



UNIVERZITET U BEOGRADU,
ELEKTROTEHNIČKI FAKULTET

DIPLOMSKI RAD

Detekcija objekata bez dodatnog treniranja korišćenjem predobučenog CLIP modela

Autor

Mihailo Grbić, 2018/0042

Mentor

Doc. dr Predrag Tadić

Beograd, septembar 2022. godine

Sadržaj

1 Zahvalnica	1
2 Uvod	2
3 Metodologija rada	5
3.1 Transformer	5
3.1.1 Preprocesiranje	5
3.1.2 Sloj pažnje	6
3.1.3 Transformer arhitektura	7
3.2 Vision Transformer	8
3.3 CLIP	8
3.3.1 Arhitektura	8
3.3.2 Verzije	9
3.4 Algoritam za detekciju objekata	10
3.4.1 Sadržaj tekstualnog upita	10
3.4.2 Inicijalna filtracija klase	10
3.4.3 Detekcija objekata	12
3.4.4 Postprocesiranje detekcija	13
4 Rezultati i diskusija	15
4.1 Rezultati inicijalne filtracije klase	15
4.2 Rezultati detekcije objekata	17
4.3 Vreme izvršavanja algoritma	21
5 Zaključak	23

1 Zahvalnica

Zahvaljujem se profesorima Elektrotehničkog fakulteta u Beogradu i katedri Signali i sistemi na posvećenosti i uloženom trudu u protekle 4 godine. Posebno se zahvaljujem profesoru Predragu Tadiću na njegovim izuzetno kvalitetnim, preciznim i jasnim predavanjima, kao i na pomoći, savetima i sugestijama pri izradi i pisanju ovog diplomskog rada.

Zahvaljujem se takođe naučnicima i inženjerima firme *OpenAI* na puštanju predobučenog CLIP modela u javnost, bez kog ovaj projekat ne bi bio moguć.

2 Uvod

Jedan od najvećih napredaka u oblasti automatske obrade prirodnog jezika postignut je otkrićem transformer arhitekture [11] i njenog korišćenja u takozvanim “velikim jezičkim modelima” (eng. *Large Language Models*). Osnovna ideja velikih jezičkih modela (VJM) je predobučavanje modela na izuzetno velikim, automatski labeliranim, obučavajućim skupovima za zadatak modelovanja jezika. Na primeru modelovanja jezika uz maskiranje (eng. *masked language modeling*) reči u ulaznoj rečenici se nasumično zamaskiraju, a model se trenira da na osnovu ostatka rečenice predvidi koje reči su se originalno nalazile na zamaskiranim mestima [12]. Pošto ovaj zadatak ne zahteva ručno labeliranje i bazira se isključivo na tekstu, moguće je veoma brzo i jeftino kreirati ogroman trening skup, prikupljanjem rečenica iz knjiga, tekstova sa Wikipedije ili čak sa celog interneta [12]–[15]. Zadatak modelovanja teksta na ovako velikim i raznovrsnim skupovima podataka je veoma kompleksan pa modeli trenirani u ovu svrhu imaju izuzetno veliki broj parametara, po čemu su i dobili naziv veliki jezički modeli. VJM GPT-2 sa okvirno 1.5 milijardi parametara je i dalje podobučen na ovom zadatku [14], što je motivisalo razvoj sve većih modela kao što je na primer GPT-3 sa otprilike 175 milijardi parametara [15].

VJM se nakon predobučavanja mogu dotreniravati (eng. *fine-tune*) za neke ciljne nadgledane zadatke, koristeći relativno male, labelirane trening skupove. Ciljni zadatak, koji je različit od zadatka korišćenog pri predtreniranju i koji zapravo želimo da model obavlja se u literaturi naziva nizvodni zadatak (eng. *downstream task*). Glavna prednost VJM se ogleda u tome da dotreniran VJM postiže bolje rezultate u praktično svakom nizvodnom zadatku obrade jezika u poređenju sa modelima treniranim isključivo na labeliranim podacima [12]–[14].

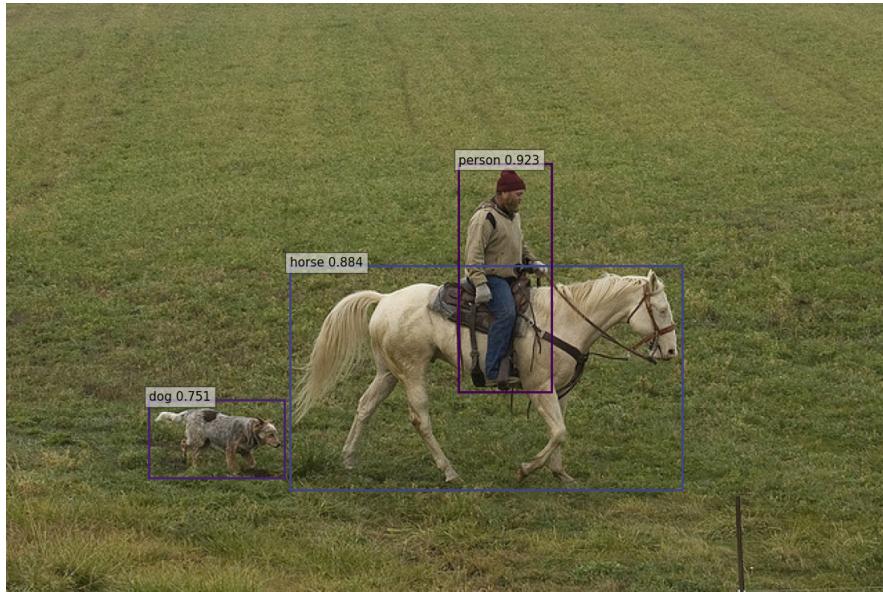
Štaviše, ispostavlja se da dovoljno veliki jezički modeli mogu da postignu kompetativne rezultate na mnogim zadacima obrade jezika bez ikakvog dotreniravanja. Parametnom konstrukcijom upita koji se prosleđuju modelu GPT-3 moguće je model koristiti za prevođenje rečenica, ispravljanje gramatičkih grešaka, generisanje novinskih članaka i u mnoge druge svrhe [15]. Kao rezultat nastala je jedna potpuno nova grana istraživanja u oblasti mašinskog učenja koja se u literaturi naziva inženjerstvo upita (eng. *prompt engineering*), koja za cilj ima da pametnom konstrukcijom upita koji se prosleđuju modelima omogući nove ili poboljša već postojeće funkcionalnosti modela. *Prompt engineering*-om su performanse *InstructGPT* modela [20] na zadatku aritmetičkog rezonovanja povećane sa 17.7% na 78.7% prostim dodavanjem rečenice “hajda da uradimo ovo korak po korak” (eng. “*let’s take this step by step*”) na početku odgovora *InstructGPT* modela [19].

Situacija u kojoj predtrenirani model koristimo za neki nizvodni zadatak bez ikakvog dotreniravanja se u literaturi naziva “*zero-shot transfer*” i kažemo da model obavlja nizvodni zadatak u “*zero-shot*” režimu. Glavna prednost modela namenjenih za *zero-shot* korišćenje jeste njihova fleksibilnost. Ako model može da obavi neki nizvodni zadatak bez dodatnog treniranja, onda je najčešće sposoban i da obavi brojne varijacije istog tog zadatka. Na primer, da bi model mogao da vrši *zero-shot* klasifikaciju, mora da bude sposoban da prepozna prethodno neviđene klase, što nam omogućava da isti

model koristimo za različite skupove klasa.

U cilju da se tehnike koje su dovele do revolucije u oblasti obrade jezika primene u oblasti kompjuterske vizije, Radford i dr. [18] predlažu novu arhitekturu CLIP, predtreniranu na velikom korpusu automatski labeliranih podataka. CLIP na ulazu prima proizvoljan broj slika i tekstova i za svaki par slika-tekst vraća verovatnoću da taj tekst opisuje tu sliku. Trening skup sadrži 400 miliona slika-tekst parova prikupljenih sa raznih izvora sa interneta. Ovako treniran model postiže kompetitivne rezultate u *zero-shot* režimu na velikom broju klasifikacionih skupova podataka, obuhvatajući zadatke kao što su prepoznavanje slova i brojeva, prepoznavanje akcija i klasifikacija objekata.

CLIP dokazuje da je princip kreiranja *zero-shot* modela predobučavanjem na velikim, automatski labeliranim trening skupovima primenljiv i u oblasti kompjuterske vizije. Međutim, CLIP je ograničen na zadatku titlovanja ili klasifikacije slike, tako da je sledeći logičan korak projektovanje *zero-shot* modela za neki kompleksniji zadatku kompjuterske vizije kao što je detekcija objekata. U zadatku detekcije objekata na slici se istovremeno može naći više objekata koje model treba da prepozna i lokalizuje. Za svaki detektovan objekat model vraća pretpostavljenu klasu objekta, pravougani okvir koji određuje poziciju i veličinu detektovanog objekta, kao i broj koji predstavlja sigurnost modela u tu detekciju (slika 1).



Slika 1: Primer detekcije objekata [5]

Uzimajući u obzir brojne funkcionalnosti koji su omogućene VJM tehnikama *prompt engineering-a*, postavlja se pitanje da li je uopšte potrebno kreirati novi model za *zero-shot* detekciju objekata ili je pametnim modifikacijama upita moguće koristiti CLIP u svrhe *zero-shot* detekcije objekata bez ikakvog dodatnog treniranja. Potencijal korišćenja CLIP-a za detekciju objekta prikazujemo kroz sledeći primer.

Ukoliko prosledimo sliku 2 CLIP modelu zajedno sa tekstualnim upitima “*a photo of a dog*”, “*a photo of a cat*” i “*a photo of a bird*” model vraća verovatnoće 0.613, 0.348 i 0.037. Pošto se na slici nalazi više objekata, model nije preterano siguran ni u jedan od ponuđenih tekstualnih opisa i vraća slične verovatnoće za klase pas i mačka. Međutim postepenim isecanjem slike oko mačke, uklanjamo psa i pticu sa slike i povećavamo sigurnost modela u klasifikaciju mačka. Potpunim isecanjem slike oko mačke (slika 3b) i prosleđivanjem takve slike modelu, uz isti tekstualni upit, sigurnost modela u klasifikaciju mačka skače na 0.997. Primenom istog postupka na psa i pticu, sigurnost modela za labelu pas skače na 0.99 za sliku 3a, a sigurnost modela za klasu ptica dostiže 0.999 za sliku 3c.



Slika 2: Originalna slika [5]



(a) *Crop-ovan pas*

(b) *Crop-ovana mačka*

(c) *Crop-ovana ptica*

Slika 3: Prikaz potencijala CLIP-a za detekciju objekata

U ovom radu je predstavljen novi algoritam koji koristi predtrenirani CLIP model, originalno namenjen za *zero-shot* titlovanje slika i vrši zadatak detekcije objekata u *zero-shot* režimu. Predstavljeni algoritam se bazira na očekivanju da će isecanje slike oko idealnog pravougaonog okvira objekta, koji obuhvata taj objekat u celosti i ne sadrži delove drugih objekata, ujedno i maksimizovati sigurnost CLIP-a u klasifikaciju tog objekta. Ukoliko je to očekivanje tačno, idealni pravougani okvir nekog objekta se može naći lokalnom pretragom, tako što se svaki pravougaoni okvir objekta ocenjuje sigurnošću koju CLIP vraća za klasifikaciju tog objekta pri prosleđivanju slike isećene oko trenutno razmatranog pravouganog okvira.

3 Metodologija rada

3.1 Transformer

3.1.1 Preprocesiranje

U slučaju transformera koji obrađuju tekstove potrebno je najpre odrediti mapiranje iz reči prirodnog jezika u vektore fiksne dužine koje transformer može da obradi. Prvi korak u određivanju ovog mapiranja je konstrukcija rečnika na osnovu trening skupa. Elementi konstruisanog rečnika se u nazivaju tokenima, a algoritam koji konstruiše rečnik zovemo algoritmom tokenizacije. Naivnim razmatranjem svake pronadene reči kao posebnog tokena bi se rečnik veoma brzo napunio rečima koje imaju drugačiji zapis, ali slično semantičko značenje kao na primer "starost", "starac" i "starica". Model bi takođe morao kroz obučavanje da nauči semantičku sličnost između ovih tokena, umesto da ona bude automatski jasna iz procesa tokenizacije. Zato se za tokene najčešće uzimaju podreči umesto celih reči, pa bi se na primer reč "starost" sastojala od tokena "star" i "ost". Postoji nekoliko široko korišćenih algoritama tokenizacije na nivou podreči, svaki sa određenim prednostima i manama [6], [10]. U slučaju CLIP-a korišćen je *Byte-Pair Encoding* (BPE) algoritam tokenizacije [6]. BPE najpre u rečnik dodaje po jedan token za svaki karakter prisutan u trening skupu, nakon čega spaja svaka dva tokena iz rečnika u parove i u rečnik dodaje onaj par koji se najčešće pojavljuje u skupu. Prethodni korak se ponavlja sve dok se rečnik ne ispuni.

Konstrukcija rečnika se vrši pre samog treniranja, nakon čega se rečnik koristi da bi se preprocesirala svaka rečenica koja se prosleđuje modelu. Na početak i kraj sekvence se u preprocesiranju mogu dodati specijalni tokeni koji označavaju početak ili kraj sekvence, u literaturi često označeni kao [SOS] i [EOS] za "*start of sequence*" i "*end of sequence*". Uloga ovih tokena će biti objašnjena u odeljku 3.1.3.

Svaki token iz rečnika se mapira u odgovarajući vektor dužine d_e u literaturi poznat kao tekstualni embedding (eng. *text embedding*). Vrednosti svakog tekstualnog embeddinga su parametri modela koji se uče kroz treniranje. Za razliku od rekurentnih i konvolucionih arhitektura, transformer nema informaciju o redosledu tokena u sekvenci koja mu se prosleđuje. Da bi se ova informacija prosledila arhitekturi definišu se pozicioni embedinzi (eng. *positional embeddings*), vektori dužine d_e koji se sabiraju element po element sa tekstualnim embedinzima. Za svaku moguću poziciju tokena u ulaznoj sekvenci postoji odgovarajući pozicioni embedding i isto kao u slučaju tekstualnih embeddinga, pozicioni embedinzi predstavljaju parametre koji se uče prilikom obučavanja. Pre prosleđivanja modelu svaki tekstualni embedding se na osnovu svoje pozicije u sekvenci sabira sa odgovarajućim pozicionim embeddingom.

Celokupan proces preprocesiranja rečenice pre ulaza u model se sastoji iz sledećih koraka: deljenje rečenice u tokene na osnovu rečnika, dodavanje specijalnih tokena na početak i/ili kraj rečenice, mapiranje tokena u tekstualne embeddinge i sabiranje tekstualnih embeddinga sa odgovarajućim pozicionim embedinzima.

3.1.2 Sloj pažnje

Glavni element transformer arhitekture je takozvani sloj pažnje (eng. *Attention layer*). Ulazne i izlazne dimenzije sloja pažnje su identične. Sloj pažnje na ulazu prima sekvencu proizvoljne dužine n , vektora dimenzija d_e : (x_1, x_2, \dots, x_n) , a na izlazu vraća sekvencu iste dužine n , vektora dimenzija d_e : $(x'_1, x'_2, \dots, x'_n)$. Iako n u teoriji može biti proizvoljne dužine, ova vrednost je u praksi ograničena i fiksirana na maksimalnu moguću dužinu sekvence n_{max} , kako bi se omogućila obrada u *batch*-evima. U slučaju da sloju pažnje želimo da prosledimo sekvencu kraću od n_{max} , onda ulaznu matricu dimenzija $n_{max} \times d_e$ dopunjavamo podrazumevanim vrednostima (uglavnom nulama) i ne uzimamo ih u obzir prilikom računjanja izlaza iz sloja pažnje.

Na svaki od ulaznih vektora x_i se primenjuju linearne transformacije W_q , W_k , W_v koje se uče, rezultujući sa 3 vektora $q_i = W_q x_i$, $k_i = W_k x_i$, $v_i = W_v x_i$, dužine d_q , d_k i d_v , sa napomenom da važi $d_q = d_k$. Vektori q , k i v se nazivaju vektorima “upita” (eng. *query*), “ključa” (eng. *key*) i “vrednosti” (eng. *value*). Za svaki vektor upita q_i računamo njegov skalarni proizvod sa svakim vektorom ključa k_j , tako dobijene vrednosti skaliramo sa $\frac{1}{\sqrt{d_q}}$ i na kraju ih normalizujemo *softmax* funkcijom, tako da se vrednosti sumiraju u 1 za fiksirano q_i . Ovako dobijene vrednosti $c_{i,j} = softmax(q_i \cdot k_j / \sqrt{d_q})$ se nazivaju vrednostima kompatibilnosti između upita q_i i ključa k_j . Skalarni proizvod $q_i \cdot k_j$ se skalira sa $\frac{1}{\sqrt{d_q}}$, kako logiti u *softmax* funkciji ne bi postali preveliki sa povećanjem dimenzije d_q , što bi rezultovalo malim vrednostima gradijenta i usporilo učenje.

Vektor pažnje a_i se računa kao suma svih vektora vrednosti v_j otežinjena vrednostima kompatibilnosti između upita i i ključa j .

$$a_i = \sum_{j=1}^n c_{i,j} v_j$$

Ukoliko sve ulazne vektore predstavimo matricom $X = [x_1^T; x_2^T; \dots; x_n^T]$ i vektore upita, ključa i vrednosti predstavimo matricama $Q = XW_q^T$, $K = XW_k^T$ i $V = XW_v^T$, postupak sloja pažnje se svodi na sledeću jednačinu:

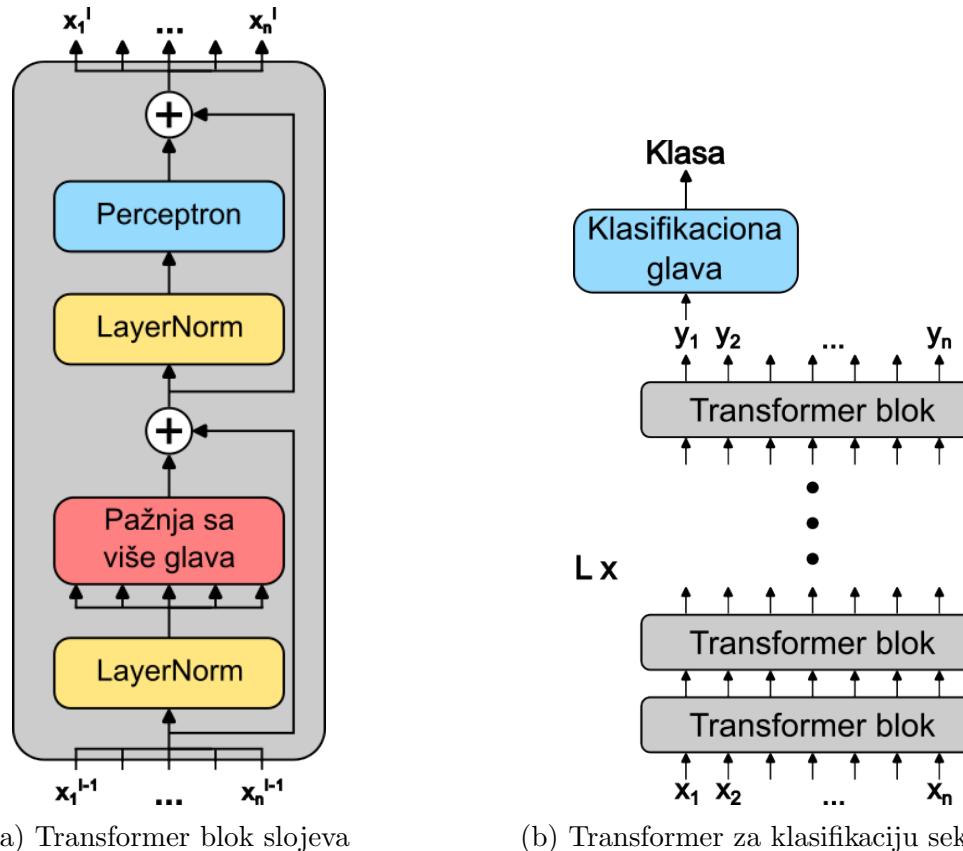
$$A = softmax \left(\frac{QK^T}{\sqrt{d_q}}, axis = 1 \right) V$$

Prethodno opisan postupak možemo ponoviti H puta za H različitih projekcionih matrica W_q , W_k i W_v , rezultujući sa po H vektora a_i , za svaki ulazni vektor x_i . U tom slučaju jedan skup matrica W_q , W_k i W_v zovemo jednom “glavom” pažnje a celokupan proces nazivamo pažnjom sa više glava (eng. *multi-head attention*). Na kraju sve vektore a_i^1 , a_i^2 , ... a_i^H konkateniramo i linearno projektujemo u vektor x'_i dimenzije d_e , koji predstavlja izlaz sloja pažnje na poziciji i .

3.1.3 Transformer arhitektura

U literaturi postoji nekoliko varijacija transformer arhitekture, a u ovom radu će biti opisana ona koja je relevantna za CLIP arhitekturu. Celokupna transformer arhitektura se sastoji od L identičnih, sekvencijalno povezanih blokova slojeva. Ulazne i izlazne dimenzije svakog bloka slojeva su identične i iste kao i kod sloja pažnje $n \times d_e$ ili preciznije $n_{max} \times d_e$. Na slici 4a je dat pregled jednog bloka.

Na svakom ulaznom vektoru x_i se najpre primenjuje *layernorm* operacija [7] koja taj vektor standardizuje, tako da mu je srednja vrednost 0 i varijansa 1. *Layer-norm* operacija čini treniranje modela stabilnijim i bržim. Standardizovani vektori se propuštaju kroz sloj pažnje sa više glava, nakon kog postoji rezidualna *skip* konekcija [8] sa ulaza u blok. Sličan postupak se zatim ponavlja u drugoj polovini bloka, osim što se umesto globalnog sloja pažnje sa više glava na svaki vektor ponaosob primenjuje perceptron sa dva sloja, čije su ulazne i izlazne dimenzije iste i jednake d_e . Između prvog i drugog sloja perceptrona se primenjuje aktivaciona funkcija koja može biti RELU ili GELU [9].



Slika 4: Prikaz transformer arhitekture

Dakle transformer sekvenci ulaznih vektora (x_1, x_2, \dots, x_n) mapira u sekvencu transformisanih vektora (y_1, y_2, \dots, y_n) . Ukoliko želimo da model obavlja zadatak klasifi-

kacije, na izlazu iz transformera dodajemo perceptron koji se naziva klasifikaciona glava (eng. *classification head*). U slučaju klasifikacije tokena, klasifikaciona glava mapira svaki izlazni vektor y_i u verovatnoće pripadnosti klasama. Međutim, u slučaju klasifikacije sekvene nezgodno je napraviti klasifikacionu glavu koja može da obradi sekvene proizvoljne dužine. U tu svrhu se tokom preprocesiranja na početak i/ili kraj sekvene dodaju specijalni [SOS] i [EOS] tokeni. Model se obučava da informacije o celoj sekvenci koduje u izlazni vektor y_1 ili y_n koji odgovara poziciji [SOS] ili [EOS] tokena, pa se glava za klasifikaciju sekvene primenjuje samo na taj izlazni vektor (slika 4b).

3.2 Vision Transformer

Primena transformera nije ograničena samo na tekstualne podatke. Glavna ideja iza *Vision Transformer*-a [16] je primena transformera arhitekture u kompjuterskoj viziji. Celokupna arhitektura modela je ista kao u slučaju transformera za obradu teksta, jedina razlika postoji u načinu preprocesiranja ulaza. Neka je ulazna slika dimenzija $H \times W \times C$ gde je H visina slike, W širina slike, a C broj kanala. Ulaznu sliku delimo na isečke dimenzija $P \times P \times C$ i svaki isečak ispravljamo u jednodimenzionalni vektor dužine $P * P * C$. Ovako ispravljene jednodimenzione vektore jednim linearnim slojem projektujemo u vektore dimenzije d_e . Ovako nastali vektori se nazivaju embedinzi isečaka (eng. *patch embeddings*) i predstavljaju ekvivalent tekstualnim embedinzima. Isto kao i kod obrade teksta, embedinzi isečaka se sabiraju sa pozicionim embedinzima i zatim propuštaju kroz model. Na početku svake sekvene se takođe dodaje specijalni [SOS] token. Izlazni vektor koji odgovara poziciji tog tokena predstavlja vektor obeležja ulazne slike.

Postoji nekoliko različitih *Vision Transformer* modela, a ovde će biti opisane 3 verzije koje su korišćene u CLIP modelu, konkretno ViT-B/32, ViT-B/16 i ViT-L/14. ViT-B označava ViT-Base model koji ima $L = 12$ bloka slojeva, dužinu embedding vektora $d_e = 768$, po $H = 12$ glava pažnje u svakom sloju i po 3072 skrivena neurona u dvoslojnim perceptronima. ViT-L označava ViT-Large model koji ima $L = 24$ bloka, dužinu embedding vektora $d_e = 1024$, $H = 16$ glava pažnje i 4096 skrivenih neurona u dvoslojnim perceptronima. Sva 3 modela koriste GELU aktivacionu funkciju u dvoslojnim perceptronima. Broj nakon kose crte u imenu modela ViT-B/32 označava da je rezolucija isečka 32×32 piksela. Manja rezolucija isečka povlači veću dužinu sekvene što povećava računsku složenost i vreme izvršavanja.

3.3 CLIP

3.3.1 Arhitektura

CLIP arhitektura na ulazu prima proizvoljan broj slika i proizvoljan broj tekstualnih opisa, pa za svaki par slika-tekst vraća verovatnoću da taj opis odgovara toj

slici. Glavni elementi CLIP arhitekture su dva modela koji se zovu enkoder slike i enkoder teksta i koji za ulazne slike i tekstove generišu njihova obeležja.

Kao tekstualni enkoder se koristi transformer za obradu teksta, opisan u odeljku 3.1 sa $L = 12$ blokova, $d_e = 512$ dimenzijom embeddinga i po $H = 8$ glava pažnje. Rečnik je konstruisan BPE algoritmom tokenizacije i sadrži 49 152 tokena. Maksimalna dužina sekvenca n_{max} je ograničena na 76 tokena. Ulazna sekvenca je okružena [SOS] i [EOS] specijalnim tokenima, a izlazni vektor na poziciji [EOS] se tretira kao vektor obeležja ulaznog teksta t_f .

Za enkoder slike se koristi *Vision Transformer* opisan u odeljku 3.2. Slike se pre deljenja na isečke, uz pomoć bikubne interpolacije [1], reskaliraju na rezoluciju 224×224 ili 336×336 u zavisnosti od verzije CLIP modela o kojima će biti reči u odeljku 3.3.2. Kao što je već rečeno, na početak sekvence isečaka slike se dodaje [SOS] token, a izlazni vektor na poziciji tog tokena predstavlja vektor obeležja slike i_f .

Vektori t_f i i_f se linearnim transformacijama W_t i W_i projektuju u vektore istih dužina t_e i i_e . Logit verovatnoće da tekst odgovara slici se određuje kao kosinusna sličnost između odgovarajućih vektora t_e i i_e skalirana sa eksponentom temperature τ . Finalne verovatnoće se dobijaju primenom *softmax* funkcije na logite između jedne slike i svih tekstova, tako da se sve verovatnoće za jednu sliku sumiraju u 1.

3.3.2 Verzije

Postoje 4 javno dostupne verzije predobučenog CLIP modela. Sve 4 verzije koriste isti tekstualni enkoder opisan u odeljku 3.3.1, ali se razlikuju po enkoderu slike koji koriste. 3 verzije koriste *Vision Transformer*-e opisane u odeljku 3.2 tj. ViT-B/32, ViT-B/16 i ViT-L/14. Sva tri modela primaju sliku dimenzija 224×224 piksela, tako da se pre ulaza u model svaka slika reskalira na tu rezoluciju. Najbolje performanse su postignute za ViT-L/14 model koji je zatim dotreniravan još jednu dodatnu epohu (prethodna 3 modela su obučavana kroz 32 epohe) na slikama veće rezolucije 336×336 što se pokazuje kao dobra praksa u obradi slike i što je dodatno povećalo performanse modela. Četvrti javno dostupni CLIP model se označava kao CLIP-ViT-L/14@336px.

Ovi modeli su testirani na 27 nizvodnih zadataka klasifikacije slike i na osnovu usrednjavanja tih rezultata su sortirani po performansama. Kao i što se moglo očekivati performanse koreliraju sa računskom kompleksnošću modela. Najbrži, ali ujedno i najgori po rezultatima je CLIP-ViT-B/32, pa CLIP-ViT-B/16, CLIP-ViT-L/14 i na kraju CLIP-ViT-L/14@336px, koji u proseku postiže najbolje rezultate ali se i najsporije izvršava. Najveća razlika između modela je uočena na *BirdSnap* skupu podataka [4], na kom najgori model postiže rezultat od 58.5, dok najbolji postiže rezultat od 79.9.

Pošto je zarad detekcije objekata kroz model potrebno propustiti veliki broj slika, za rezultate prikazane u ovom radu je uglavnom korišćen najbrži model sa najgorim performansama CLIP-ViT-B/32. Takođe je rađena dodatna analiza potencijalnog poboljšanja performansi algoritma za detekciju objekata pri korišćenju drugih CLIP modela.

3.4 Algoritam za detekciju objekata

Zadatak algoritma za detekciju objekata je da na slici detektuje postojanje i položaj objekata iz nekog skupa klasa. Algoritam na ulazu prima sliku i skup imena klasa koje treba da detektuje, a na izlazu vraća proizvoljan broj detekcija koje se sastoje od imena klase objekata koji je detektovan, položaj i veličina pravougaonika koji okružuje objekat na slici i broja od 0.5 do 1 koji predstavlja sigurnost modela u tu detekciju (detekcije sa sigurnošću ispod 0.5 se ne razmatraju). Celokupan algoritam se može grubo podeliti u 3 koraka:

1. Inicijalna filtracija klasa
2. Detekcija objekata
3. Postprocesiranje detekcija

3.4.1 Sadržaj tekstualnog upita

Prilikom testiranja CLIP modela za brojne zadatke klasifikacije, otkriveno je da je konstrukcija rečenica koje se prosleđuju modelu izuzetno bitna za performanse tog modela. Jedan konkretni problem koji je pomenut u originalnom radu je da se u trening skupu CLIP-a relativno retko opis slike sastojao od samo jedne reči, već se uglavnom sastojao od potpunih rečenica. Radford i dr. [18] su ustanovili da su rezultati CLIP-a bolji i stabilniji ukoliko se modelu prosleđuje niz rečenica oblika “*a photo of a* *ime klase*” umesto niza koji se sastoji samo od imena klase. Isti način konstruisanja upitnih rečenica je korišćen i u ovom radu. Konkretno kada kažemo da ispitujemo klasu X ili da je tekstualni upit sadržao klasu X , to znači da je jedna od prosleđenih rečenica bila “*a photo of a* X ”.

U ovom radu nije razmatrano korišćenje više alternativnih tekstualnih upita koji se zasnivaju na sinonimima ili korišćenje tekstualnih upita koji obuhvataju više klase istovremeno. Istraživanje ovakvog pristupa je ostavljeno za budući rad.

3.4.2 Inicijalna filtracija klasa

Prvi korak algoritma je inicijalna filtracija klasa, koja za cilj ima da ispitivane klase razdvaja u one koje se potencijalno nalaze na ulaznoj slici i one koje se na njoj ne nalaze. U daljem tekstu će se dva rezultujuća skupa inicijalne filtracije nazivati obećavajuće klase i negativne klase. Testirano je više načina inicijalne detekcije, od kojih je najprostiji prosleđivanje cele, nepromenjene, ulazne slike zajedno sa tekstualnim upitom za sve ispitivane klase. Na verovatnoće koje model vrati se zatim primenjuje prag t_{ini} , tako da su klase čije su verovatnoće iznad praga obećavajuće, a sve ostale negativne. Glavni problem sa ovim pristupom je činjenica da jaka prisustnost jedne ili

nekoliko klasa može negativno da utiče na verovatnoće prisustnosti drugih klasa. Na primeru slike 5, model može da dodeli toliko veliku verovatnoću klasi bicikl, da sve ostale padnu ispod praga, iako se na slici nalaze drugi objekti kao što su automobil i kamion. Za ovaj problem predlažemo 2 rešenja u vidu modifikacija na prosti način filtracije.



Slika 5: Slika iz COCO skupa podataka [5]

Prvo rešenje nazivamo “inicijalna filtracija sa ponavljanjem” i ono se sastoji od izbacivanja verovatnoća koje su iznad praga, normalizacija preostalih verovatnoća i ponovna primena praga na njih. Tako da ukoliko jedna ili nekoliko klasa imaju preveliku verovatnoću, one će biti proglašene za obećavajuće i izbačene iz daljeg razmatranja, nakon čega se opet primenjuje prag što daje priliku i drugim klasama da budu proglašene za obećavajuće. Broj ponavljanja je ograničen na 3, jer bi za neograničen broj vrlo lako došlo do situacije u kojoj su nakon velikog broja ponavljanja sve klase proglašene za obećavajuće. Pošto se određivanje obećavajućih klasa vrši u više iteracija prag mora biti veći od praga u prostoj filtraciji. Problem sa ovim pristupom može nastati ukoliko se na slici nalazi veliki broj klasa i model podjednako raspodeli verovatnoće među njima, što može da rezultuje situacijom u kojoj nijedna od klase nije iznad praga. Međutim očekujemo da su ovakve situacije retke.

Drugo rešenje nazivamo “segmentacija do $S \times S$ ” i predstavlja deljenje ulazne slike na segmente i prosleđivanje tih segmenata modelu. Slika se deli na 2×2 , 3×3 , ... $S \times S$ isečaka i onda se modelu kao slikovni upit prosleđuje originalna slika zajedno sa svim isećcima. Model sada vraća verovatnoće prisustnosti klase za svaku ulaznu sliku, ali pošto je dovoljno da objekat bude prisutan samo na jednom segmentu slike, kao

finalna verovatnoća prisutnosti klase se uzima maksimalna od svih tih verovatnoća. Sa ovim pristupom, ukoliko se na slici nalazi manji objekat pored nekog većeg, izraženijeg objekta, računamo da će na jednom od segmenata manji objekat biti izraženiji od većeg. Glavni problem sa ovim rešenjem je mogućnost da na manjim segmentima, usled niske rezolucije, model pogrešno detektuje prisustvo neke klase. Ovaj pristup je takođe sporiji i računski složeniji, zato što sada model mora da obradi više slika.

Filtracija sa ponavljanjem i segmentacija do $S \times S$ su međusobno nezavisna poboljšanja na prostu inicijalnu filtraciju i mogu se primeniti istovremeno.

3.4.3 Detekcija objekata

Nakon što je urađena inicijalna filtracija klasa iterira se kroz sve obećavajuće klase i za svaku razmatranu klasu se primenjuje lokalna pretraga za detekciju objekata. Jedan konkretan pravougaoni okvir ocenjujemo tako što isečemo sliku oko tog pravougaonog okvira i zatim isečenu sliku prosledimo CLIP-u zajedno sa nizom tekstova od kojih se jedan odnosi na trenutno razmatranu klasu. Verovatnoća koju model vrati za taj tekst predstavlja vrednost ciljne funkcije za razmatrani pravougani okvir.

Najpre se konstruiše tekstualni upit koji će se koristiti za detekciju. U ovom radu su testirane 4 strategije za konstrukciju tekstualnog upita. U sve 4 strategije jedna od ulaznih rečenica se odnosi na trenutno razmatranu klasu, dok strategije variraju u ostalim rečenicama koje se prosleđuju modelu.

U strategiji “negativni tekst”, ostatak tekstualnog upita čine rečenice koje označavaju manjak prisustva neke klase. Neke od korišćenih rečenica su “other”, “background”, “a photo of nothing” i “a photo of clutter”. Mana ovog pristupa je da čak i kada se na slici ne nalazi objekat, model daje nisku verovatnoću ovim negativnim opisima, najverovatnije zato što nije postojalo preterano puno sličnih opisa u trening skupu.

U strategiji “negativne kategorije” ostatak upita čine kategorije koje su proglašene za negativne u inicijalnoj filtraciji tj. one za koje prepostavljamo da nisu prisutne na slici. Ideja iza ovog pristupa je da će za loše isečke koje ne sadrže objekte razmatrane klase model dati sličnu verovatnoću razmatranoj klasi i negativnim klasama, dok će za dobre okvire verovatnoća razmatrane klase biti veća. Problem koji može nastati sa ovom strategijom je situacija u kojoj model povezuje okolinu sa objektom, čak iako se objekat zapravo ne nalazi u toj okolini. Na primer ukoliko pravougani okvir okružuje nebo ali ne i pticu, model i dalje može dati veliku verovatnoću za klasu ptica prosti zato što povezuje slike neba sa pticama.

U strategiji “obećavajuće kategorije” ostatak upita čine kategorije koje su proglašene za obećavajuće u inicijalnoj filtraciji tj. one za koje slutimo da jesu prisutne na slici. Ideja ove strategije je da bi u teoriji trebalo da bude bolja za lokalizaciju objekata nego ostale strategije. Ukoliko je jedan objekat blizu nekog drugog objekta različite klase očekujemo da će pri ovoj strategiji ciljna funkcija imati najveću vrednost kada pravougaoni okvir sadrži što manje delova drugog objekta. Glavna mana ove strategije

je što broj upitnih rečenica zavisi od broja obećavajućih klasa, koji dosta varira od slike do slike, što može prouzrokovati nestabilnost i neočekivano ponašanje.

Poslednja strategija se zove “sve kategorije” i pri njoj se tekstualni upit sastoji od svih ispitivanih kategorija. Ova strategija predstavlja kompromis između prethodne dve strategije.

Nakon što je konstruisan tekstualni upit, postavljaju se 3×3 inicijalna pravougana okvira, koji dele ulaznu sliku na 9 jednakih delova. Svaki inicijalni okvir predstavlja jedan mogući početak lokalne pretrage. Za svaki inicijalni okvir se računa ciljna funkcija i ukoliko ona prelazi određeni prag započinje se lokalna pretraga sa tim inicijalnim okvirom. Ukoliko za nijedan od inicijalnih okvira ciljna funkcija nije iznad praga, lokalna pretraga se započinje samo jednom za onaj inicijalni okvir sa najvećom ciljnom funkcijom.

Lokalna pretraga se vrši algoritmom pretrage po snopu. Jedinka predstavlja jedan pravougaoni okvir i opisuje se sa 4 parametara. U svakom koraku svaka jedinka iz populacije generiše po b novih jedinki nasumičnim mutacijama njenih parametara. Konkretno na svaki parametar dodajemo uniformno raspodeljenu nasumičnu vrednost iz opsega $(-30, +30)$. Sve jedinke zatim ocenjujemo ciljnom funkcijom i zadržavamo samo p najboljih. Prethodni korak se ponavlja sve dok ne dostignemo maksimalni broj od 200 koraka ili dok se u 10 koraka ciljna funkcija najbolje jedinke ne poveća za 0.001. Na kraju lokalne pretrage najbolju jedinku tj. pravougaoni okvir zadržavamo kao detekciju ako je vrednost njegove ciljne funkcije iznad 0.5.

3.4.4 Postprocesiranje detekcija

Detekcije koje dobijamo na kraju prethodnog koraka i dalje nisu idealne, pa zato nad njima vršimo dodatan korak postprocesiranja. Jedan konkretan problem koji može da se javi je da se isti objekat detektuje više puta kao što je to prikazano na slici 6. Zato se prilikom postprocesiranja primenjuje metod potiskivanja nemaksimuma (eng. *non-maximum suppression*). U potiskivanju nemaksimuma iterativno uzimamo detekciju sa najvećom sigurnošću i stavljamo je u skup finalnih detekcija. Kada jednu detekciju proglašimo za finalnom, iz originalnog skupa detekcija brišemo sve one koje su iste klase kao proglašena detekcija i kojima je *intersection over union* (IoU) sa proglašenom detekcijom veći od nekog praga. Taj prag je u ovom radu postavljen na 0.3. IoU je vrednost količnika površine preseka dva pravougaona okvira i površine unije tih okvira. Što se više pravougani okviri preklapaju to je IoU bliži jedinici, a što se manje preklapaju to je on bliži nuli.



Slika 6: Primer više detekcija za jedan objekat [5]

Čak i nakon potiskivanja nemaksimuma algoritam detekcije pravi mnogo više detekcija nego što je zapravo prisutno na slici. Često se dešava da se modelu pričini neki objekat na slici i da u procesu lokalne pretrage pronađe pravougani okvir za koji je ciljna funkcija veća od 0.5, što rezultira detekcijom. Olakšavajući faktor je to što je sigurnost takvih detekcija u većini slučajeva manja nego sigurnost u detekcije koje se zapravo nalaze na slici. Da bi se ovaj problem otklonio pored potiskivanja nemaksimuma se vrši dodatno odbacivanje detekcija i reskaliranje sigurnosti. Sve detekcije sa sigurnošću ispod praga t_{post} se odbacuju, a preostalima se sigurnost skalira tako da bude između 0.5 i 1 po sledećoj formuli.

$$s_{nova} = 0.5 + 0.5(s_{stara} - t_{post})/(1 - t_{post})$$

Ovaj prag je drugačiji za svaku strategiju konstrukcije tekstuallnog upita i određen je tako da prosečan broj detekcija na slici bude 50% veći nego prosečan broj objekata po slici u validacionom skupu.

4 Rezultati i diskusija

Algoritam za *zero-shot* detekciju objekata je testiran na COCO skupu podataka za detekciju i segmentaciju koji sadrži ukupno 80 raznovrsnih klasa objekata [5]. Korišćen je samo validacioni skup iz 2017. godine sa ukupno 5000 slika.

4.1 Rezultati inicijalne filtracije klasa

Korak inicijalne filtracije zapravo rešava konkretni problem mašinskog učenja poznat kao klasifikacija sa više labela (eng. *multi-label classification*) u kom svaka instanca može imati više labela istovremeno. U našem slučaju svaka slika može sadržati više različitih klasa a cilj inicijalne filtracije je da detektuje te klase. Samim tim se različite metode inicijalne filtracije mogu evaluirati odvojeno od ostatka algoritma, evaluacijom njihovih performansi na zadatku klasifikacije sa više labela. Pošto je inicijalna filtracija samo prvi korak u detekciji objekata nije potrebno da prisutne klase budu savršeno detektovane u ovom koraku, već smo spremni da žrtvujemo preciznost zarad osetljivosti. Idealan način da poredimo različite metode inicijalne filtracije je uz pomoć ROC krive (eng. *reciever operating characteristic curve*) [2], međutim konstrukcija ROC krive za metode koje koriste poboljšanje “sa ponavljanjem” je komplikovano, zato što se filtracija vrši u više koraka, tokom kojih se sigurnosti menjaju. Zato se metode inicijalne filtracije porede na malo prostiji način. Prosečan broj različitih klasa prisutnih na slici na COCO validacionom skupu je okvirno 3. Za svaki razmatrani metod inicijalne filtracije određena su 3 praga t_{ini} za koje metod u proseku detektuje po 6, 12 i 18 klasa na slici. Ove pragove redom nazivamo visokim, srednjim i niskim pragom. Performanse određene metode inicijalne filtracije merimo kao mikro usrednjenu osetljivost te metode za visoki, srednji i niski prag. U tabelama 1 i 2 su dati rezultati različitih metoda inicijalne filtracije.

Bez ponavljanja				
Mikro osetljivost	Prosti metod	Segmentacija do 3×3	Segmentacija do 4×4	Segmentacija do 5×5
Visoki prag	0.6018	0.6843	0.6979	0.7025
Srednji prag	0.7157	0.8028	0.8151	0.8216
Niski prag	0.7861	0.8633	0.8769	0.8829

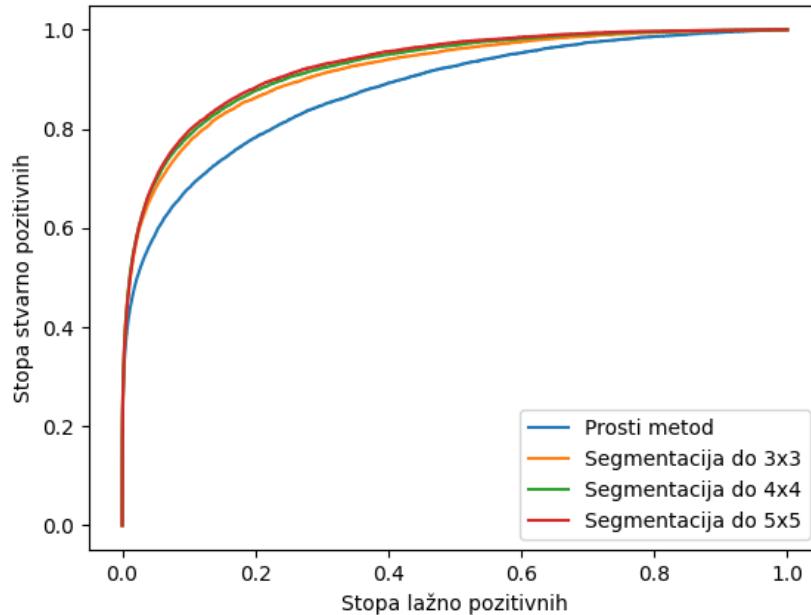
Tabela 1: Rezultati metoda inicijalne filtracije bez ponavljanja

Sa ponavaljanjem				
Mikro osetljivost	Prosti metod	Segmentacija do 3×3	Segmentacija do 4×4	Segmentacija do 5×5
Visoki prag	0.6237	0.6969	0.7037	0.7040
Srednji prag	0.7582	0.8317	0.8435	0.8463
Niski prag	0.8274	0.8920	0.9008	0.9049

Tabela 2: Rezultati metoda inicijalne filtracije sa ponavljanjem

Uočava se da ukoliko je jedan metod bolji od drugog za jednu vrednost praga onda je on bolji i za ostale 2 vrednosti praga. Takođe se primećuje da poboljšanja “segmentacija do $S \times S$ ” i “ponavljanje” oba povećavaju performanse inicijalne filtracije i da performanse rastu sa povećanjem broja segmenta. Najbolje rezultate ima metoda segmentacija do 5×5 sa ponavljanjem, praćena metodom segmentacije do 4×4 sa ponavljanjem. Pošto su im performanse bliske i pošto je prva od ove dve metode skoro duplo računski složenija (prva obrađuje 55, a druga 30 slika) u ostatku rada je korišćena metoda segmentacije do 4×4 sa ponavljanjem i nju nazivamo podrazumevanom metodom inicijalne filtracije. Za podrazumevanu vrednost praga je uzet odgovarajući niski prag koji iznosi $t_{ini} = 0.1805$.

Na grafiku 7 su date ROC krive metoda koje ne koriste poboljšanje “sa ponavljanjem”. Površina ispod krive za prosti metod iznosi 0.877, dok su površine ispod metoda sa segmentacijom do 3×3 , 4×4 i 5×5 redom 0.917, 0.925 i 0.928, što dodatno potvrđuje prethodno donesene zaključke.



Slika 7: Grafik ROC krivih za različite metode inicijalne filtracije

4.2 Rezultati detekcije objekata

Pošto se algoritam za detekciju objekata dugo izvršava, evaluacija je rađena na nasumično odabranom podskupu od 100 slika iz COCO validacionog skupa. Rezultate prikazujemo u vidu 6 metrika koje se zasnivaju na najzastupljenijoj metrići za evaluaciju detekcije objekata - *mean average precision* (mAP) [3]. Detekcija se proglašava tačnom ukoliko je IoU između pravouganog okvira detekcije i labele veći od praga t_{IoU} . Dve razmatrane metrike postavljaju ovaj prag na 0.5 i 0.75 i označavaju se sa AP_{blago} i AP_{strog} . Preostale 4 metrike primenjuju 10 razlicitih pragova t_{IoU} od 0.5 do 0.95 sa korakom 0.05 i tako izračunate mAP vrednosti usrednjavaju. Glavna metrika na COCO bazi podataka se računa na taj način i označava sa AP . Preostale 3 metrike se nazivaju AP_{mali} , $AP_{srednji}$ i AP_{veliki} . One se računaju na isti način kao AP ali razmatraju samo podskup objekata određene veličine iz cele baze podataka. AP_{mali} računa AP samo za objekte iz validacionog skupa čija je površina manja od 32^2 piksela, a ostale ignorise. $AP_{srednji}$ razmatra samo objekte sa površinom većom od 32^2 i manjom od 96^2 piksela, dok AP_{veliki} razmatra sve objekte sa površinom iznad 96^2 piksela.

U tabeli 3 su prikazani rezultati detekcije objekata za različite strategije konstrukcije tekstualnog upita, pri podrazumevanim podešavanjima inicijalne filtracije klasa i fiksiranim parametrima pretrage po snopu $p = 5$ i $b = 5$.

Strategija	Negativni tekst	Negativne kategorije	Obećavajuće kategorije	Sve kategorije
AP	0.0028	0.0039	0.0046	0.0044
AP_{blago}	0.0148	0.0173	0.0154	0.0139
AP_{strog}	0.0000	0.0008	0.0000	0.0036
AP_{mali}	0.0000	0.0000	0.0000	0.0000
$AP_{srednji}$	0.0000	0.0000	0.0000	0.0000
AP_{veliki}	0.0074	0.0162	0.0158	0.0111

Tabela 3: Rezultati algoritma za različite strategije tekstualnog upita. $p = 5$, $b = 5$

Primećuje se da su rezultati algoritma veoma loši. Radi poređenja, trenutno najbolji modeli za detekciju objekata postižu preko 0.63 AP na COCO bazi podataka [21], [22]. Pored strategije "negativni tekst" koja ima najgore performanse, sve strategije postižu slične rezultate na svim metrikama, sa strategijom "obećavajuće kategorije" u maloj prednosti. Strategija "obećavajuće kategorije" je proglašena za podrazumevanu strategiju konstrukcije tekstualnih upita.

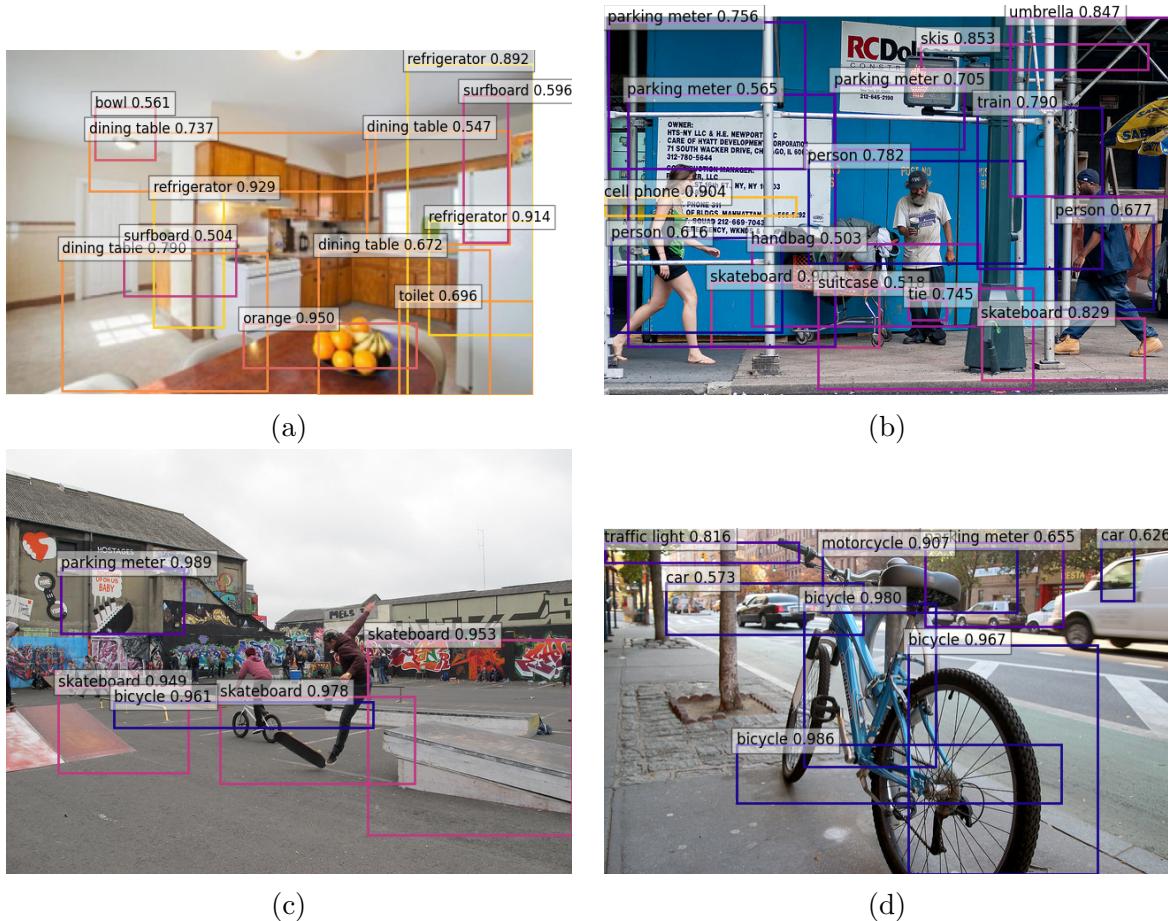
U tabeli 4 su prikazani rezultati detekcije objekata pri podrazumevanim podešavanjima za različite parametre pretrage po snopu.

Parametri	$p = 3, b = 3$	$p = 5, b = 5$	$p = 10, b = 3$	$p = 10, b = 10$
AP	0.0046	0.0046	0.0009	0.0013
AP_{blago}	0.0246	0.0154	0.0041	0.0046
AP_{strog}	0.0000	0.0008	0.0000	0.0014
AP_{mali}	0.0000	0.0000	0.0000	0.0000
$AP_{srednji}$	0.0000	0.0000	0.0000	0.0000
AP_{veliki}	0.0098	0.0158	0.0021	0.0040

Tabela 4: Rezultati algoritma za različite parametre lokalne pretrage

Zaključuje se da algoritam ima loše performanse nezavisno od parametara lokalne pretrage i strategije tekstualnog upita. Pošto algoritam postiže neznatno bolje rezultate za parametre $p = 5, b = 5$ oni su uzeti za podrazumevane.

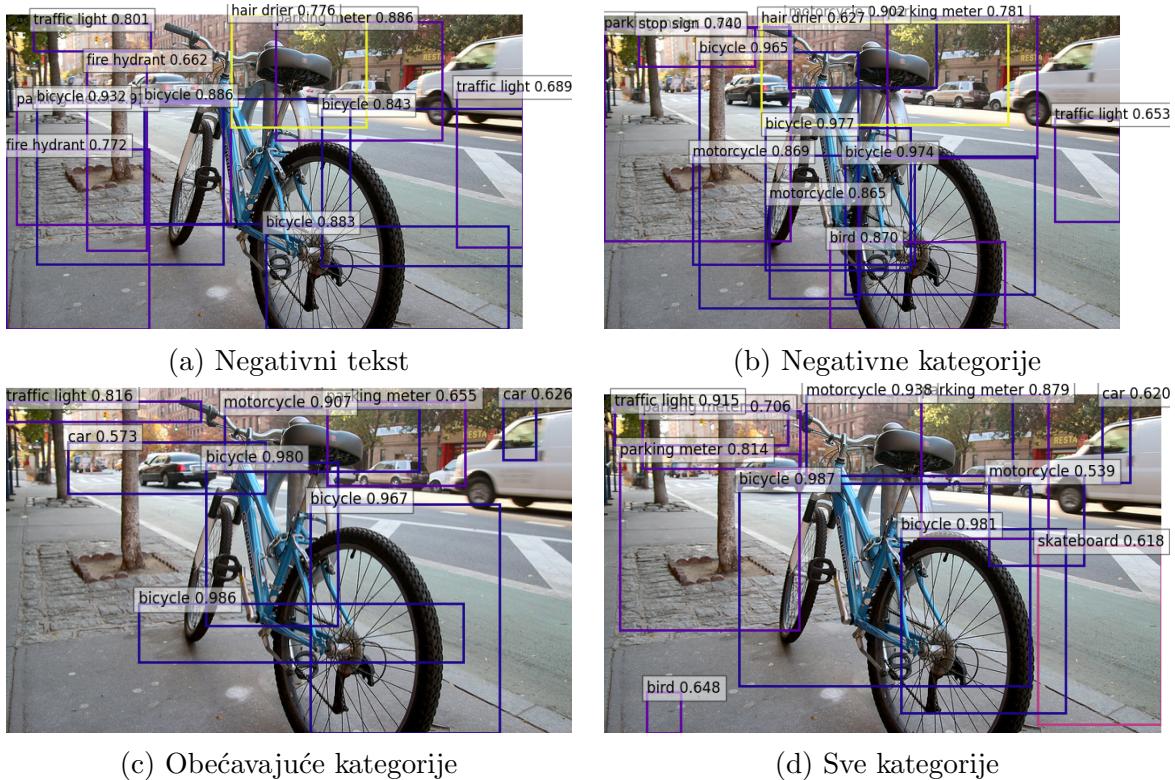
Na slikama 8 su prikazani primeri detekcije algoritma pri podrazumevanim podešavanjima. Iako algoritam ponekad pravi dobre detekcije, na slikama se uočavaju dva velika problema.



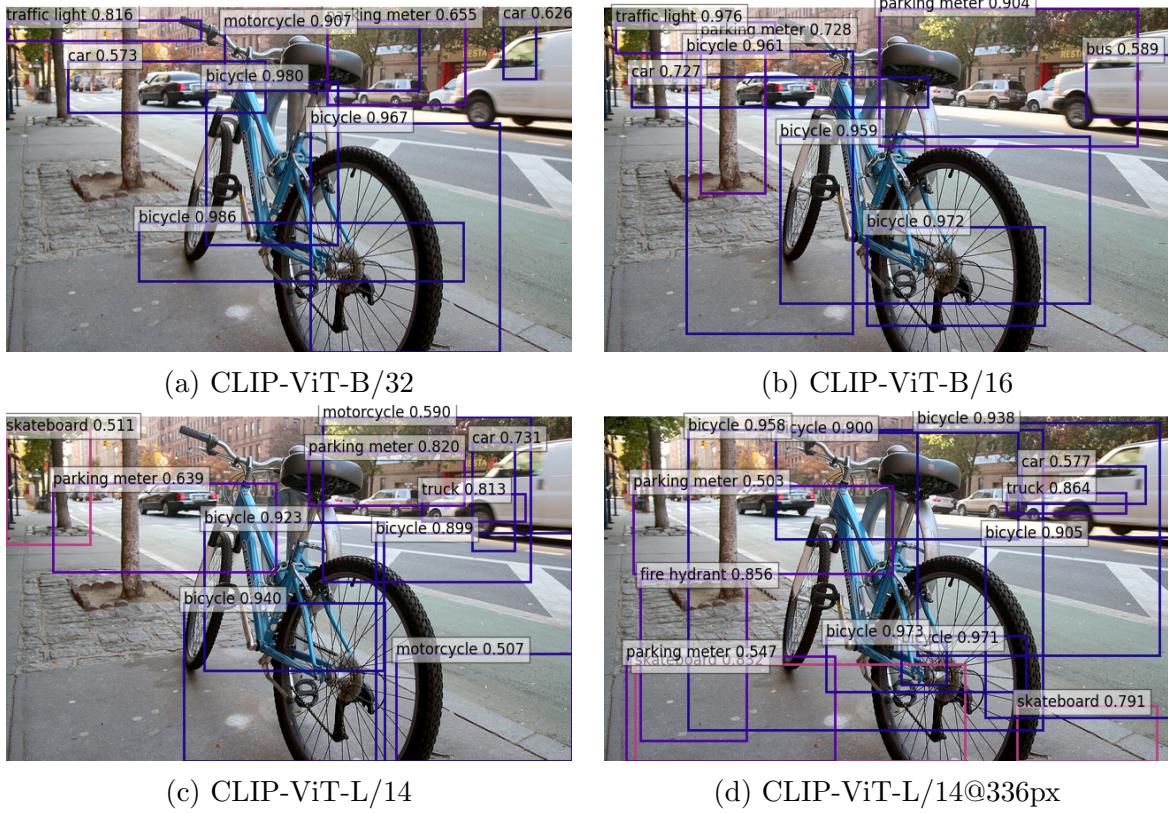
Slika 8: Rezultati algoritma za detekciju objekata pri podrazumevanim podešavanjima

Prvi problem je veliki broj lažno pozitivnih detekcija sa visokom sigurnošću. Ovaj problem se u literaturi ponekad naziva "halucinacija" [17]. Na primer, na slici 8b algoritam halucinira mobilni telefon sa sigurnošću 0.904, na slici 8c parking metar sa sigurnošću 0.989, a na slici 8d semafor sa sigurnošću 0.816. Određeni broj lažno pozitivnih detekcija se može otkloniti povećanjem praga t_{post} ali značajan broj halucinacija ima veoma visoku sigurnost, često veću nego tačne detekcije, tako da povećanje praga može istovremeno da smanji broj korektnih detekcija.

Ovaj problem nije specifičan za strategiju "obećavajuće kategorije", što se može videti na slikama 9 na kojima su prikazane detekcije za različite tekstualne strategije. Interesantno je da čak i pri strategiji "sve kategorije" algoritam pravi istu grešku, što znači da za određene besmislene isecke, čak i kada CLIP modelu prosledimo 80 tekstualnih upita sa svim kategorijama on i dalje jednoj od njih daje verovatnoću preko 0.5. Problem halucinacije, nažalost, nije specifičan ni za korišćenu verziju CLIP-a, već je prisutan i pri korišćenju boljih verzija kao što se može videti na slikama 10. Halucinacija predstavlja inherentan problem predtreniranog CLIP modela i čini ga nepodesnim za korišćenje u svrhe detekcije objekata. Ovaj problem je potencijalno moguće otkloniti korišćenjem drugačijih tekstualnih upita ili dotreniravanjem modela, što je ostavljeno za buduće istraživanje.



Slika 9: Rezultati algoritma za detekciju objekata pri podrazumevanim podešavanjima za različite strategije konstrukcije tekstualnog upita



Slika 10: Rezultati algoritma za detekciju objekata pri podrazumevanim podešavanjima za različite verzije CLIP modela

Na slici 8d se primećuje da čak i kada je klasa objekta tačno detektovana, algoritam ima problema sa lokalizacijom objekta. Idealni isečak bicikla prikazan na slici 11b je proglašen za bicikl sa verovatnoćom od 0.981, dok je verovatnoća bicikla na lošem isečku 11a jednaka 0.996. Ispostavlja se da idealni pravougaoni isečak objekta ne predstavlja nužno i maksimum funkcije dobitka lokalne pretrage. Može se pretpostaviti da je sigurnost na idealnom isečku 11b manja zbog prisustva drugih objekata na slici, međutim čak i na isečku 11c koji ne sadrži nijedan drugi objekat sigurnost je 0.987 i manja nego na isečku 11a. Model rado daje veću verovatnoću isečcima objekta koji sadrže upečatljive elemente tog objekata, nego isečcima koji u potpunosti obuhvataju posmatrani objekat. To nažalost krši osnovnu pretpostavku na kojoj se algoritam bazira, zato što nema smisla raditi lokalnu pretragu kada se željena vrednost parametara ne poklapa sa maksimumom funkcije dobitka. Kao što se može primetiti na slikama 9 i 10 ovaj problem postoji nezavisno od strategije tekstualnog upita i verzije korišćenog CLIP modela. Da bi se ovaj problem otklonio potrebno je na neki način uskladiti funkciju dobitka sa kvalitetom pravouganog okvira. Ovo se potencijalno može postići dotreniranjem modela, promenom tekstualnih upita ili drugim modifikacijama funkcije dobitka, koje su ostavljene za budući rad.



(a) Optimalni isečak po funkciji dobiti. Sigurnost 0.996



(b) Idealni pravougaoni isečak. Sigurnost 0.981



(c) Prikladan pravougaoni isečak. Sigurnost 0.987

Slika 11: Prikaz neusklađenosti funkcije dobiti i kvaliteta pravouganog okvira

4.3 Vreme izvršavanja algoritma

Treći veliki problem sa predstavljenim algoritmom za detekciju objekata je izuzetno dugo vreme izvršavanja. Većina vremena izvršavanja se troši na obradu koju CLIP model vrši. Značajno vreme se takođe troši na pripremu slika za ulaz u model, konkretno na reskaliranje slika na rezoluciju 224×224 ili 334×334 , zato što se koristi računski kompleksna bikubna interpolacija. U odnosu na ova dva izvora računske složenosti ostatak algoritma troši zanemarljivo malo vremena.

Vreme izvršavanja algoritma je mereno na računaru sa AMD Ryzen 7 5800H procesorom i NVIDIA GeForce RTX 3070 grafičkom karticom. U tabeli 5 su dati okvirni brojevi slika koje različite verzije CLIP-a mogu da obrade u jednoj sekundi.

Verzije	CLIP-ViT-B/32	CLIP-ViT-B/16	CLIP-ViT-L/14	CLIP-ViT-L/14@336px
Slika po sekundi	180	25	17	8

Tabela 5: Okvirni brojevi slika koje različite verzija CLIP-a obrađuju po sekundi

Na osnovu prikazanih rezultata se zaključuje da bi korišćenje nekog modela većeg od CLIP-ViT-B/32 značajno usporilo izvršavanje već sporog algoritma. Nažalost ne postoji ništa što se može uraditi da bi se ove brzine povećale, osim da se koristi jači hardver ili da se razvije manji, destilovani CLIP model, što je van okvira ovog rada. Celokupan algoritam se može ubrzati smanjivanjem parametara pretrage p i b i samim tim smanjivanjem broj slika koji se prosleđuju modelu tokom lokalne pretrage. To bi u generalnom slučaju trebalo da pogorša rezultate lokalne pretrage, ali to nije tačno u slučaju predstavljenog algoritma, zbog neusklađenosti idealnih pravougañih okvira i funkcije dobitka.

Promena rezolucije slike se izvršava okvirno brzinom od 170 slika po sekundi. Postupak se može ubrzati korišćenjem prostije metode interpolacije, kao što je na primer najbliži sused, ali očekujemo da će u tom slučaju performanse algoritma opasti, pogotovo zato što je ponekad potrebno reskalirati isečke slika koji su vrlo male rezolucije. Alternativno poboljšanje, koje bi u značajnoj meri ubrzalo ovaj deo algoritma, je da se slika na koju se primenjuje algoritam detekcije samo jednom reskalira na željenu rezoluciju, a kada je potrebno da se modelu prosledi isečak te slike, umesto da se slika seče, svi pikseli koji ne pripadaju tom isečku se zacrne. Na ovaj način je potrebno samo jednom promeniti rezoluciju slike, a svako sledeće isecanje i reskaliranje menjamo sa zacrnjivanjem koje ima značajno manju računsku složenost. Zacenjivanje slike je u inicijalnom testiranju davalо lošije rezultate nego isecanje, pogotovo za male objekte, koji postanu jedva vidljivi kada se slika oko njih zacrni. Pošto pretprocesiranje slika predstavlja samo jedno usko grlo algoritma i pošto je drugo usko grlo praktično neuklonjivo, ekstenzivnije testiranje ovakvih poboljšanja je ostavljeno za budući rad.

Vreme izvršavanja celog algoritma dosta varira pri korišćenju različitih podešavanja algoritma. Pri podrazumevanim podešavanjima algoritmu je u proseku potrebno 70 sekundi da izvrši detekciju na jednoj slici.

5 Zaključak

U ovom radu je testirana mogućnost korišćenja predobučenog CLIP modela zarad *zero-shot* detekcije objekata. Predložen je i testiran algoritam za *zero-shot* detekciju objekata zasnovan na lokalnoj pretrazi. Predloženi algoritam postiže veoma loše rezultate i pati od 3 velika problema: halucinacija objekata, neusklađenost kvaliteta lokalizacije sa funkcijom dobiti i dugo vreme izvršavanja. Navedeni problemi su prisutni nezavisno od brojnih podešavanja i modifikacija algoritma. Iako ovaj rad nije iscrpan, 3 uočena problema dovode u pitanje validnost celokupnog postupka. Pretpostavlja se da je samo inženjerstvom upita, bez dodatnog treniranja modela, izuzetno teško rešiti probleme halucinacije i neusklađenosti funkcije dobiti, što u krajnjoj liniji znači da je predobučeni CLIP najverovatnije nepodesan za korišćenje u svrhe *zero-shot* detekcije objekata.

Literatura

- [1] R. Keys, „Cubic convolution interpolation for digital image processing,” *IEEE transactions on acoustics, speech, and signal processing*, sv. 29, br. 6, str. 1153–1160, 1981.
- [2] T. Fawcett, „An introduction to ROC analysis,” *Pattern recognition letters*, sv. 27, br. 8, str. 861–874, 2006.
- [3] M. Everingham, L. Van Gool, C. K. Williams, J. Winn i A. Zisserman, „The pascal visual object classes (voc) challenge,” *International journal of computer vision*, sv. 88, br. 2, str. 303–338, 2010.
- [4] T. Berg, J. Liu, S. Woo Lee, M. L. Alexander, D. W. Jacobs i P. N. Belhumeur, „Birdsnap: Large-scale fine-grained visual categorization of birds,” u *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014., str. 2011–2018.
- [5] T.-Y. Lin, M. Maire, S. Belongie i dr., „Microsoft COCO: Common objects in context,” u *European conference on computer vision*, Springer, 2014., str. 740–755.
- [6] R. Sennrich, B. Haddow i A. Birch, „Neural machine translation of rare words with subword units,” *arXiv preprint arXiv:1508.07909*, 2015.
- [7] J. L. Ba, J. R. Kiros i G. E. Hinton, „Layer normalization,” *arXiv preprint arXiv:1607.06450*, 2016.
- [8] K. He, X. Zhang, S. Ren i J. Sun, „Deep residual learning for image recognition,” u *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016., str. 770–778.
- [9] D. Hendrycks i K. Gimpel, „Gaussian error linear units (GELUs),” *arXiv preprint arXiv:1606.08415*, 2016.
- [10] Y. Wu, M. Schuster, Z. Chen i dr., „Google’s neural machine translation system: Bridging the gap between human and machine translation,” *arXiv preprint arXiv:1609.08144*, 2016.
- [11] A. Vaswani, N. Shazeer, N. Parmar i dr., „Attention is all you need,” *Advances in neural information processing systems*, sv. 30, 2017.
- [12] J. Devlin, M.-W. Chang, K. Lee i K. Toutanova, „Bert: Pre-training of deep bidirectional transformers for language understanding,” *arXiv preprint arXiv:1810.04805*, 2018.
- [13] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever i dr., „Improving language understanding by generative pre-training,” 2018.
- [14] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever i dr., „Language models are unsupervised multitask learners,” *OpenAI blog*, sv. 1, br. 8, str. 9, 2019.

-
- [15] T. Brown, B. Mann, N. Ryder i dr., „Language models are few-shot learners,” *Advances in neural information processing systems*, sv. 33, str. 1877–1901, 2020.
 - [16] A. Dosovitskiy, L. Beyer, A. Kolesnikov i dr., „An image is worth 16x16 words: Transformers for image recognition at scale,” *arXiv preprint arXiv:2010.11929*, 2020.
 - [17] O. S. Kayhan, B. Vredebregt i J. C. van Gemert, „Hallucination In Object Detection—A Study In Visual Part VERIFICATION,” u *2021 IEEE International Conference on Image Processing (ICIP)*, IEEE, 2021., str. 2234–2238.
 - [18] A. Radford, J. W. Kim, C. Hallacy i dr., „Learning transferable visual models from natural language supervision,” u *International Conference on Machine Learning*, PMLR, 2021., str. 8748–8763.
 - [19] T. Kojima, S. S. Gu, M. Reid, Y. Matsuo i Y. Iwasawa, „Large Language Models are Zero-Shot Reasoners,” *arXiv preprint arXiv:2205.11916*, 2022.
 - [20] L. Ouyang, J. Wu, X. Jiang i dr., „Training language models to follow instructions with human feedback,” *arXiv preprint arXiv:2203.02155*, 2022.
 - [21] W. Wang, H. Bao, L. Dong i dr., „Image as a Foreign Language: BEiT Pretraining for All Vision and Vision-Language Tasks,” *arXiv preprint arXiv:2208.10442*, 2022.
 - [22] H. Zhang, F. Li, S. Liu i dr., „Dino: Detr with improved denoising anchor boxes for end-to-end object detection,” *arXiv preprint arXiv:2203.03605*, 2022.