Prepoznavanje spora nozema na mikroskopskim slikama

1. Definicija problema

Prepoznavanje spora nozema na mikroskopskim snimcima složen je problem iz razloga što su spore veoma mali objekti u odnosu na veličinu slike. Pored toga na slici nalazi se i veliki broj drugih spora koje imaju sličan izgled. Takođe, spore mogu da se nalaze u raznim položajima, ne samo horizontalnom i vertikalnom. Problem detekcije objekata na slikama može se podeliti u 4 različite kategorije: klasifikacija, klasifikacija i lokalizacija, detekcija objekata i segmentacija. Razlika između tih slučajeva prikazana je na Slici 1.1.

Slika 1.1 Kategorije detekcije objekata na slikama od najlakše do najteže (prema desno)

Najjednostavniji od ta četiri problema jeste problem klasifikacije. U njemu je potrebno samo detektovati kojoj klasi pripada slika zavisno o objektu koji se nalazi na njoj. Malo komplikovaniji problem u odnosu na čistu klasifikaciju jeste klasifikacija kombinovana sa lokalizacijom. U ovome slučaju se osim rešavanja problema kojoj klasi pripada slika mora definisati i gde se objekat koji pripada toj klasi nalazi na slici. Treći problem je detekcija objekata. Ovaj problem je vrlo sličan klasifikaciji i lokalizaciji samo što je dodatno otežan zahtevom da se na jednoj slici mora pronaći više objekata koji mogu pripadati različitim klasama. Najteži od ova četiri problema je segmentacija objekata. Segmentacija je problem gde se zahteva klasifikacija u pripadajuću klasu za svaki pojedinačni piksel na slici. Konačno rešenje ne zahteva segmentaciju, pa će u ovom radu biti opisana detekcija objekata. U području detekcije objekata koristile su se neke konvencionalne metode, međutim puno bolji rezultati postignuti su primenom neuronskih mreža. U slučajevima kada ne postoji adekvatna arhitektura neuronske mreže za dati domenski problem, adaptacija neke postojeće arhitekture koja se pokazala kao efikasna u rešavanju sličnog problema, vrlo je jasan pristup.

2. Klasifikacioni i detekcioni modeli

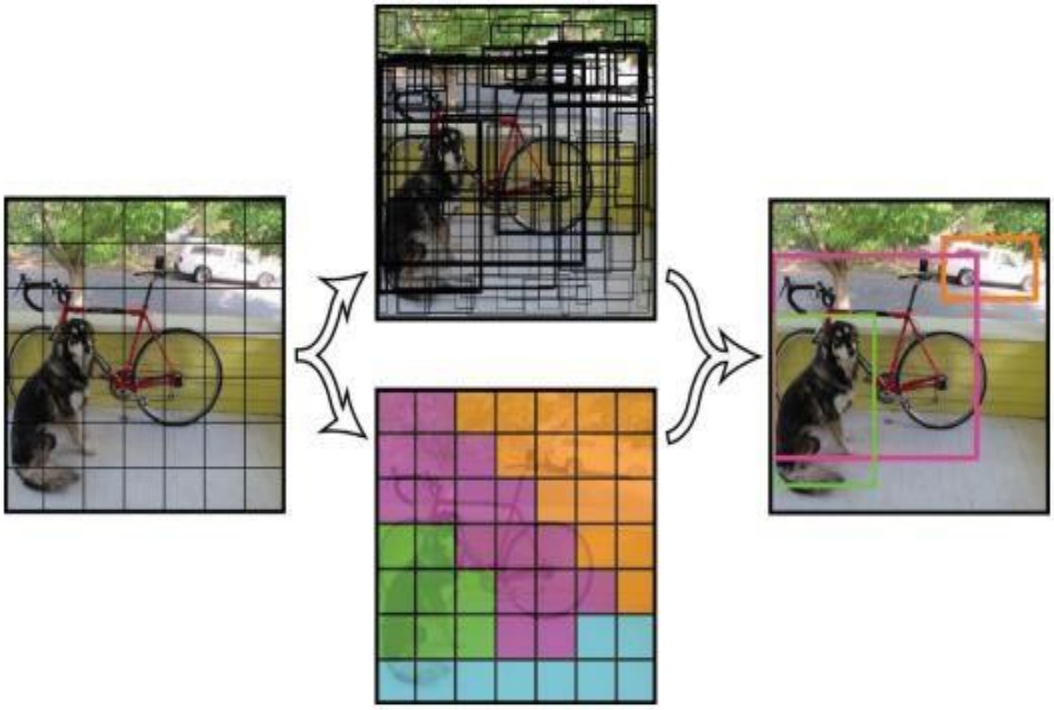
Daleko najistraženiji i najkorišteniji modeli neuronskih mreža su klasifikacioni modeli koji ulaze različitih tipova preslikavaju u labele koje označavaju kategorije ili klase.
Proces učenja ovakvih neuronskih mreža odvijao se nad velikim skupovima podataka kao što je ImageNet. ImageNet je baza sa preko 14 miliona ručno označenih slika sa ukupno 1000 kategorija. Performanse klasifikacionih modela izmerene su na ImageNet takmičenju gde je prvo mesto zauzela GoogleNet ili Inception mreža. Ovakve već obučene mreže moguće je iskoristi za domenski problem. Na primer, lakše je obučiti već obučeni sistem za detektovanje automobila da prepoznaje kamione nego ispočetka obučavati sistem za detekciju kamiona. Klasifikacioni modeli omogućavaju takozvani transfer znanja, odnosno, mreža obučena na jednom problemu klasifikacije se može iskoristiti za izračunavanje deskriptora slika koji će se koristiti u drugom, nepovezanom problemu. Prethodno spomenuti modeli u stanju su da klasifikuju slike, međutim potrebno je implementirati sistem koji će detektovati instance individualnih klasa na slikama, te da odrede njihovu lokaciju. Postoji nekoliko glavnih modela koji se u praksi najčešće koriste za ovu svrhu.

2.1 R-CNN

Regionalna konvoluciona neuronska mreža ili R-CNN je jedan takav model, zajedno sa njemu sličnim modelima: Brzi R-CNN (*eng. Fast R-CNN*) i Brži R-CNN (*eng. Faster R-CNN*). Osnovna ideja ove arhitekture je iskoristiti duboku neuronsku mrežu koja je već istrenirana za klasifikaciju slika i modifikovati je za detekciju objekata. Kao korak predobrade, algoritamski se pronalazi 2000 kandidatskih regiona na ulaznoj slici, nakon čega se svaki od regiona individualno klasifikuje neuronskom mrežom. Glavni problem kod ovog pristupa su u vremenu potrebnom da se svi regioni klasifikuju i u činjenici da obučavanje mreže može da traje izuzetno dugo. Algoritam za predselekciju regiona je klasičan deterministički. Kod Bržeg R-CNN, ovaj problem je prevaziđen uvođenjem dodatne mreže koja služi za selekciju regiona.

2.2 YOLO

Spomenuti pristup je spor i jako ga je teško optimizovati zbog složenosi. Zbog toga postoje pristupi kao što je YOLO (*eng. You Only Look Once*) koji formulišu problem na način da se pikseli na slici pretvaraju u regije i pripadajuće klase. Za razliku od R-CNN gde se prvo predviđaju regije pa se na njima radi klasifikacija, u YOLO pristupu znatno se smanjuje broj pogrešno pozitivnih regija. Najveća prednost ovog pristupa je u njegovoj brzini. Pomoću YOLO arhitekture slika se pogleda samo jednom kako bi se predvidelo koji su predmeti prisutni na toj slici. Regije koje nisu maksimalno dobro postavljene u odnosu na posmatrani objekat se odbacuju. Iako može brzo prepoznati objekte na slikama, još uvek se muči prilikom precizne lokalizacije pojedinih objekata. Način dobijanja detekcije na slici metodom YOLO prikazan je na Slici 2.1.



Slika 2.1 Detekcija objekata metodom YOLO

2.3 Detectron

Pored prethodna dva spomenuta modela, bitno je spomenuti još jedan alat za detekciju objekata, Detectron. Detectron je projekat koji je razvijen u Facebook-ovoj AI Research grupi. Detectron je izuzetno koristan alat koji omogućava lakše učenje modela i eksperimentisanje sa podacima. U ovom radu je korištena poboljšana druga verzija softvera nazvana Detectron2. Detectron2 je programski alat koji implementira razne algoritme visokog kvaliteta za detekciju objekata i segmentaciju a uključuje R-CNN i sve njegove naslednike. Implementiran je pomoću PyTorch biblioteke otvorenog koda koja se koristi za mašinsko učenje.

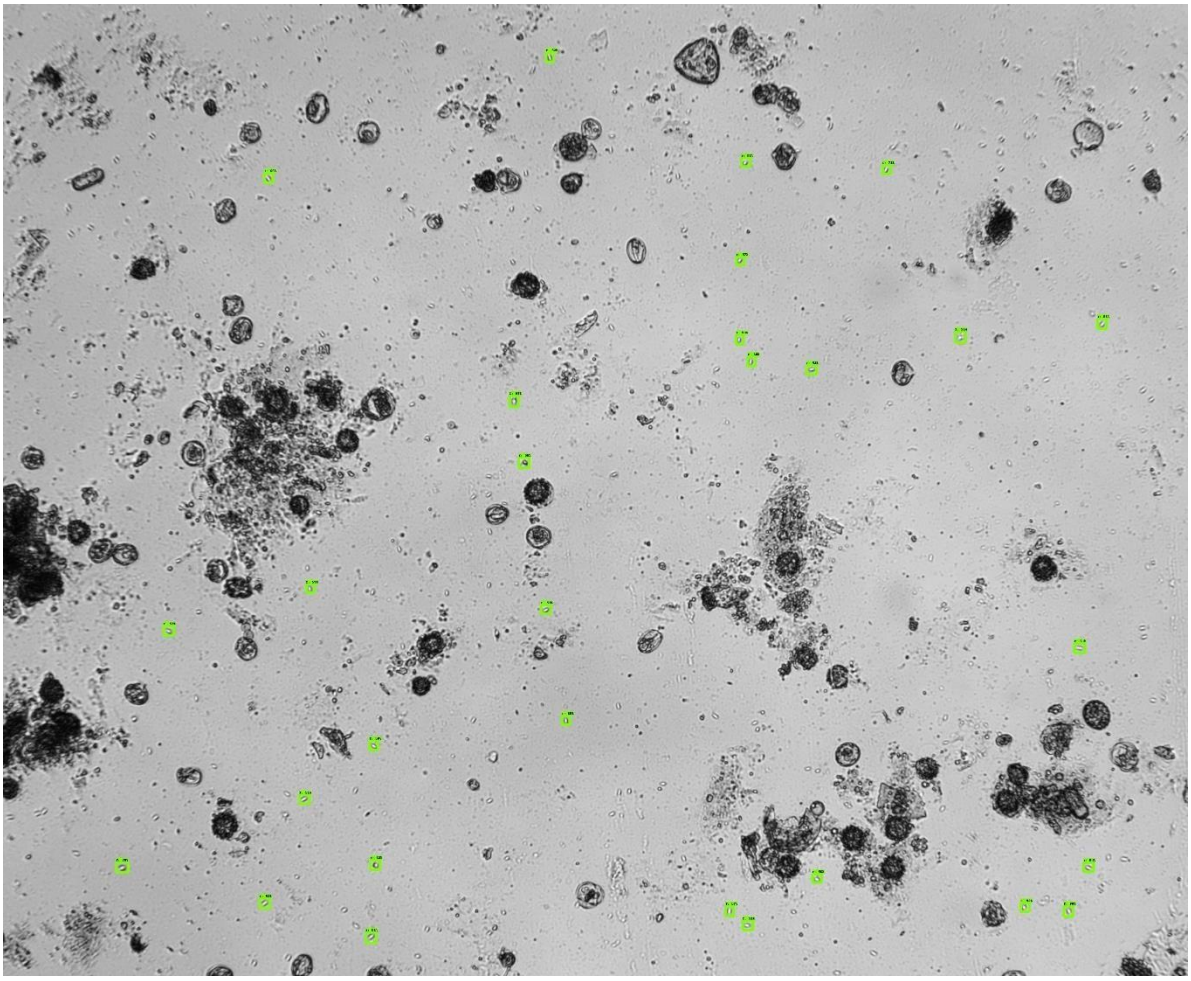
3. Programska implementacija

3.1 Priprema skupa podataka

Prvi korak u eksperimentu bio je labelovanje svih slika kako bi se mogle koristiti kao skup podataka prilikom učenja i evaluacije modela. Pošto su originalne slike velikih dimenzija, podeljene su na slike dimenzija 416x416, na takav način da su se iz originalnih slika uzimali delovi na kojima su se nalazile spore. Dalje, da bih mogao koristiti vlastiti skup podataka, bilo ga je potrebno prilagoditi odgovarajućem formatu. Detectron2 koristio je JSON (*eng. JavaScript Object Notation*) format, dok su R-CNN i YOLO koristili zasebne formate. Taj prilagođeni format je u stvari zapravo skup podataka koji sadrži listu rečnika, a svaki rečnik zasebno opisuje jednu sliku i položaj spore nozema na njoj. Sledeći korak bio je vršenje augmentacije nad celim skupom podataka za obučavanje, korištenjem nasumičnog rotiranja i isecanja podslika iz slika iz skupa za obučavanje. Sve ovo odrađeno je uz pomoć specijalizovanog alata Roboflow. Programski kod za sve aplikacije je napisan na Google Colab platformi zato jer pružaju opciju za pokretanje koda na njihovim serverima koji imaju opciju korištenja grafičkih kartica.

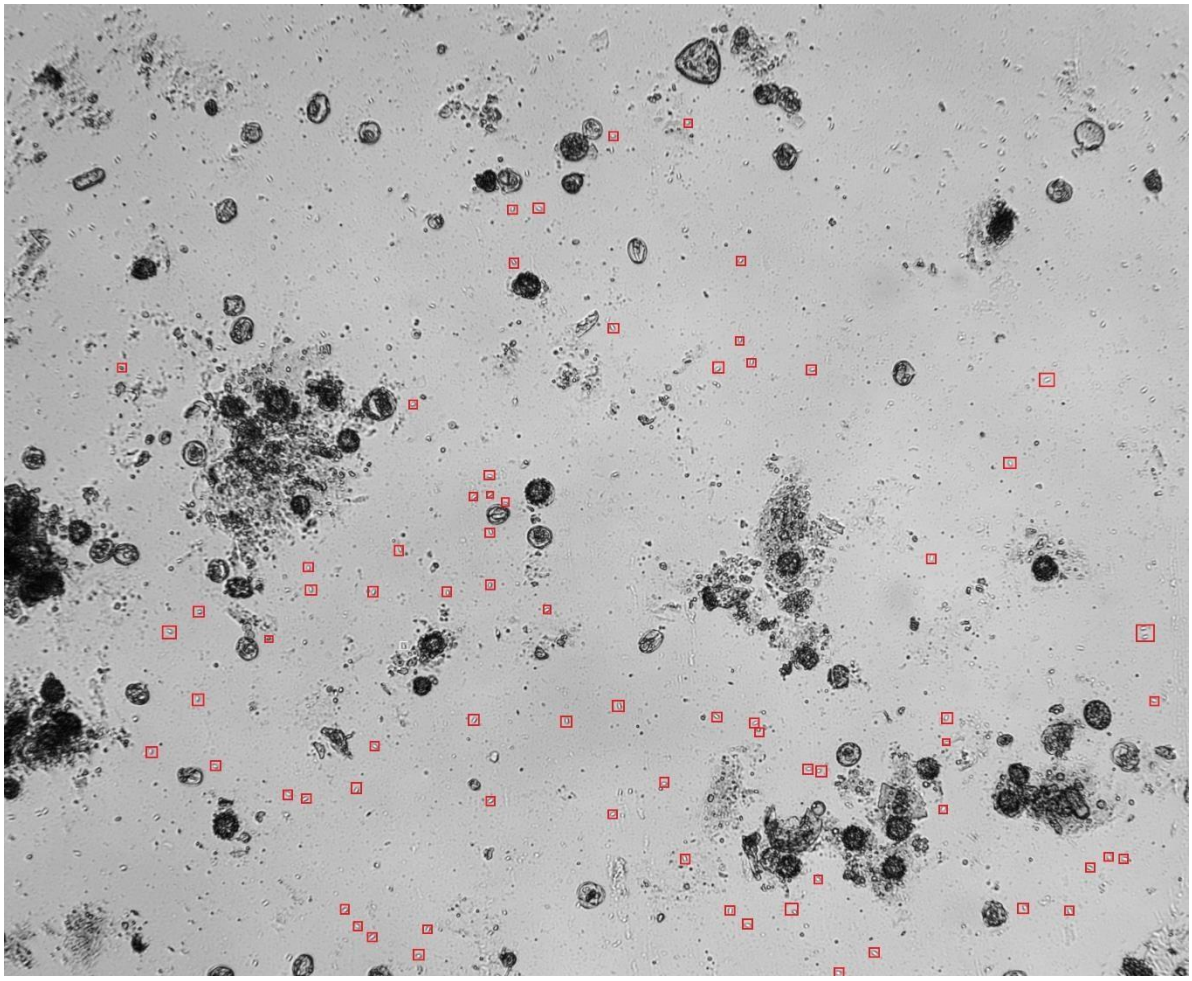
3.2 Implementacija pomoću Faster R-CNN modela

Obzirom da je skup podataka za obučavanje relativno mali, da bi se složene arhitekture obučile, neophodno je da se postojeće težine naučene na sličnom problemu, prenesu u novu arhitekturu. Kao polazni model izabrana je Faster R-CNN arhitektura, a za klasifikaciju regiona izabrana je Inception mreža, jer postoje već obučeni modeli ove mreže na ImageNet skupu, koji uključuje i mikroskopske snimke. Program pretežno koristi TensorFlow biblioteku. TensorFlow je programska biblioteka otvorenog koda namenjena za numeričko računanje uz pomoć grafova. Razvijena je u Google-ovom razvojnom timu za potrebe mašinskog učenja i istraživanja dubokih modela neuronskih mreža. Hiperparametri koji su bitni za ovaj model su batch_size i num_steps. Prvi hiperparametar daje oznaku modelu koliko iteracija treba proći dok ne pokuša napraviti neku predikciju dok drugi, broj koraka za obučavanje, označava broj koliko puta treba ponoviti prolazak kroz sve ulazne podatke. Što je veći batch_size treniranje će biti brže ali prezisnost lošija. Obučavanje je isprobano na 10000 i 20000 epoha, međutim obučavanje na 20000 nije dalo značajno bolje rezultate. Za ovaj model korišten je batch_size = 12. Korak obučavanja postavljen je na 50, a sa prosečnim vremenom od 110 sekundi po koraku, obučavanje je trajalo oko 6 sati. Testiranje je izvršeno na slici koja nije bila deo trening skupa. Ona je podeljena u blokove veličine 416x416 i na svakom bloku se posebno vršila detekcija. Na Slici 3.1 prikazani su rezultati detekcije, a na Slici 3.2 prikazani su idealni rezultati, odnosno na njoj su označene sve spore nozema.



Slika 3.1 Rezultat detekcije Faster R-CNN modelom

Detektor je prepoznao 29 objekata od kojih su 17 spore nozema, a preostalih 12 veoma podsećaju na njih. Ipak, nije detektovao većinu spora jer se na slici nalaze ukupno 72 spore. Bolji rezultati bi se dobili kada bi te lažno detektovane spore svrstali u negativnu klasu, pa mrežu obučili da prepoznaje i lažne spore. Neki od mogućih razloga za mali broj detekcija može biti u tome da su se u trening skupu nalazile i neke neoznačene spore, kao i to što su se spore nozema na trening slikama nalazile u sredini, pa je algoritam stvorio tendenciju da se objekat nalazi u sredini.

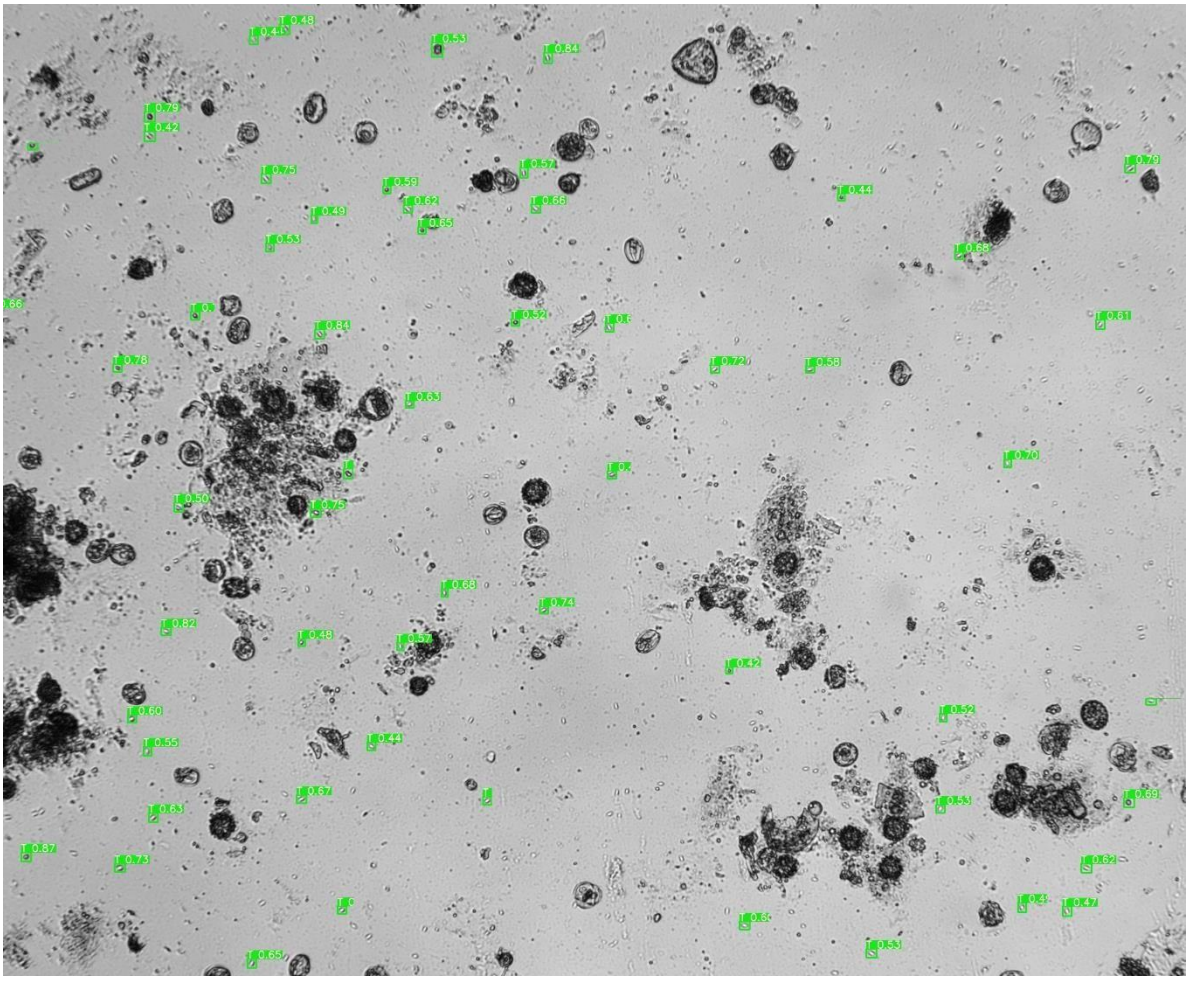


Slika 3.2 Idealni rezultati detekcije

3.3 Implementacija pomoću YOLO

YOLO arhitektura predstavljena je u nekoliko verzija. Cilj svake novije verzije bio je povećanje brzine detekcije, pa ne mora da znači da novije verzije bolje detektuju objekte. S obzirom da je glavni cilj ovog projekta kvalitetna a ne brza detekcija, eksperimentisano je sa više verzija YOLO arhitekture. Izvršeno je testiranje na YOLOv3, YOLOv4 i YOLOv5 arhitekturama. Najbolje rezultate dala je poslednja verzija pa su na njoj vršena dalja testiranja. YOLOv5 zahteva podatke za treniranje u odgovarajućem formatu, pa je pomoću Roboflow alata skup podataka pripremljen za taj format. Nakon učitavanja podataka, konfigurisani su parametri za obučavanje. Obučavanje je vršeno u 100 epoha a batch_size je postavljen na 16. Razlog sa ovaj mali broj epoha je u tome što AP dostigne svoju maksimalnu vrednost već oko 60-te epohe. Zbog malog broja epoha, obučavanje je završeno relativno brzo.

Testiranje je izvršeno na istoj slici kao i u prethodnom slučaju. Slika 3.3 pokazuje rezultate detekcije.

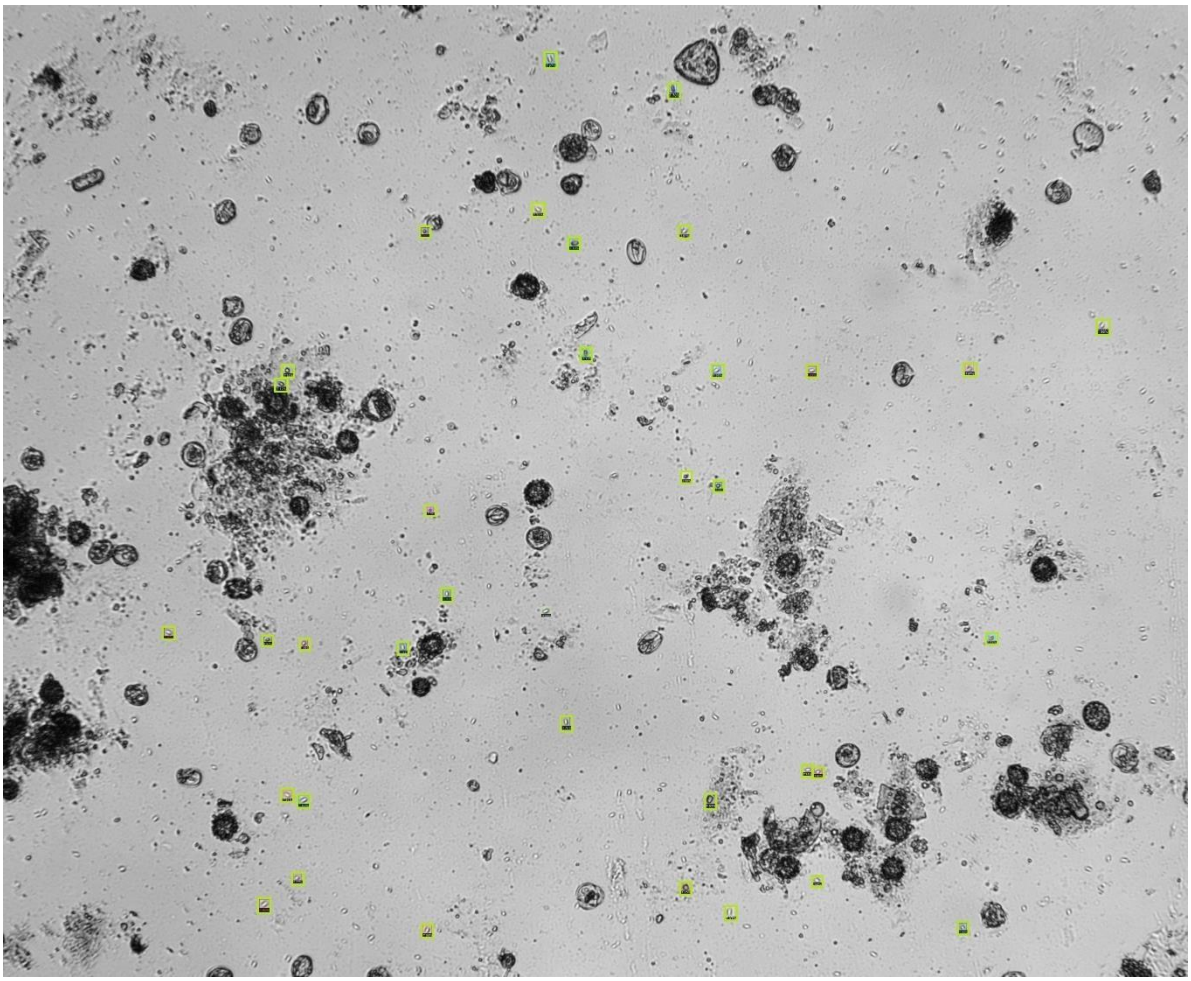


Slika 3.3 Rezultat detekcije YOLO modelom

Može se reći da YOLO dobro detektuje ali ima i puno lažnih detekcija. Detektovao je 24 spore nozema i 35 drugih objekata koje je takođe klasifikovao kao spore. Kao što se i pretpostavljalo, YOLO arhitektura namenjena je brzoj detekciji nekih većih i izraženijih objekata. Poboljšanje bi moglo biti ostvareno kada bi trening skup bio znatno veći. Ovakav model se može iskoristiti za pretpostavku položaja spore nozema, međutim potrebna je dalja analiza tog položaja.

3.4 Implementacija pomoću Detectron2

Nakon instalacije i učitavanja potrebnih biblioteka, učitani su već pripremljeni podaci. Za prikaz labelovanog objekta na slici iskorištena je klasa Visualizer pomoću koje se odmah može proveriti da li su anotacije dobro učitane u memoriju. Prije pokretanja učenja modela, potrebno je dohvatiti konfiguraciju i težine baznog modela nad kojim će biti pokrenut postupak učenja. Detectron2 pruža različite konfiguracione datoteke prednaučenih modela za detekciju i segmentaciju objekata, dostupnih u model_zoo modulu. U ovom eksperimentu izabran je MASK R-CNN model kojem je isključena opcija za segmentaciju. MASK R-CNN je poslednji model u R-CNN familiji koji sadrži i opciju da segmentira detektovani objekat. Korištena je implementacija sa rezidualnom neuronskom mrežom (ResNet), koja se dobro pokazala u detekciji malih objekata. Za ovaj skup podataka, broj područja od interesa postavljen je na 64. Bazna stopa učenja je 0.001 koja eksperimentalno daje najbolje rezultate.

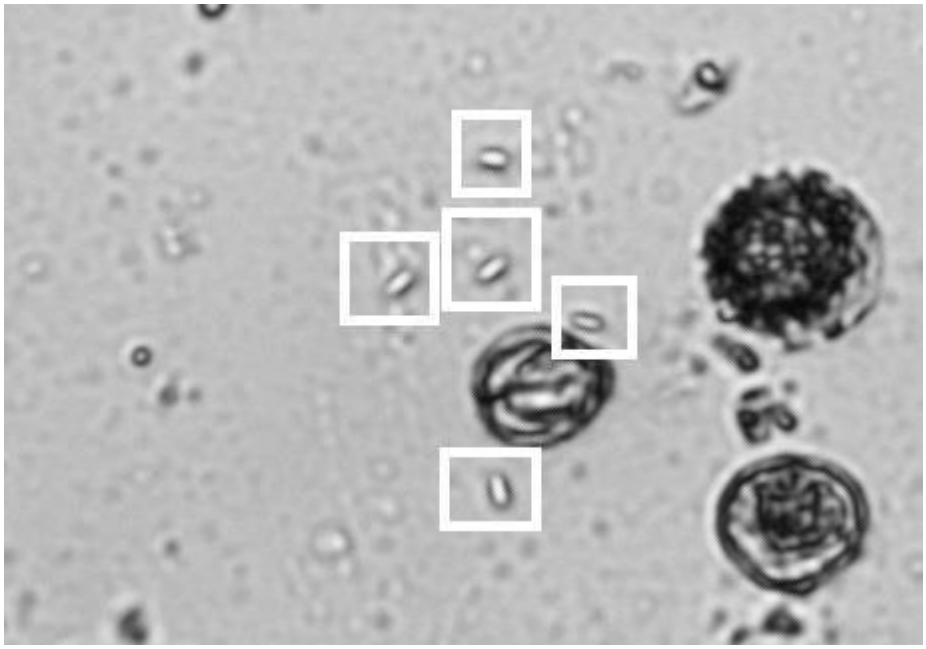


Slika 3.4 Rezultat detekcije Detectron2 modelom

Broj iteracija učenja je 1500, a evaluacijski period postavljen je na 500, što znači da će se evaluacija pozivati na svakih 500 iteracija. Ostali parametri su postavljeni na podrazumevane vrednosti. Za učenje modela korištena je zadana funkcionalnost DefaultTrainer prilagođena potrebama projekta preko konfiguracione postavke. Kao i u prethodna dva slučaja, i ovde je ista slika podeljena u blokove 416x416 i iskorištena je za testiranje. Za vizuelizaciju dobijenih podataka koristi se klasa Visualizer čija metoda draw_instance_predictions vraća ulaznu sliku sa nacrtanim prediktiranim okvirima. Na slici 3.4 prikazani su rezultati detekcije. Primeti se da je detektovano 36 objekata od kojih su 16 spore nozema a preostalih 20 su lažne detekcije. Ovaj rezultat je znatno lošiji od rezultata polaznog modela.

3.5 Upoređivanje rezultata

Svi objekti koje je Faster R-CNN detektovao ili su bile spore nozema ili su veoma podsećali na njih. YOLO je detektovao malo više pravih spora, međutim i mnogo lažnih među kojima se nalaze i objekti koji uopšte ne liče na spore nozema. Detectron2 je uglavnom detektovao iste spore kao i Faster R-CNN ali imao je i mnogo više lažnih predikcija. Treba napomenuti da ni jedan od prethodnih eksperimenata nije uspeo da detektuje neke veoma izražene spore, kao što su prikazane na Slici 3.5.



Slika 3.5 Primer spora nozema

Nakon završetka učenja evaluirane su predikcije koje je računar napravio nad validacionim skupom podataka. Metrika koja je izabrana za određivanje performansi je AP (*eng. average precision*). Definiše se na sledeći način:

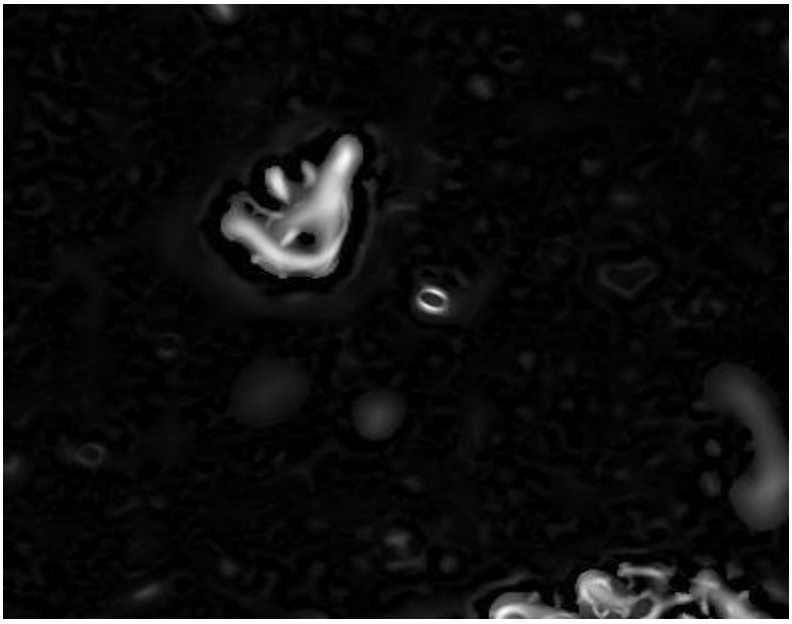
$$Precision = \frac{TP}{TP + FP}$$

TP (*eng. true positive*) predstavlja ukupan broj tačno pozitivnih testnih uzoraka, a FP (*eng. false positive*) ukupan broj lažno pozitivnih. Najbolju sliku o konačnom rezultatu daje mera koja se računa na način da se uzme prosek svih AP vrednosti počevši od praga prolaznosti jednakom 0.5 do 0.95 sa korakom od 0.05, što znači da se računa prosek 10 različitih vrednosti praga prolaznosti nad celim validacionim skupom. U sledećoj tabeli prikazani su statistički podaci o evaluaciji predikcije koristeći prethodno opisanu meru:

Metrika	Faster R-CNN	YOLOv5	Detectron2
AP	0.59	0.4	0.45

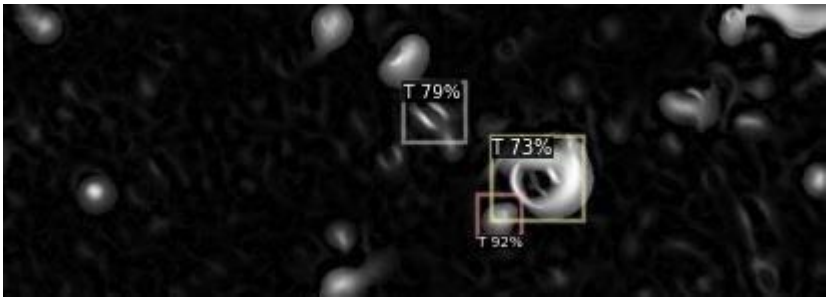
3.6 Filtriranje slike

Pošto prethodni rezultati nisu zadovoljavajući, prirodni sledeći korak je pokušaj filtriranja slike kako bi se spore naglasile, a ostali delovi nekako potisnuli. Isticanje spora pokušano je na nekoliko načina, raznim filtrima, ekvalizacijom histograma, a najbolje rezultate dao je Meijering filtar.



Slika 3.6 Slika filtrirana Meijering filtrom

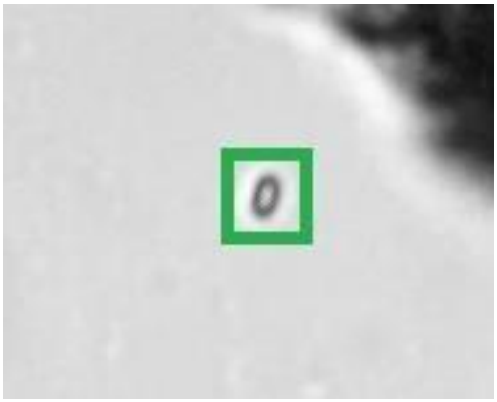
Ovaj filtar konkretno je implementiran za isticanje detalja na mikroskopskim slikama. Primer filtriranja ovim filtrom prikazan je na Slici 3.6. Spora nozema nalazi se u sredini i primetno je istaknuta u odnosu na ostale delove slike. Tako filtrirane slike korištene su kao trening skup Detectron2 modela na bazi Faster R-CNN arhitekture. Nakon obučavanja izvršeno je testiranje na slikama koje su takođe filtrirane ovim filtrom. Uprkos očekivanjima optimističnih rezultata, rezultati ni ovaj put nisu bili zadovoljavajući. Detektor nije detektovao puno objekata. Detektovao je nekoliko spora ali sa druge strane detektovao je i nekoliko objekata koji ni ne liče na spore. Jedan od primera loše detekcije prikazan je na Slici 3.7.



Slika 3.7 Primer detekcije Detectron2 modela nad filtriranom slikom

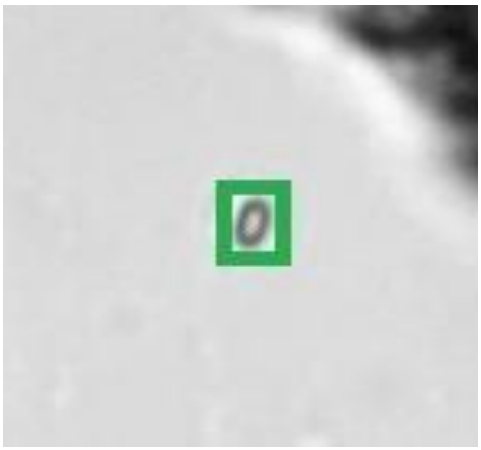
3.7 Korigovanje skupa podataka

S obzirom da filtriranje slika nije donelo očekivane rezultate, treba pogledati šta je srž problema loše detekcije. Pretpostavlja se da je uzrok to što skup podataka koji je korišten za treniranje nije dobro označen. U postupku označavanja labele su postavljene tako da obuhvataju i beli obruč oko spore nozema kao što je prikazano na Slici 3.8.



Slika 3.8 Primer loše labelovane spore

Čitav skup podataka za treniranje izmenjen je na takav način da su labele postavljene uz samu sporu, što se može videti na slici 3.9.



Slika 3.9 Primer dobro labelovane spore

Pored toga smanjene su dimenzije trening slika na 128x128, pa se svaka slika detaljnije obrađuje, odnosno postoji više slika u skupu podataka. Posledica toga je i što testne slike moraju da budu te veličine. To znači da sliku koju želimo da testiramo moramo da podelimo u blokove dimenzija 128x128. To će nam doneti bolju predikciju ali i sporiju obradu. Zbog toga što se najbolje pokazala u prethodnim razmatranjima, iskorištena je Faster R-CNN arhitektura na bazi Inception mreže za istraživanje problema detekcije nad izmenjenim skupom podataka.

4. Zaključak

Detekcija spora nozema jako je složen zadatak. Ovaj zadatak nije pogodan ni za ljude pa nije realno očekivati da nešto što je čak i ljudima teško, bude automatizovano. U ovom radu predstavljeno je nekoliko rešenja problema detekcije spora nozema na mikroskopskim slikama. Predstavljeni su rezultati izvršavanja implementacija raznih opisanih rešenja. Dat je uopšten pregled oblasti mašinskog učenja sa najpoznatijim modelima za detekciju objekata. Utvrđeno je da kvalitet labelovanja veoma utiče na konačan rezultat.

Najbitnija stvar prilikom rešavanja problema detekcije spora nozema jeste dobra priprema podataka. Skup oznaka, odnosno labela, treba da bude formiran od strane stručnjaka iz oblasti. Potrebno je svaku sporu labelovati linijama koje će ići uz njenu ivicu, a ne pravougaonim linijama. Nakon toga, treba istrenirati mrežu sa takvim skupom podataka, pa sve lažne predikcije označiti i svrstati u negativnu klasu. Zatim ponovo istrenirati mrežu da

detektuje i pozitivnu i negativnu klasu. To je veoma obiman posao koji je bio isproban prilikom izrade ovog rada, ali zbog ograničenog vremena odbačen je, međutim sigurno bi dao dobre rezultate te to treba pokušati u budućnosti.

Pošto su lažne detekcije uglavnom podsećale na prave spore, bilo bi dobro imati veći skup podataka na kojem bi arhitektura bolje naučila šta je prava spora nozema a šta ne. Pored toga moguće je istražiti koji modeli mreža su trenirani nad većim i sličnim skupom podataka, pa ih dodatno obučiti za ovaj problem. Takođe, bolji rezultati bi se mogli dobiti nastavkom eksperimentisanja sa različitim parametrima. Jedno potencijalno rešenje je kombinovanje dve ili više postojećih arhitektura u jedinstvenu arhitekturu, čuvajući naučene parametre za postojeće arhitekture, uz dodatno obučavanje.

Uz veći i dobro označeni skup podataka, fino podešen Faster R-CNN model na bazi Inception mreže, mogao bi se koristiti za automatsku detekciju spora nozema na mikroskopskim snimcima.