

Motivacija

Ideja je bila napraviti Machine Learning algoritam koji će biti u mogućnosti da čita i oživljava stripove.
Izabran je strip domaćeg izdavača Alan Ford.
Kompletan projekat podrazumeva:
1. prepoznavanje teksta iz oblačića
2. prepoznavanje glavnih likovima
3. pripisivanje oblačića likovima
4. text to speech sistem koji izgovara pročitano
5. "oživljavanje" junaka koji govori na neki način
(neophodno da bi se znalo koji junak je taj koji govori)

*u ovom projektu podržane su samo prve dve tačke,
prepoznavanje teksta i prepoznavanje četiri glavna lika



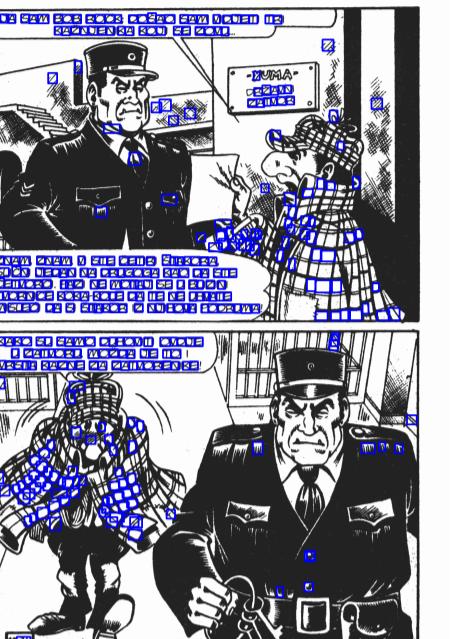
Fragment iz data set-a
(ceo sadrži 50 brojeva
sa po ~120 stranica stripa)



Prepoznavanje teksta u stripu

Ovaj zadatak se pokazao kao teži nego što je bilo očekivano, s obzirom na prirodu stripa u crno belom formatu koja sa sobom nosi mnogo distrakcija koje standarnim metodama za prepoznavanje određenih kontura otežavaju posao.

Prvi korak bio je pronaći skupove slova koji zajedno čine jedan oblačić.
Za tako nešto bilo je neophodno prvo pronaći konture čije dimenzije odgovaraju dimenzijama slova:



Slika 1

Da bi se izdvojena slova stopila u jednu konturu uradeno je sledeće.
Izdvojeni su plavi pikseli sa slike, pretvoreni u bele i postavljeni na novu crnu sliku jedanke veličine.

Nakon toga izvršena je dilektacija po horizontalnoj osi u cilju gubljenja razmaka između reči i dobijanja spojenih kontura koje mogu biti prepoznate kao oblačići:



Slika 2

To još uvek nije bilo dovoljno za određivanje oblačića na slici. Ostajao je problem sa konturama koje svojim dimenzijama zadovoljavaju uslove da budu prihvadena kao oblačići, a da to nisu.
U cilju otklanjanja ovakvih kontura dodate su još dve provere:

1. Da li je procenat belih piksela u konturi dovoljno veliki
2. Da li je ugao pod kojim se nalazi kontura jednak 2π

Kao što se vidi na slici, ovim postupkom izolovane su same one konture koje stvarno predstavljaju oblačiće

(crvenom bojom su obeležene konture koje bi bez ovih dodatnih provera bile okarakterisane kao oblačići):



Slika 3

Nakon što su regioni koji predstavljaju oblačiće uspešno izdvojeni:

JA SAM BOB ROCK DOŠAO SAM VIDJETI TRI KAŽNJENIKA KOJI SE ZOVU...
DRAŽAVNI ZATVOR
ZAJAM, ZNATE VI STE ĆETIRI ŠTAKORA, EUCI JEDAN NA DRUGOGA KAO DA STE ĆETVORCI, PAZI NE MOTAJ SE U BLIZINI TVORNICE KOKA-KOLE, DA TE NE UHVATE MLEŠEĆI DA SI ŠTAKOR IZ NJUHOVA PODRUMA!
KAKO SU SAMO DUHOVITI OVOJE U ZATVORU, MOŽDA JE TO I VRSTA KAZNE ZA ZATVORENIKE!

Slika 4

Bilo je potrebno izvršiti sortiranje regiona sa leva desno i od gore na dolje.
Tu su pojavili novi problemi, naime, koliko god pokušavali da izdvojimo samo one konture koji odgovaraju slovima, uvek bismo u specijalnim slučajevima zajedno pokupili i po neku neželjenu konturu koja bi onemogućila uspešan rad algoritma za sortiranje.



Slika 5

Za rešavanje ovog problema, uveden je DBSCAN algoritam koji izdvaja samo slova tako što formira klaster koji mora zadovoljiti dva uslova:

1. mora sarzati dovoljno veliki broj kontura
2. konture medusobno moraju biti dovoljno blizu

Ovim smo konačno došli do trenutka kada iz oblačića možemo izdvojiti samo slova i propustiti ih kroz veštaku neuronsku mrežu koju smo prethodno obučili korisćenjem obeleženog seta svih slova iz abecede koje smo prethodno sveli na jednake dimenzije 28x28 piksela.

0 1 2 3 4 5 6 7 8 9
A B C Č Č Č D E F G
H I J K L M N O P R
S T U V W X Y Z Z

Slika 6

Konačno, algoritam uspešno uspeva da pročita tekst iz strip-a, osim u retkim situacijama kada sortiranje ne može biti uspešno izvršeno.

Primeri kada dolazi do neuspelog sortiranja:

ZELITE LI CMJEĆE ZA
ROBODAN, CMJEĆE ZA
GRAB POŠTUNJE PUNICE
CMJEĆE ZA PRVI
POLJUBAC, CMJEĆE ZA...

Slika 7

Gornja granica slova Č niza je od donje granice nekog reda ili slova iz prethodnog reda

Sljika 8
U otkriven oblačiću koji želimo da obradujemo našao se deo drugog oblačića zbog koga je nemoguće sortirati konture.

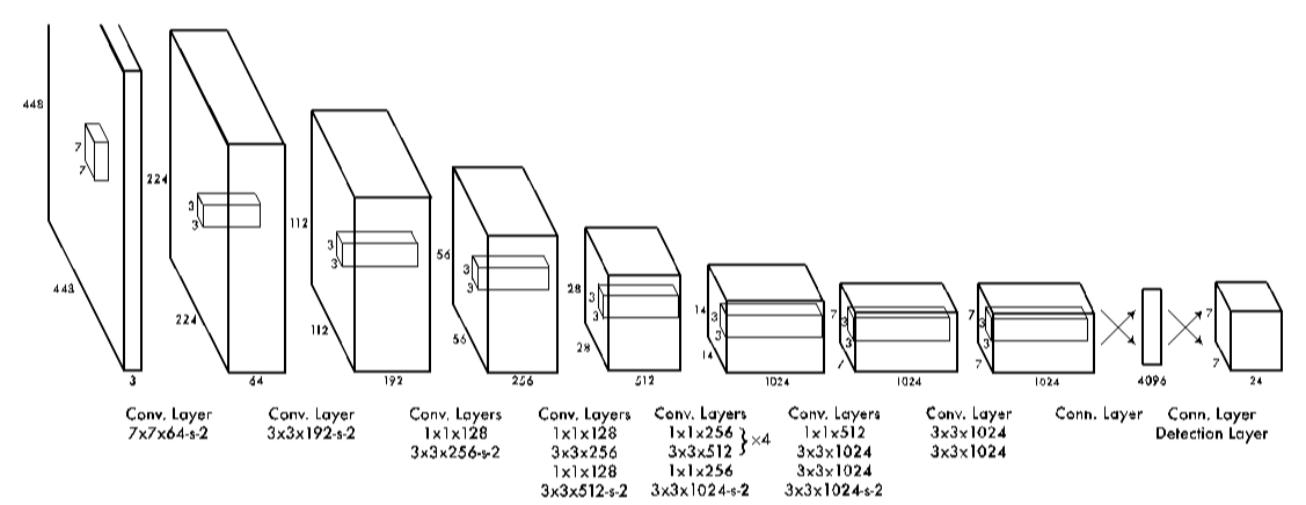
Prepoznavanje lica - korišćenjem YOLO algoritma

YOLO (You Only Look Once) je algoritam za detekciju i klasifikaciju objekata napisan u Darknet framework-u u programskom jeziku C. U ovom projektu korišćen je oblik Darkneta koji je preveden u Python i prilagođen za Tensorflow - Darkflow.

Pretходnici ovog algoritma su za detektovanje objekta na slici morali svaki pojedinačni segment slike slati na predikciju u prethodno obučenu konvolucionu neuronsku mrežu i takav pristup je zahtevaо veliki broj iteracija i veliki broj predikcija.

Ideja iz YOLO(You Only Look Once) algoritma je u tome da se slika za obradu samo jednom provlači kroz konvolucionu neuronsku mrežu.

Arhitektura mreže:



Mreža se sastoji iz dva glavna dela, konvolucionih slojeva za ekstrakovanje značajnih delova slike (feature-a) i dva potpuno spojena sloja.

Način na koji YOL O obavlja ceo postupak detektovanja objekata na slici u jednom prolazu kroz CNN je sledeći:

Podaci (slike) za obučavanje mreže se označavaju na poseban način:

- slika se podeli na MxM regiona (u ovom slučaju M je 19)
- za svaki objekat na slici određuje se centralna tačka, visina i širina
- s obzirom na to da svaka centralna tačka pripada tačno jednom od regiona, tom regionu se dati objekat i pripisuje
- za svaki region odredujemo i standardnim opisujućim pravougaonika (bounding box) (u ovom slučaju i je 5)
- svakom objektu čija je centralna tačka u datom regionu, pripisujemo jedan od standardnih opisujućih pravougaonika, onaj koji se najviše poklapa sa oblikom objekta.
- formiramo ulazni vektor koji izgleda ovako:

$$y = \begin{bmatrix} p_{c1} & p_{c2} & p_{c3} & p_{c4} & p_{c5} \\ b_{x1} & b_{y1} & b_{w1} & b_{h1} & c_{11} \\ c_{12} & c_{13} & c_{14} & c_{15} & c_{21} \\ c_{22} & c_{23} & c_{24} & c_{25} & c_{31} \\ c_{32} & c_{33} & c_{34} & c_{35} & c_{41} \\ c_{42} & c_{43} & c_{44} & c_{45} & c_{51} \\ c_{52} & c_{53} & c_{54} & c_{55} & c_{61} \\ c_{62} & c_{63} & c_{64} & c_{65} & c_{71} \\ c_{72} & c_{73} & c_{74} & c_{75} & c_{81} \\ c_{82} & c_{83} & c_{84} & c_{85} & c_{91} \\ c_{92} & c_{93} & c_{94} & c_{95} & c_{101} \\ c_{102} & c_{103} & c_{104} & c_{105} & c_{111} \\ c_{112} & c_{113} & c_{114} & c_{115} & c_{121} \\ c_{122} & c_{123} & c_{124} & c_{125} & c_{131} \\ c_{132} & c_{133} & c_{134} & c_{135} & c_{141} \\ c_{142} & c_{143} & c_{144} & c_{145} & c_{151} \\ c_{152} & c_{153} & c_{154} & c_{155} & c_{161} \\ c_{162} & c_{163} & c_{164} & c_{165} & c_{171} \\ c_{172} & c_{173} & c_{174} & c_{175} & c_{181} \\ c_{182} & c_{183} & c_{184} & c_{185} & c_{191} \\ c_{192} & c_{193} & c_{194} & c_{195} & c_{201} \\ c_{202} & c_{203} & c_{204} & c_{205} & c_{211} \\ c_{212} & c_{213} & c_{214} & c_{215} & c_{221} \\ c_{222} & c_{223} & c_{224} & c_{225} & c_{231} \\ c_{232} & c_{233} & c_{234} & c_{235} & c_{241} \\ c_{242} & c_{243} & c_{244} & c_{245} & c_{251} \\ c_{252} & c_{253} & c_{254} & c_{255} & c_{261} \\ c_{262} & c_{263} & c_{264} & c_{265} & c_{271} \\ c_{272} & c_{273} & c_{274} & c_{275} & c_{281} \\ c_{282} & c_{283} & c_{284} & c_{285} & c_{291} \\ c_{292} & c_{293} & c_{294} & c_{295} & c_{301} \\ c_{302} & c_{303} & c_{304} & c_{305} & c_{311} \\ c_{312} & c_{313} & c_{314} & c_{315} & c_{321} \\ c_{322} & c_{323} & c_{324} & c_{325} & c_{331} \\ c_{332} & c_{333} & c_{334} & c_{335} & c_{341} \\ c_{342} & c_{343} & c_{344} & c_{345} & c_{351} \\ c_{352} & c_{353} & c_{354} & c_{355} & c_{361} \\ c_{362} & c_{363} & c_{364} & c_{365} & c_{371} \\ c_{372} & c_{373} & c_{374} & c_{375} & c_{381} \\ c_{382} & c_{383} & c_{384} & c_{385} & c_{391} \\ c_{392} & c_{393} & c_{394} & c_{395} & c_{401} \\ c_{402} & c_{403} & c_{404} & c_{405} & c_{411} \\ c_{412} & c_{413} & c_{414} & c_{415} & c_{421} \\ c_{422} & c_{423} & c_{424} & c_{425} & c_{431} \\ c_{432} & c_{433} & c_{434} & c_{435} & c_{441} \\ c_{442} & c_{443} & c_{444} & c_{445} & c_{451} \\ c_{452} & c_{453} & c_{454} & c_{455} & c_{461} \\ c_{462} & c_{463} & c_{464} & c_{465} & c_{471} \\ c_{472} & c_{473} & c_{474} & c_{475} & c_{481} \\ c_{482} & c_{483} & c_{484} & c_{485} & c_{491} \\ c_{492} & c_{493} & c_{494} & c_{495} & c_{501} \\ c_{502} & c_{503} & c_{504} & c_{505} & c_{511} \\ c_{512} & c_{513} & c_{514} & c_{515} & c_{521} \\ c_{522} & c_{523} & c_{524} & c_{525} & c_{531} \\ c_{532} & c_{533} & c_{534} & c_{535} & c_{541} \\ c_{542} & c_{543} & c_{544} & c_{545} & c_{551} \\ c_{552} & c_{553} & c_{554} & c_{555} & c_{561} \\ c_{562} & c_{563} & c_{564} & c_{565} & c_{571} \\ c_{572} & c_{573} & c_{574} & c_{575} & c_{581} \\ c_{582} & c_{583} & c_{584} & c_{585} & c_{591} \\ c_{592} & c_{593} & c_{594} & c_{595} & c_{601} \\ c_{602} & c_{603} & c_{604} & c_{605} & c_{611} \\ c_{612} & c_{613} & c_{614} & c_{615} & c_{621} \\ c_{622} & c_{623} & c_{624} & c_{625} & c_{631} \\ c_{632} & c_{633} & c_{634} & c_{635} & c_{641} \\ c_{642} & c_{643} & c_{644} & c_{645} & c_{651} \\ c_{652} & c_{653} & c_{654} & c_{655} & c_{661} \\ c_{662} & c_{663} & c_{664} & c_{665} & c_{671} \\ c_{672} & c_{673} & c_{674} & c_{675} & c_{681} \\ c_{682} & c_{683} & c_{684} & c_{685} & c_{691} \\ c_{692} & c_{693} & c_{694} & c_{695} & c_{701} \\ c_{702} & c_{703} & c_{704} & c_{705} & c_{711} \\ c_{712} & c_{713} & c_{714} & c_{715} & c_{721} \\ c_{722} & c_{723} & c_{724} & c_{725} & c_{731} \\ c_{732} & c_{733} & c_{734} & c_{735} & c_{741} \\ c_{742} & c_{743} & c_{744} & c_{745} & c_{751} \\ c_{752} & c_{753} & c_{754} & c_{755} & c_{761} \\ c_{762} & c_{763} & c_{764} & c_{765} & c_{771} \\ c_{772} & c_{773} & c_{774} & c_{775} & c_{781} \\ c_{782} & c_{783} & c_{784} & c_{785} & c_{791} \\ c_{792} & c_{793} & c_{794} & c_{795} & c_{801} \\ c_{802} & c_{803} & c_{804} & c_{805} & c_{811} \\ c_{812} & c_{813} & c_{814} & c_{815} & c_{821} \\ c_{822} & c_{823} & c_{824} & c_{825} & c_{831} \\ c_{832} & c_{833} & c_{834} & c_{835} & c_{841} \\ c_{842} & c_{843} & c_{844} & c_{845} & c_{851} \\ c_{852} & c_{853} & c_{854} & c_{855} & c_{861} \\ c_{862} & c_{863} & c_{864} & c_{865} & c_{871} \\ c_{872} & c_{873} & c_{874} & c_{875} & c_{881} \\ c_{882} & c_{$$