Usporedba performansi YOLOv5 i YOLOv8 algoritama u zadacima klasifikacije

Drašković, Ema Andrea

SVEUČILIŠTE U SPLITU

PRIRODOSLOVNO-MATEMATIČKI FAKULTET

ZAVRŠNI RAD

USPOREDBA PERFORMANSI YOLOv5 I YOLOv8 ALGORITAMA U ZADACIMA KLASIFIKACIJE

Ema Andrea Drašković

Split, 2024.

Temeljna dokumentacijska kartica

Završni rad

Sveučilište u Splitu
Prirodoslovno-matematički fakultet
Odjel za informatiku
Ruđera Boškovića 33, 21000 Split, Hrvatska

USPOREDBA PERFORMANSI YOLOv5 I YOLOv8 ALGORITAMA U ZADACIMA DETEKCIJE

Ema Andrea Drašković

SAŽETAK

Umjetna inteligencija predstavlja široko interdisciplinarno područje informatike koje stvara sustave sposobne demonstrirati ljudske vještine poput percepcije, učenja i rješavanja problema. Rad uvodi u koncepte ključne za razvoj AI sustava, kao strojno učenje i duboko učenje. Posebno se ističe YOLO algoritam dubokog učenja za detekciju objekata, razvijen radi brze i precizne analize slika. Pregledavajući arhitekturu varijanti v5 i v8 YOLO modela, ističe se njegova sposobnost prilagodbe različitim zahtjevima te podrška za vlastiti skup podataka. Analiza metrika poput srednje prosječne preciznosti i F1 pruža uvid u evaluaciju performansi modela. Treniranje modela, ključno za postizanje visokih performansi, ističe važnost kvalitetnih podataka i finog podešavanja (engl. *fine-tuning*) za specifične potrebe aplikacije. Detekcija objekata kroz klasifikaciju i regresiju objašnjava kako YOLO model brzo i precizno detektira objekte u slikama. Kroz primjere primjene, rad ilustrira kako pripremiti skup podataka i trenirati na njemu YOLO model, te uspoređuje različite inačice algoritma preko metrika i performansi.

Ključne riječi: računalni vid, duboko učenje, konvolucijske neuronske mreže, detekcija objekata, klasifikacija, lokalizacija

Rad je pohranjen u knjižnici Prirodoslovno-matematičkog fakulteta, Sveučilišta u Splitu

Rad sadrži:	41 stranicu, 21 grafički prikaz i 27 literaturnih navoda. Izvornik je na hrvatskom jeziku.
Mentor:	**Saša Mladenović, prof. dr. sc.**, *redoviti profesor Prirodoslovno-matematičkog fakulteta u Splitu, Sveučilišta u Splitu*
Neposredni voditelj:	**Dino Nejašmić, mag. educ. math. et inf.,** *predavač Prirodoslovno-matematičkog fakulteta u Splitu, Sveučilišta u Splitu*
Ocjenjivači:	**Saša Mladenović, prof. dr. sc.**, *redoviti profesor Prirodoslovno-matematičkog fakulteta u Splitu, Sveučilišta u Splitu*
	Dino Nejašmić, mag. educ. math. et inf., *predavač Prirodoslovno-matematičkog fakulteta u Splitu, Sveučilišta u Splitu*
	Nika Jerković, mag. ing, comp., *asistent Prirodoslovno-matematičkog fakulteta u Splitu, Sveučilišta u Splitu*

Rad prihvaćen: **Rujan 2024.**

Basic documentation card

Bachelor Thesis

University of Split

Faculty of Science

Department of Informatics

Ruđera Boškovića 33, 21000 Split, Croatia

PERFORMANCE COMPARISON OF YOLOv5 AND YOLOv8 ALGORITHMS IN OBJECT CLASSIFICATION TASKS

Ema Andrea Drašković

ABSTRACT

Artificial intelligence represents a broad interdisciplinary field of computer science that creates systems capable of demonstrating human-like skills such as perception, learning and problem solving. The paper introduces key concepts essential for the development of AI systems, such as machine learning and deep learning, where the YOLO algorithm for object detection is notable. By reviewing the architecture of the YOLO v5 and v8 model variants, the focus is on its adaptability to various requirements and support for custom datasets. The analysis of metrics such as mean average precision and F1 provides insight into the evaluation of model performance. Model training, crucial for achieving high performance, emphasizes the importance of quality data and fine-tuning for specific application needs. Object detection through classification and regression explains how YOLO model quickly and accurately detects objects in images. Through practical application, the paper illustrates how to prepare a dataset and train the YOLO model on it, comparing different versions of the algorithm based on metrics and performance.

Key words: computer vision, deep learning, convolutional neural networks (CNN), object detection, classification, localization

Thesis deposited in library of Faculty of science, University of Split

Thesis consist of:	41 pages, 21 figures and 27 references. Original language: Croatian.
Mentor:	**Saša Mladenović, Ph.D.**, *Professor at the Faculty of Science and Mathematics, University of Split*
Supervisor:	**Dino Nejašmić, mag. educ. math. et inf.,** *Lecturer at the Faculty of Science and Mathematics, University of Split*
Reviewers:	**Saša Mladenović, Ph.D.**, *Professor at the Faculty of Science and Mathematics, University of Split*
	Dino Nejašmić, mag. educ. math. et inf., *Lecturer at the Faculty of Science and Mathematics, University of Split*
	Nika Jerković, mag. ing. comp., *Instructor at the Faculty of Science and Mathematics, University of Split*

Thesis accepted: **September 2024**

IZJAVA

Kojom izjavljujem s punom materijalnom i moralnom odgovornošću da sam završni rad s naslovom Usporedba performansi YOLOv5 i YOLOv8 algoritama u zadacima detekcija izradila samostalno pod voditeljstvom Dina Nejašmića, mag. educ. math. et inf., pred. U radu sam primijenila metodologiju znanstvenoistraživačkog rada i koristila literaturu koja je navedena na kraju završnog rada. Tuđe spoznaje, stavove, zaključke, teorije i zakonitosti koje sam izravno ili parafrazirajući navela u završnom radu na uobičajen, standardan način citirala sam i povezala s fusnotama s korištenim bibliografskim jedinicama. Rad je pisan u duhu hrvatskog jezika.

Studentica

Ema Andrea Drašković

Sadržaj

1. UVOD

Umjetna inteligencija (engl. *Artificial intelligence*, AI) obuhvaća široko interdisciplinarno područje informatike. Bavi se razvojem sustava i algoritama koji su sposobni obavljati zadatke koji tradicionalno zahtijevaju ljudsku inteligenciju. Da bi se to postiglo razvijaju se vještine poput percepcije, učenja, razumijevanja jezika te sposobnost prilagodbe, donošenja odluka i rješavanja problema. (*What Is Artificial Intelligence (AI)?*, n.d.)

Za realizaciju tih sposobnosti ključni su koncepti strojnog učenja i dubokog učenja. Strojno učenje (engl. *machine learning*) predstavlja paradigmatski pristup kojim sustav umjetne inteligencije uči iz iskustva, prilagođava se novim podacima i obavlja zadatke bez eksplicitnog programiranja. Duboko učenje (engl. *deep learning*) je područje strojnog učenja, a koristi duboke neuronske mreže za izvlačenje složenih značajki iz podataka. Takve su mreže inspirirane strukturom ljudskog mozga i mogu analizirati i interpretirati podatke na različitim razinama apstrakcije. Sastoje se od više međusobno povezanih slojeva koji se tijekom treniranja prilagođavaju. Slojeve čine umjetni neuroni koji primaju ulazne podatke, obrađuju ih i prosljeđuju izlaz drugim neuronima. Duboko učenje koristi ove kompleksne neuronske arhitekture za usvajanje reprezentacija podataka koje se mogu koristiti za rješavanje različitih zadataka.

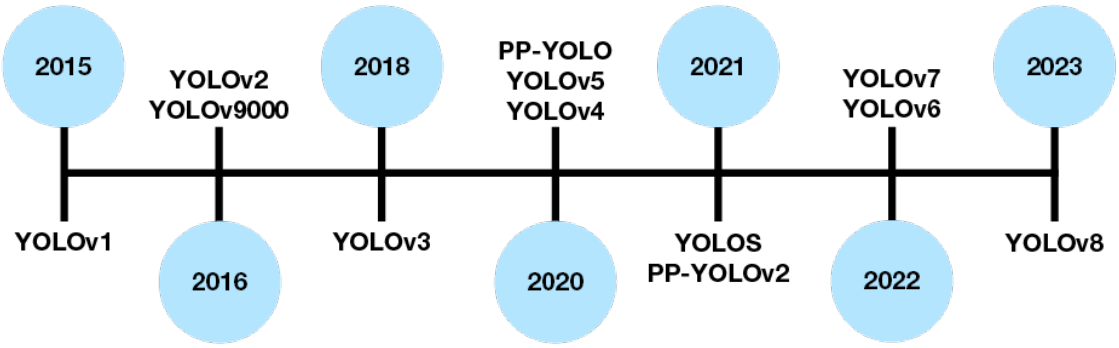
Slika 1.1 - Odnos umjetne inteligencije, strojnog i dubokog učenja

Primjena dubokog učenja postala je ključna u ostvarivanju visokih performansi u raznim područjima umjetne inteligencije pa tako i kod prepoznavanja objekata. Ovaj rad za cilj ima istražiti konkretnu primjenu umjetne inteligencije u prepoznavanju obrazaca i klasifikaciji podataka, a fokusira se na praktičnu implementaciju i treniranje modela umjetne inteligencije na stvarnim podacima korištenjem YOLO algoritma dubokog učenja. Uspoređuju se modeli *s* i *l* iteracija v5 i v8 treningom na istom skupu podataka za identifikaciju redova člankonožaca. Demonstrira se još kako obraditi i analizirati velike količine podataka kako bi se mogli koristiti sa modelima YOLO algoritma.

2. YOLO ALGORITAM

YOLO je jedan od najpoznatijih algoritama dubokog učenja poznat kao ***You Only Look Once***. Koristi konvolucijske neuronske mreže (engl. *convolutional neural network*, skraćeno CNN), koje su specifična vrsta neuronskih mreža dizajnirana za rad s vizualnim podacima. CNN-ovi primjenjuju posebne filtere, poznate kao konvolucije, koje pomoću matematičkih operacija stvaraju skup težina kojim se reprezentiraju dijelovi slike. Izlaz tih filtara je dvodimenzionalni niz koji predstavlja ulaznu sliku, a naziva se karta značajki. Na ovaj način CNN-ovi prepoznavanju i izdvajaju značajke i obrasce, kao što su rubovi, teksture i oblici (Craig & Awati, 2024).

YOLO algoritam radi na način da predviđa okvire koji omeđuju objekte direktno iz slike, odnosno analizira cijelu sliku u jednom prolazu kroz model kako bi otkrio postoje li i gdje su razni objekti. Zbog toga je YOLO jedan od bržih metoda detekcije objekata, posebice kada se koristi u stvarnom vremenu (Ayush, 2024).

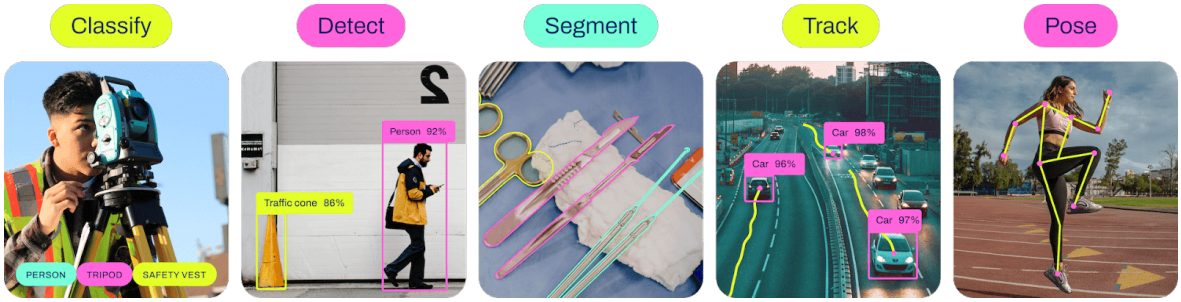


Slika 2.1 - Razvoj YOLO iteracija

Ovaj algoritam trenutno razvija Ultralytics te postoji mnogo verzija razvijanih tokom zadnjih desetak godina (Slika 2.1), a prva je iteracija YOLOv1 predstavljena 2015. godine. Danas se posebno ističu verzije YOLOv5 i YOLOv8 koje su zbog svoje brzine i preciznosti idealni za upotrebu u video nadzoru, automatskoj vožnji i sličnim situacijama gdje je bitna brza reakcija. Neki od primjera primjene su detekcija vozila, pješaka i prometnih znakova u području prometa i transporta, analiza sigurnosnih snimki u području video nadzora, praćenje inventara, analiza ponašanja kupaca te poboljšanje iskustva kupovine preko personaliziranih preporuka u industriji maloprodaje.

YOLO algoritam je dizajniran da bude jednostavan za korištenje pa privlači i šire zajednice korisnika, od programera, inženjera, istraživača pa sve do amatera i entuzijasta. Zbog toga što dopušta treniranje sa vlastitim podacima, dosta je fleksibilan i prilagodljiv potrebama korisnika pa se može koristiti u gotovo svim industrijama.

YOLOv8 u nekim primjenama postiže bolje rezultate sa većom točnosti, no to ne znači nužno da je uvijek najbolje koristiti baš tu iteraciju umjesto YOLOv5. Obje verzije imaju svoje jedinstvene prednosti i upotrebe. Osim što v8 ima više načina rada (Slika 2.2), ključna razlika stoji u prednostima troškova računanja između v5 i v8. Druga razlika je jaz u izvedbi na različitim hardverskim platformama. Zbog stohastičke prirode dubokog učenja, možda se jednostavno može trenirati bolji v5 model na nekim skupovima podataka nego v8.

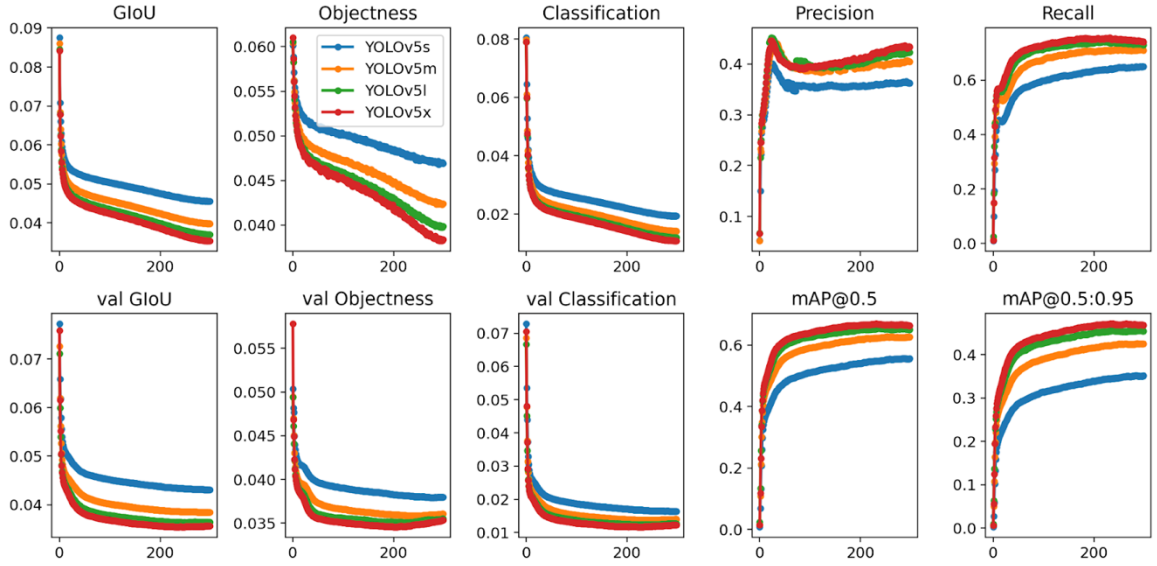


Slika 2.2 - Načini rada YOLOv8 (Ultralytics, 2024d)

Unatoč impresivnim postignućima, YOLO modeli se suočavaju s nekoliko izazova. Primjerice prilagodljivost na različite uvjete osvjetljenja, sjenčanje i raznolike perspektive, limitacija detekcije malih objekata (Deci, 2023), problem određivanja visine i širine detektiranih objekata, posebice ako je više objekata detektirano na slici (Mirani et al., 2022). Osim tehničkih izazova, sve relevantnija postaju pitanja privatnosti i etičnosti, posebice u području video nadzora (*Ethical Implications of Computer Vision Technology in 2023*, 2023).

3. ARHITEKTURA YOLO ALGORITMA

YOLOv5 i v8 se sastoje od pet varijanti modela (n, s, m, l, x), a razlikuju se po veličini i kompleksnosti modela što rezultira u varijaciji u performansama (Slika 3.1). Jedine razlike u arhitekturi varijanta modela su broj slojeva i parametara. Ovisno o specifičnim zahtjevima zadatka, može se odabrati odgovarajuća varijanta za postizanje optimalnih performansi uz balansiranje vremena potrebnog modelu za treniranje i predikciju (engl. *inference*).



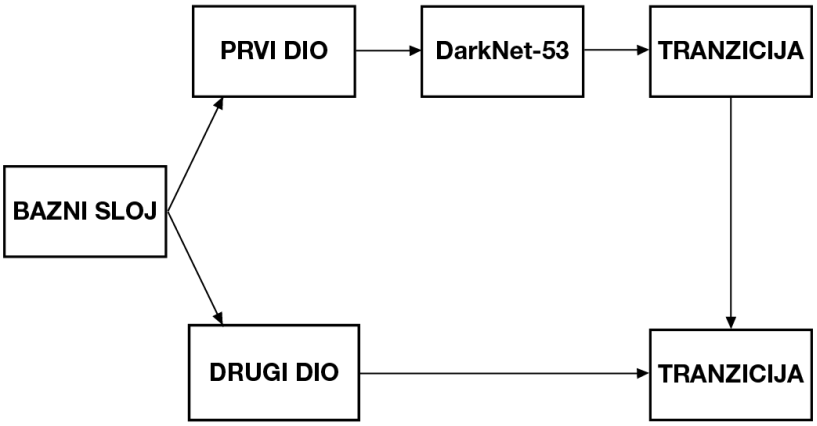
Slika 3.1 - Usporedba varijanti YOLOv5 modela

Sve varijante YOLOv5 modela se sastoje od iste tri komponente: CSP-Darknet53 kao osnova, SPP i PANet kao vrat modela i glave modela (Cherifi, 2022). YOLOv8 uvodi promjene u vidu osnove gdje koristi C2f dizajn, umjesto SPP koristi SPPF te prelazi na *anchor-free* detekciju (Casas et al., 2024).

3.1. Osnova modela

Osnova (engl. *backbone*) YOLOv5 modela je **CSP-Darknet53**, konvolucijska neuronska mreža koja koristi ***Cross Stage Partial network strategy*** (skraćeno CSPNet). Ona dijeli mape značajki baznog sloja na dva dijela te zatim spaja što rezultira boljim protokom informacija kroz mrežu (Slika 3.2) (Bochkovskiy et al., 2020). Primjena CSPNet-a pomaže

smanjiti broj parametara i smanjuje količinu računanja što vodi do povećanja brzine detekcije, što je ključno za modele detekcije objekata u realnom vremenu (Cherifi, 2022).



Slika 3.2 - Struktura CSP-DarkNet53 mreže

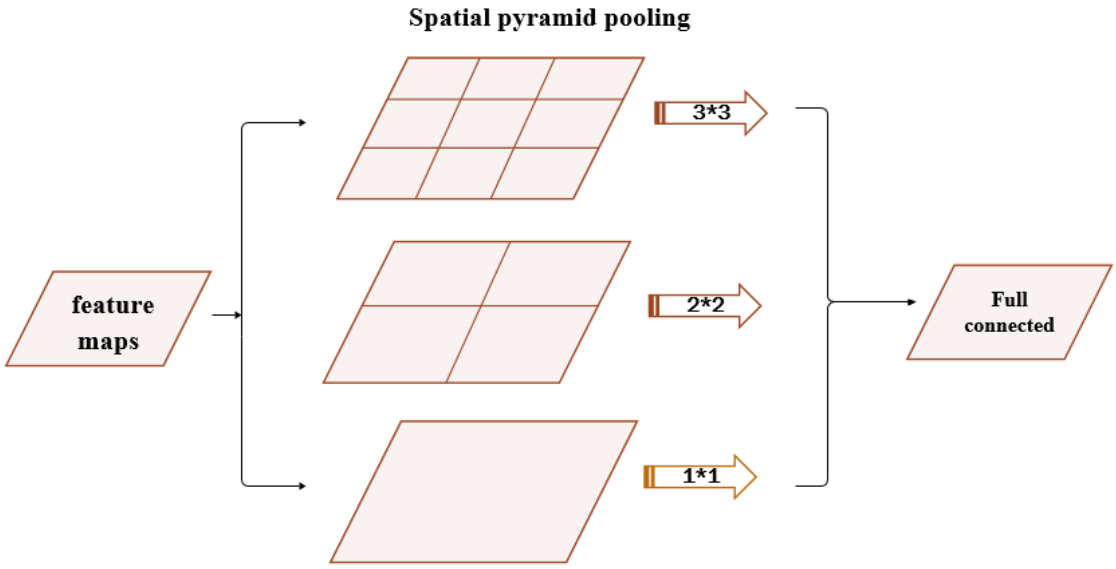
YOLOv8 uvodi poboljšanje zamjenom 6x6 konvolucije u stablu sa efikasnijom 3x3 konvolucijom (Casas et al., 2024). Koristi novi dizajn **C2f** kojim se poboljšava detekcija malih objekata. C2f je brža implementacija modula C2 koja je zapravo *CSP Bottleneck* sa dvije konvolucije.

3.2. Vrat modela

YOLOv5 kao vrat modela (engl. *model neck*) koristi ***Spatial Pyramid Pooling*** (skraćeno SPP) i ***Path Aggregation Network*** (skraćeno PANet) modificiran uključivanjem ***BottleNeckCSP*** u svoju arhitekturu. PANet je mreža uvedena u arhitekturu YOLOv4 kako bi poboljšala protok informacija i pomogla u pravilnom lokaliziranju piksela u zadatku predviđanja maski (Cherifi, 2022). Omogućuje da svaka predložena regija izravno pristupa informacijama iz svih razina značajki, što poboljšava predviđanje (S. Liu et al., 2018). U YOLOv5 je ova mreža modificirana primjenom CSPNet strategije, kako je prikazano na slici (Slika 3.2) arhitekture mreže (Cherifi, 2022).

SPP blok izvodi agregaciju informacija koje prima od ulaza i vraća izlaz fiksne duljine bez obzira na veličinu ili skalu slike, uklanjajući potrebu za fiksnom veličinom ulazne slike. Na ulaznu sliku se primjenjuje konvolucijska neuronska mreža kako bi se izračunale značajke. Nakon toga se primjenjuje SPP sloj koji razdvaja sliku na više područja različitih veličina (npr. 1x1, 2x2, 4x4). Za svako područje se izračunava maksimalna vrijednost značajki. Zatim se maksimalne vrijednosti iz svih područja spajaju u vektor fiksne

duljine koji predstavlja reprezentaciju cijele slike, neovisno o njenoj veličini (Slika 3.3). Ovaj se vektor na kraju koristi za detekciju ili klasifikaciju objekata, ovisno o modelu (He et al., 2014).



Slika 3.3 - Mrežna struktura s slojem prostorne piramide za grupiranje (Yang et al., 2018)

YOLOv8 uvodi **SPPF** (engl. *Spatial Pyramid Pooling Feature*), poboljšanu verziju SPP-a. Dizajniran je da ubrza predikciju i optimizira performanse mreže (Shi et al., 2022).

3.3. Glava modela

YOLOv5 koristi istu glavu (engl. *model head*) kao YOLOv3 i YOLOv4 (Rajput, 2020). Sastoji se od tri konvolucijska sloja koji predviđaju lokaciju okvira (x, y, visina, širina) (Cherifi, 2022), klasu objekta i vjerojatnosti klasa, odnosno vjerojatnosti da su objekti unutar okvira (Ultralytics, 2024c).

YOLOv8 prelazi na *anchor-free* detekciju koja omogućuje direktno predviđanje centara objekata. To znači da glava modela ne koristi unaprijed definirane "*anchor*" okvire kao referentne točke za daljnje predviđanje okvira, već izravno predviđa okvire objekata na temelju značajki slike (Casas et al., 2024).

3.4. Aktivacijska funkcija

Postoji niz aktivacijskih funkcija koje razni YOLO modeli upotrebljavaju, a neke od najkorištenijih su ***Sigmoid-weighted Linear Unit*** (skraćeno SiLU ili Swish) aktivacijska

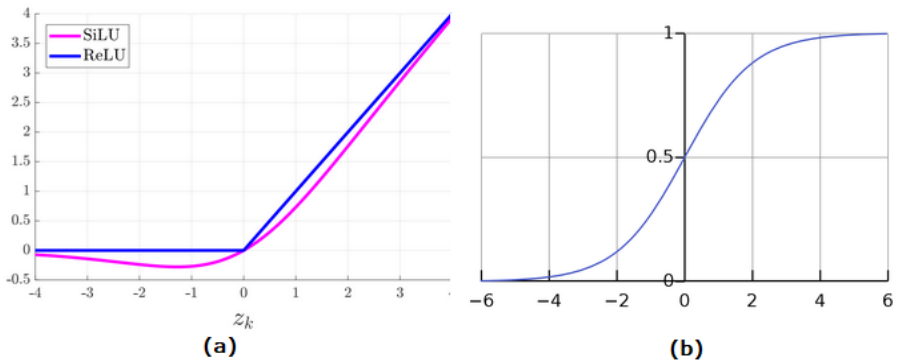
funkcija s operacijama konvolucije u skrivenim slojevima (1) i **sigmoidna** aktivacijska funkcija (2) s operacijama konvolucije u izlaznom sloju za predviđanje koordinata omeđujućih okvira (Cherifi, 2022). Grafovi SiLU i sigmoidne funkcije su prikazani na slici ispod (Slika 3.4).

$$Swish = x \cdot (1 + e^{-x})^{-1}, \qquad raspon(x) = [-0.28, +\infty] \tag{1}$$

SiLU je relativno nova aktivacijska funkcija koja je postala popularna zbog svojih svojstava i performansi u dubokom učenju te se smatra kontinuiranom aproksimacijom standardne linearne funkcije i ReLU (engl. *rectified linear unit*) aktivacijske funkcije. Ima svojstvo postupnog zasićenja, što može pomoći u sprječavanju problema poput mrtvih neurona i stabilizaciji gradijenata tijekom treniranja (Doherty et al., 2022).

$$Sigmoid = \frac{1}{1 + e^{-x}}, \qquad raspon(x) = [0, 1] \tag{2}$$

Sigmoidna aktivacijska funkcija je dosta složenija od linearne pa je i „skuplja“ po pitanju performansi, a uz to ima problem zasićenja gradijenata (Doherty et al., 2022).



Slika 3.4 - (a) Graf funkcije SiLU (b) Graf funkcije Sigmoid (Cherifi, 2022)

3.5. Funkcija gubitka

YOLO koristi različite metrike unutar modela za optimizaciju svojih predikcija. Konačni gubitak modela sastoji se od više komponenti, a dosta pojednostavljena formula prikazana je u formuli (3).

$$GUBITAK = \lambda_1 L_{cls} + \lambda_2 L_{obj} + \lambda_3 L_{loc} \tag{3}$$

Ova pojednostavljena formula gubitka sastoji se od tri glavna dijela. Prvo, gubitak klasifikacije L_{cls}, koji uspoređuje rezultate predikcije sa stvarnim vrijednostima. Za računanje gubitka klasifikacije koriste se funkcije poput **kategoričke unakrsne entropije** (engl. *categorical cross-entropy*) ili **binarne unakrsne entropije** (engl. *binary cross-entropy*). Zatim gubitak objekta L_{obj} koji računa koliko dobro model detektira objekte na slici. Naposljetku je lokalizacijski gubitak L_{loc} koji uspoređuje predviđene omeđujuće okvire sa stvarnim omeđujućim okvirima, na način da računa razliku predviđenih i stvarnih koordinata omeđujućih okvira. Za to se obično koristi ***Complete Intersection over Union*** (skraćeno CIoU) (Cherifi, 2022). Koeficijentima λ_1, λ_2, λ_3 se skalira svaki dio gubitka kako bi se uravnotežili. Ovi koeficijenti su hiperparametri koji se mogu eksperimentalno podešavati za postizanje optimalnih performansi modela.

4. METRIKE

Za procjenu performansi YOLO modela koriste se različite metrike za analizu treninga i validacije. Time se identificiraju slabosti modela koje se mogu poboljšati. Neke od često korištenih metrika u evaluaciji YOLO modela su preklapanje prema uniji, preciznost, odziv, F1 ocjena, srednja apsolutna pogreška, srednja kvadratna pogreška (Ultralytics, 2024a). Matrica konfuzije pomaže jednostavno vizualizirati performanse modela (Tablica 1).

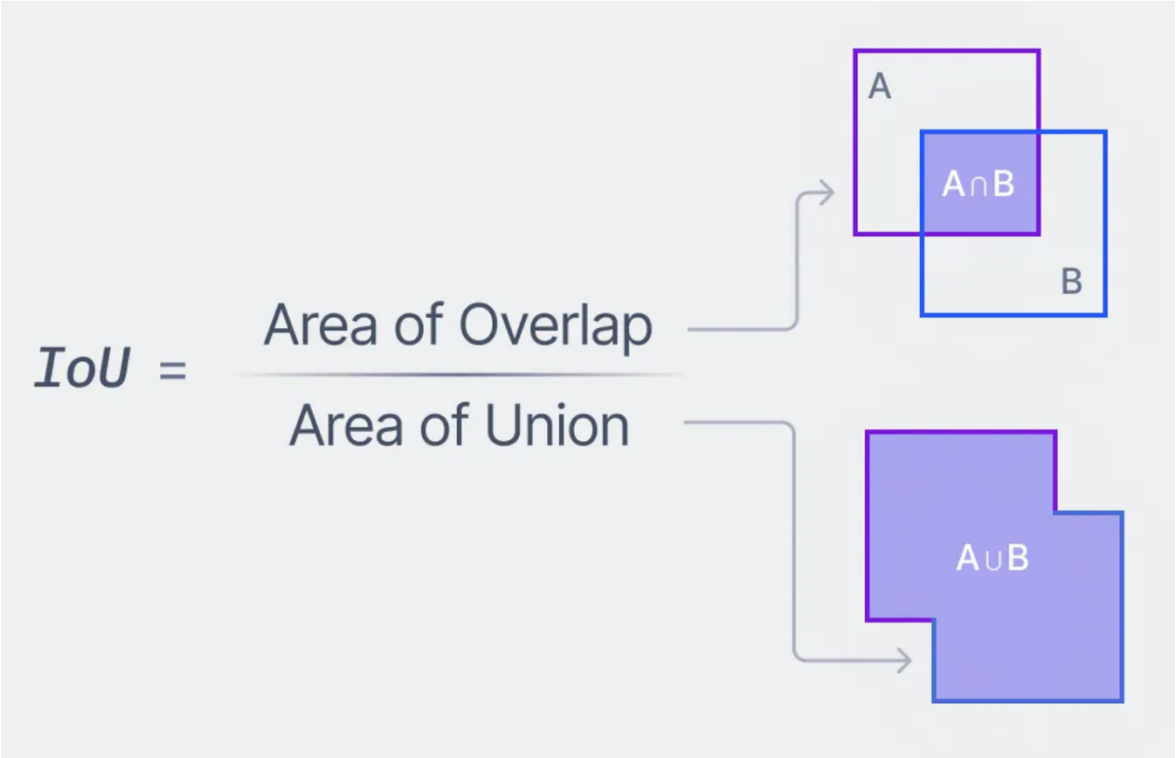
		Stvarne vrijednosti	
		Pozitivna	Negativna
Predviđene vrijednosti	Pozitivna	**Stvarno Pozitivna**	**Lažno Pozitivna**
	Negativna	**Lažno Negativna**	**Stvarno Negativna**

Tablica 1 – Definicija matrice konfuzije

4.1. Preklapanje prema uniji

Preklapanje prema uniji (engl. *intersection over union*, skraćeno IoU) je mjera koja računa preklapanje između predviđenih omeđujućih okvira i omeđujućih okvira stvarnih podataka. IoU vrijednosti 1 predstavlja savršeno preklapanje, dok vrijednost 0 znači da uopće nema preklapanja (Slika 4.1).

Najčešće se bira prag koji će za predviđene omeđujuće okvire koji su manje vrijednosti od praga odbaciti. Biranje praga je osjetljivo jer će sa višim pragom biti detektirano manje lažno pozitivnih (FP), ali potencijalno biti propušteni neki stvarno pozitivni (TP) (Shah, 2023). Za balansiranje rezultata koristi se još metrika prosječne preciznosti u sklopu metrike preklapanja prema uniji. IoU igra bitnu ulogu u procjeni točnosti lokalizacije objekata.



Slika 4.1 - Preklapanje prema uniji (Shah, 2023)

4.2. Preciznost i odziv

Preciznost (engl. *precision*) je omjer pravilno detektiranih objekata i ukupnog broja detekcija, dok je odziv (engl. *recall*) omjer detektiranih objekata i stvarnog broja objekata u skupu podataka (Y. Liu, 2020). Ove metrike su bitne za procjenu performansi modela jer pokazuju veću sliku kod ispravnog identificiranja pozitivnih detekcija, a uz to pomažu minimizirati lažno pozitivne detekcije (Ultralytics, 2024a). Preciznost i odziv se obično računaju na razini klase i mogu se izraziti jednadžbama (4) i (5).

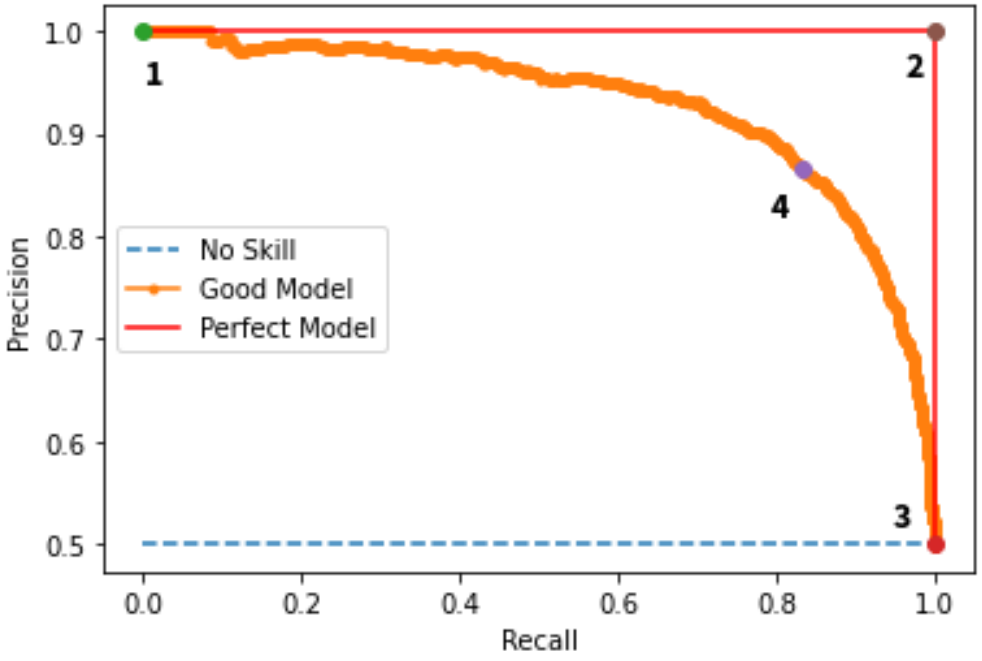
$$Preciznost = \frac{True\ Positives}{True\ Positives + False\ Positives} \tag{4}$$

$$Odziv = \frac{True\ Positives}{True\ Positives + False\ Negatives} \tag{5}$$

4.3. Prosječna preciznost

Prosječna preciznost (engl. *average precision*, skraćeno AP) računa područje ispod *precision-recall* (skraćeno PR) krivulje gdje je y-os preciznost a x-os odziv (Slika 4.2). Može se izračunati za različite IoU pragove, kao što su AP50, AP75 ili AP50-95 (redom AP za prag 50%, AP za prag 75% i prosjek AP sa rasponom pragova 50%-95%). AP mjeri sposobnost modela da otkrije sve pojave određene klase.

Srednja prosječna preciznost (engl. *mean average precision,* skraćeno mAP) je prosjek svih AP vrijednosti za sve klase i sve pragove IoU, također široko korištena metrika za procjenu ukupne performanse modela detekcije objekata (Y. Liu, 2020).



Slika 4.2 – PR krivulja

4.4. F1 ocjena

F1 ocjena (engl. *F1-score*) je harmonijska sredina preciznosti i odziva. Pruža uravnoteženu procjenu performansi modela jer uzima u obzir lažno pozitivne i lažno negativne detekcije (Ultralytics, 2024a). F1 ocjena se računa jednadžbom:

$$F1 = 2 \cdot \frac{Preciznost \cdot Odziv}{Preciznost + Odziv} \tag{6}$$

5. TRENIRANJE

Treniranje YOLO modela započinje odabirom odgovarajućeg skupa podataka, definiranjem parametara treniranja i praćenjem performansi tijekom epoha. Trenira se slikama i pripadajućim podacima; lokacija, veličina i klasa objekata na slikama. Model se trenira tako da bude minimizirana funkcija gubitka koja mjeri razliku predviđene i stvarne labele klase, lokacije objekata i njihove veličine.

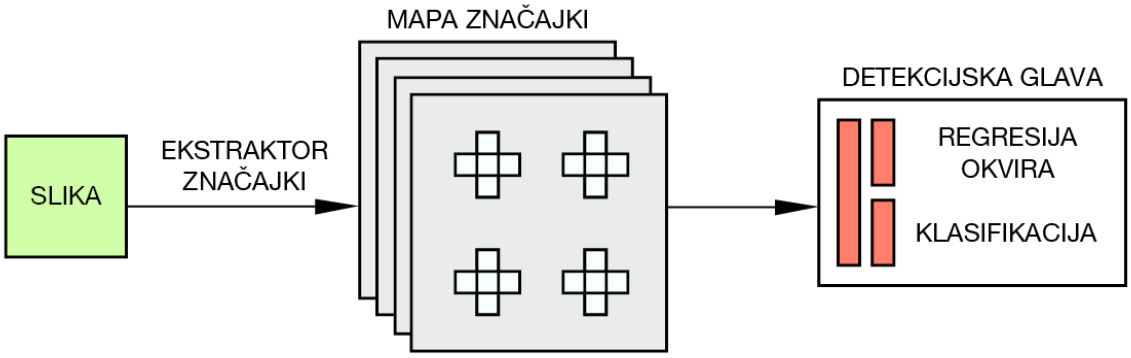
Sam postupak treniranja započinje inicijalizacijom modela i postavljanjem optimizatora prema definiranim parametrima. Zatim se podaci obrađuju u svrhu povećanja učinkovitosti učenja modela. Tijekom treniranja, model se optimizira, a rezultati se prate i *logiraju*, uključujući gubitak, točnost i druge metrike. Raspored učenja prolazi kroz transformacije tijekom epoha, a najvažniji trenuci modela pohranjuju se kako bi se sačuvali najbolji rezultati. Krajem treniranja, korisniku se pruža informacija o trajanju treniranja, a model i metrike uz ostale relevantne podatke o treniranju pohranjuju se u direktorij također specificiran u ispisanim informacijama.

Da bi detekcije bile što uspješnije bitno je trenirati mrežu na velikim skupovima podataka koji pokrivaju različite scenarije i uvjete koji se očekuju u primjerni modela. Skupovi podataka idealno trebaju imati kvalitetne i raznovrsne podatke da bi generalizacija modela bila što bolja. Osim samih skupova podataka, za optimizaciju performansi modela bitni su *fine-tuning* postupci, odnosno prilagodba hiperparametara jer se tako model može prilagoditi specifičnim potrebama primjene.

U svrhu pripreme kvalitetnog skupa podataka za treniranje može se koristiti Roboflow. To je platforma koja osim za izgradnju vlastitog skupa podataka nudi pretragu brojnih skupova podataka te mnoge alate za manipulaciju i obradu; na primjer učitavanje, organiziranje, označavanje, proširivanje skupa podataka te suradnja sa drugim korisnicima. Osim toga, može pojednostaviti proces treniranja s *Roboflow Train* solucijom i ubrzati implementaciju s mnogim opcijama implementacije koristeći *Roboflow Deploy*. Ključna stvar kod pripreme kvalitetnog skupa podataka, osim kvalitetnih slika, je pravilno i dobro označiti te slike kako bi model mogao što bolje učiti (Skalski, 2023), a Roboflow upravo taj proces olakšava i ubrzava.

6. DETEKCIJA OBJEKATA

Klasifikacija i regresija su ključni mehanizmi YOLO modela. Regresijom model lokalizira objekte na način da predviđa koordinate omeđujućih okvira, odnosno koordinate gornjeg lijevog i donjeg desnog kuta. Mreža za klasifikaciju koristi *softmax* funkciju kojom pretvara izlazne vrijednosti u vjerojatnost pripadnosti određenoj klasi. Mreža istovremeno izračunava vjerojatnosti pripadnosti objekta različitim klasama i koordinate omeđujućih okvira. Ovaj pristup omogućuje učinkovitu obradu slike ili videa u jednom prolazu kroz mrežu (Slika 6.1), što za znači da je postupak detekcije brži (Sharma, 2022). To je ujedno i porijeklo naziva algoritma *"You Only Look Once"*, samo jednom pogledaj sliku i odredi gdje su i koji su objekti prisutni.

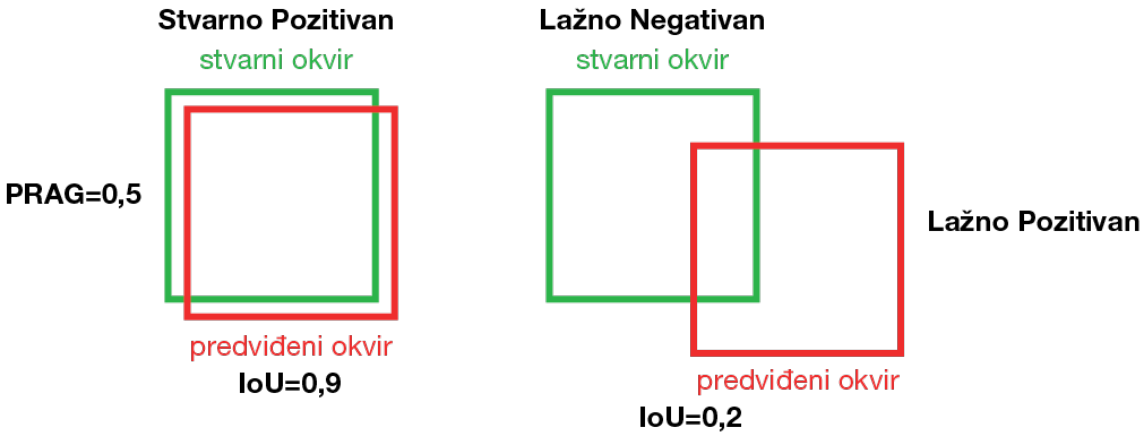


Slika 6.1 - YOLO je model detekcije objekata u jednoj fazi

Mreža za razlikovanje i lokalizaciju objekata koristi konvolucijske slojeve koji ekstrahiraju značajke iz ulazne slike. Detekcijom na različitim razinama mreže omogućava prepoznavanje objekata različitih veličina. Izlazne predikcije mreže sadrže informacije o prisutnosti objekata, njihovim klasama, lokacijama ili omeđujućim okvirima te vjerojatnostima pripadnosti određenoj klasi. Osim klasifikacije i segmentacije za detekciju kao kod v5, v8 iteracija podržava i praćenje objekata i detektiranje poza ljudskih tijela te korištenje pravokutnih okvira koji se prilagođavaju orijentaciji objekta, *oriented bounding boxes* (skraćeno OBB) (Ultralytics, 2024b).

YOLO koristi tehnike post-procesiranja da bi poboljšao performanse. Da bi uklonio duplicirane okvire iz rezultata koristi tehniku *non-maximum suppression* (skraćeno NMS), koja zadržava samo okvir sa najvećom ocjenom sigurnosti za svaku klasu objekta (Bhalerao, 2023). Model također filtrira detekcije sa niskom ocjenom sigurnosti, odnosno

zadržava samo one sa ocjenom većom od praga (engl. *threshold*) ocjene (Slika 6.2). *IoU Threshold* se koristi za filtriranje detekcija koje se previše preklapaju sa ostalim detekcijama, tako da se izbjegne detekcija istog objekta više puta (Bhalerao, 2023).

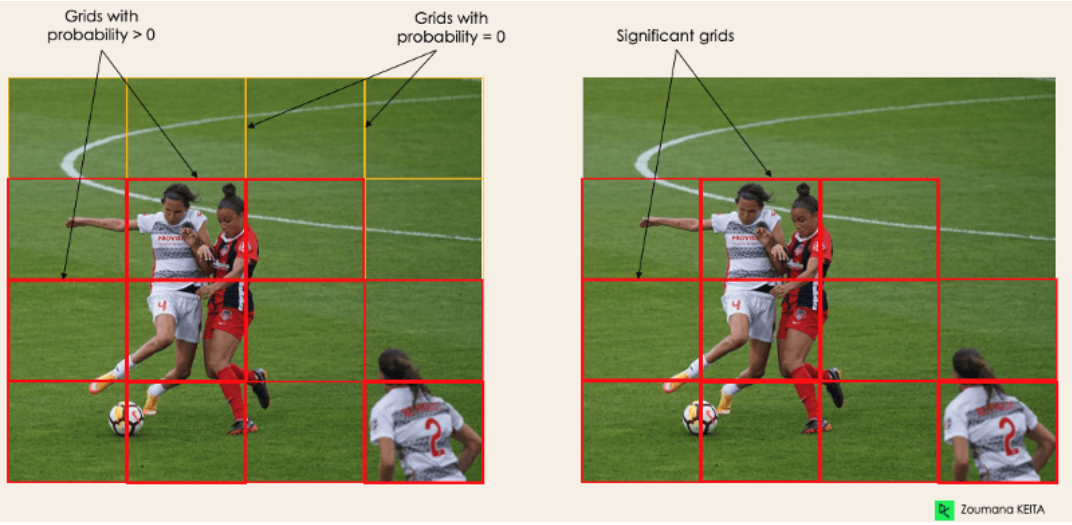


Slika 6.2 - Preklapanja unija; ako je prag 0.5, detekcija lijevo će se prihvatiti, a detekcija desno odbaciti

6.1. Postupak detekcije

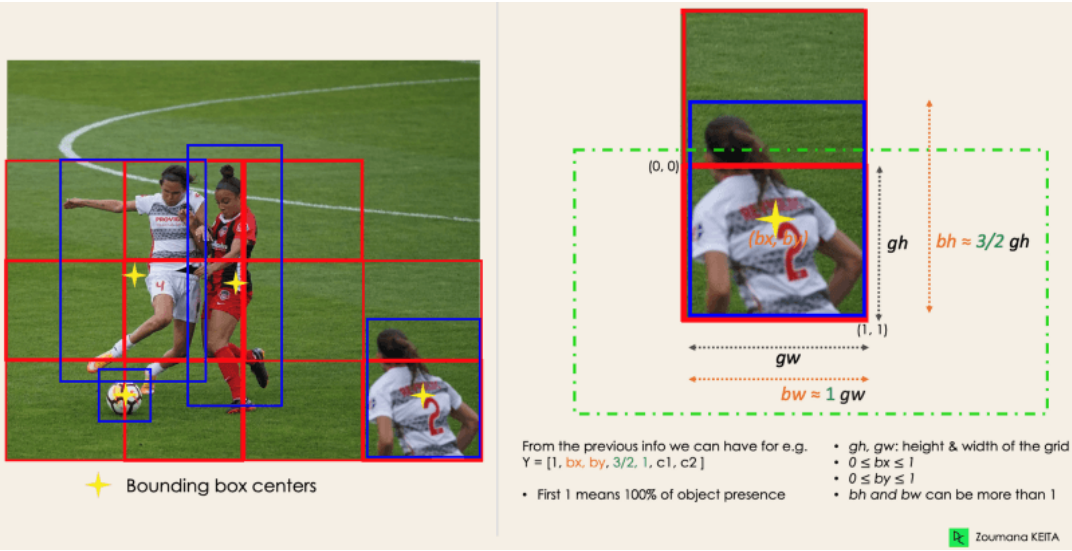
Unutar YOLO duboke neuronske mreže koriste se slojevi konvolucije, *pooling-a* i *upsampling-a* kako bi izvukle značajke iz ulazne slike. Ulazni sloj (engl. *input layer*) modela je slika. Veličina slike može varirati, ali za svaku sliku mora biti unaprijed određena što pomaže mreži prepoznati različite obrasce i strukture unutar slike poput prepoznavanja rubova, tekstura i drugih vizualnih značajki. Prvi korak je ekstrakcija značajki (engl. *feature extraction*).

Slika 6.3 - Podjela ulazne slike na ćelije (Keita, 2022)



Slika 6.4 - Značajne su ćelije koje imaju vjerojatnost pripadanja nekoj klasi (Keita, 2022)

Prvo se originalna slika podijeli na ćelije mreže NxN jednakih formi (Slika 6.3). Svaka ćelija u mreži ima zadaću lokalizacije i predikcije klase objekta koji pokriva, zajedno s vjerojatnosti pripadanja klasi. Sljedeći korak je određivanje okvira omeđivanja koji odgovaraju pravokutnicima koji ističu sve objekte na slici. YOLO određuje atribute tih okvira omeđivanja koristeći jedan regresijski modul. Često se može javiti velik broj kandidata okvira za predikciju, iako nisu svi relevantni. Cilj IoU-a (vrijednosti između 0 i 1) je odbaciti neke okvire kako bi se zadržale samo oni koji su relevantni (Slika 6.4) (Keita, 2022).



Slika 6.5 - Postavljanje omeđujučih okvira oko detektiranih objekata (Keita, 2022)

Nakon što su značajke ekstrahirane iz ulazne slike pomoću slojeva konvolucije, *pooling-a* i *upsampling-a*, dolazi faza skaliranja značajki. Skaliranje značajki osigurava

smanjenje dimenzionalnosti izlučenih značajki smanji i poboljšava sposobnost modela da generalizira i prepozna objekte različitih veličina. Konačno, postavljanje praga za IoU nije uvijek dovoljno jer objekt može imati više okvira s IoU-om iznad praga, a ostavljanje svih tih okvira može uključiti šum. Zato se koristi NMS kako bi se zadržali samo okviri s najvišom vjerojatnošću otkrivanja (Slika 6.5). Na kraju, model generira izlazne predikcije koje sadrže informacije o detektiranim objektima, njihovim lokacijama, klasama i pouzdanostima (Keita, 2022).

Drugim riječima, proces kako mreža obrađuje ulazne podatke i daje izlazne predikcije može biti opisan na sljedeći način: ulazna slika prolazi kroz slojeve konvolucije i *pooling-a*. Svaki sloj u neuronskoj mreži obrađuje podatke na sve složeniji način. Na različitim razinama dubine, mreža generira detekcije objekata. Svaka razina odgovara određenom rasponu veličina objekata. Detekcije s različitih razina povezuju za dobivanje konačnog rezultata koji sadrži sve detektirane objekte. Izlazni sloj generira informacije o detektiranim objektima, uključujući njihove lokacije, odnosno omeđujuće okvire, klase i vjerojatnosti pripadnosti klasi.

7. TRENIRANJE YOLO MODELA NA PRILAGOĐENOM SKUPU PODATAKA

Za demonstraciju rada YOLO algoritma u ovom radu koristit će se modeli YOLOv5s, YOLOv5l, YOLOv8s i YOLOv8l te je potrebno pripremiti i vlastiti skup podataka. Skup podataka *Arthropod Taxonomy Orders Object Detection Dataset* (Drange, n.d.) koji će se koristiti je preuzet sa online platforme Kaggle, pouzdanog izvora skupova podataka koji je namijenjen za istraživanje, analizu i treniranje modela.

7.1. Priprema skupa podataka

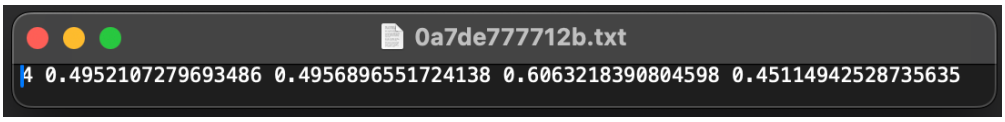
Preuzeti skup podataka ima strukturu koja nije prilagođena za treniranje sa modelima YOLO algoritama. Podijeljen je u direktorije po klasifikaciji, a anotacije su u JSON formatu. Iz originalnih anotacija treba izvući samo podatke potrebne YOLO modelu (Slika 7.1) te ih smjestiti u tekstualne datoteke u obliku **klasa x-centar y-centar širina visina** (Kod 1), gdje su klase isključivo pozitivni cijeli brojevi koji kreću od 0. Sve instance okvira za istu sliku su u istoj tekstualnoj datoteci, a svaka mora biti svom redu. Koordinate centra se računaju iz apsolutnih koordinata iz originalnih anotacija i normaliziraju se. Širina i visina su normalizirane dimenzije omeđujućeg okvira. Normalizirane vrijednosti samo znači da se pretvaraju u relativne koordinate, tako da se daljnjom obradom slika ne trebaju ažurirati.

```
import os
import json

def convert_bbox_to_yolo(size, box):
    dw = 1. / size[0]
    dh = 1. / size[1]
    x = (box[0] + box[2]) / 2.0
    y = (box[1] + box[3]) / 2.0
    w = box[2] - box[0]
    h = box[3] - box[1]
    x = x * dw
    w = w * dw
    y = y * dh
    h = h * dh
    return (x, y, w, h)
```

```
classes_map = {
    'Araneae': 0,
    'Coleoptera': 1,
    'Diptera': 2,
    'Hemiptera': 3,
    'Hymenoptera': 4,
    'Lepidoptera': 5,
    'Odonata': 6
}

dataset_root = '{putanja-do-skupa-podataka}'
labels_dir = '{putanja-labele}'

os.makedirs(labels_dir, exist_ok=True)

for class_name in os.listdir(dataset_root):
    class_dir = os.path.join(dataset_root, class_name)
    if os.path.isdir(class_dir):
        annotations_dir = os.path.join(class_dir, 'annotations')
        if os.path.isdir(annotations_dir):
            for json_file in os.listdir(annotations_dir):
                if json_file.endswith('.json'):
                    json_path = os.path.join(annotations_dir, json_file)
                    with open(json_path) as f:
                        data = json.load(f)

                    image_name = data['asset']['name']
                    img_width = data['asset']['size']['width']
                    img_height = data['asset']['size']['height']
                    txt_file_name = os.path.splitext(image_name)[0] + '.txt'
                    txt_file_path = os.path.join(labels_dir, txt_file_name)

                    with open(txt_file_path, 'w') as out_file:
                        for region in data['regions']:
                            class_id = classes_map[region['tags'][0]]
                            x_min = region['boundingBox']['left']
                            y_min = region['boundingBox']['top']
                            x_max = x_min + region['boundingBox']['width']
                            y_max = y_min + region['boundingBox']['height']
                            b = (x_min, y_min, x_max, y_max)
```

```
                            bb = convert_bbox_to_yolo((img_width, img_height), b)
                            out_file.write(f"{class_id} {bb[0]} {bb[1]} {bb[2]}
{bb[3]}\n")
```

Kod 1 - Pretvaranje anotacija u format prilagođen YOLO algoritmu



Slika 7.1 - Primjer anotacije koju očekuje YOLO

Slike su različitih dimenzija, ali su dosta velike za efikasno treniranje pa se trebaju obraditi na način da se smanje na maksimalnu širinu i visinu od 640 piksela (Kod 2). Time se smanjila veličina skupa podataka sa preko 11 GB na 658 MB, a posljedično i brzina treniranja.

```
from PIL import Image
import os

def resize_to_max(image, max_size):
    width, height = image.size

    if max(width, height) > max_size:
        if width >= height:
            new_width = max_size
            new_height = int(height * (max_size / width))
        else:
            new_height = max_size
            new_width = int(width * (max_size / height))

        resized_image = image.resize((new_width, new_height), Image.ANTIALIAS)
    else:
        resized_image = image

    return resized_image

def process_images(input_dir, output_dir, max_size):
    os.makedirs(output_dir, exist_ok=True)

    for filename in os.listdir(input_dir):
        if filename.endswith('.jpg') or filename.endswith('.png'):
            image_path = os.path.join(input_dir, filename)
```

```
            try:
                image = Image.open(image_path)
                resized_image = resize_to_max(image, max_size)
                resized_image.save(os.path.join(output_dir, filename))
            except Exception as e:
                print(f"Error processing image {filename}: {e}")

dataset_dir = '{putanja-do-skupa-podataka}'
dest_dir = '{izlazna-putanja}'
max_size = 640

process_images(dataset_dir, dest_dir, max_size)
```

Kod 2 - Kod za obradu slika

Kako bi se moglo trenirati ovim podacima treba prilagoditi i hijerarhiju direktorija. Za početak slike treba smjestiti u jedan direktorij *images,* a anotacije u drugi direktorij *labels.* Za svaku instancu ime slike i anotacije se trebaju podudarati. Nakon toga podaci trebaju biti podijeljeni na skupove za treniranje, validaciju i testiranje (Slika 7.2). Taj korak se može dosta olakšati korištenjem Roboflow platforme (Solawetz & Nelson, 2020). Pripremljeni skup podataka kojem su se u prethodnom koraku obradile slike i prilagodile anotacije se može prenijeti na platformu Roboflow, a onda tamo dodatno napraviti razna predprocesiranja te podijeliti podatke na skupove za treniranje, validaciju i testiranje sa željenim omjerima, tako da sami alat pazi na omjere klasa u svakom skupu.

Slika 7.2 - Struktura direktorija skupa podataka

Korijenski direktorij treba sadržavati YAML datoteku sa metapodacima koji su potrebni YOLO modelu: broj klasa, imena klasa poredanih usklađeno sa brojem koji je u

anotacijama (u ovom slučaju abecedno poredani, pa dodijeljeni uzlazni brojevi od 0), putanje do slika za treniranje, validaciju i testiranje (Slika 7.3). Roboflow sam generira YAML datoteku na temelju opcija koje korisnik odabere i vodi računa o takvom imenovanju klasa. Nakon toga skup podataka je spreman za korištenje u Google Colab okruženju.

```yaml
names:
- Araneae
- Coleoptera
- Diptera
- Hemiptera
- Hymenoptera
- Lepidoptera
- Odonata
nc: 7
roboflow:
  license: CC BY 4.0
  project: arthropod-taxonomy-orders
  url: https://universe.roboflow.com/dojo-0jx9t/arthropod-taxonomy-orders/dataset/1
  version: 1
  workspace: dojo-0jx9t
test: /content/arthropod-taxonomy-orders-1/test/images
train: /content/arthropod-taxonomy-orders-1/train/images
val: /content/arthropod-taxonomy-orders-1/valid/images
```

Slika 7.3 - Sadržaj YAML datoteke

7.2. Treniranje modela

Zbog razlike u veličini, kompleksnosti i arhitekturi modela YOLO algoritma za usporedbu treniraju se modeli YOLOv5s, YOLOv5l, YOLOv8s i YOLOv8l na prethodno pripremljenom skupu podataka koji se u Google Colab okruženje jednostavno preuzme sa Roboflow-a. Moguće je pratiti performanse, metrike tijekom treniranja i rezultate treniranja korištenjem Comet ML alata. Praktičan je jer vizualizira metrike generiranjem grafova, sprema najbolje težine treniranja te pohranjuje metrike i razne metapodatke za daljnju analizu.

```
!pip install roboflow
!pip install -q comet_ml
import comet_ml
import torch
from roboflow import Roboflow
import shutil
import os

comet_ml.login() # pri prvom pokretanju zatražit će Comet ML API ključ

api_key = "roboflow-api-kljuc" # ovdje postaviti svoj Roboflow API ključ
```

```
rf = Roboflow(api_key=api_key)
project = rf.workspace().project("arthropod-taxonomy-orders")
```

Kod 3 – Kod za uvoz potrebnih alata, vlastitog skupa podataka sa Roboflow-a i aktivacija Comet alata u Colab okruženje

Nakon instalacije potrebnih alata, potrebno je instalirati YOLO u Colab okruženje. Kod za instalaciju YOLOv5 (Kod 4) je nešto drukčiji od onog za instalaciju YOLOv8 (Kod 5).

```
!git clone https://github.com/ultralytics/yolov5
%cd yolov5
!pip install -qr requirements.txt

dataset = project.version(1).download("yolov5")
```

Kod 4 - Instalacija YOLOv5 u Colab okruženje

```
!pip install ultralytics
from ultralytics import YOLO

dataset = project.version(1).download("yolov8")
```

Kod 5 - Instalacija YOLOv8 u Colab okruženje

U svrhu bolje usporedbe za treniranje svakog modela se koriste isti parametri, 25 epoha, *batch size* 16, maksimalna veličina slika 640 piksela.

7.2.1. YOLOv5s

Nakon što je YOLOv5 instaliran u Colab okruženje, treniranje se jednostavno može pokrenuti jednom instrukcijom (Kod 6). Osim osnovnih parametara (`img`, `batch`, `epochs`) treba postaviti putanju do YAML datoteke skupa podataka preko parametra `data` te odrediti koja varijanta modela se koristi preko parametra `weights`. Parametar `bbox_interval` govori koliko često se bilježe ili procjenjuju granice okvira tijekom treniranja, a parametar `rect` omogućava skaliranje slika tako da zadrže svoj izvorni omjer. Parametri `project` i `name` određuju putanju gdje se spremaju rezultati i bilješke treniranja.

```
!python train.py \
  --img 640 \
```

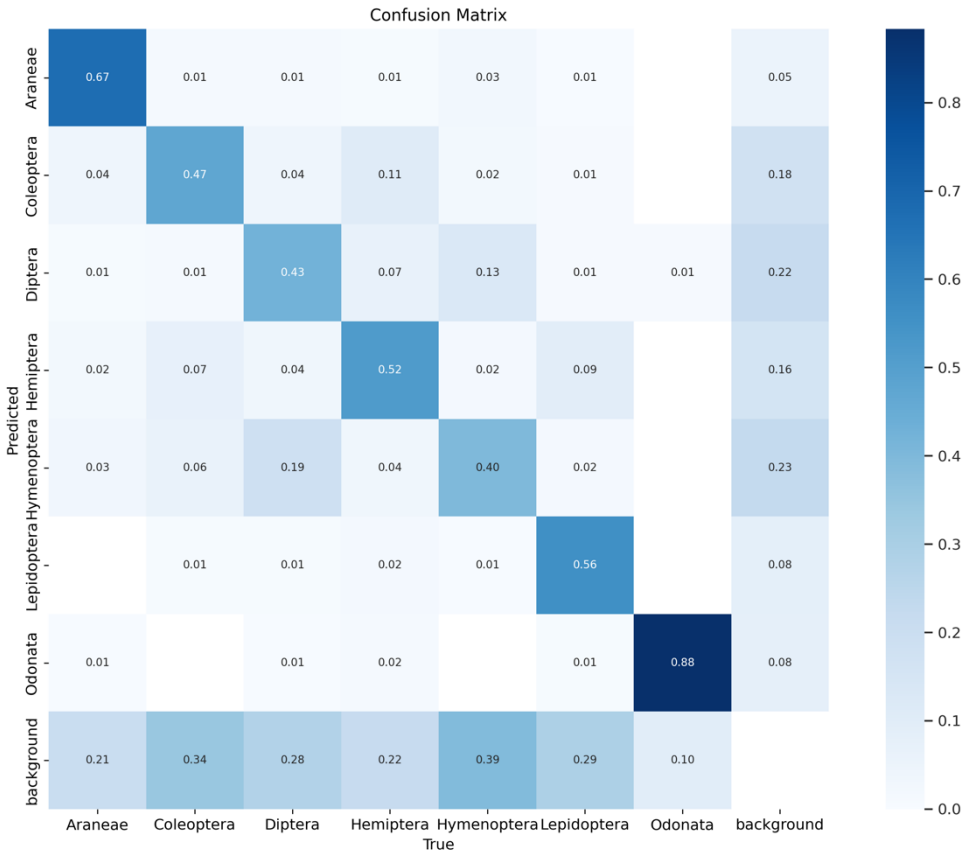
```
--batch 16 \
--epochs 25 \
--data "/content/arthropod-taxonomy-orders-1/data.yaml" \
--weights yolov5s.pt \
--bbox_interval 1 \
--rect \
--project "/content/yolov5/runs/train_yolov5s" \
--name "exp"
```

Kod 6 - Kod za treniranje YOLOv5s modela

YOLOv5s je najmanja i najbrža inačica modela iz YOLOv5 serije, ali pod cijenu malo lošijih rezultata koji su prikazani u tablici (Tablica 2). Treniranje modela je trajalo 1h. Gledajući mAP50, preciznost modela za različite zadatake detekcije varira, a najbolja je za prepoznavanje vretenaca (lat. *Odonata*) i pauka (lat. *Araneae*). Međutim, mAP@0.5:0.95 od 0.405 ukazuje na značajan pad kada se uzme u obzir stroži kriterij (različite vrijednosti IoU praga), što sugerira da model možda ima problema s točnom lokalizacijom objekata na različitim razinama preciznosti. Ovi rezultati općenito ukazuju na potrebu za dodatnom obukom ili prilagodbom hiperparametara.

Tablica 2 - Rezultati validacije modela YOLOv5s

Sažetak modela: 157 slojeva, 7029004 parametara, 0 gradijenata, 15.8 GFLOPs						
Klasa	**Slike**	**Instance**	**P**	**R**	**mAP50**	**mAP50-95**
sve	2000	2646	0.658	0.608	0.619	0.405
Araneae	2000	329	0.775	0.687	0.766	0.479
Coleoptera	2000	400	0.581	0.534	0.539	0.356
Diptera	2000	344	0.519	0.532	0.495	0.356
Hemiptera	2000	408	0.624	0.583	0.56	0.378
Hymenoptera	2000	456	0.501	0.461	0.431	0.241
Lepidoptera	2000	374	0.779	0.586	0.646	0.428
Odonata	2000	335	0.828	0.874	0.896	0.596



Slika 7.4 - Matrica konfuzije za model YOLOv5s

7.2.2. YOLOv5l

Treniranje YOLOv5l se pokreće isto kao varijanta v5s, samo se za argument `weights` postavi YOLOv5l (Kod 7).

```
!python train.py \
  --img 640 \
  --batch 16 \
  --epochs 25 \
  --data "/content/arthropod-taxonomy-orders-1/data.yaml" \
  --weights yolov5l.pt \
  --bbox_interval 1 \
  --rect \
  --project "/content/yolov5/runs/train_yolov5l" \
  --name "exp"
```

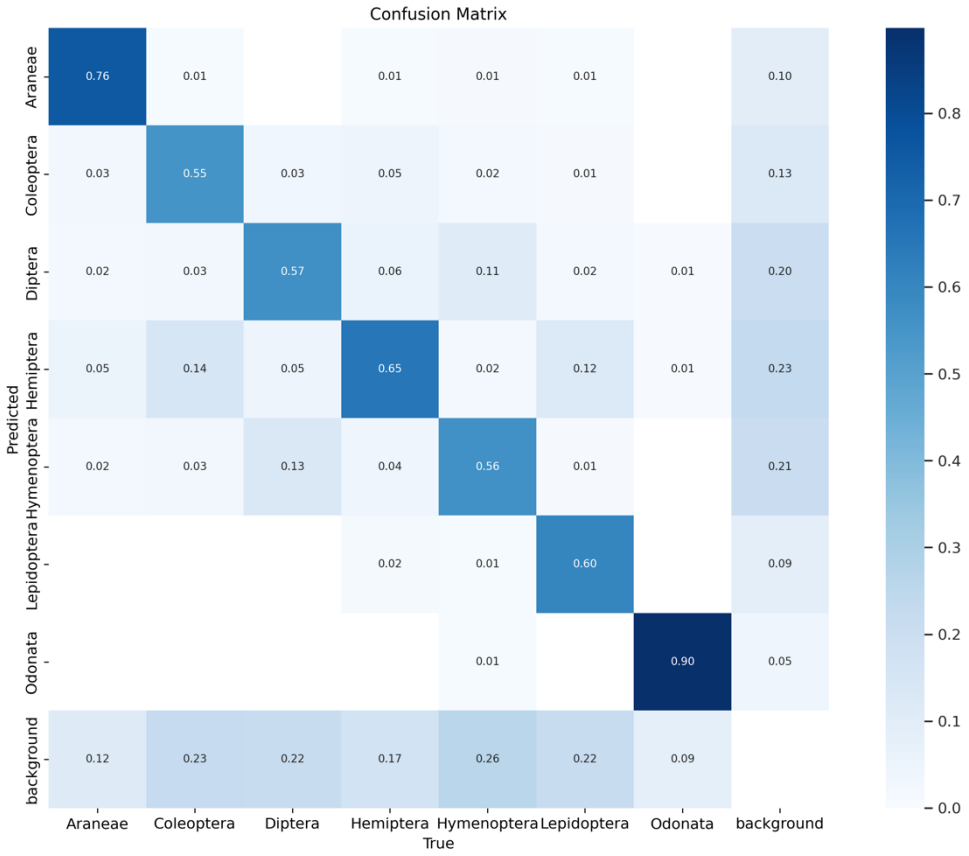
Kod 7 - Kod za treniranje YOLOv5l modela

Rezultati YOLOv5l modela pokazuju poboljšanje u preciznosti (mAP@50 je 74.1%) i općim performansama u odnosu na YOLOv5s model što se vidi u tablici (Tablica 3). Nešto

je malo bolja i lokalizacija objekata gledajući na metriku mAP@0.5:0.95 koja je 56.2%. I ovaj model najbolje rezultate ima kod prepoznavanja paukova i vretenaca, dok za druge klase ima niže performanse i to se lakše vidi iz matrice konfuzije (Slika 7.5). Primjerice, klase opnokrilci (lat. *Hymenoptera*) i dvokrilci (lat. *Diptera*) ukazuju na potrebu za daljnjim poboljšanjima. Pošto je YOLOv5l model složeniji, povećanje epoha treniranja bi dalo još bolje rezultate, pogotovo jer je skup podataka velik; 7002 slike za treniranje, 2000 slika za validaciju te 1006 slika za testiranje. Ovo treniranje je trajalo 1h i 48 min.

Tablica 3 - Rezultati validacije modela YOLOv5l

Sažetak modela: 267 slojeva, 46140588 parametara, 0 gradijenata, 107.7 GFLOPs						
Klasa	**Slike**	**Instance**	**P**	**R**	**mAP50**	**mAP50-95**
sve	2000	2646	0.781	0.664	0.741	0.562
Araneae	2000	329	0.884	0.766	0.876	0.662
Coleoptera	2000	400	0.767	0.557	0.698	0.521
Diptera	2000	344	0.634	0.602	0.625	0.494
Hemiptera	2000	408	0.641	0.672	0.672	0.535
Hymenoptera	2000	456	0.736	0.566	0.624	0.423
Lepidoptera	2000	374	0.868	0.596	0.754	0.541
Odonata	2000	335	0.934	0.886	0.935	0.762



Slika 7.5 - Matrica konfuzije za model YOLOv5l

7.2.3. YOLOv8s

Naredba za pokretanje treniranja YOLOv8 malo se razlikuje od v5 varijanti, ali u pravilu se isti parametri određuju (Kod 8). Parametar `imgsz` je isto što i `img` kod v5, a parametar `plots` se koristi za omogućavanje generiranja različitih grafičkih prikaza vezanih za treniranje.

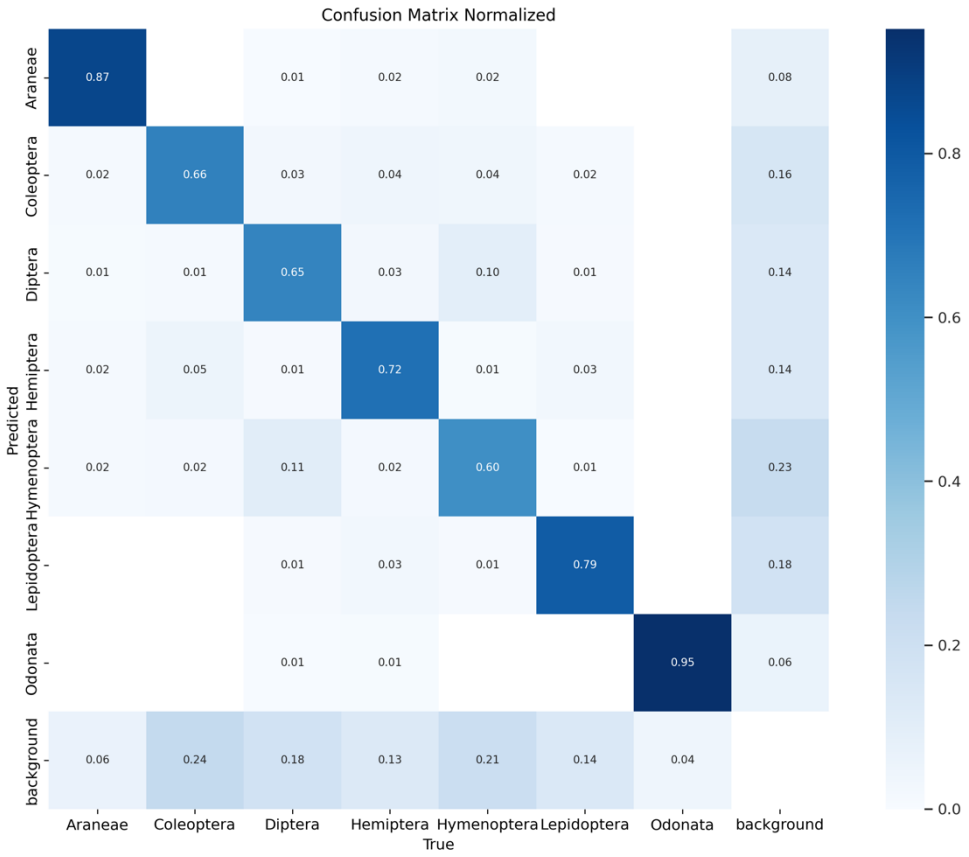
```
!yolo task=detect \
  mode=train \
  model=yolov8s.pt \
  data='/content/arthropod-taxonomy-orders-1/data.yaml' \
  epochs=25 \
  imgsz=640 \
  plots=True
```

Kod 8 - Kod za treniranje YOLOv8s modela

YOLOv8s je mali model iz YOLOv8 serije i ima skok u rezultatima u odnosu na v5 modele, vidljivo iz tablice (Tablica 4). Metrika mAP@0.5: 0.813 pokazuje visoku preciznost modela za detekciju objekata. Ponavlja se obrazac dobrih performansi kod klasa vretenci i pauci te nižih performansi kod klasa opnokrilci i dvokrilci (Slika 7.6).

Tablica 4 - Rezultati validacije modela YOLOv8s

Sažetak modela (fused): 168 slojeva, 11128293 parametara, 0 gradijenata, 28.5 GFLOPs						
Klasa	**Slike**	**Instance**	**P**	**R**	**mAP50**	**mAP50-95**
sve	2000	2646	0.867	0.719	0.813	0.662
Araneae	322	329	0.921	0.854	0.932	0.762
Coleoptera	283	400	0.834	0.627	0.729	0.582
Diptera	278	344	0.807	0.628	0.716	0.611
Hemiptera	315	408	0.858	0.696	0.811	0.674
Hymenoptera	294	456	0.829	0.562	0.688	0.503
Lepidoptera	270	374	0.868	0.743	0.842	0.655
Odonata	311	335	0.951	0.924	0.972	0.843



Slika 7.6 - Matrica konfuzije za model YOLOv8s

7.2.4. YOLOv8l

Naredba za treniranje v8l modela je, opet ista kao kod v8s, uz promjenu argumenta `model` (Kod 9).

```
!yolo task=detect \
  mode=train \
  model=yolov8l.pt \
  data='/content/arthropod-taxonomy-orders-1/data.yaml' \
  epochs=25 \
  imgsz=640 \
  plots=True
```

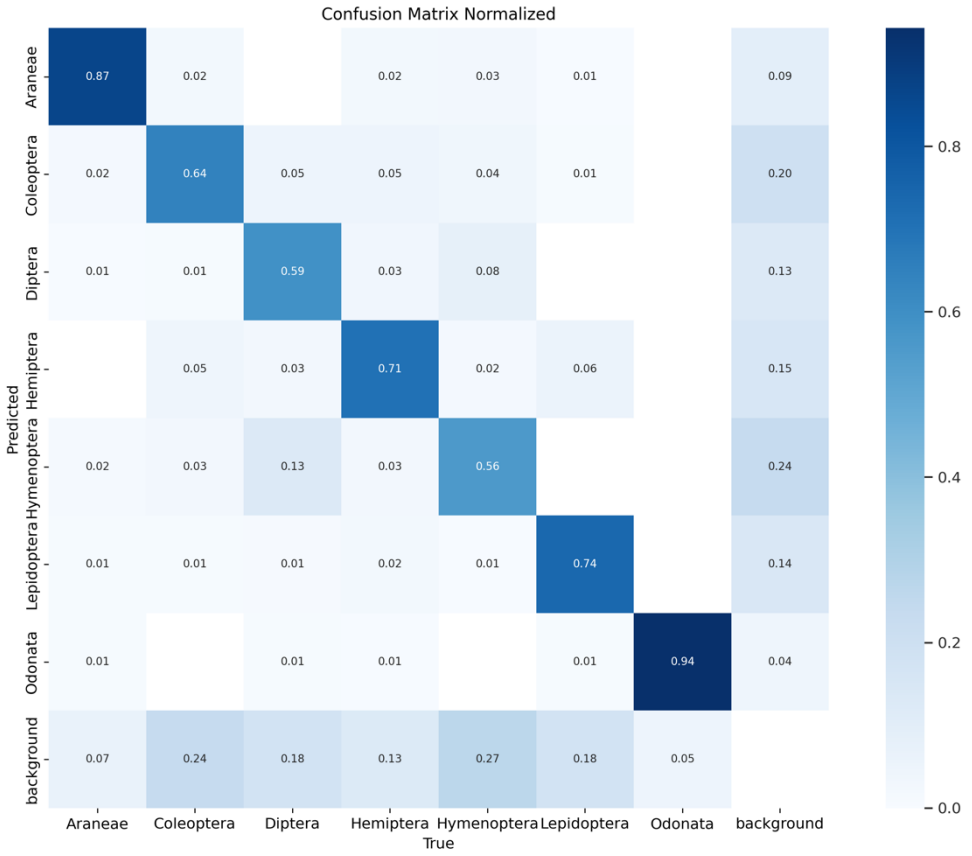
Kod 9 - Kod za treniranje YOLOv8l modela

YOLOv8l je veći model iz YOLOv8 serije, a fokus mu je na maksimalnoj preciznosti i točnosti. Ovaj model bi bio idealan za složene zadatke detekcije gdje je ključna visoka točnost, ali zahtijeva i više treniranja, pogotovo sa kompleksnijim i većim skupovima podataka. U ovoj usporedbi se nije istakao svojim performansama i nije dosegao svoj

potencijal. Ako se radi sa većim skupovima podataka sa više klasa, raznovrsnijim objektima ili kompleksnijim scenama, ovaj model bi mogao bolje iskoristiti svoju veću arhitekturu (više slojeva, više parametara) za postizanje boljih rezultata.

Tablica 5 - Rezultati validacije modela YOLOv8l

Sažetak modela (fused): 268 slojeva, 43612005 parametara, 0 gradijenata, 164.8 GFLOPs						
Klasa	**Slike**	**Instance**	**P**	**R**	**mAP50**	**mAP50-95**
sve	2000	2646	0.811	0.719	0.796	0.66
Araneae	322	329	0.843	0.857	0.913	0.764
Coleoptera	283	400	0.747	0.665	0.734	0.587
Diptera	278	344	0.779	0.619	0.705	0.603
Hemiptera	315	408	0.807	0.686	0.783	0.681
Hymenoptera	294	456	0.734	0.546	0.662	0.481
Lepidoptera	270	374	0.826	0.722	0.81	0.645
Odonata	311	335	0.939	0.937	0.966	0.86



Slika 7.7 - Matrica konfuzije za model YOLOv8l

7.2.5. Usporedba performansi

Modeli koji su trenirani za ovu usporedbu daju različite rezultate. YOLOv5 ima solidne rezultate s obzirom na veliki skup podataka, ali YOLOv8 se pokazao efikasnijim po pitanju performansi. Drugim riječima, v8 modeli za iste parametre treniranja (25 epoha, 16 *batch size*, 2 *dataloader workers*) imaju bolje rezultate po metrikama, ali zato duže traje treniranje zbog razlika u arhitekturi. Slabiji rezultati po nekim klasama kod svih modela ukazuju na slabost u skupu podataka više nego modela, jer su ti razredi člankonožaca sami po sebi teži za razlikovati od ostalih.

Tablica 6 - Usporedba metrika treniranih modela

	YOLOv5s	**YOLOv5l**	**YOLOv8s**	**YOLOv8l**
Trajanje treninga	1h	1h 48min	1h 18min	2h 35min

Preciznost	0,658	0,781	0,867	0,811
Odziv	0,608	0,664	0,719	0,719
Max F1 na pragu	0,63 na 0,251	0,71 na 0,352	0,78 na 0,472	0,76 na 0,388
mAP50	0,619	0,741	0,813	0,796
mAP50-95	0,405	0,562	0,662	0,66
Predprocesiranje	0,4 ms	0,4 ms	0,4 ms	0,3 ms
Detekcija	9,1 ms	21,5 ms	6,8 ms	6,8 ms
Postprocesiranje	1,4 ms	1,8 ms	1,8 ms	1,8 ms
Ukupna brzina	10,9 ms	23,7 ms	9 ms	8,9 ms
Broj parametara	7 029 004	46 140 588	11 128 293	43 612 005
Veličina modela	14,4 MB	92,8 MB	22,5 MB	87,6 MB

YOLOv5l i YOLOv8l imaju značajno veću veličinu modela u usporedbi s YOLOv5s i YOLOv8s, što može utjecati na zahtjeve za pohranu i distribuciju modela. Sami izbor varijante modela, s ili l, ovisi o kompleksnosti zadatka i skupa podataka, a i hardveru na kojem se izvodi.

YOLOv8s se po rezultatima usporedbe čini kao odličan izbor za većinu zadataka jer pruža dobar balans između brzine, performansi i trajanja treninga, a YOLOv5s je odličan izbor kad je važna manja veličina modela. Modeli v5 su dobar izbor i po pitanju jednostavnosti implementacije, bržeg prototipiranja i kada su projekti manje zahtjevni. S druge strane, YOLOv8l ima najbrže detekcije i zahtijeva najviše vremena za treniranje. Zbog svoje kompleksnije arhitekture može se izabrati kad se radi sa puno složenijim i većim skupom podataka gdje je cijena vremena i performansi manje bitna.

Zaključak

U ovom radu istražena je primjena YOLO algoritma za detekciju objekata u kontekstu dubokog učenja. Potvrđeno je da su YOLO modeli učinkoviti za detekciju objekata u stvarnom vremenu i da pružaju fleksibilnost potrebnu za primjenu u različitim industrijama i situacijama. Unatoč njihovim prednostima, preostaje izazov prilagodbe modela različitim uvjetima i potrebama da bi se postigli optimalni rezultati.

Analiza verzija YOLOv5 i YOLOv8 pokazala je kako njihove razlike u arhitekturi mogu utjecati na rezultate i performanse, posebno pri korištenje vlastitog skupa podataka. Rad također naglašava važnost pažljivog odabira odgovarajućeg modela i pripreme podataka kako bi se balansirala brzina treninga, brzina detekcije, dobrih metrika i veličine modela. Usporedba performansi između navedenih verzija YOLO modela pokazala je da nijedan model nije univerzalno najbolji, već da su specifični zahtjevi zadatka presudni u odabiru odgovarajućeg modela.

Popis kratica

AI	*Artificial Intelligence*	umjetna inteligencija
AP	*Average Precision*	prosječna preciznost
CLoU	*Complete Intersection over Union*	potpuna unija presjeka
CNN	*Convolutional Neural Network*	konvolucijska neuronska mreža
CSPNet	Cross Stage Partial Network	djelomična mreža kroz faze
FN	*False Negative*	lažno negativan
FP	*False Positive*	lažno pozitivan
IoU	*Intersection over Union*	unija presjeka
MAE	*Mean Average Error*	srednja apsolutna pogreška
mAP	*Mean Average Precision*	srednja prosječna preciznost
MSE	*Mean Squared Error*	srednja kvadratna pogreška
NMS	*Non-Maximum Suppression*	suzbijanje ne-maksimuma
OBB	O*riented Bounding Boxes*	orijentirani granični okviri
P	*Precision*	preciznost
PANet	*Path Aggregation Network*	mreža za agregaciju putanja
PR	*Precision-Recall curve*	graf preciznost-odziv
R	*Recall*	odziv
ReLU	*Rectified Linear Unit*	rektificirana linearna jedinica
SiLU	*Sigmoid-weighted Linear Unit*	sigmoidno ponderirana linearna jedinica
SPP	*Spacial Pyramid Pooling*	prostorno piramidno grupiranje
SPPF	*Spacial Pyramid Pooling Feature*	značajka prostornog piramidnog grupiranja
YOLO	*You Only Look Once*	pogledaš samo jednom

Popis slika

Popis tablica

Popis kôdova

Literatura

Ayush, P. (2024, May 8). *Introduction to YOLO Object Detection: Understanding the Basics*. Pareto. https://pareto.ai/blog/yolo-object-detection

Bhalerao, C. (2023, November 2). *A Deep Dive Into Non-Maximum Suppression (NMS)*. Built In. https://builtin.com/machine-learning/non-maximum-suppression

Bochkovskiy, A., Wang, C.-Y., & Liao, H.-Y. M. (2020). *YOLOv4: Optimal Speed and Accuracy of Object Detection* (arXiv:2004.10934; Version 1). arXiv. https://doi.org/10.48550/arXiv.2004.10934

Casas, E., Ramos, L., Bendek, E., & Rivas-Echeverria, F. (2024). YOLOv5 vs. YOLOv8: Performance Benchmarking in Wildfire and Smoke Detection Scenarios. *Journal of Image and Graphics*, *12*(2), 127–136. https://doi.org/10.18178/joig.12.2.127-136

Cherifi, I. (2022, October 28). *YOLO v5 model architecture [Explained]*. OpenGenus IQ: Learn Algorithms, DL, System Design. https://iq.opengenus.org/yolov5/

Craig, L., & Awati, R. (2024, January). *What is a convolutional neural network (CNN)?* Enterprise AI. https://www.techtarget.com/searchenterpriseai/definition/convolutional-neural-network

Deci. (2023, December 4). *YOLOv8 vs. YOLO-NAS Showdown: Exploring Advanced Object Detection*. Deci. https://deci.ai/blog/yolov8-vs-yolo-nas-showdown-exploring-advanced-object-detection/

Doherty, J., Gardiner, B., Kerr, E., Siddique, N., & Manvi, S. (2022). *Comparative Study of Activation Functions and Their Impact on the YOLOv5 Object Detection Model* (pp. 40–52). https://doi.org/10.1007/978-3-031-09282-4_4

Drange, G. (n.d.). *Arthropod Taxonomy Orders Object Detection Dataset*. Kaggle. Retrieved July 10, 2024, from https://www.kaggle.com/datasets/mistag/arthropod-taxonomy-orders-object-detection-dataset

Ethical Implications of Computer Vision Technology in 2023. (2023, June 13). AUGMENTED A.I. https://www.augmentedstartups.com/blog/exploring-the-ethical-implications-of-computer-vision-technology-in-2023

He, K., Zhang, X., Ren, S., & Sun, J. (2014). *Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition* (Vol. 8691, pp. 346–361). https://doi.org/10.1007/978-3-319-10578-9_23

Keita, Z. (2022). *YOLO Object Detection Explained: A Beginner's Guide*. Datacamp. https://www.datacamp.com/blog/yolo-object-detection-explained?dc_referrer=https%3A%2F%2Fwww.bing.com%2F

Liu, S., Qi, L., Qin, H., Shi, J., & Jia, J. (2018). *Path Aggregation Network for Instance Segmentation* (arXiv:1803.01534). arXiv. http://arxiv.org/abs/1803.01534

Liu, Y. (2020, May 18). The Confusing Metrics of AP and mAP for Object Detection. *Medium*. https://yanfengliux.medium.com/the-confusing-metrics-of-ap-and-map-for-object-detection-3113ba0386ef

Mirani, I. K., Tianhua, C., Khan, M. A. A., Aamir, S. M., & Menhaj, W. (2022). *Object Recognition in Different Lighting Conditions at Various Angles by Deep Learning Method*. *1*.

Rajput, M. (2020, July 1). YOLO V5—Explained and Demystified – Towards AI. *Towards AI*. https://towardsai.net/p/computer-vision/yolo-v5%e2%80%8a-%e2%80%8aexplained-and-demystified, https://towardsai.net/p/computer-vision/yolo-v5%e2%80%8a-%e2%80%8aexplained-and-demystified

Shah, D. (2023, May 30). *Intersection over Union (IoU): Definition, Calculation, Code*. V7LABS. https://www.v7labs.com/blog/intersection-over-union-guide, https://www.v7labs.com/blog/intersection-over-union-guide

Sharma, A. (2022, April 11). Understanding a Real-Time Object Detection Network: You Only Look Once (YOLOv1). *PyImageSearch*. https://pyimagesearch.com/2022/04/11/understanding-a-real-time-object-detection-network-you-only-look-once-yolov1/

Shi, Z., Sang, M., Huang, Y., Xing, L., & Liu, T. (2022). Defect Detection of MEMS Based on Data Augmentation, WGAN-DIV-DC, and a YOLOv5 Model. *Sensors*, *22*(23), Article 23. https://doi.org/10.3390/s22239400

Skalski, P. (2023, January 10). *How to Train YOLOv8 Object Detection on a Custom Dataset*. Roboflow Blog. https://blog.roboflow.com/how-to-train-yolov8-on-a-custom-dataset/

Solawetz, J., & Nelson, J. (2020, June 10). *How to Train a YOLOv5 Model On a Custom Dataset*. Roboflow Blog. https://blog.roboflow.com/how-to-train-yolov5-on-a-custom-dataset/

Ultralytics. (2024a, June 2). *YOLO Performance Metrics*. Ultralytics. https://docs.ultralytics.com/guides/yolo-performance-metrics

Ultralytics. (2024b, June 10). *OBB*. Ultralytics. https://docs.ultralytics.com/tasks/obb

Ultralytics. (2024c, June 19). *Architecture Summary*. https://docs.ultralytics.com/yolov5/tutorials/architecture_description

Ultralytics. (2024d, July 4). *Ultralytics YOLOv8 Tasks*. Ultralytics. https://docs.ultralytics.com/tasks

What Is Artificial Intelligence (AI)? (n.d.). Google Cloud. Retrieved August 15, 2024, from https://cloud.google.com/learn/what-is-artificial-intelligence

Yang, W., Chen, Y., Huang, C., & Gao, M. (2018). Video-Based Human Action Recognition Using Spatial Pyramid Pooling and 3D Densely Convolutional Networks. *Future Internet*, *10*, 115. https://doi.org/10.3390/fi10120115