preciznost, koja se računa kao zbroj svih prosječnih preciznosti podijeljenih sa brojem klasa. Metrike koje se koriste prilikom evaluacije rješenja su sljedeće:

- AP50 – ako je presjek preko unije (engl. *Intersect over Union, IoU)* detekcije nekog objekta veća od 50% uzima se kao pozitivan rezultat, zatim se računa prosječna preciznost
- AP75 – ako je *IoU* detekcije nekog objekta veća od 75% uzima se kao pozitivan rezultat, zatim se računa prosječna preciznost

- APs – je prosječna preciznost pri detekciji malih objekata, malim objektima se smatraju objekti površinom manji od 32^2 elemenata slike
- APm – je prosječna preciznost pri detekciji objekata srednje veličine, objektima srednje veličine se smatraju objekti površine između 32^2 i 96^2 elemenata slike
- APl – je prosječna preciznost pri detekciji velikih objekata, velikim objektima se smatraju objekti površine veće od 96^2 elemenata slike

4.1. Postignuti rezultati na precizno označenim skupovima

Odlučeno je da će svako treniranje sadržavati 60 epoha, stoga je kod svakog skupa broj iteracija prilagođen da bude konstantan broj epoha. Kod skupa koji se sastoji od 1029 slika, određeno je da proces treniranja mreže ima 61470 iteracija. Proces treniranja mreže na skupu od 3265 slika ima 195900 iteracija. Proces treniranja mreže na skupu od 5099 slika ima 305940 iteracija. Modeli se spremaju nakon svake epohe. Tri najbolja modela su izabrana na temelju krivulje gubitka na određenom skupu. Primjer krivulje gubitka na skupu Precizni5000 prikazana je na slici 4.1. Za testiranje su izabrani modeli koji su tokom treniranja imali najmanje gubitke. Rezultati testiranih modela na testnom skupu su vidljivi u tablici 4.1. Modeli u skupu od 1029 slika koji su se pokazali kao najbolji su modeli nakon 30 epoha, nakon 57 epoha i nakon 58 epoha. Modeli u skupu od 3265 slika koji su se pokazali kao najbolji su nakon 13 epoha, nakon 14 epoha i nakon 60 epoha. Modeli u skupu od 5099 slika koji su se pokazali kao najbolji su nakon 11 epoha, nakon 45 epoha i nakon 59 epoha. Iz rezultata je vidljivo da povećanjem broja slika u skupovima preciznost detekcije teksta raste. Vidljivo je da mreža poprilično dobro prepoznaje velike i srednje objekte na slici, ali očito mreža ima problema s detekcijom malih objekata jer su oni teže vidljiviji na slici i lakše se uklapaju u pozadinu.



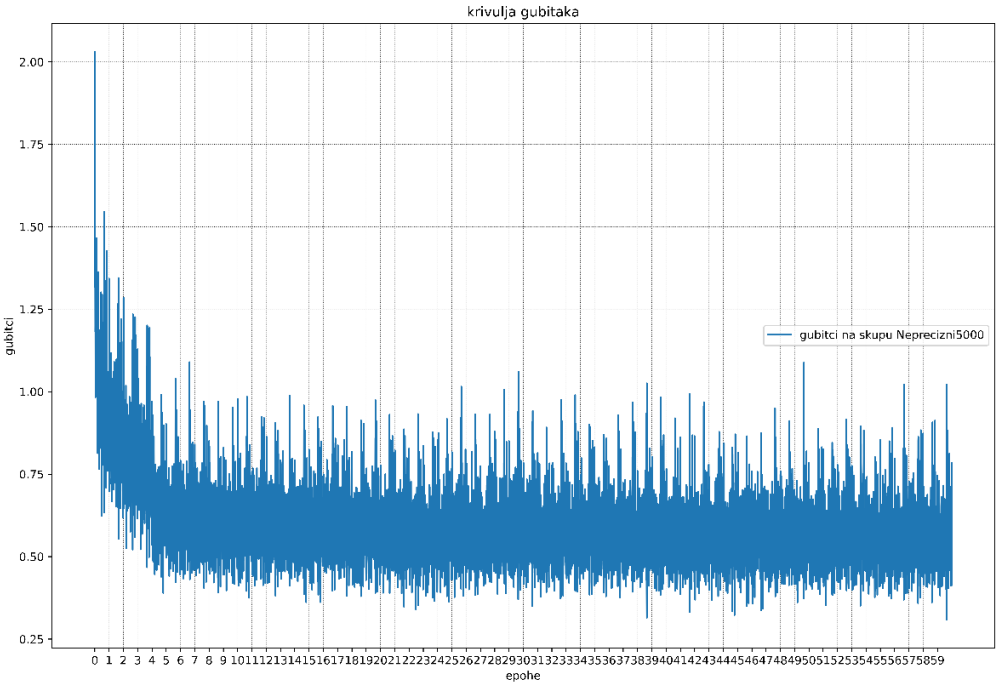
Slika 4.1. Krivulja gubitka na skupu Precizni5000

Tab. 4.1. Rezultati testiranja izgrađenih detektora teksta na testnom skupu

Metrike - detekcija objekata						
Model	AP	AP50	AP75	APs	APm	APl
Precizni1000(1)	54.873	68.825	65.645	53.872	81.379	87.887
Precizni1000(2)	54.757	68.732	65.540	53.762	81.538	87.712
Precizni1000(3)	54.424	68.658	65.539	53.483	80.747	87.376
Precizni3000(1)	56.349	69.816	67.065	55.308	84.644	89.324
Precizni3000(2)	56.169	69.555	66.877	55.121	84.686	89.239
Precizni3000(3)	56.107	69.829	67.027	55.067	84.171	88.947
Precizni5000(1)	57.088	70.499	67.825	56.068	84.859	89.298
Precizni5000(2)	56.824	70.271	67.521	55.823	84.377	89.34
Precizni5000(3)	56.423	70.269	67.422	55.396	83.551	88.773

4.2. Postignuti rezultati na neprecizno označenim skupovima

Kod skupa podataka treniranih sa nepreciznim oznakama broj iteracija i spremanje modela ostaju isti s obzirom da se radi o istom broju slika u određenim skupovima. Princip biranja najboljih modela također je isti. Prikaz krivulje gubitaka na skupu Neprecizni5000 vidljiv je na slici 4.2. Rezultati testirani na testnom skupu su vidljivi u tablici 4.2. Modeli u skupu od 1029 slika koji su se pokazali kao najbolji su modeli nakon 30 epoha, nakon 37 epoha i nakon 36 epoha. Modeli u skupu od 3265 slika koji su se pokazali kao najbolji su nakon 58 epoha, nakon 60 epoha i nakon 54 epoha. Modeli u skupu od 5099 slika koji su se pokazali kao najbolji su nakon 59 epoha, nakon 60 epoha i nakon 57 epoha.



Slika 4.2. Krivulja gubitaka na skupu Neprecizni5000

Tab. 4.2. Rezultati testiranja izgrađenih detektora teksta na testnom skupu

Metrike – detekcija objekata						
Model	AP	AP50	AP75	APs	APm	APl
Neprecizni1000(1)	54.525	68.786	65.461	49.997	80.598	88.595
Neprecizni1000(2)	54.604	68.669	65.601	50.136	80.387	90.073
Neprecizni1000(3)	54.796	68.914	65.868	50.308	80.621	91.521
Neprecizni3000(1)	56.055	69.446	66.783	51.502	82.368	92.478
Neprecizni3000(2)	56.140	69.523	66.857	51.552	82.398	92.595
Neprecizni3000(3)	56.161	69.558	66.809	51.587	82.373	92.497
Neprecizni5000(1)	56.921	70.314	67.818	52.361	83.287	92.433
Neprecizni5000(2)	56.968	70.401	67.791	52.410	83.336	92.430
Neprecizni5000(3)	57.074	70.596	68.010	52.534	83.125	92.330

4.3. Analiza rezultata

Sam proces treniranja mreže je poprilično dug i računalno zahtjevan. Za skup od 1000 slika proces treninga traje nešto manje od 5 sati, a zauzima 100% GPU memorije. Povećanjem broja slika linearno raste i vrijeme izvođenja treninga. Analizom svih rezultata utvrđeno je da povećanjem broja slika dolazi do nelinearnog rasta prosječne preciznosti detekcije teksta. Ne vidi se zamjetna razlika u prosječnoj preciznosti između modela treniranih na preciznim skupovima i modela treniranih na nepreciznim skupovima. Skup sa preciznim oznakama preciznije detektira male objekte, vidljiva je razlika od otprilike 4%. AP75 je nešto manji od AP50 jer zahtijeva veći *IoU*. Na primjeru slike 4.3.a se vidi teško prepoznatljiv mali objekt. Na slici 4.3.b se vidi uspješna detekcija teksta sa modelom treniranim na nepreciznim oznakom. Na slici 4.3.c se vidi uspješna detekcija teksta te uspješna detekcija slova 'd' te njihova relativno precizna segmentacija sa modelom treniranim na preciznim oznakama. Primjer cijele scene s detekcijom teksta vidljiv je na slici 4.4.

a b c

Slika 4.3. Primjer detekcije pojedine instance teksta

Slika 4.4. Primjer detekcije teksta

5. ZAKLJUČAK

U ovom diplomskom radu ispitana je efikasnost detektora teksta temeljenog na dubokim neuronskim mrežama na način da se poveća preciznost označavanja slika na sintetički generiranom skupu slika u kojem su riječi smještene u slike prirodnih scena. Kreirani su vlastiti skupovi slika s preciznim oznakama i s nepreciznim oznakama kako bi se mogla vidjeti poboljšanja prilikom treniranja modela na skupu s preciznim oznakama. Prilikom generiranja umjetnih podataka s precizno označenom segmentacijskom maskom pojavio se problem s izdvajanjem teksta od pozadine stoga je izgrađena konvolucijska neuronska mreža za klasifikaciju koja odvaja tekst od pozadine. Prosječna preciznost detektora teksta je ispitana na šest različitih skupova podataka, koji su se razlikovali po broju slika i oznakama teksta. Najveća prosječna preciznost detekcije teksta na testnom skupu podataka iznosi 57.088%. Mreža izrazito dobro prepoznaje velike objekte, gdje u 92.433% dobro detektira tekst. Mreža najteže detektira male objekte, sa prosječnom preciznošću pod 56.068%. Mali objekti se najčešće spoje sa šarenom pozadinom te to stvara problem pri samom odvajanju teksta od pozadine, stoga može doći i do problema pri označavanju takve vrste teksta. No kod malih objekata se najviše vidi rast dodavanjem većeg broja slika u skupove podataka i na tom skupu se može vidjeti prednost preciznog označavanja skupova podataka za treniranje. Može se zaključiti da precizne oznake teksta nisu pretjerano utjecale na precizniju detekciju teksta no ipak su dale vidljivo poboljšanje rezultata pri prepoznavanju malih objekata. Preciznost detekcije teksta bi se mogla poboljšati povećanjem broja slika u skupovima, dodatnim poboljšanjem preciznosti oznaka slika u skupovima podataka, korištenjem jače grafičke kartice mogla bi se povećati veličina ulaznih slika pri treniranju, a i ubrzalo bi sam proces treniranja.

LITERATURA

[1] D. Karatzas *et al.*, "ICDAR 2013 Robust Reading Competition," in *2013 12th International Conference on Document Analysis and Recognition*, Washington, DC, USA, Aug. 2013, pp. 1484–1493. doi: 10.1109/ICDAR.2013.221.

[2] A. Gupta, A. Vedaldi, and A. Zisserman, "Synthetic Data for Text Localisation in Natural Images," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, Jun. 2016, pp. 2315–2324. doi: 10.1109/CVPR.2016.254.

[3] A. Veit, T. Matera, L. Neumann, J. Matas, and S. Belongie, "COCO-Text: Dataset and Benchmark for Text Detection and Recognition in Natural Images." arXiv, Jun. 19, 2016. Accessed: Aug. 23, 2022. [Online]. Available: http://arxiv.org/abs/1601.07140

[4] N. Nayef *et al.*, "ICDAR2019 Robust Reading Challenge on Multi-lingual Scene Text Detection and Recognition -- RRC-MLT-2019." arXiv, Jul. 01, 2019. Accessed: Nov. 10, 2022. [Online]. Available: http://arxiv.org/abs/1907.00945

[5] D. Karatzas *et al.*, "ICDAR 2015 competition on Robust Reading," in *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*, Aug. 2015, pp. 1156–1160. doi: 10.1109/ICDAR.2015.7333942.

[6] 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.

[7] J. Ye, Z. Chen, J. Liu, and B. Du, "TextFuseNet: Scene Text Detection with Richer Fused Features," in *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, Yokohama, Japan, Jul. 2020, pp. 516–522. doi: 10.24963/ijcai.2020/72.

[8] Q. Ye, Q. Huang, W. Gao, and D. Zhao, "Fast and robust text detection in images and video frames," *Image Vis. Comput.*, vol. 23, no. 6, pp. 565–576, Jun. 2005, doi: 10.1016/j.imavis.2005.01.004.

[9] S. Bonechi, P. Andreini, M. Bianchini, and F. Scarselli, "COCO_TS Dataset: Pixel-level Annotations Based on Weak Supervision for Scene Text Segmentation," vol. 11730, 2019, pp. 313–325. doi: 10.1007/978-3-030-30490-4_26.

[10] A. Gupta, "SynthText." Aug. 23, 2022. Accessed: Aug. 23, 2022. [Online]. Available: https://github.com/ankush-me/SynthText

[11] "About the Ubuntu project," *Ubuntu*. https://ubuntu.com/about (accessed Aug. 23, 2022).

[12] "Python PIP." https://www.w3schools.com/python/python_pip.asp (accessed Aug. 23, 2022).

[13] "What is K-Means Clustering? - Definition from Techopedia," *Techopedia.com*. http://www.techopedia.com/definition/32057/k-means-clustering (accessed Aug. 23, 2022).

SAŽETAK

Cilj ovog diplomskog rada je pokazati hoće li precizno označavanje skupova za učenje poboljšati preciznost detekcije teksta. Opisani su problemi koji nastaju pri označavanju skupova podataka. Dan je pregled postojećih podatkovnih skupova za detekciju teksta i rješenja za detekciju teksta. Opisan je proces izrade vlastitih skupova podataka. Opisan je proces treniranja duboke neuronske mreže te su dane upute za pokretanje testiranja. Evaluirani su i analizirani dobiveni rezultati i rješenje za preciznu detekciju teksta.

Ključne riječi: precizno označavanje skupova podataka, detekcija teksta, duboka neuronska mreža

IMPROVING TEXT DETECTION ON IMAGES BY PRECISELY LABELING LEARNING DATASETS

ABSTRACT

The main goal of this thesis is to show whether precise labeling of learning datasets will improve the accuracy of text detection. The problems that arise when labeling data sets are described. A review of existing data sets for text detection and solutions for precise text detection is given. The process of creating your own data sets is described. The process of training a deep neural network is described and instructions for testing are given. The obtained results and the solution for precise text detection were evaluated and analyzed

Keywords: precise labeling of datasets, text detection, deep neural network

ŽIVOTOPIS

Ivan Dorkić rođen je 11. studenog 1996. godine u Osijeku. 2003. godine započinje obrazovanje u OŠ „Matija Gubec“ Magadenovac te nakon toga upisuje III. gimnaziju u Osijeku. 2015. upisuje Fakultet elektrotehnike, računarstva i informacijskih tehnologija u Osijek, preddiplomski smjer Računarstvo. 2020. godine stječe akademski naziv sveučilišni prvostupnik (lat. *baccalaureus*) inženjer računarstva. Iste godine upisuje diplomski sveučilišni studij automobilsko računarstvo i komunikacije.

PRILOZI

- P.3.1. Kod za konvolucijsku neuronsku mrežu za klasifikaciju

```
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from matplotlib import pyplot as plt
import numpy as np
import cv2
from keras.utils.vis_utils import plot_model

img_size = (64,64)
no_epochs = 40
batch_size = 32
train_datagen = ImageDataGenerator(
    rotation_range=10,
    width_shift_range=0.1,
    height_shift_range=0.1,
    rescale=1./255,
    shear_range=0.2,
    zoom_range=0.1,
    horizontal_flip=True,
    vertical_flip=False,
    fill_mode='nearest')
test_datagen = ImageDataGenerator(rescale=1./255)
train_generator = train_datagen.flow_from_directory(
    directory="train",
    target_size=img_size,
    color_mode="rgb",
    batch_size=batch_size,
    class_mode="categorical",
    shuffle=True,
    seed=42)
validation_generator = test_datagen.flow_from_directory(
        directory="val",
        target_size=img_size,
        batch_size=batch_size,
        class_mode='categorical',
        shuffle = False)
test_generator = test_datagen.flow_from_directory(
        directory="test",
        target_size=img_size,
        batch_size=1,
        class_mode='categorical',
        shuffle=False)
```

```
batch = train_generator.next()
plt.subplots(figsize=(10, 8))
for i in range(1,batch_size+1):
  plt.subplot(4,8,i)
  image = (batch[0][i-1]*255).astype('uint8')
plt.show()
inputs = keras.Input(shape=(img_size[0],img_size[1],3))
x = layers.Conv2D(32, kernel_size=(3,3), padding='same',
activation="relu")(inputs)
x = layers.Conv2D(32, kernel_size=(3,3), padding='valid', activation="relu")(x)
x = layers.MaxPool2D(pool_size=(2,2), strides=(2,2))(x)
x = layers.Dropout(rate = 0.2)(x)
x = layers.Conv2D(64, kernel_size=(3,3), padding='same', activation="relu")(x)
x = layers.Conv2D(64, kernel_size=(3,3), padding='valid', activation="relu")(x)
x = layers.MaxPool2D(pool_size=(2,2), strides=(2,2))(x)
x = layers.Dropout(rate = 0.2)(x)
x = layers.Flatten()(x)
x = layers.Dense(512, activation="relu")(x)
x = layers.Dropout(rate = 0.5)(x)
outputs = layers.Dense(2, activation="softmax")(x)
model = keras.Model(inputs=inputs, outputs=outputs, name="model")
model.summary()
model.compile(loss="categorical_crossentropy",
                        optimizer="adam",
                        metrics=["accuracy",])
my_callbacks = [
    keras.callbacks.EarlyStopping(monitor="val_loss", patience=7),
    keras.callbacks.ModelCheckpoint(filepath='checkpoints/model.{epoch:02d}-
{val_loss:.2f}.h5',
                                    save_best_only=True,
                                    monitor = "val_accuracy",
                                    mode="max"),
    keras.callbacks.TensorBoard(log_dir='logs',
                                update_freq=100),
    keras.callbacks.ReduceLROnPlateau(monitor="val_loss",
                                    factor = 0.1,
                                    patience = 5,
                                    cooldown = 1)
]

plot_model(model, to_file='model_plot.png', show_shapes=True,
show_layer_names=True)
```

```
history = model.fit(
    train_generator,
    steps_per_epoch = train_generator.samples // batch_size,
    validation_data = validation_generator,
    validation_steps = validation_generator.samples // batch_size,
    callbacks = my_callbacks,
    epochs=no_epochs)

test_generator.reset()
score = model.evaluate(test_generator, batch_size=None,
steps=test_generator.samples )
print("Test loss:", score[0])
print("Test accuracy:", score[1])
```

- P.3.2. funkcija za računanje vjerojatnosti da se na ulaznim slikama nalazi tekst

```
def mreza(self):

        test_datagen = ImageDataGenerator(rescale=1./255)

        test_generator = test_datagen.flow_from_directory(
                directory="set",
                target_size=(64, 64),
                batch_size=batch_size,
                class_mode='categorical',
                shuffle=False)

        test_generator.reset()

        for j in range (0,4):
            batch = test_generator.next()

        for j in range(0,1):
                batch = test_generator.next()
                probab = model.predict(batch[0])

        return probab[0][1],probab[1][1],probab[2][1],probab[3][1]
```

- P.3.3. konfiguracijska datoteka *textfusnet_requirements.yml* potrebna za stvaranje programskog okruženja

```yaml
name: textfusenet3
channels:
  - pytorch
  - defaults
dependencies:
  - _libgcc_mutex=0.1=main
  - _openmp_mutex=4.5=1_gnu
  - blas=1.0=mkl
  - brotlipy=0.7.0=py39h27cfd23_1003
  - bzip2=1.0.8=h7b6447c_0
  - ca-certificates=2022.4.26=h06a4308_0
  - certifi=2021.10.8=py39h06a4308_2
  - cffi=1.15.0=py39hd667e15_1
  - charset-normalizer=2.0.4=pyhd3eb1b0_0
  - cryptography=36.0.0=py39h9ce1e76_0
  - cudatoolkit=11.3.1=h2bc3f7f_2
  - ffmpeg=4.3=hf484d3e_0
  - freetype=2.11.0=h70c0345_0
  - giflib=5.2.1=h7b6447c_0
  - gmp=6.2.1=h2531618_2
  - gnutls=3.6.15=he1e5248_0
  - idna=3.3=pyhd3eb1b0_0
  - intel-openmp=2021.4.0=h06a4308_3561
  - jpeg=9e=h7f8727e_0
  - lame=3.100=h7b6447c_0
  - lcms2=2.12=h3be6417_0
  - ld_impl_linux-64=2.35.1=h7274673_9
  - libffi=3.3=he6710b0_2
  - libgcc-ng=9.3.0=h5101ec6_17
  - libgomp=9.3.0=h5101ec6_17
  - libiconv=1.16=h7f8727e_2
  - libidn2=2.3.2=h7f8727e_0
  - libpng=1.6.37=hbc83047_0
  - libstdcxx-ng=9.3.0=hd4cf53a_17
  - libtasn1=4.16.0=h27cfd23_0
  - libtiff=4.2.0=h85742a9_0
  - libunistring=0.9.10=h27cfd23_0
  - libuv=1.40.0=h7b6447c_0
  - libwebp=1.2.2=h55f646e_0
  - libwebp-base=1.2.2=h7f8727e_0
  - lz4-c=1.9.3=h295c915_1
  - mkl=2021.4.0=h06a4308_640
  - mkl-service=2.4.0=py39h7f8727e_0
  - mkl_fft=1.3.1=py39hd3c417c_0
  - mkl_random=1.2.2=py39h51133e4_0
```

```
- ncurses=6.3=h7f8727e_2
- nettle=3.7.3=hbbd107a_1
- numpy=1.21.5=py39he7a7128_2
- numpy-base=1.21.5=py39hf524024_2
- openh264=2.1.1=h4ff587b_0
- openssl=1.1.1n=h7f8727e_0
- pillow=9.0.1=py39h22f2fdc_0
- pip=21.2.4=py39h06a4308_0
- pycparser=2.21=pyhd3eb1b0_0
- pyopenssl=22.0.0=pyhd3eb1b0_0
- pysocks=1.7.1=py39h06a4308_0
- python=3.9.12=h12debd9_0
- pytorch=1.11.0=py3.9_cuda11.3_cudnn8.2.0_0
- pytorch-mutex=1.0=cuda
- readline=8.1.2=h7f8727e_1
- requests=2.27.1=pyhd3eb1b0_0
- setuptools=61.2.0=py39h06a4308_0
- six=1.16.0=pyhd3eb1b0_1
- sqlite=3.38.3=hc218d9a_0
- tk=8.6.11=h1ccaba5_0
- torchaudio=0.11.0=py39_cu113
- torchvision=0.12.0=py39_cu113
- typing_extensions=4.1.1=pyh06a4308_0
- tzdata=2022a=hda174b7_0
- urllib3=1.26.9=py39h06a4308_0
- wheel=0.37.1=pyhd3eb1b0_0
- xz=5.2.5=h7f8727e_1
- zlib=1.2.12=h7f8727e_2
- zstd=1.4.9=haebb681_0
- pip:
  - absl-py==1.0.0
  - antlr4-python3-runtime==4.8
  - appdirs==1.4.4
  - black==21.4b2
  - cachetools==5.0.0
  - click==8.1.3
  - cloudpickle==2.0.0
  - cycler==0.11.0
  - fonttools==4.33.3
  - future==0.18.2
  - fvcore==0.1.5.post20220506
  - google-auth==2.6.6
  - google-auth-oauthlib==0.4.6
  - grpcio==1.46.0
  - hydra-core==1.1.2
```

```
    - importlib-metadata==4.11.3
    - iopath==0.1.9
    - kiwisolver==1.4.2
    - markdown==3.3.7
    - matplotlib==3.5.2
    - mypy-extensions==0.4.3
    - oauthlib==3.2.0
    - omegaconf==2.1.2
    - opencv-python==4.5.5.64
    - packaging==21.3
    - pathspec==0.9.0
    - portalocker==2.4.0
    - protobuf==3.20.1
    - pyasn1==0.4.8
    - pyasn1-modules==0.2.8
    - pycocotools==2.0.4
    - pydot==1.4.2
    - pyparsing==3.0.8
    - python-dateutil==2.8.2
    - pyyaml==6.0
    - regex==2022.4.24
    - requests-oauthlib==1.3.1
    - rsa==4.8
    - scipy==1.8.0
    - tabulate==0.8.9
    - tensorboard==2.9.0
    - tensorboard-data-server==0.6.1
    - tensorboard-plugin-wit==1.8.1
    - termcolor==1.1.0
    - toml==0.10.2
    - tqdm==4.64.0
    - werkzeug==2.1.2
    - yacs==0.1.8
    - zipp==3.8.0
prefix: /home/mattex/miniconda3/envs/textfusenet
```

- P.3.4. konfiguracijska datoteka *synthtext_precise_101_FPN.yaml* koja služi za pokretanje treninga mreže (priloženo na DVD-u uz rad)

```yaml
_BASE_: "./Base-RCNN-FPN.yaml"
MODEL:
  MASK_ON: True
  TEXTFUSENET_MUTIL_PATH_FUSE_ON: True
  WEIGHTS:
"/home/idorkic/Downloads/tf/TextFuseNet/out_dir_r101/precizni5000/model_0081583.p
th"  # "./out_dir_r101/icdar2013_model/model_ic13_r101.pth" za treniranje
  PIXEL_STD: [57.375, 57.120, 58.395]
  RESNETS:
    STRIDE_IN_1X1: False  # this is a C2 model
    NUM_GROUPS: 32
    WIDTH_PER_GROUP: 8
    DEPTH: 101
  ROI_HEADS:
    NMS_THRESH_TEST: 0.35
  TEXTFUSENET_SEG_HEAD:
    FPN_FEATURES_FUSED_LEVEL: 2
    POOLER_SCALES: (0.0625,)
DATASETS:
  TRAIN: ("synthtext_imprecise5000",)#na kojem skupu zelimo trenirati
  TEST: ("synthtext_test_precise",) #testiranje ide test, trening ide valid
SOLVER:
  IMS_PER_BATCH: 1
  BASE_LR: 0.001
  STEPS: (20000,40000,)
  MAX_ITER: 305940  # ovisno koji skup koristimo  61470   195900   305940
  CHECKPOINT_PERIOD: 5099                         # 1029    3265      5099
INPUT:
  MIN_SIZE_TRAIN: (400,500,600)
  MAX_SIZE_TRAIN: 1000
  MIN_SIZE_TEST: 500
  MAX_SIZE_TEST: 1000

OUTPUT_DIR: "./out_dir_r101/icdar2015_model/"
```