S V E U Č I L I Š T E U Z A G R E B U

FAKULTET STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

Filip Zadro

Zagreb, 2024.

SVEUČILIŠTE U ZAGREBU

FAKULTET STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

Mentor:

Izv. prof. dr. sc. Tomislav Stipančić, dipl. ing.

Student:

Filip Zadro

Zagreb, 2024.

Izjavljujem da sam ovaj rad izradio samostalno koristeći znanja stečena tijekom studija i navedenu literaturu.

Zahvaljujem mentoru izv. prof. dr. sc. Tomislavu Stipančiću, mag. ing. te asistentu Leonu Korenu, mag. ing. na dostupnosti, pristupačnosti, korisnim savjetima i pomoći prilikom izrade ovog diplomskog rada.

Zahvaljujem svojim roditeljima Danijelu i Katici, sestri Eni, bakama i djedovima te cijeloj obitelji Zadro na podršci tijekom svih godina studiranja.

Zahvaljujem svojim prijateljima s fakulteta i iz studentskog doma što su mi uljepšali boravak na fakultetu i izvan fakulteta.

Zahvaljujem svojim cimerima: Josipu Hraniću, Antoniju Zadri, Valentini Zadro, Steli Zadro, Luki Kolaru i Domagoju Filaru, koji su sa mnom dijelili dobre i loše trenutnke.

Filip Zadro

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE
Središnje povjerenstvo za završne i diplomske ispite
Povjerenstvo za diplomske ispite studija strojarstva za smjerove:
Proizvodno inženjerstvo, inženjerstvo materijala, industrijsko inženjerstvo i menadžment, mehatronika i robotika, autonomni sustavi i računalna inteligencija

Sveučilište u Zagrebu Fakultet strojarstva i brodogradnje	
Datum	Prilog
Klasa: **602 - 04 / 24 - 06 / 1**	
Ur.broj: **15 - 24 -**	

DIPLOMSKI ZADATAK

Student: **Filip Zadro** JMBAG: 0035217548

Naslov rada na hrvatskom jeziku: **Razvoj sustava za prepoznavanje geometrijskih tijela korištenjem YOLOv8 modela**

Naslov rada na engleskom jeziku: **Development of a system for recognizing geometric bodies using the YOLOv8 model**

Opis zadatka:

Računalni vid se koristi u različite svrhe, uključujući rješavanje zadataka koji uključuju detekciju, klasifikaciju, prepoznavanje i slično. Pritom se često koriste različite metode umjetne inteligencije koje prilikom treninga modela koriste odgovarajuće podatke.

Cilj ovog rada je razviti sustav za prepoznavanje geometrijskih tijela za popularnu igru Dungeons & Dragons (D&D) koristeći YOLOv8 duboku neuronsku mrežu za detekciju objekata. Potrebno je razviti vlastitu neuronsku mrežu i istrenirati je kako bi prepoznala geometrijske oblike s 4, 6, 8, 10, 12 i 20 stranica, koji se često koriste u igri D&D.

U radu je potrebno:

- proučiti YOLOv8 arhitekturu za detekciju objekata te istražiti proces treniranja neuronskih mreža
- pripremiti skup podataka koji sadrži slike različitih geometrijskih tijela s 4, 6, 8, 10, 12 i 20 stranica za treniranje i evaluaciju modela
- trenirati mrežu koristeći pripremljeni skup podataka kako bi se postigla visoka točnost prepoznavanja geometrijskih tijela
- evaluirati razvijeni model u sklopu Laboratorija za projektiranje izradbenih i montažnih sustava te dati kritički osvrt.

U radu je potrebno navesti korištenu literaturu i eventualno dobivenu pomoć.

Zadatak zadan:	Datum predaje rada:	Predviđeni datumi obrane:
9. svibnja 2024.	11. srpnja 2024.	15. – 19. srpnja 2024.

Zadatak zadao: Izv. prof. dr. sc. Tomislav Stipančić

Predsjednik Povjerenstva: Prof. dr. sc. Ivica Garašić

SADRŽAJ

POPIS SLIKA

POPIS TABLICA

SAŽETAK

Računalni vid se koristi u različite svrhe, uključujući prepoznavanje, detekciju, klasifikaciju. Prilikom treniranja neuronske mreže koriste se različite metode umjetne inteligencije, koje koriste prikupljene podatke za učenje modela. Cilj ovog diplomskog rada je razviti sustav prepoznavanje geometrijskih tijela za društvenu igru Dungeons&Dragons (D&D) koristeći YOLOv8 paket. Potrebno je razviti neuronsku mrežu. prikupiti potrebne podatke te prikupljene podatke je potrebno obilježiti i klasificirati, nakon toga provesti treniranje te izvršiti analizu rezultata. Skup podataka se sastoji od „kockica“ D4, D6, D8, D10, D12, D20 i D100. Treniranje modela se provodi s varijabilnim skupom podataka, tako da će se skup podataka povećavati novim podacima nakon svakog treninga, s ciljem praćenja poboljšavanja neuronske mreže.

Ključne riječi: Računalni vid, Neuronska mreža, YOLOv8, Klasifikacija

SUMMARY

Computer vision is used for various purposes, including recognition, detection, and classification. During the training of a neural network, different artificial intelligence methods are utilized, using collected data to train the model. The aim of this thesis is to develop a geometric shape recognition system for the board game Dungeons & Dragons (D&D) using the YOLOv8 package. It is necessary to develop a neural network, collect the required data, label and classify the collected data, conduct the training, and perform an analysis of the results. The dataset consists of "dice" D4, D6, D8, D10, D12, D20, and D100. The model training is conducted with a variable dataset, in such a way that the dataset will be increased with new data after each training session, with the goal of monitoring the improvement of the neural network.

Keywords: Computer vision, Neural network, YOLOv8, Classification

1. UVOD

Razvoj neuronskih mreže je krenuo još 1950-tih pojavom perceptora, ali zbog nemogućnosti računala da barata s velikim brojem podataka, nije zaživio veliki uspjeh. Kako u današnje vrijeme računala mogu obrađivati vrlo velike količine podataka u kratkome vremenu (značajno poboljšanje procesora, grafičkih kartica i pojava mikroprocesora), došlo je do značajnoga razvoja neuronskih mreža i pojave velikih neuronskih mreža s mnogo slojeva, takozvane duboke neuronske mreže (engl. „deep neural networks“). Kod umjetne inteligencije, vrlo je čest i bitan računalni vid, koji je u posljednjem desetljeću postao vrlo efikasan u rješavanju inženjerskih problema u robotici, zdravstvu, medicini, razvoju autonomnih vozila, pametnom upravljanju i nadzoru. Zadaće vizualnog raspoznavanja, kao što su klasifikacija slika, te lokalizacija i detekcija objekata, čine temelj pri rješavanju spomenutih problema.

Zbog toga su neuronske mreže ključni dio dubokih algoritama u području računalnog vida. Kako bi se bolje razumjeli modeli neuronskih mreža, rad će započeti s prikazom klasičnih neuronskih mreža. Na jasan i slikovit način bit će objašnjeni model neurona, arhitektura mreže te metode učenja neuronskih mreža. Cilj ovog poglavlja je uvesti temeljne koncepte dubokih modela poput slojeva mreže, bioloških i umjetnih neurona, arhitekture umjetnih mreža te njihovih funkcija aktivacije. Priprema slika za treniranje neuronske mreže, treniranje neuronske mreže i na kraju implementacija neuronske mreže na stvarni problem.

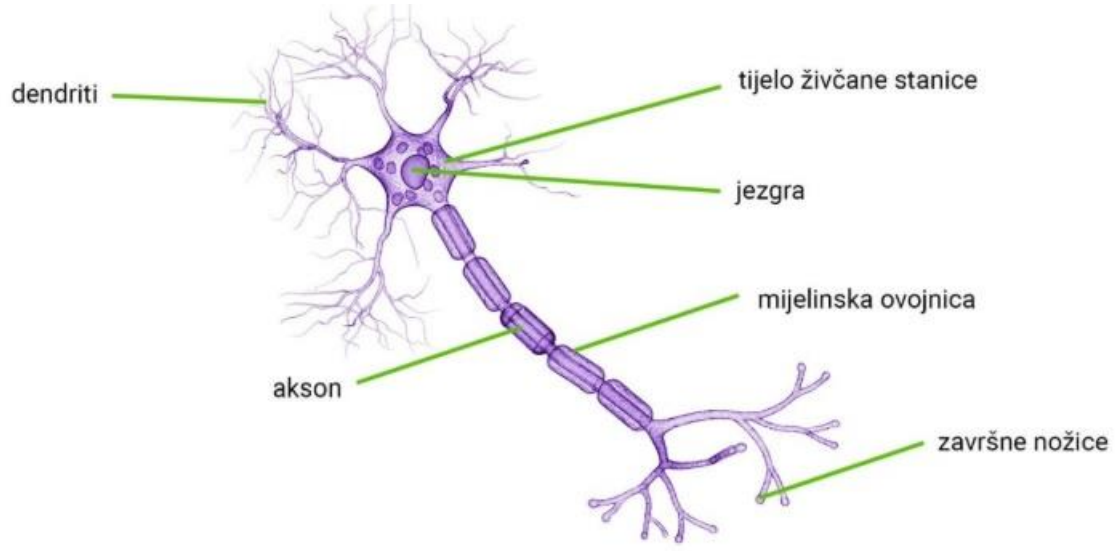
2. NEURONSKE MREŽE

2.1. Umjetne neuronske mreže

2.1.1. Princip rada

Umjetne neuronske mreže sastoje se od građevnih jedinica, organiziranih u različite slojeve, koji se nazivaju umjetni neuroni. Sama ideja umjetne neuronske mreže izvedena je iz bioloških neurona mozga. Neuroni u mozgu povezani su sinapsama i neprestano komuniciraju s tisućama drugih neurona koji ih okružuju. Potrebno je vrlo pažljivo birati arhitekturu i parametre umjetne neuronske mreže kako bi se što vjernije reproduciralo biološko učenje, koje se temelji na prijenosu informacija između neurona.

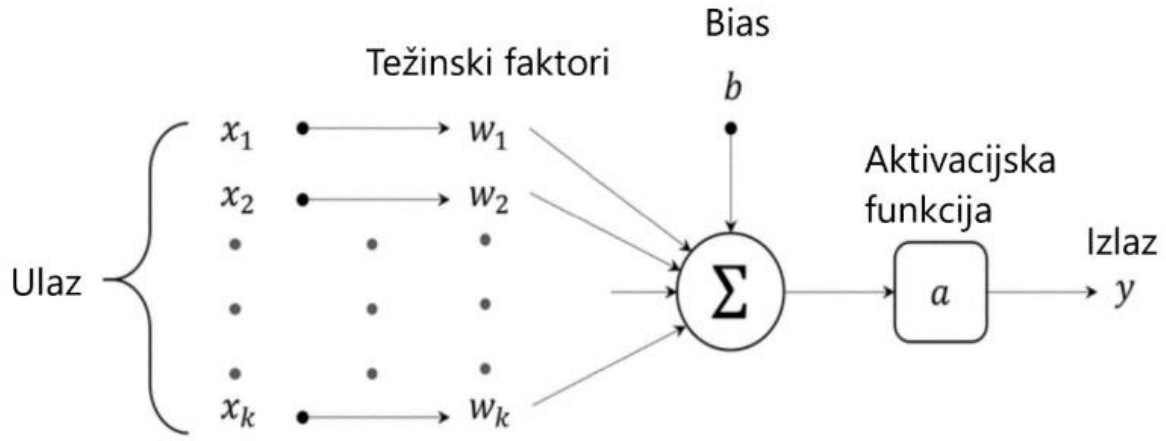
Svaki biološki neuron prima dolazne signale od susjednih neurona putem dendrita. Izmijenjeni izlazni signal se zatim šalje drugim neuronima preko aksona, koji je na svojim krajevima povezan s dendritima drugih neurona. Na slici 1 možemo vidjeti pojednostavljenu strukturu biološkog neurona s prikazom njegovih glavnih dijelova.



Slika 1. Građa biološkog neurona [9]

Umjetni neuron prima potencijale od susjednih neurona, ali umjesto praga funkcije koristi drugu funkciju nazvanu funkcija aktivacije. Označimo ulazne signale kao x1, x2, ..., xn, težine sinapsi kao w1, w2, ..., wn, i pomak kao b. Budući da su krajevi aksona povezani s dendritima drugih neurona, njihova međusobna interakcija predstavljena je množenjem ulaznih signala s težinama sinapsi. Svaki neuron računa težinsku sumu. Ova suma zatim prolazi kroz funkciju aktivacije, što konačno predstavlja izlazni signal. Neuroni se aktiviraju na temelju vrijednosti izlaznog signala.

Na slici 2. prikazana je građa umjetnog neurona i proces prijenosa signala.



Slika 2. Građa umjetnog neurona [11]

2.1.2. Arhitektura umjetne neuronske mreže

Razlikuju se tri dijela/sloja u neuronskim mrežama: ulazni sloj, skriveni sloj i izlazni sloj. Prvi je odgovoran za prihvaćanje ulaza - informacija, značajki ili mjera vanjskog okruženja čija normalizacija poboljšava točnost matematičkih funkcija u mreži. Drugi sloj sastoji se od neurona koji izvlače obrasce povezane s analiziranim procesom. Treći sloj sadrži neurone koji prikazuju izlaz mreže. Stoga, osnovna arhitektura neuronskih mreža prema položaju i povezanosti neurona dijeli se na jednoslojne i višeslojne neuronske mreže, kao i rekurentne mreže. Ovisno o broju slojeva i smjeru veza između neurona, definirana su tri osnovna tipa neuronskih mreža:

1. Jednoslojne mreže bez povratnih veza (eng. Single-Layer Feedforward Networks)
2. Višeslojne mreže bez povratnih veza (eng. Multilayer Feedforward Networks)
3. Mreže s povratnim vezama (eng. Reccurrent Networks)

2.2. Računalni vid

Računalni vid je znanost koja nastoji kopirati ili funkcionalno nadmašiti ljudski vizualni sustav, omogućujući računalima da vizualno razumiju okolinu. Istraživači nastoje postići ovo razvijanjem matematičkih postupaka koji transformiraju dvodimenzionalne prikaze u trodimenzionalne. Unatoč postignutom napretku u ovoj grani računalne znanosti, približavanje računalne percepcije ljudskoj razini i dalje je izazovno. Zahvaljujući razvoju algoritama za procesiranje, analizu i interpretaciju multimedijskih podataka, računalni vid igra ključnu ulogu u razvoju multimedijalnih aplikacija, omogućujući, primjerice, pretraživanje milijuna srodnih videozapisa na temelju kratkih video isječaka..

Obrada slika predstavlja korak koji prethodi računalnom vidu. Konkretno, cilj obrade slika je izdvajanje njenih osnovnih elemenata poput rubova, kutova, filtara i sličnih karakteristika. Prije semantičke segmentacije, potrebno je filtrirati sliku. Glavna razlika između obrade slika i računalnog vida je povratna informacija, koja je ključna za računalni vid, ali nije prisutna u obradi slika. Dvije ključne faze računalnog vida su ekstrakcija i klasifikacija karakteristika. Kvaliteta karakteristika i klasifikatora utječe na točnost i učinkovitost cjelokupnog vizualnog sustava. Karakteristika je bilo koja osobina koja se koristi za rješavanje specifičnog računalnog zadatka u određenoj primjeni. Kombinacija n karakteristika čini n-dimenzionalni vektorski prostor nazvan "vektor karakteristika". Kvaliteta vektora karakteristika ovisi o njegovoj sposobnosti razlikovanja uzoraka slika različitih klasa, gdje bi uzorci iste klase trebali imati slične vrijednosti karakteristika.

Zadatak klasifikatora je svrstati sliku ili "područje interesa" (ROI) u odgovarajuću kategoriju na temelju vektora značajki, pri čemu težina klasifikacije ovisi o značajkama slika iste kategorije u odnosu na razlike značajki slika različitih kategorija. Zbog prisutnosti sjena, prepreka, problematičnih perspektiva, elemenata koji ne pripadaju i drugih nejasnoća, savršena klasifikacija nije moguća i ostaje veliki izazov.

2.3. Konvolucijske neuronske mreže

Konvolucijske neuronske mreže (CNN) su specijalizirane neuronske mreže u obliku rešetke dizajnirane za obradu podataka. One funkcioniraju slično kao standardne neuronske mreže, s ključnom razlikom da je svaka jedinica u sloju CNN-a (minimalno) dvodimenzionalni filter

primijenjen na ulaz sloja. Ovo je ključno za prepoznavanje uzoraka u ulaznom vizualnom ili video materijalu. CNN-ovi su korisni kako za nadzirano, tako i za nenadzirano učenje. U nadziranom učenju, ulazni i izlazni podaci su poznati, a model uči mapirati ulaze na izlaze. U nenadziranom učenju, model nema pristup izlaznim podacima za svaki skup ulaza; umjesto toga, nastoji procijeniti temeljnu distribuciju ulaznih podataka. CNN-ovi uče povezivati pojedinačne poglede s odgovarajućim kategorijama ekstrahiranjem reprezentacija značajki, koje se zatim koriste unutar mreže za predviđanje ispravne kategorije izlaznog pogleda.

2.3.1. Konvolucija

Konvolucija je operacija između dvije funkcije s realnim domenama, koja daje modificiranu verziju jedne od dviju originalnih funkcija. Operaciju konvolucije između dvije funkcije možemo zamisliti kao mjeru preklapanja između funkcije i verzije pomaknute i invertirane u odnosu na drugu funkciju. Primjer gdje se operacija konvolucije koristi je obrada signala. Konvolucija bilo kojeg signala s drugim signalom proizvodi treći signal koji može pružiti više informacija o prvom signalu nego što bi prvi signal sam mogao. Na primjer, primjenom konvolucije na 2D signal koji predstavlja crno-bijelu sliku s drugim signalom nazvanim filter ili jezgra, izlazni signal može sadržavati rubove originalne slike, što može biti korisno u određenim aplikacijama. Konvolucija ima tri vrlo značajna svojstva koja uvelike mogu poboljšati sustav učenja:

1. Raspršenu povezanost

2. Dijeljenje parametara

3. Translatornu ekvivarijantnost

Slika 3. Ulazno-izlazni signal

2.3.2. Vizualizacija

Za vizualizaciju onoga što su konvolucijske neuronske mreže naučile tijekom i nakon procesa učenja, koriste se različiti pristupi za prikazivanje značajki i aktivacija, uzimajući u obzir kriterije poput težinskih vrijednosti, aktivacija i gradijenata. Jedan od najjednostavnijih pristupa za vizualizaciju onoga što je CNN naučio jest pregledavanje konvolucijskih filtera koji ilustriraju vrstu uzoraka koje svaki konvolucijski sloj traži u ulaznim podacima. Aktivacije značajki kroz slojeve CNN-a također pružaju korisne informacije o kvaliteti naučenih reprezentacija. Vizualizacija aktivacija izlaza pruža uvid u obrasce izlaznih prikaza koji su izvučeni kao korisne značajke klasifikacije. Vizualizacija neuronskih mreža je ključni korak za razumijevanje njihovog ponašanja, dijagnostiku problema i optimizaciju performansi.

Metode za izradu vizualizacije neuronskih mreža:

1. **Struktura mreža:** Vizualizacija arhitekture mreže pomaže u razumijevanju slojeva i veza između njih. Alati kao što su „TensorFlow“ i „Keras“ nude ugrađene funkcionalnosti za grafičko prikazivanje modela.
2. **Aktivacije slojeva:** Vizualizacija aktivacija slojeva može pomoći u razumijevanju kako se podaci transformiraju kroz mrežu. To je posebno korisno za slojeve konvolucije kod konvolucijskih neuronskih mreža
3. **Gubitak i točnost tijekom treniranja:** Praćenje gubitka i točnosti tijekom treniranja pomaže u razumijevanju učinka modela i u otkrivanju potencijalnih problema kao što su prenaučenost ili podučenost.
4. **Vizualizacija težina:** Vizualizacija težina može otkriti obrasce u učenju mreže, poput simetričnih ili nepotrebnih veza.
5. **Vizualizaciju pomoću specijaliziranih alata:** Postoji nekoliko specijaliziranih alata za vizualizaciju neuronskih mreža, uključujući TensorBoard (TensorFlow) i Netron (alat za vizualizaciju neuronskih mreža za pregled arhitekture mreža).

Kombinaciju ovih tehnika i alata može pružiti uvid u duboki rad neuronskih mreža, olakšavajući dijagnostiku i poboljšanje performansi.

2.4. YOLO

YOLO (You Only Look Once) je tip neuronske mreže za detekciju objekata u slikama i videima. Umjesto tradicionalnih metoda koji koriste višestruke korake (kao što su regije interesa i klasifikatori), YOLO koristi jednostavan pristup - izravno predviđanje okvira objekata i vjerojatnosti klasa za sve objekte u slici u jednom prolazu kroz neuronsku mrežu.

Evo nekoliko ključnih značajki YOLO neuronskih mreža:

1. **Brzina:** YOLO je poznat po svojoj brzini jer može istovremeno predvidjeti objekte u slici. Ovo čini YOLO pogodnim za stvarne primjene, poput analize videozapisa u stvarnom vremenu.
2. **Jedan prolaz**: Za razliku od drugih metoda koje zahtijevaju višestruke prolaze kroz sliku, YOLO radi samo jedan prolaz, što ga čini jednostavnijim i bržim.
3. **Detekcija višestrukih objekata**: YOLO može otkriti više objekata u jednoj slici i dodijeliti im odgovarajuće okvire i vjerojatnosti klasa.
4. **Nedostatak konteksta:** Iako je brz i učinkovit, YOLO može imati poteškoća u prepoznavanju manjih objekata ili objekata koji su blizu jedan drugome, jer nedostaje kontekst koji bi mogao biti dostupan u višeprolaznim metodama.
5. **YOLOv5, YOLOv8**: Postoje različite verzije YOLO-a, a svaka nova verzija donosi poboljšanja u performansama, brzini ili točnosti detekcije. YOLOv5 i YOLOv8 su neke od najpoznatijih verzija.

2.4.1. YOLOv8

YOLOv8 je najnovija iteracija YOLO (You Only Look Once) modela, predstavljena kao poboljšanje u odnosu na prethodne verzije.

Evo nekoliko ključnih karakteristika YOLOv8:

1. **Jednostavnost i brzina**: Kao i prethodne verzije YOLO-a, YOLOv8 se ističe svojom jednostavnošću i brzinom. Uspoređujući s prethodnim verzijama, YOLOv8 je optimiziran kako bi postigao još veću brzinu i efikasnost u detekciji objekata.
2. **Arhitektura:** YOLOv8 koristi arhitekturu temeljenu na konvolucijskim neuronskim mrežama (CNN). To obično uključuje razne konvolucijske slojeve, slojeve grupne normalizacije (Batch Normalization), aktivacijske funkcije (npr. ReLU) te tehnike kao što su skip connections.

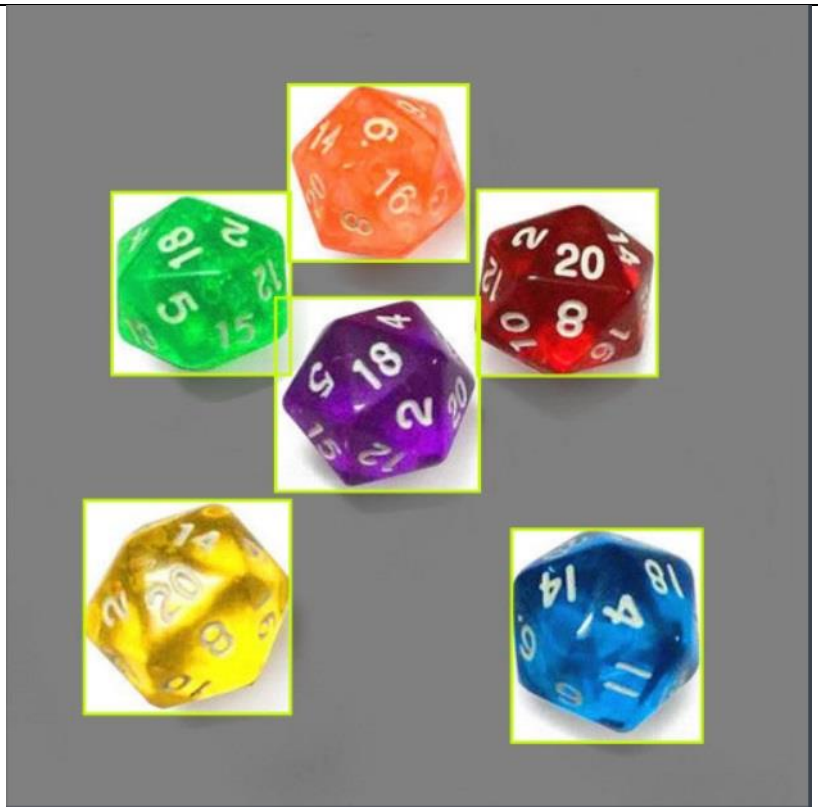
3. **Poboljšana točnost:** Iako je brzina i efikasnost ključna karakteristika, YOLOv8 također donosi poboljšanja u točnosti detekcije objekata. Ova poboljšanja mogu proizaći iz optimizacija u arhitekturi modela, boljeg pristupa podacima ili primjenom novih tehnika učenja.

Implementacija i dostupnost: YOLOv5 je dostupan kao open-source projekt, što znači da je kod otvoren za pregled, prilagodbu i korištenje od strane zajednice. To olakšava istraživačima i praktičarima u području računalnog vida da koriste YOLOv5 u svojim projektima.

2.4.2. Vizualizacija YOLOv8

Vizualizacija rezultata detekcije YOLOv5 modela obično uključuje iscrtavanje okvira (bounding boxes) oko detektiranih objekata u slikama ili videozapisima, zajedno s oznakama klasa i vjerojatnostima. Ove vizualne oznake pomažu korisnicima da vizualno razumiju što je model detektirao i s kojom sigurnošću.

1. **Bounding boxes:** Svaki detektirani objekt je obično označen okvirom koji ga ograničava. Ovi okviri često su obojeni različitim bojama kako bi se razlikovali različiti objekti ili klase objekata.
2. **Oznaka klase:** Pored svakog okvira obično se prikazuje oznaka klase kojoj pripada detektirani objekt. Na primjer, ako je detektiran automobil, oznaka klase će biti "automobil"
3. **Vjerojatnosti:** Uz svaki okvir često se prikazuje vjerojatnost (ili sigurnost) s kojom je model identificirao objekt kao pripadajući određenoj klasi. Ovo može biti prikazano kao postotak ili neki drugi format, omogućujući korisnicima da procjene koliko su pouzdani rezultati detekcije



Slika 4. Granični okvir (Bounding box)

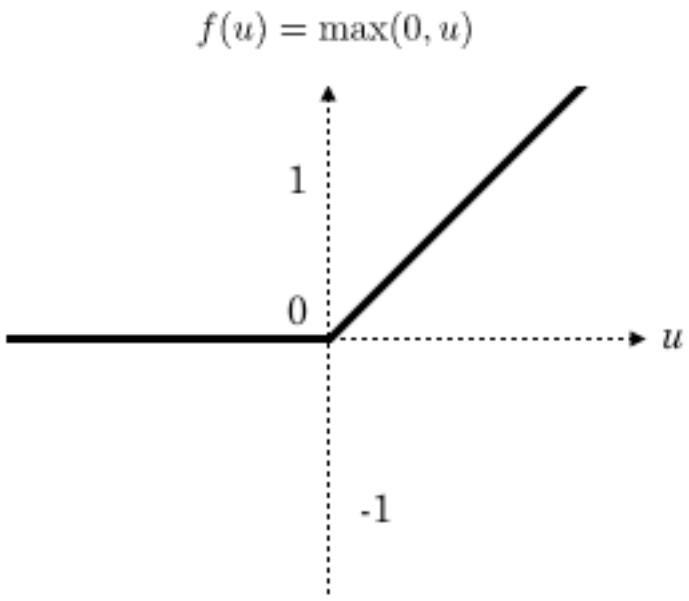
2.5. ReLU aktivancijska funkcija

ReLU (Rectified Linear Unit) je aktivacijska funkcija koja se koristi u neuronskim mrežama kako bi unijela nelinearnost u model i pomogla mu da nauči složene obrasce u podacima.

Njena formula je jednostavna:

$$f(u) = \max(0, u)$$

ReLU aktivacijska funkcija vraća za pozitivne vrijednosti parametra u, samu ulaznu vrijednost, a za negativne vrijednosti parametra u vraća nulu.



Slika 5. Grafički prikaz ReLU funkcije [1]

Na slici 5 se vidi kako je funkcija *f(u)* za sve negativne vrijednosti nula, za pozitivne vrijednosti linearno raste.

Prednosti ReLU aktivacijske funkcije:

1. **Nelinearnost**: ReLU unosi nelinearnost u mrežu, omogućujući joj da nauči složene obrasce
2. **Jednostavnost:** Izračunavanje ReLU funkcije je jednostavno i efikasno, što smanjuje računsku složenost.
3. **Ublažava problem nestajućeg gradijenta:** Za razliku od sigmoidne ili tanh funkcije, ReLU ublažava problem nestajućeg gradijenta, omogućujući brže i učinkovitije treniranje dubokih mreža**.**

Nedostaci ReLU aktivacijske funkcije:

1. **Umiranje ReLU:** Ako previše neurona postane trajno neaktivno (izlaz je uvijek 0), to može smanjiti kapacitet učenja mreže. Ovaj problem se naziva "umiranje ReLU" (dying ReLU)

3. PRIKUPLJANJE I OBRADA PODATAKA

3.1. Dungeons & Dragons

Dungeons & Dragons (D&D) je društvena fantazijska igra poznata po svojoj kreativnosti, složenosti i priči. Objavljena je 1974.godine, a osmislili su je Gary Gygax i Dave Arneson. D&D je uvertira u svijet modernih igara i ima velik utjecaj na mnoge aspkete pop kulture, uključujući video igre, književnost i filmove. U D&D igrači preuzimaju uloge avanturista u izmišljenim svjetovima koje vodi Dungeon Master (DM). DM je zadužen za vođenje price, stvaranje svijeta. Igrači kreiraju svoje likove, biraju njihove vještine i opremu te prolaze kroz razne izazove i avanture koje DM postavlja. Jedan od ključnih aspekata igre je sloboda mašte koja se pruža igračima, ne postoje stroga pravila koja ograničavaju igrače, već oni mogu koristit maštu i kreativnost da pristupe različitim situacijama na različite načine. Igra ima velik društveni aspekt, igra se u grupama, bilo uživo il online, što omogućava igračima razvijanje timskog rada, komunikaciju i sklapanje novih prijateljstava. Igra ima i veliki društveni aspekt. Često se igra u grupama, bilo uživo ili online, što omogućava igračima da razviju timski rad, komunikaciju i prijateljstva. Igra se minimalno u dvoje, ali većinom je to grupa od četiri ili šest igrača.

Osnovna pravila i uvjeti igre su:

1. **Kreiranje likova**:
    - **Vrsta**: Igrači biraju vrstu svog lika, kao što su ljudi, divovi, vilenjaci i mnogi drugi, a svaka rasa ima svoje jedinstvene sposobnosti i karakteristike.
    - **Atributi**: Svaki lik ima šest osnovnih atributa, kao što su snaga, spretnost, mudrost, konstitucija, inteligencija i karizma.
2. **Bacanje kockica:**
    - D&D koristi razne vrste „kockica“ (tetraedar (D4), heksaedar (D6), oktaedar (D8), dekaedar (D10), dodekaedar (D12), duodekaedar (D20).
    - Bacanje kockica se koristi za određivanje uspjeha i neuspjeha u raznim situacijama, kao što su borba, otpor čarolijama itd.

3. **Borba:**
    - Borba se odvija u potezima, gdje svaki igrač ima priliku za akciju. Akcije ukljucuju napade, kretanje, izvođenje posebnih vještina. Redoslijed akcija se određuje bacanjem kockica
4. **Oprema:**
    - Osnovni set uključuje knjige s pravilima, kockice, figure te mape i planove. Postoje i mnogu drugi dodatni priručnici i setovi koji proširuju igru.
5. **Vrijeme trajanja:**
    - D&D obično traje po nekoliko sati ovisno o složenosti priče i raspoloženju igrača.

Ideja za ovaj diplomski rad se pojavila igrajući D&D, teško je pratiti oblik i brojeve na kockica te se želi automatizirati i olakšati prepoznavanje kockica i brojeva na kockicama putem neuronskih mreža.

Slika 6. Dungeons&Dragons (D&D) [2]

3.2. Kamera za prikupljanje podataka

Prikupljanje podataka je najveći i najzahtjevniji dio diplomskog rada. Što više podataka se prikupi, neuronska mreža će bolje naučiti i biti primjenjivija u kompleksnijim situacijama. Za prikupljanje podataka se koristi digitalni fotoaparta i kamera na mobitelu.

3.2.1. Canon PowerShot SX740 HS

Canon PowerShot SX740 HS je digitalni fotoaparat poznat po svojim naprednim značajkama i svestranosti, idealan za svakodnevnu upotrebu. Odlikuje ju 20.3 megapikselni CMOS senzor te impresivni 40x optički zoom idealan za širokokutne snimke. Prijenos slika je olakšan zbog mogućnosti povezivanja fotoaparata i laptopa putem Wi-Fi i bluetooth. Jedan od nedostataka Canon PowerShot SX740 HS digitalnog fotoaparata je slabija kvaliteta slike pri lošijem osvjetljenju, performanse slike nisu na razini DSLR ili mirrorless fotoaparata s većim senzorom. Canon PowerShot SX740 HS je odličan izbor za putnike, obiteljske fotografije i svakodnevno fotografiranje. Upravo navedeni nedostatak digitalnog fotoaparata je korišten namjerno u sustavu prikupljanja podataka, nisu sve slike napravljene u idealnim i laboratorijskim uvjetima, s ciljem neuronskoj mreži predstaviti što više različitih podataka i u različitim okolnostima kako bi se dobio veći broj uzoraka koje će neuronska mreža naučiti i kasnije lakše ovladati prepoznavanjem ako uzorak koji mora analizirati ne bude idealno predstavljen.

Slika 7. Canon PowerShot SX740 HS [3]

3.3. Stvaranje klasa i labelovanje prikupljenih podataka (RoboFlow)

U ovom poglavlju će biti prikazan način stvaranja klasa i metode kojim će se vršiti labelovanje prikupljenih slika i njihova implementacija u RoboFlow.

3.3.1. RoboFlow

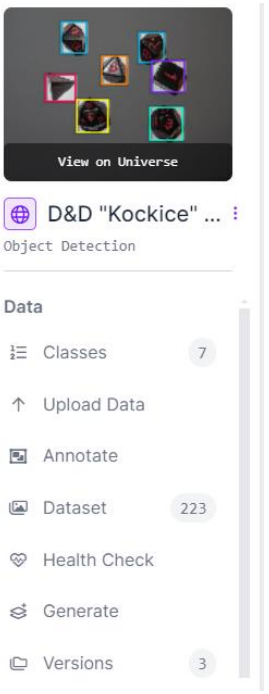
RoboFlow je alat za osobe koje se bave razvojem aplikacija za računalni vid. Njegove značajke omogućavaju jednostavno treniranje modela, upravljanje podacima i njihovu implementaciju u stvarne aplikacije. RoboFlow podržava uvoz slika i anotacija iz različitih izvora i formata, podržava različite vrste računalnog vida kao što su detekcija objekata, klasifikacija slika i segmentacija. Jedna od velikih prednosti RoboFlow-a je ta što daje mogućnost grupnog rada, sve potrebne datoteke se nalaze na platformi i svi koji sudjeluju na nekom projektu i imaju pristup platformi mogu međusobno dijeliti i prikupljati informacije.

Neke od prednosti platforme RoboFlow su:

- **Jednostavnost upotrebe**: Intuitivno korisničko sučelje koje olakšava upravljanje podacima i treniranje modela.
- **Fleksibilnost**: Podržava razne tipove modela.
- **Štednja vremena**: Automatizacija mnogih koraka u razvoju modela

Slika 8. Roboflow platforma [4]

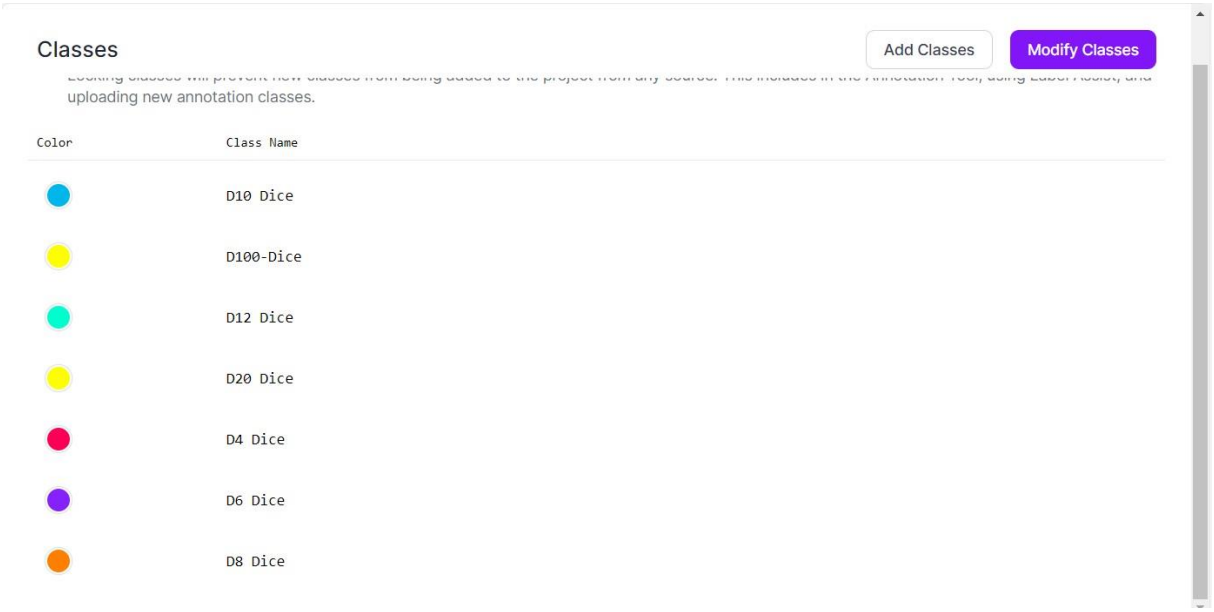
Za pristup RoboFlow platformi potrebno je napraviti korisnički račun na službenoj stranici platforme RoboFlow i nakon prijave, ponuđen je pristup nekim javnim projektima ili postoji mogućnost otvoriti vlastiti projekt preko kojeg će se raditi neki budući projekti. Za potrebe ovog rada je bilo potrebno kreirati vlastiti projekt te mu je dodijeljen naziv „D&D „kockice“ (Dice)“.



Slika 9. Roboflow projekt D&D „Kockice“ (Dice)

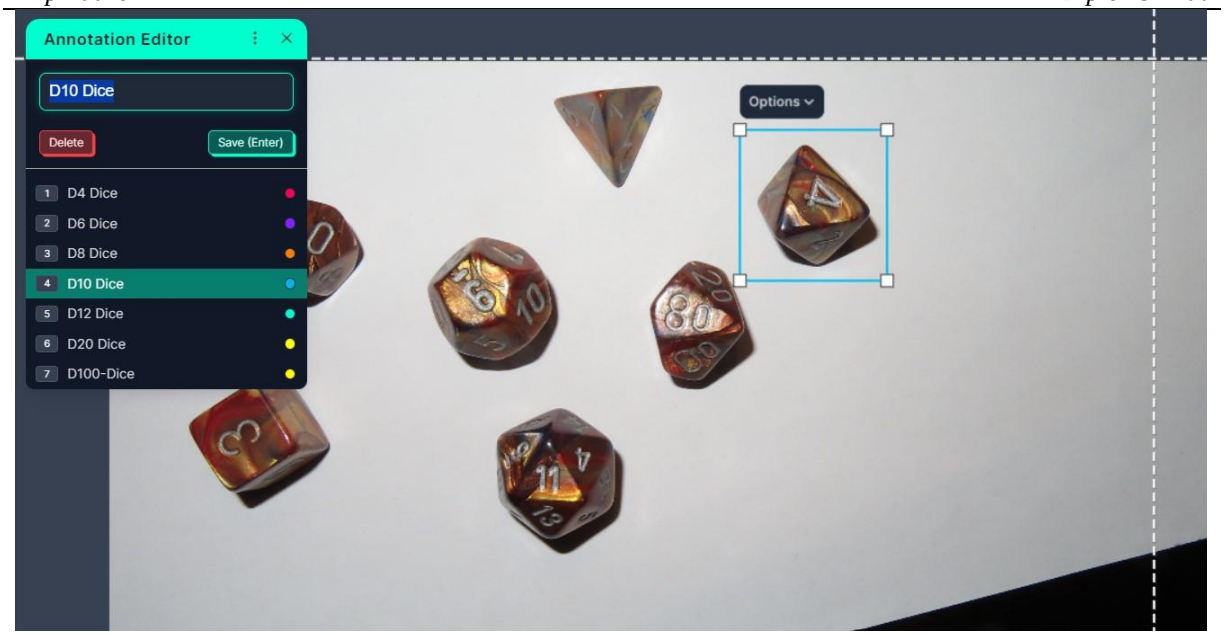
Nakon izrade i imenovanja projekta potrebno je napraviti klase koje su potrebne za provedbu osmišljenog projekta. U ovom projektu koristi se 7 klasa, za svaku vrstu „kockice“ po jedna, tako postoje klase: D4 Dice, D6 Dice, D8 Dice, D10 Dice, D12 Dice, D20 Dice i D100 Dice. Kockica D100 ima identičan oblik kao i kockica D10, samo se na njoj nalaze brojevi od 10 do 100 pa će se kasnije vidjeti kada započne postupak treniranja neuronske mreže, hoće li neuronska mreža sa sigurnošću raspoznavati te dvije kockice. Zatim se stvara jedan vlastiti skup podataka (eng. Dataset) u koju se učitavaju svi prikupljeni podaci, taj skup podataka će biti namijenjena isključivo za treniranje neuronske mreže i količina podataka u tom skupu podataka ovisi koliko će se neuronska mreža kvalitetno naučiti i biti primjenjiva za buduću primjenu. Iako prevelika količina podataka u skupu podataka često znači bolju kvalitetu neuronske mreže, ali se može dogoditi da se neuronska mreža već dovoljno dobro naučila na trenutnom broju prikupljenih podataka i svaki novi učitani podatak će neuronsku mrežu zanemarivo malo

poboljšati, a može drastično usporiti njezin rad, zbog toga treba biti oprezan s količinom podataka.



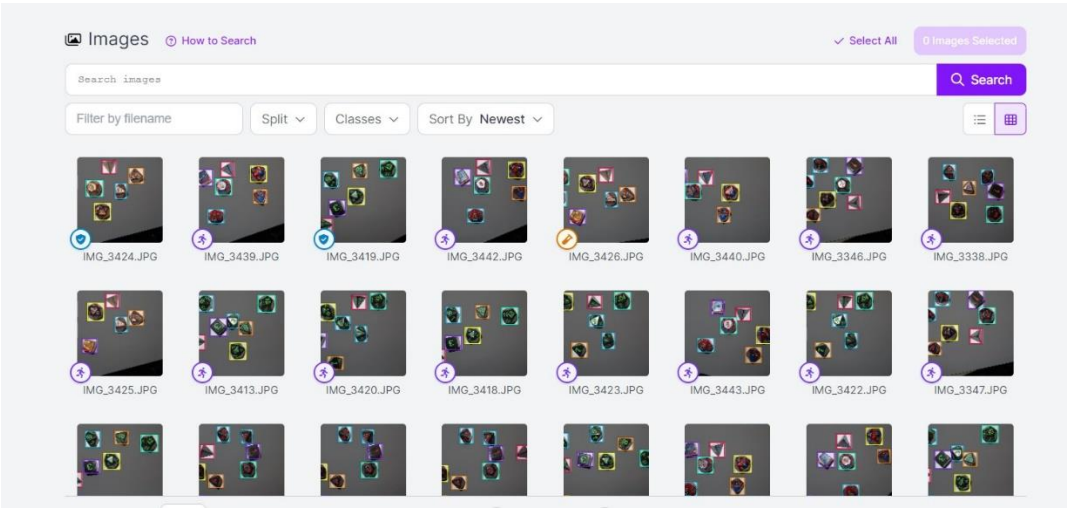
Slika 10. Imenovanje svih klasa u skupu podataka

U skup podataka se dodaju slike koje su napravljene tijekom izrade diplomskog rada, na slikama su vidljive kockice u različitim kombinacijama. Stoga postoje slike gdje je vidljiva samo jedan kockica na slici, sve kockice odjednom, dvije identične kockice, slaba vidljivost, loše pozadinsko osvjetljenje idt. Plan je naučiti neuronsku mrežu prepoznavanju, kada nisu idealni uvjeti, jer što neuronska mreža bolje „pliva“ u nepredvidivim situacijama, to je kvalitetnije napravljena i može kompleksnije stvari raditi.. Da bi se došlo do neuronske mreže koja radi neki konkretan zadatak, sve slike koje su učitane u skup podataka treba obilježiti, to se radi tako što se svakoj slici pristupa pojedinačno, dodaju se gore navedene klase, ovisno o tome što se od elemenata koje pripadaju nekoj klasi nalazi na pojedinoj slici. Postupak dodavanja i obilježavanja slika nije kompleksan, ali je potrebno uložiti mnogo vremena, ako će skup podataka imati veliku količinu podataka, u ovom slučaju slika.



Slika 11. Anotacija i klasifikacija objekata na slikama

Na slici 11 je vidljivo kako se odabranom elementu slike napravi okvir koji se u Roboflow naziva bounding box i nakon toga je potrebno odrediti kojoj klasi pripada odabrani element, u ovom slučaju element pripada klasi D10 Dice, isto treba napraviti za ostalih šest elemenata. Isti postupak treba provesti za sve učitane slike u podatkovnoj mapi, nažalost ne postoji neki automatizirani način učitavanja klasa, svaka slika se mora ručno obilježiti i pripremiti za treniranje neuronske mreže. Postoji opcija automatskog obilježavanja, kada se stvori neuronska mreža kojoj se mogu učitati nove slike i dopustit da neuronska mreža sama obilježava potrebne slike za daljnje učenje, što je ujedno i dobar test za provjeru kvalitete rada neuronske mreže.



Slika 12. Skup podataka sa svim klasama i graničnim okvirima

3.3.2. Postavljanje radne okoline

Sve dosada navedeno je bila samo priprema za treniranje neuronske mreže putem modela YOLOv8, koji u sebi već sadrži neke elemente neuronske mreže. Za upravljanje s YOLOv8 i izgradnju neuronske mreže potrebno je poznavanje Python programskog jezika. Osnovna verzija Python programskog jezika ne sadrži u sebi potrebne pakete za rad s YOLOv8, zbog toga se moraju posebno instalirati putem „Command Prompt (CMD)".

Za instalaciju potrebnog paketa potrebno je otići na službenu stranicu za rad s YOLOv8 „Ultralytics" [7] i pratiti upute za instalaciju. Za pokretanje instalacije potrebno je u „Command Prompt" upisati: **pip install ultralytics.**

Slika 13. Postupak instalacije Ultralytics paketa za programski jezik Python

Nakon što se izvršila instalacija ultralyticsa, potrebno je otići na internet stranicu „PyTorch" [5] i u traci s padajućim izbornicima odabrati „Start Locally". U izborniku „Start Locally" je potrebno podesiti preference koje odgovaraju računalu na kojem se izvodi projekt.

Tablica 1. Kombinacija preferenci za instalaciju [5]

PyTorch verzija	Stable (2.3.1.)
Operacijski sustav	Windows
Paket	Pip
Jezik	Python
Računalna platforma	CUDA 12.1

Nakon odabranih kombinacija iz desnog stupca tablice dobiveni link (pip3 install torch torchvision torchaudio --index-url https://download.pytorch.org/whl/cu121) je potrebno zalijepiti u CMD i nastaviti instalaciju. CUDA 12.1 dopušta neuronskoj mreži treniranje pomoću grafičke kartice računala.

```
C:\Users\Domagoj>pip3 install torch torchvision torchaudio --index-url https://download.pytorch.org/whl/cu121
Looking in indexes: https://download.pytorch.org/whl/cu121
Requirement already satisfied: torch in c:\users\domagoj\appdata\local\programs\python\python312\lib\site-packages (2.2.0+cu121)
Requirement already satisfied: torchvision in c:\users\domagoj\appdata\local\programs\python\python312\lib\site-packages (0.17.0+cu121)
Requirement already satisfied: torchaudio in c:\users\domagoj\appdata\local\programs\python\python312\lib\site-packages (2.2.0+cu121)
Requirement already satisfied: filelock in c:\users\domagoj\appdata\local\programs\python\python312\lib\site-packages (from torch) (3.13.1)
Requirement already satisfied: typing-extensions>=4.8.0 in c:\users\domagoj\appdata\local\programs\python\python312\lib\site-packages (from torch) (4.9.0)
Requirement already satisfied: sympy in c:\users\domagoj\appdata\local\programs\python\python312\lib\site-packages (from torch) (1.12)
Requirement already satisfied: networkx in c:\users\domagoj\appdata\local\programs\python\python312\lib\site-packages (from torch) (3.2.1)
Requirement already satisfied: jinja2 in c:\users\domagoj\appdata\local\programs\python\python312\lib\site-packages (from torch) (3.1.2)
Requirement already satisfied: fsspec in c:\users\domagoj\appdata\local\programs\python\python312\lib\site-packages (from torch) (2024.2.0)
Requirement already satisfied: numpy in c:\users\domagoj\appdata\local\programs\python\python312\lib\site-packages (from torchvision) (1.26.3)
Requirement already satisfied: requests in c:\users\domagoj\appdata\local\programs\python\python312\lib\site-packages (from torchvision) (2.31.0)
Requirement already satisfied: pillow!=8.3.*,>=5.3.0 in c:\users\domagoj\appdata\local\programs\python\python312\lib\site-packages (from torchvision) (9.5.0
)
Requirement already satisfied: MarkupSafe>=2.0 in c:\users\domagoj\appdata\local\programs\python\python312\lib\site-packages (from jinja2->torch) (2.1.3)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\domagoj\appdata\local\programs\python\python312\lib\site-packages (from requests->torchv
ision) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in c:\users\domagoj\appdata\local\programs\python\python312\lib\site-packages (from requests->torchvision) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\domagoj\appdata\local\programs\python\python312\lib\site-packages (from requests->torchvision)
 (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\domagoj\appdata\local\programs\python\python312\lib\site-packages (from requests->torchvision)
 (2023.7.22)
Requirement already satisfied: mpmath>=0.19 in c:\users\domagoj\appdata\local\programs\python\python312\lib\site-packages (from sympy->torch) (1.3.0)
```

Slika 14. Završni korak instalacije Ultralytics paketa

3.3.3. Treniranje vlastitog skupa podataka

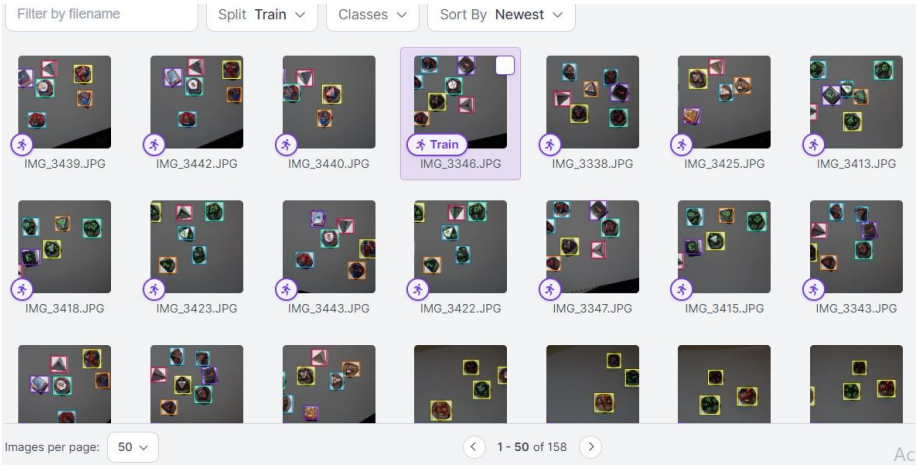
Nakon učitavanja i obilježavanja svih slika, skup podataka slučajnim odabirom slike spremi u tri podskupa podataka: trening (train), validacija (valid) i testiranje (test). Najviše slika odlazi u podskup podataka trening, jedan dio u validaciju, a najmanji dio slika pripada podskupu za testiranje neuronske mreže.

Treniranje (Train) modela u YOLOv8 odnosi se na proces obučavanja neuronske mreže za detekciju objekata koristeći skup podataka s anotiranim slikama. Prije treniranja potrebno je pripremiti skup podataka koji sadrži slike s odgovarajućim oznakama.Prije početka treniranja potrebno je definirati konfiguraciju modela te je potrebno odrediti broj epoha, veličinu batch-a i stopu učenja.

Konfiguracija modela se sastoji od:

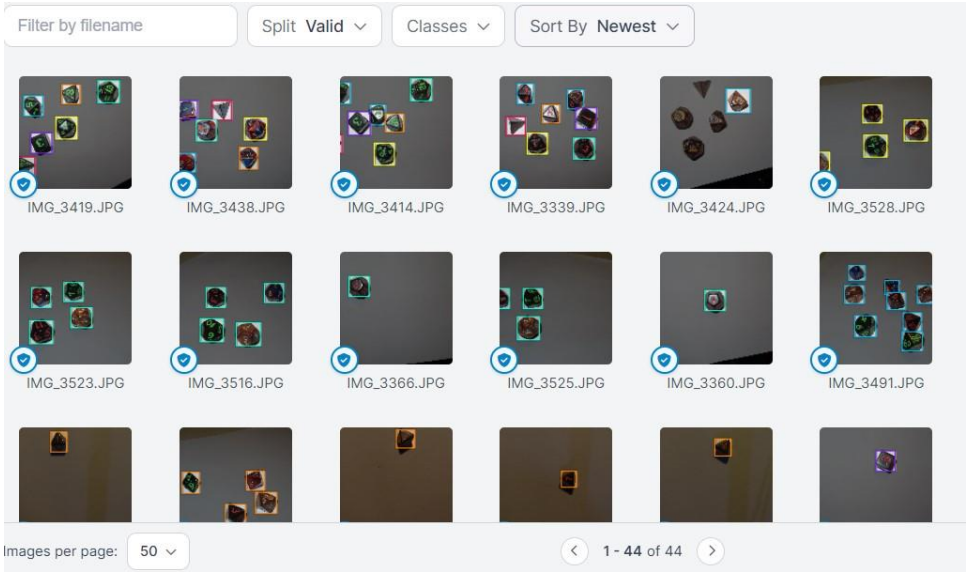
- **Broj epoha**: Koliko puta će model proći kroz cijeli skup podataka tijekom treniranja, najveći dopušteni broj epoha je 100.
- **Veličina batch-a:** Ukupan broj slika koje obrađuje u jednom prolazu kroz neuronsku mrežu.
- **Stopa učenja:** Brzina kojom model ažurira svoje težine tijekom treninga.

Proces treniranja obuhvaća prolazak kroz slike iz skupa podataka i njihovo propuštanje kroz neuronsku mrežu. Model predviđa granične okvire (bounding box) i klase objekata, a zatim izračunava pogrešku između stvarnih oznaka i predikcija. Svaku novu epohu neuronska mreža je naučila nešto novo i svakom novom epohom se dolazi do boljih rezultata i smanjenja pogreške između stvarne oznake i predikcije. Tijekom procesa treniranja potrebno je pratiti performanse modela u validacijskom skupu podataka kako bi se osigurala generalizacija i spriječila „pretreniranost“ neuronske mreže. Prekomjerno prilagođavanje (overfitting) može imati za posljedicu to, da neuronska mreža perfektno radi na uzorku po kojem je trenirana, a za testne slike daje loše rezultate.



Slika 15. Skup podataka za proces treniranja modela

Validacija (Valid) u YOLOv8 odnosi se na proces provjere treniranog modela koji na posebnom skupu podataka koji nije korišten tijekom treniranja. Cilj validacije je procijeniti koliko dobro neuronska mreža generalizira na neviđene podatke i spriječavanje prekomjernog prilagođavanja u procesu treniranja. Skup podataka za validaciju mora sadržavati slike i njihove anotacije, slike trebaju sadržavati stvarne primjere s kojim će se naučena neuronska mreža susretat u praksi. Nakon svake epohe treniranja neuronska mreža se evoluira u validacijskom skupu podataka, slike iz validacijskog skupa podataka se propuštaju kroz neuronsku mrežu te model predviđa gdje bi se trebali nalaziti granični okviri i svakom graničnom okviru dodijelit odgovarajuću klasu. Rezultati validacije se prate za vrijeme procesa treniranja kako bi se vidjela razina poboljšanja neuronske mreže, često se rezultati validacije prokazuju grafikonima i tablicama.



Slika 16. Skup podataka za proces validacije modela

Testiranje modela (test) u YOLOv8 je proces procjene performasi izvršen na skupu podataka koji nije prošao kroz proces treniranja i validacije. Cilj testiranja je procijeniti koliko dobro neuronska mreža generalizira potpuno nove podatke i izračunava njegove metrike: točnost, odziv, preciznost i mAP (mean Average Precision).

Objašnjena metrika su:

- **Preciznost:** Omjer točnih pozitivnih predikcija u odnosu na ukupan broj pozitivnih predikcija, govori koliko su predikcije pouzdane.

$$\boldsymbol{Preciznost = \frac{Broj\ točno\ detektiranih\ objekata}{Ukupan\ broj\ detektiranih\ objekata}} \quad (\mathbf{1})$$

- **Odziv (Recall):** Omjer točno detektiranih objekata u odnosu na ukupan broj stvarnih objekata. Mjeri koliko je model uspješan u pronalasku stvarnih objekata.

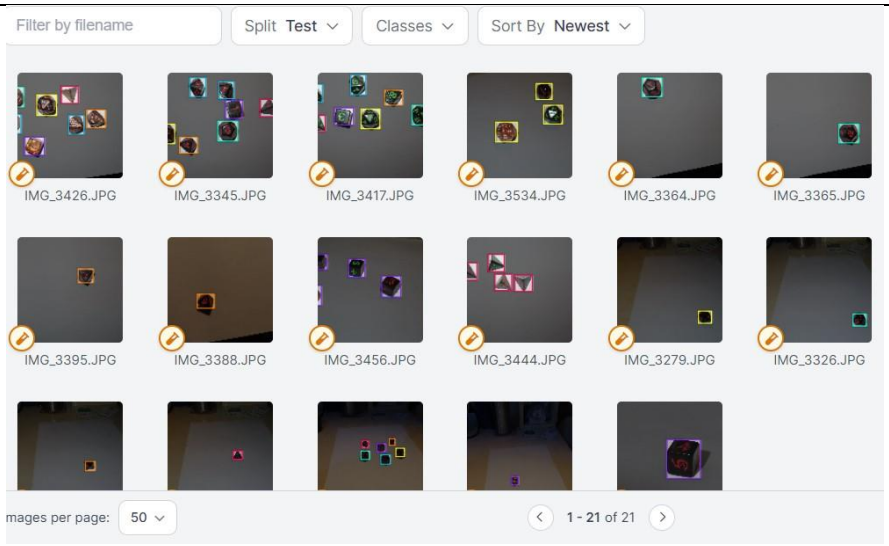
$$\boldsymbol{Odziv = \frac{Broj\ točno\ detektiranih\ objekata}{Ukupan\ broj\ stvarnih\ \ objekata}} \quad (\mathbf{2})$$

- **mAP (mean Average Precision):** Kombinira preciznost i odziv za sve klase**,** predstavlja prosječnu preciznost modela.

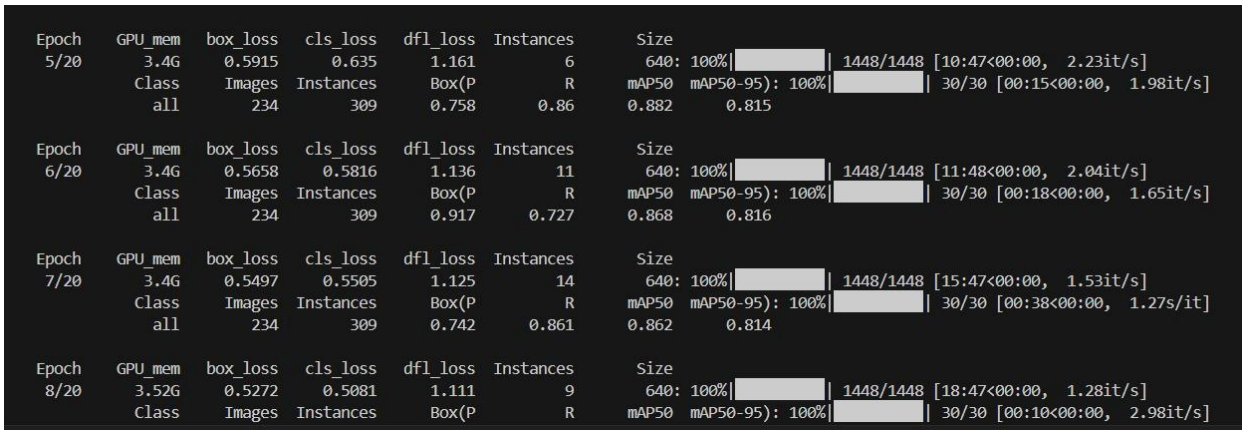
$$\boldsymbol{mAP = \frac{1}{C}\sum_{C=1}^{C} AP_C} \quad (\mathbf{3})$$

Gdje su oznake:

- C- prestavlja ukupan broj klasa,
- AP_c – prosječna preciznost za klasu C



Slika 17. Skup podataka za proces testiranja modela



Slika 18. Parametri treniranja tijekom učenja neuronske mreže

Na slici 18 su vidljivi parametri koji se prate za vrijeme treniranja neuronske mreže, prikazuje se koliko je epoha neuronska mreža odradila u odnosu na zadani broj epoha, prikazani su još parametri kao što su: gubitak okvira (box_loss), gubitak klasifikacije (cls_loss), dfl_loss i nstaces.

Parametri prikazani na slici su:

1. Gubitak okvira (box_loss) je ključna komponenta u treniranja modela za detekciju objekta, jer osigurava da previđeni okviri (bounding box) što bolje odgovaraju stvarnim okvirima objekata. Što neuronska mreža više minimizira ovaj gubitak treniranjem , rezultirat će boljom preciznošću modela u detekciji i lokalizaciji objekata.

   Gubitak okvira prati 3 vrste gubitaka koje se pokušava tijekom treniranja minimizirati, a to su:

   a. **Regresijski gubitak**: Ova komponenta mjeri razliku između koordinata stvarnog okvira i okvira stvorenog prilikom predikcije modela neuronske mreže.

   b. **Koordinate centra (cx, cy):** Neuronska mreža predviđa gdje se nalazi centar okvira objekta, a gubitak se računa razlikom izmedu koordinata centra predikcije i stvarnih koordinata centra.

   c. **Širina i visina (w, h):** Model predviđa visinu i širinu okvira, a gubitak se računa između stvarnih dimenzija okvira i okvira koji nastaje predikcijom.

   Formula putem koje se izračuva gubitak okvira glasi:

$$\boldsymbol{box_{loss}} = \boldsymbol{\lambda_{coord}} \sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} ((\boldsymbol{cx_i} - \boldsymbol{\widehat{cx}_i})^{\boldsymbol{2}} + (\boldsymbol{cy_i} - \boldsymbol{\widehat{cy}_i})^{\boldsymbol{2}} + (\boldsymbol{w_i} - \boldsymbol{\widehat{w}_i})^{\boldsymbol{2}} + \left(\boldsymbol{h_i} - \boldsymbol{\widehat{h}_i}\right)^{\boldsymbol{2}}) \quad (\mathbf{4})$$

Gdje su:

$\boldsymbol{cx_i}, \boldsymbol{cy_i}, \boldsymbol{\widehat{cx}_i}, \boldsymbol{\widehat{cy}_i}$ – stvarne i predviđene koordinate centra okvira objekta i.

$\boldsymbol{w_i}, \boldsymbol{h_i}, \boldsymbol{\widehat{w}_i}, \boldsymbol{\widehat{h}_i}$ – stvarne i predviđene dimenzije okvira i

$\boldsymbol{\lambda_{coord}}$ – težinski faktor koji kontrolira važnost ove komponente u ukupnom gubitku.

2. Gubitak klasifikacije (cls_loss) je komponenta ukupnog gubitka koja se odnosi na točnost klasifikacije predviđenih objekata u odgovarajuće klase, ovaj gubitak mjeri koliko su predikcije određivanja klasa točne u odnosu na stvarnu klasifikaciju objekata. Prilikom treniranja gubitak klasifikacije se prati u validacijskom procesu i cilj je minimizirati gubitak

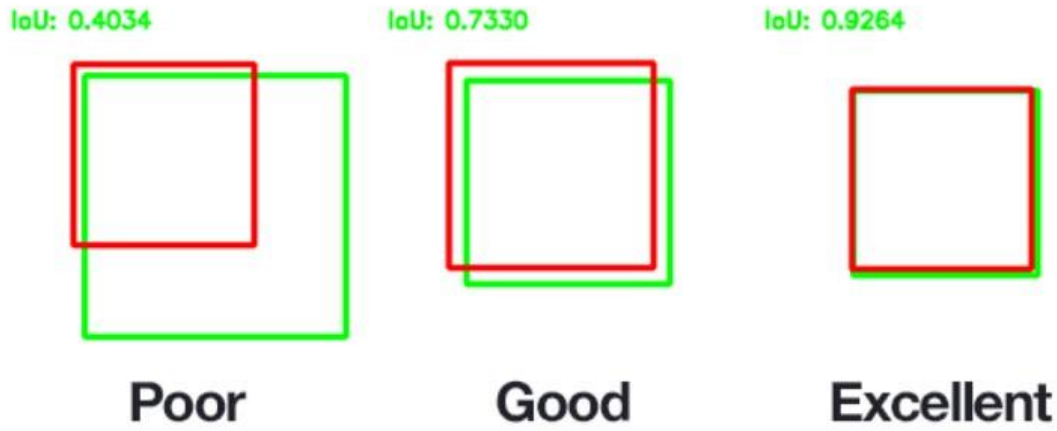
klasifikacije u svrhu poboljšanja rada neuronske mreže. Za izračunavanje gubitka klasifikacije se obično koristi formula za unakrsnu entropiju.

$$\boldsymbol{cls_{loss}} = -\sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} [\boldsymbol{y_i}\, \boldsymbol{log}(\boldsymbol{\hat{y}_i}) + (\mathbf{1} - \boldsymbol{y_i})\, \boldsymbol{log}(\mathbf{1} - \boldsymbol{\hat{y}_i})] \qquad (\mathbf{5})$$

Gdje su : $\boldsymbol{y_i}, \boldsymbol{\hat{y}_i}$ - stvarne i predviđene oznaka klase

3. DFL (Distribution Focal Loss) gubitak je gubitak osmišljen kako bi pomogao modelu bolje naučiti raspodjelu graničnih okvira objektra, DFL kombinira dvije funkcije, uzima ideje iz focal los funkcije koja se fokusira na teže primjere, smanjivajući težinu lako klasificirajućim objektima i ideje iz distribucijskog aspekta, što znači da umjesto predviđanja samo jedne vrijednosti za svaku koordinatu graničnog okvira, DFL predviđa distribuciju vrijednosti.

4. U okvirima računalnog vida i rada s neuronskim mrežama riječ „Instance“ se odnosi na pojedinačne objekte koji se prepoznaju i klasificiraju na slici. U slučaju YOLOv8 pojam Instance se odnosi na svaki detektirani objekt na slici, uključujući njegovu klasu i granični okvir. Prilikom korištenja YOLOv8, neuronska mreža će dati predikciju za svaku instancu objekta na slici, a to znači koordinate graničnog okvira, odabranu klasu objekta i koliko je povjerenje modela u tu predikciju.



Slika 19. Prikaz minimizacije gubitka okvira [6]

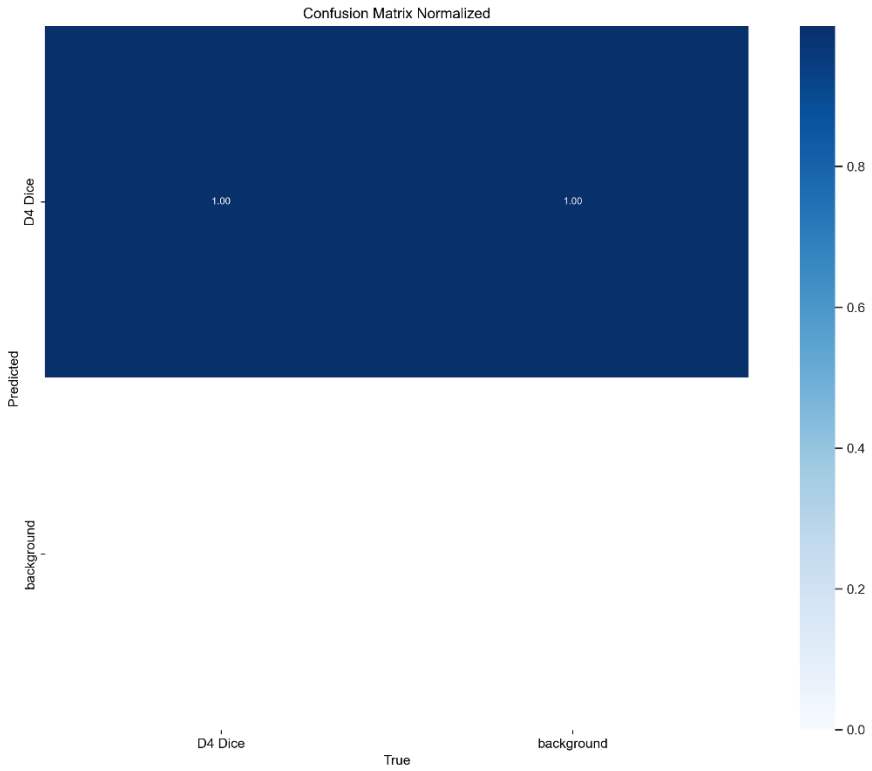
4. ANALIZA REZULTATA

U ovom poglavlju će biti prikazani rezultati treniranja neuronske mreže, pratit će se poboljšanje kvalitete rada neuronske mreže s obzirom na broj objekata i slika koje sadrži skup podataka te broj epoha i batch-eva koji su korišteni za treniranje.

4.1. Rezultati učenja

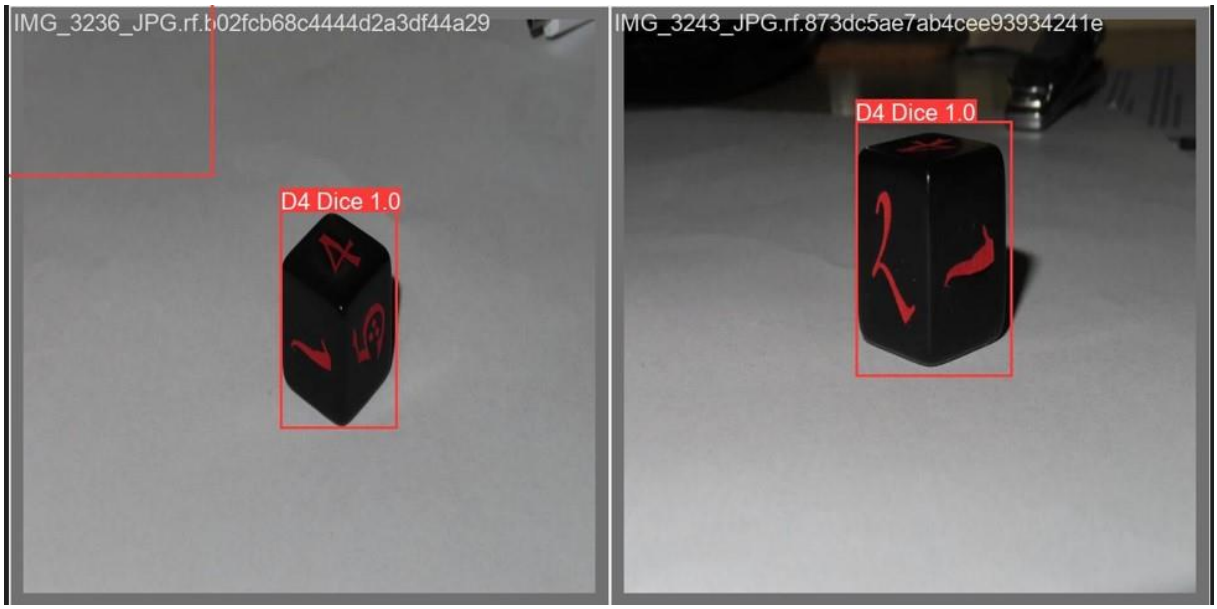
4.1.1. Slučaj 1.

Slučaj 1. se odnosi na treniranje neuronske mreže s malim brojem slika u skupu podataka i na samo jednoj klasi, cilj je provjeriti kako će se neuronska mreža ponašati kad ima mali broj ulaznih podataka.



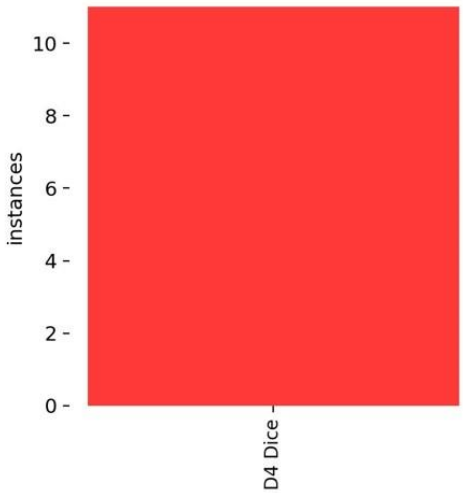
Slika 20. Grafički prikaz matrice konfuzije za slučaj 1.

Na slici 20 je vidljivo kako neuronska mreža iako ima samo jednu klasu, ne može dovoljno precizno naučiti i prepoznati tu klasu na svakom testom primjeru. Događa se da je model neuronske mreže uvijek 100% siguran u klasu D4 Dice, ali zna dolaziti do preklapanja klase i pozadine na kojoj se objekt s klasom D4 Dice nalazi, to znači da neuronska mreža zna prepoznati klasu D4 Dice na mjestu gdje se ne nalazi, nego ju zamijeni s nečim na pozadini.



Slika 21. Prikaz performanse predikcije modela za slučaj 1.

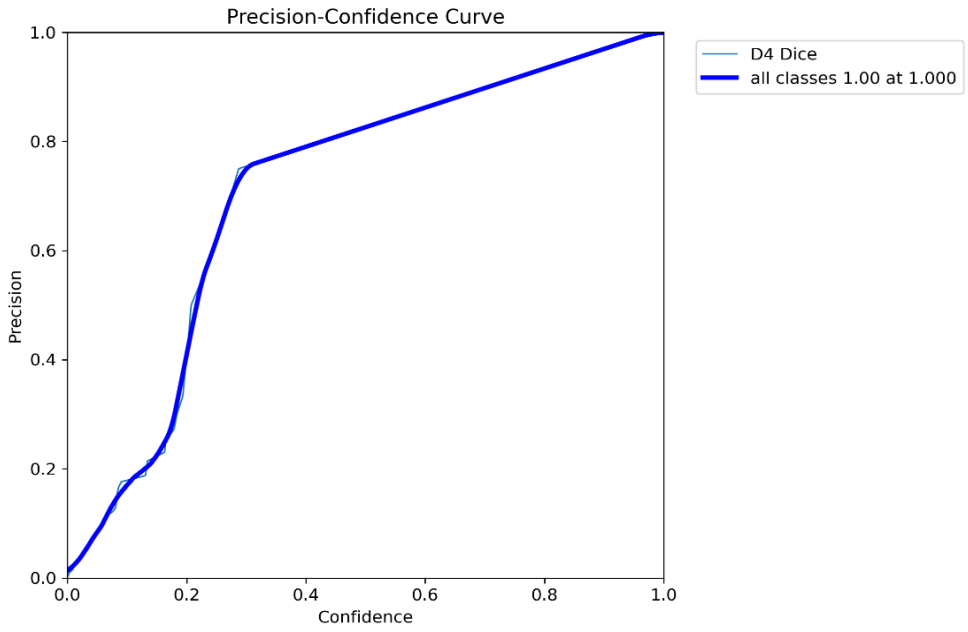
Na slici 21 je vidljivo kako je napravljen bounding box na mjestu gdje se uopće ne nalazi objekt. To može biti zbog lošeg osvjetljenja i zbog jako malo ulaznih podataka, te se neuronska mreža nije prilagodila netipičnim situacijama. Obje kockice je ispravno prepoznao i u predikciji vratio kolika je sigurnost u tu klasu.



Slika 22. Prikaz instanci klasa u skupu podataka za slučaj 1.

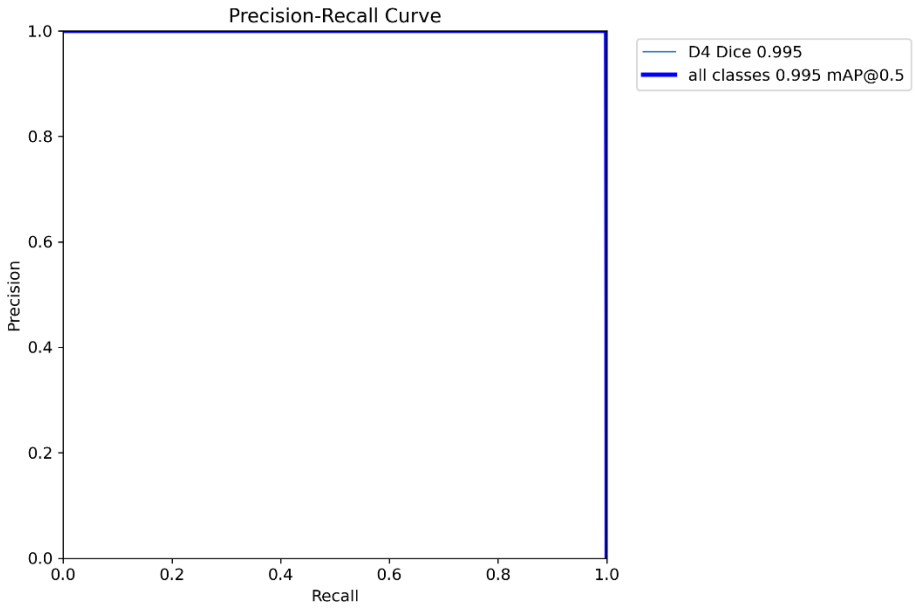
Model je tijekom treniranja zabilježio kako se klasa D4 Dice u skupu podataka pojavila deset puta, što nije potpuno ispravno, budući da su neke predikcije za granične okvire krive i model

je postavio granične okvire gdje se objekt sa zadanom klasom uopće ne nalazi. To je prikazano na slici.



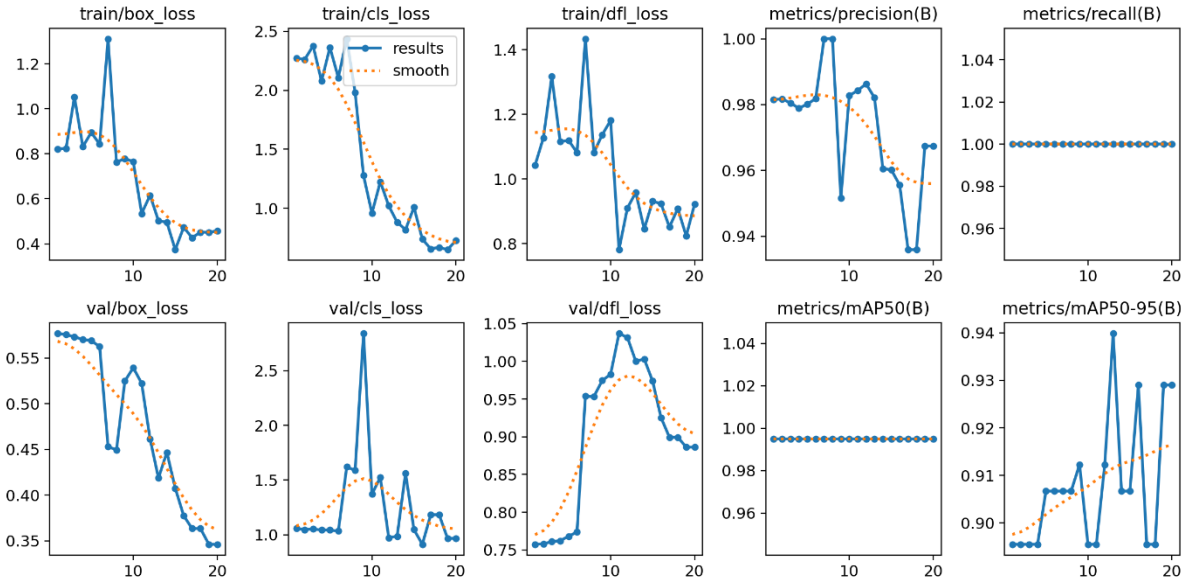
Slika 23. Grafički prikaz preciznost-sigurnost za slučaj 1.

Na apscisi grafičkog prikaza prikazuje se omjer sigurnosti modela u vlastitu predikciju i preciznosti same predikcije. Sigurnost modela u vlastitu predikciju govori koliko je model uvjeren da prepoznati objekt odgovara jednoj od klasa (u ovom konkretnom slučaju postoji samo jedna klasa) za čije prepoznavanje je model istreniran. Preciznost odgovara postotku slučajeva u kojima prepoznata klasa odgovara stvarnoj anotaciji. Ovim grafom se dobiva, kojoj sigurnosti modela odgovara preciznost same predikcije.



Slika 24. Grafički prikaz preciznost-odziv za slučaj 1.

Grafički prikaz prikazuje omjer preciznost i odziva, vidljivo je kako je neuronska mreža detektirala svaki navedeni objekt koji se nalazio na slikama, uključujući i granične okvire i klase. Budući da se u modelu nalaze objekti sa samo jednom klasom, preciznost je jednaka odzivu, što u budućim situacijama neće biti slučaj.

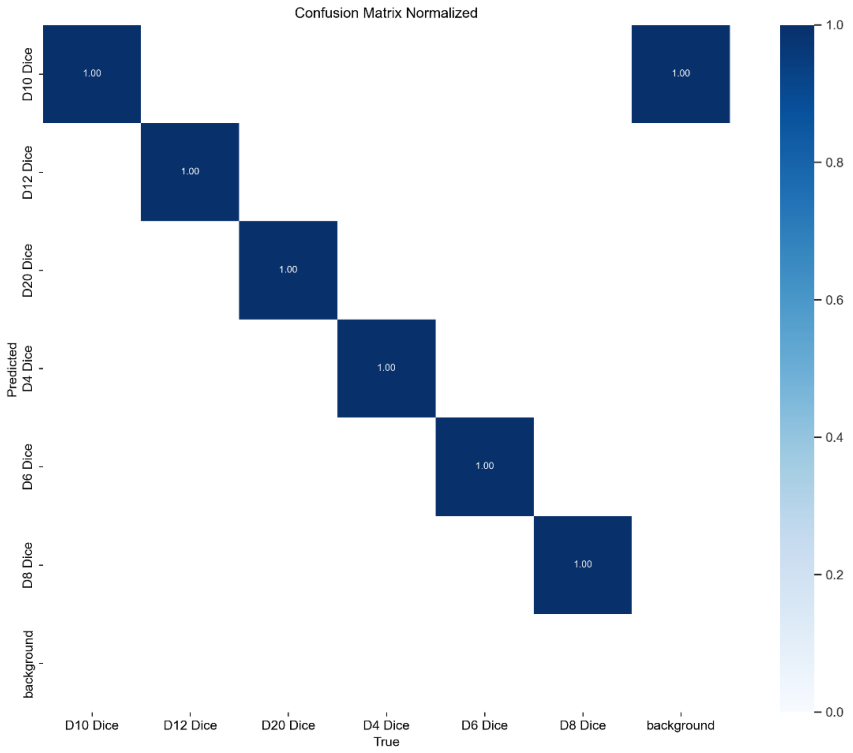


Slika 25. Grafički prikaz metričkih vrijednosti modela u procesu treniranja i validacije

Na apscisama svih navedenih grafičkih prikaza se nalazi kroz koliki je broj epoha prošla neuronska mreža. Grafikon train/box_loss prikazuje gubitke preciznosti između predikcijskog i stvarnog graničnog okvira, kao što je vidljivo na grafu, graf ima lagane oscilacije koje mogu biti posljedica razlike u bias i weith, lošem pozadinskom osvjetljenju, no graf ima silaznu putanju tijekom većine epoha, što znači da se gubitak koordinata graničnog okvira između predikcije i stvarnog objekt minimizirao. Grafikon train/cls_loss prikazuje gubitke u klasifikaciji između predikcije i stvarnog objekta, kao i grafički prikaz train/box_loss i ovaj graf je silazne putanje s obzirom na broj epoha, što znači da model sve bolje prepoznaje pojedine klase. Graf metrics/preciznost(B) prikazuje preciznost modela kroz pojedine epohe, vidljivo je kako model ima velike oscilacije s obzirom na broj epoha. S obzirom kako je neuronska mreža trenirala na malom skupu podataka, svaka slika je neuronskoj mreži bila nova situacija, zbog toga umjesto da preciznost raste s epohama, u ovom slučaju je imala laganu silaznu putanju. Graf val/box_loss prikazuje ponašanje gubitaka tijekom procesa validacije, gdje neuronska mreža nakon svake epohe uzima podatke iz validacijskog skupa podataka i provjerava ponašanje neuronske mreže, u ovom slučaju su se gubitci u razlici između graničnih okvira predikcije i stvarne slike minimizirale. Na grafu val/cls_loss je vidljivo kako klasifikacija raste između pete i desete epohe, što predstavlja loše naučen model, što znači kako za bolje rezultate treba povećati skup podataka s klasama i graničnim okvirima za bolje rezultate. Graf mAP50 predstavlja prosječnu vrijednost između preciznosti i odziva i ovdje iznosi otprilike 0.995, mAP tijekom epoha mora težiti prema broju jedan tj. 100%.

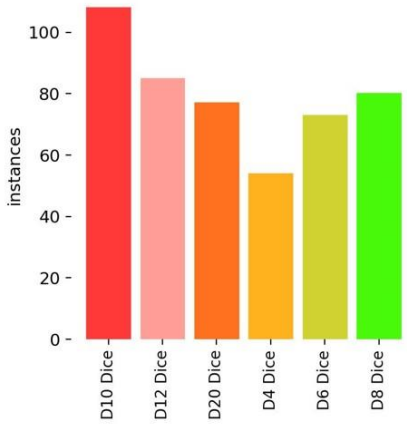
4.1.2. Slučaj 2.

Treniranje modela neuronske mreže je provedeno na malom skupu podataka, koji posjeduje sve zadane klase. Treniranje nove neuronske mreže je bio dug proces, jer je skup prošao kroz 200 epoha učenja. Cilj ovog treniranja je usporedba ponašanja neuronske mreže trenirane s skupom podataka koji sadrže veliki broj podataka i učenje s malim brojem epoha te skup podataka s malim brojem podataka i veliki broj epoha.



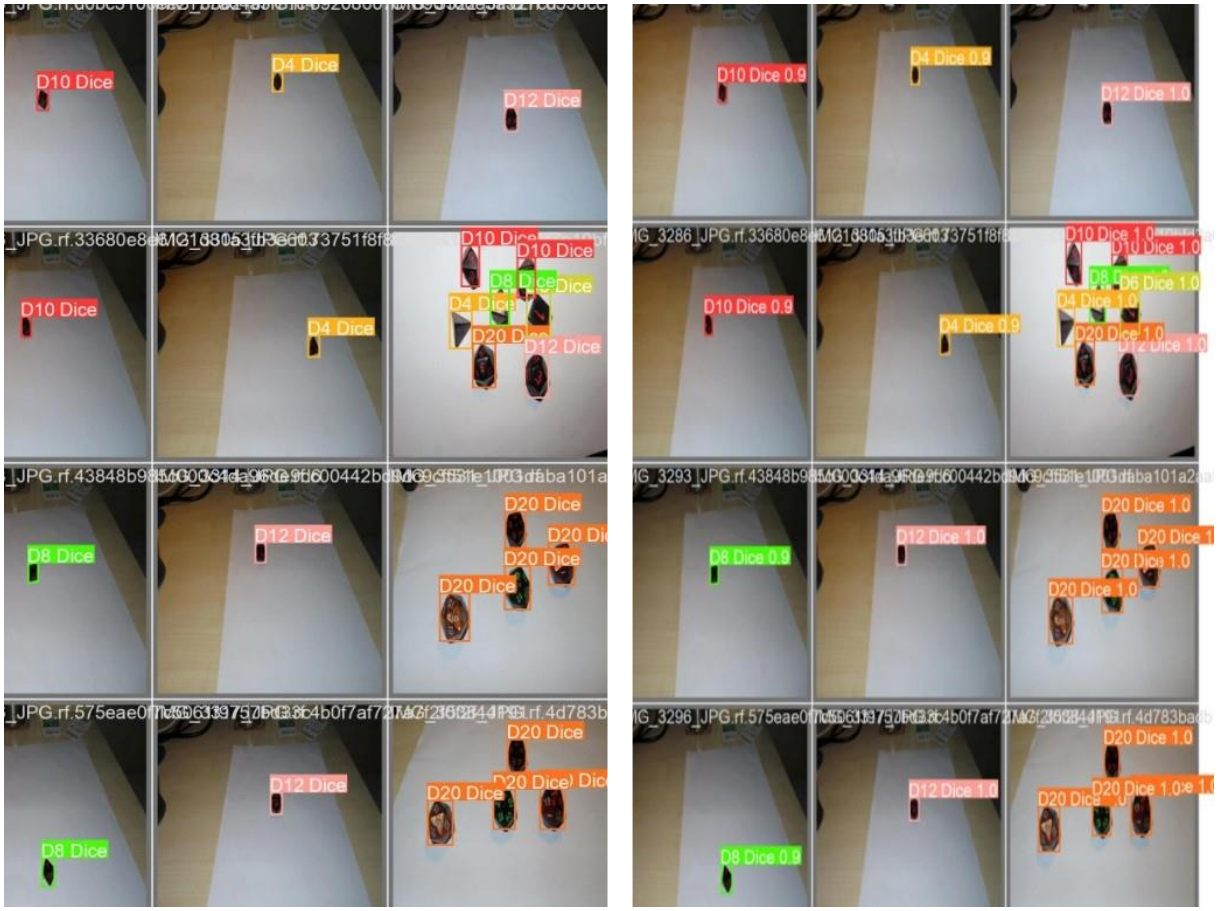
Slika 26. Grafički prikaz matrice konfuzija za 2.slučaj

Iz matrice konfuzija je vidljivo kako model neuronske mreže sa sigurnošću od 100% prepoznaje sve klase, jedino klasu D10 zna zamijeniti s pozadinom na kojoj se nalazi. Ovdje se može govoriti o pretreniranosti neuronske mreže, neuronske mreža je izgubila na robusnosti i fleksibilnosti, zbog toga što su se bias-i i weith-ovi prilagodili za točno određene slike na kojima je model trenirao, za trenirane slike radi perfektno, ali za slike koje neuronska mreža prije nije vidjela, neće dati tako dobre predikcije kao što je na matrici konfuzija prikazano. 200 epoha s batch-om koji je zadan kao broj 1 je prevelik broj ponavljanja, za mali skup podataka.



Slika 27. Prikaz broja instanci klasa u skupu podataka za slučaj 2.

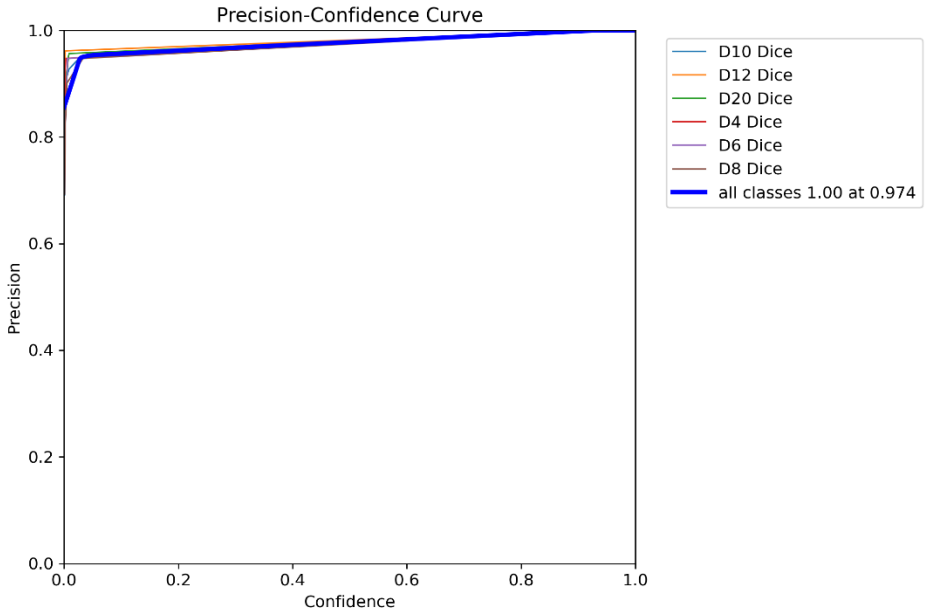
Iz grafičkog prikaza je vidljivo kako je model zabilježio pojavljivanje klase D10 Dice oko 100 puta u skupu podataka, što je ujedno i klasa s najvećim brojem pojavljivanja u modelu tijekom treniranja. Jedan od razloga može biti, kako se stvarno u skupu podataka najviše puta pojavljuje objekt kojemu odgovara klasa D10 Dice, drugi razlog je taj, što model neuronske mreže klasu D10 Dice zna zamijeniti s pozadinom na kojoj se objekt nalazi(vidljivo iz slike 26.), što automatski povećava broj pojavljivanja klase D10 Dice u procesu treniranja.



Slika a) Slika b)

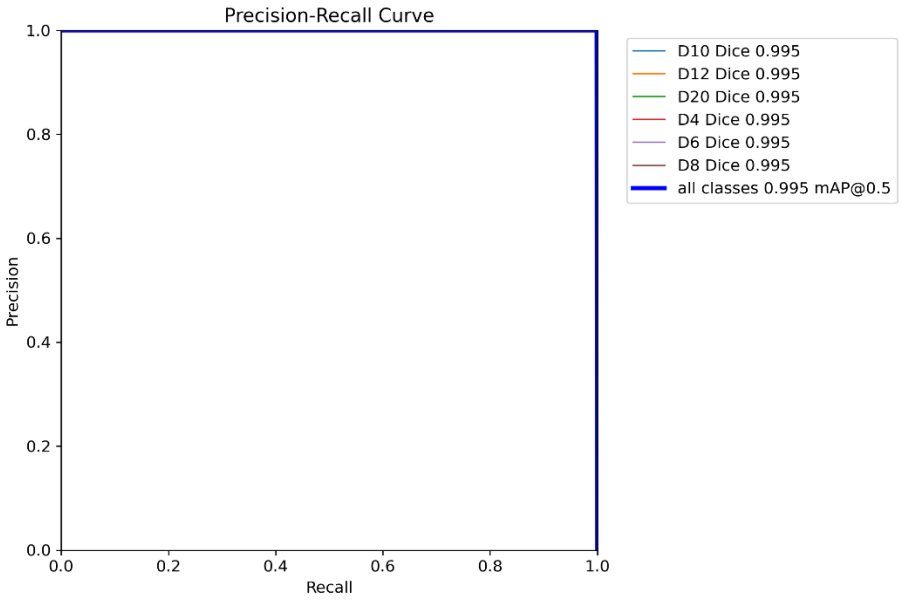
Slika 28. Slika a) (prikaz slika iz validacijskog skupa podataka) i slika b) (predikcija modela)

Na slici A je prikazan niz slika ručno anotiranih te smještenih u validacijski skup podataka, slika B prikazuje predikcije modela na istom skupu slika. Uspoređivanjem slike A i slike B, dolazi se do zaključka kako je neuronska mreža ima visoku preciznost, ako se razmatra isključivo validacijski skup podataka s točnosti preko 90%.



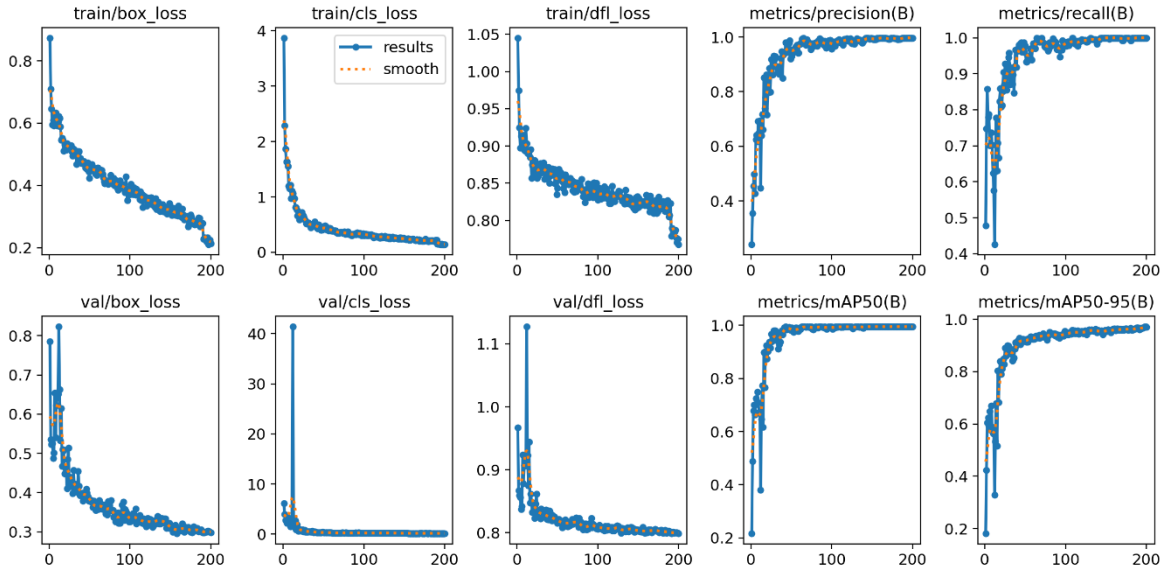
Slika 29. Grafički prikaz preciznost-sigurnost za slučaj 2.

Grafički prikaz govori kolika je preciznost modela i njegova sigurnost u predikciju. Krivulja preciznosti započinje na 0.85 i konstatno raste prema preciznosti od 100%. Neuronska mreža kada ima preciznost detekcije objekta s određenim graničnim okvirima i klasama 100%, javlja u predikciji sigurnost od 100%, u ostalim situacijama daje predikcije s velikom sigurnošću u određenu klasu.



Slika 30. Grafički prikaz preciznost-odziv za slučaj 2.

Grafikon preciznost-odziv prikazuje kako preciznost i odziv imaju prosječnu vrijednost oko 99.5%, što znači da model savršeno prepoznaje svaki objekt s klasama i graničnim okvirima i rijetko stvara pogrešku između stvarne slike i predikcije, ako model radi na treniranom skupu podataka.

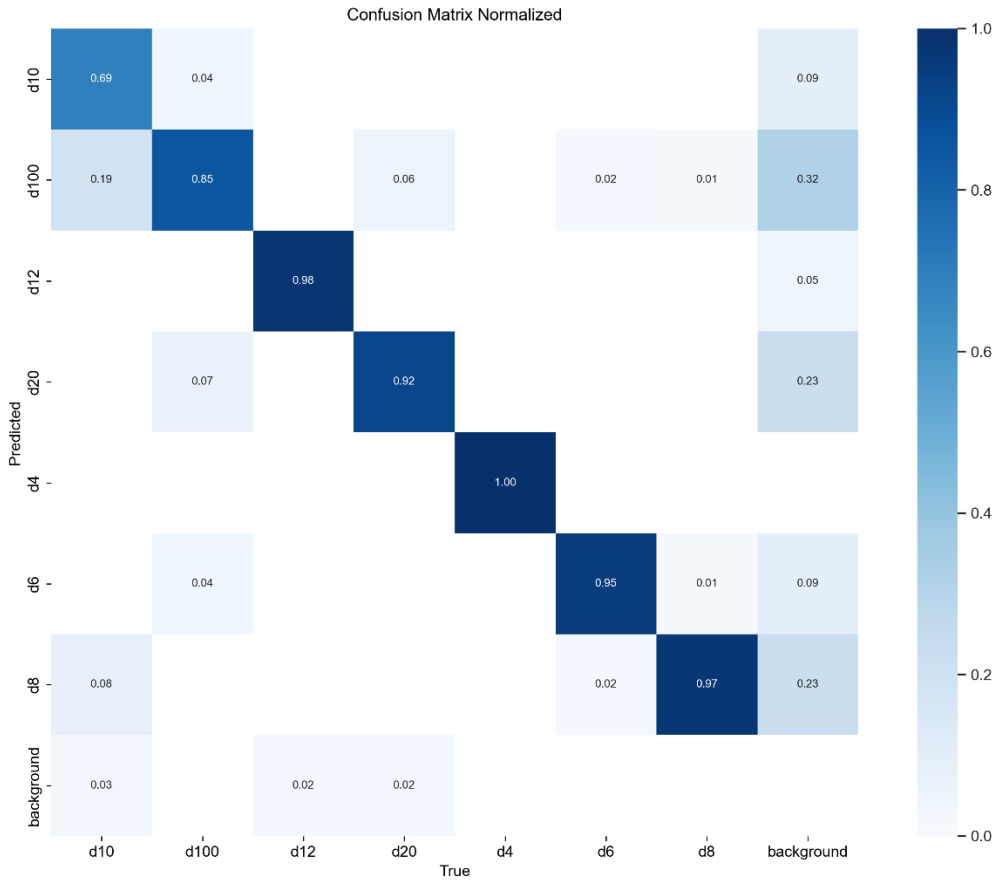


Slika 31. Grafički prikaz metričkih vrijednosti modela u procesu treniranja i validacije

Gubitak graničnog okvira za trening i validaciju, gubitak klasifikacije za trening i neuronsku mrežu imaju silaznu putanju, što znači da se taj gubitak minimizira tijekom svake epohe. Razlika između 100-te i 200-te epohe u minimiziranju greške je zanemariva te se broj epoha mogao smanjiti radi bržeg treniranja neuronske mreže. Prosječna vrijednost preciznosti i odziva (mAP50) teži u jedinicu već nakon 50 epoha, što daje veliku točnost neuronske mreže.

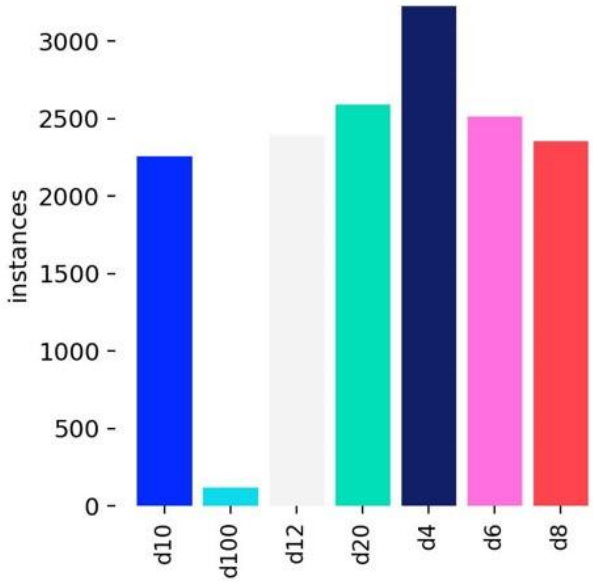
4.1.3. Slučaj 3.

Slučaj 3. sadrži sve zadane klase i veliki skup podataka, model neuronske mreže prolazi kroz 20 epoha učenja s batch-om koji je odabran kao 16, što znači kako neuronska mreža u isto vrijeme prolazi kroz 16 slika.



Slika 32. Matrica konfuzija za slučaj 3.

Iz matrice konfuzija se može vidjeti kako model ima neke nedostatke, sve klase osim klase D4 Dice model zna zamijenit s pozadinom. D100 i D10 su dvije klase kod kojih model ima najviši stupanj nesigurnosti, u nekim situacijama model zna poslati krivu predikciju i zamijeniti D100 s D10 i obrnuto. Za klasu D4 je jedino 100% siguran u svakoj predikciji.

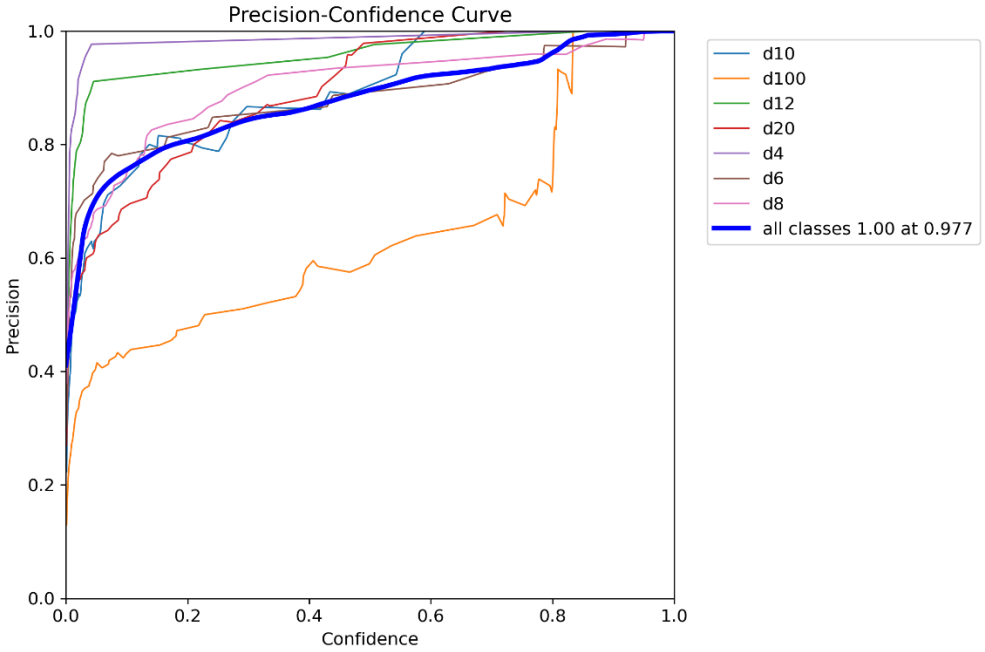


Slika 33. Grafički prikaz instanci modela za slučaj 3.

Objekata s klasom D100 ima najmanje u skupu podataka, zbog toga se neuronska mreža nije mogla najbolje naučiti razlikovati klasu D100 od klase D10, postoji još jedan problem, klasa D100 ima identičan oblik dekaedra kao i klasa D10, samo na njoj postoje dvoznamenkasti brojevi. Neuronska mreža bi se mogla poboljšati, kada bi naučila raspoznavati i brojeve. Iz grafikona je vidljivo kako u ovom skupu podataka postoji velik broj instanci, za razliku od slučaja 1. i slučaja 2.

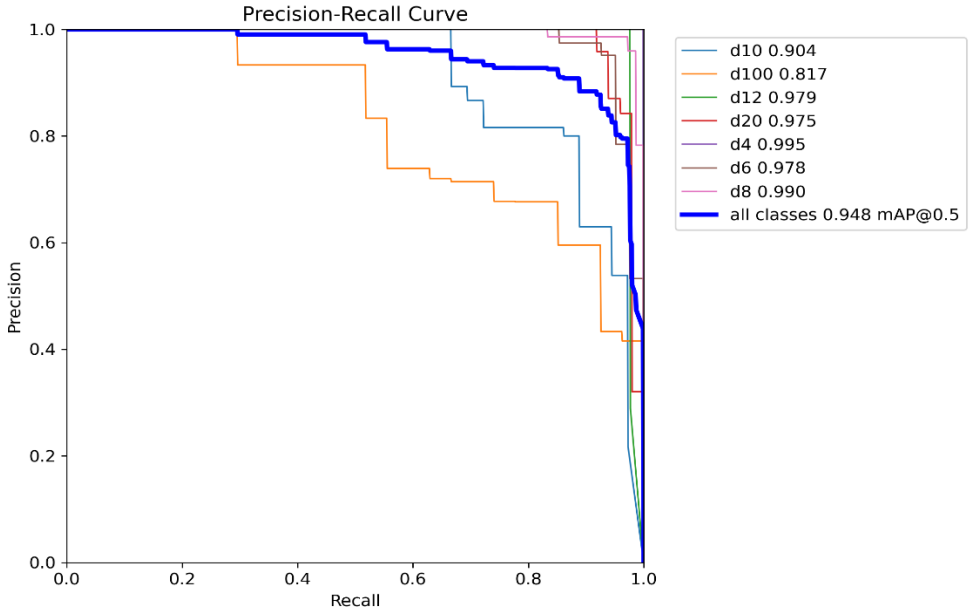
Slika 34. Slika a) (prikaz slika iz validacijskog skupa podataka) i slika b) (predikcija modela)

Sve klase i granične okvire je model neuronske mreže prepoznao i poslao predikciju sa sigurnošću u istu. Jedino za klasu D100 je siguran 40%, ali ju je dobro prepoznao.



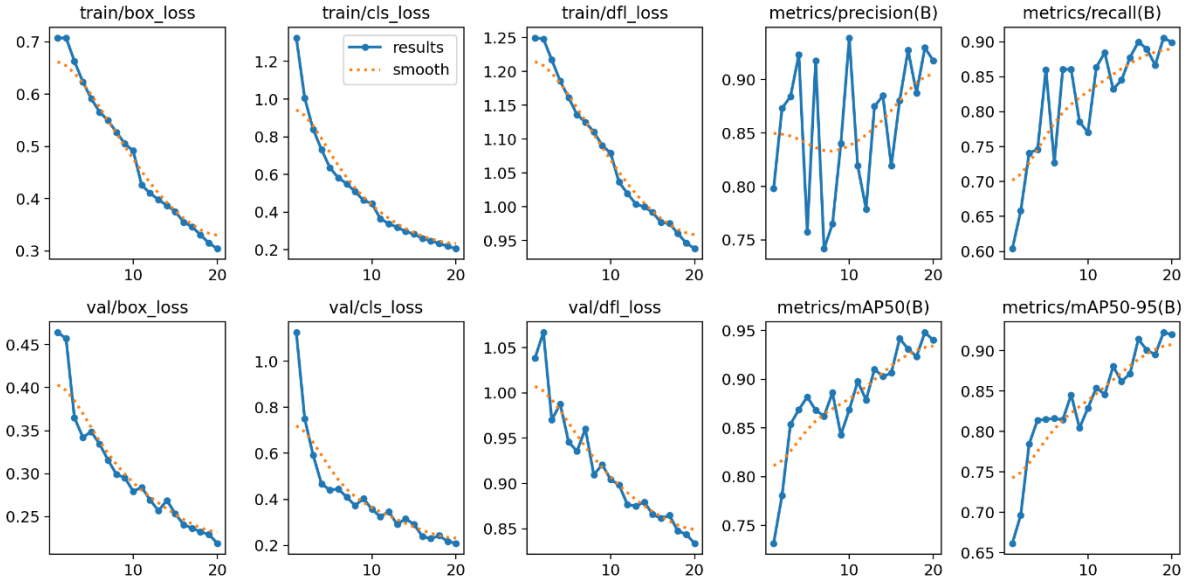
Slika 35. Grafički prikaz preciznost-sigurnost za slučaj 3.

Grafički prikaz preciznost-sigurnost samo potvrđuje sve navedeno u matrici konfuzija, model neuronske mreže je najmanje precizan i ima najmanju sigurnost u klasu D100, u D4 je siguran gotovo 100%. Sveukupno model je siguran u svoje predikcije oko 97%, što je više nego zadovoljavajuće.



Slika 36. Grafički prikaz preciznost-odziv za slučaj 3.

Iz grafičkog prikaza na slici 36. se vidi kako je najveća razlika između preciznosti i odziva na klasi D100 i D10, što prikazuje i matrica konfuzija. Model nije idealan, jer ima protora za poboljšavanje, ali zadovoljavajućih parametara.

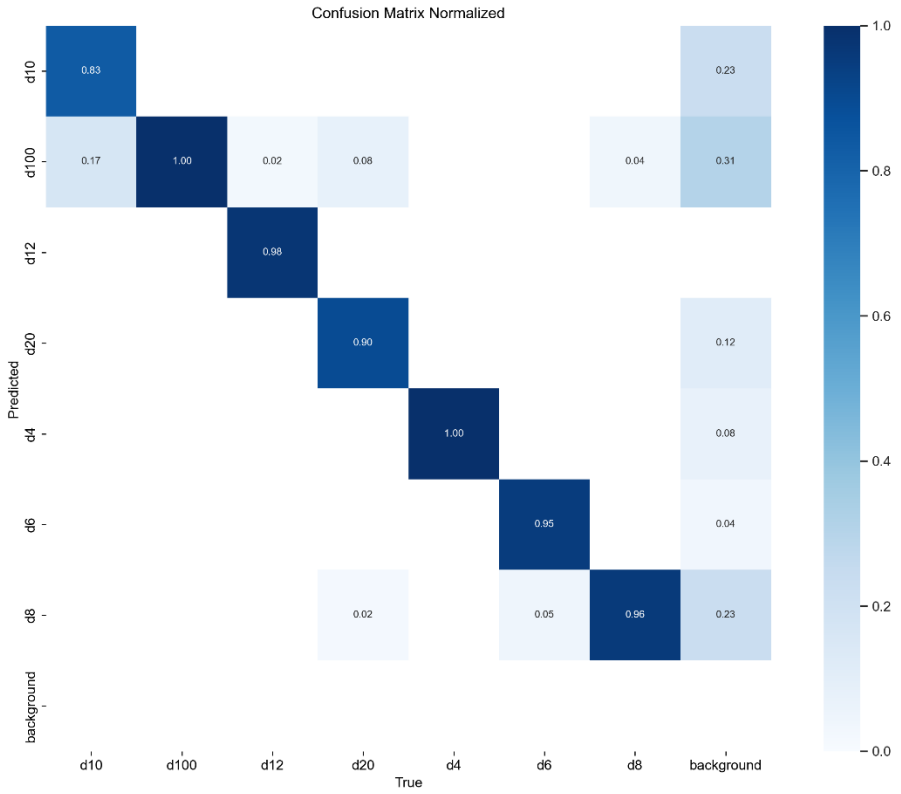


Slika 37. Grafički prikaz metričkih vrijednosti modela u procesu treniranja i validacije

Grafovi povezani s treniranjem: train/box_loss, train/cls_loss i train/dfl_loss, kao i grafovi za validaciju: val/box_loss, val/cls_loss, val/dfl_loss imaju silaznu putanju tijekom epoha, vidljivo je još kako se krivulja nije ustalila na neku vrijednost minimiziranog gubitka, što znači kako je bilo potrebno provesti još nekoliko epoha treniranja, kako bi se maksimalno minimizirao gubitak i poboljšao rad modela neuronske mreže. Preciznost i odziv imaju velike oscilacije između silazne i uzlazne putanje, ali ipak crvena isprekidana krivulja ima uzlaznu putanju, što znači kako preciznost i odziv modela neuronske mreže i unatoč velikim oscilacijama bilježi poboljšanje tijekom svake epohe.

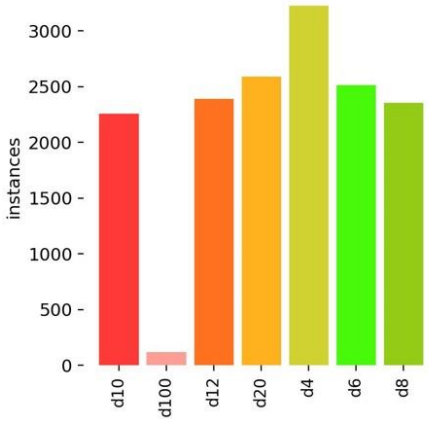
4.1.4. Slučaj 4.

U slučaju 4. treniranje modela neuronske mreže je provedeno na istom skupu podataka kao slučaj 3., samo sa 100 epoha i batch-om koji odabran kao 1. Ovaj model će provesti detaljnu analizu svake slike posebno pa će se usporediti rezultati sa slučajem 3..



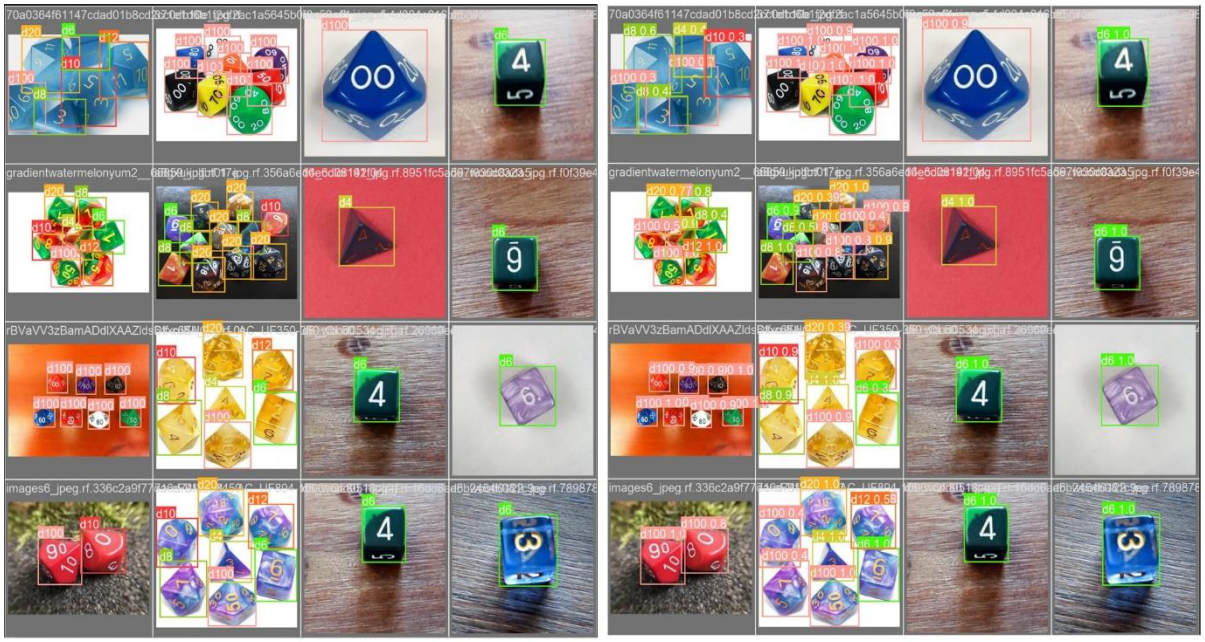
Slika 38. Matrica konfuzija za slučaj 4.

Iz matrice konfuzija je vidljivo kako je model neuronske mreže poboljšao svoje specifikacije i povećao sigurnost u prepoznavanje klasa D10 i D100. To poboljšanje nije znatno bolje, s obzirom kako je 80 epoha treniranja više napravljeno nego u slučaju 3., za ostale klase slučaj 3. je već imao više nego zadovoljavajuće rezultate, što je kroz 80 epoha zanemarivajuće poboljšano. S ovolikim povećanjem broja epoha samo je model neuronske mreže puno duže prolazio kroz treniranje, usporen je rad neuronske mreže i kao rezultat se dobilo malo poboljšanje.



Slika 39. Grafički prikaz instanci modela za slučaj 4.

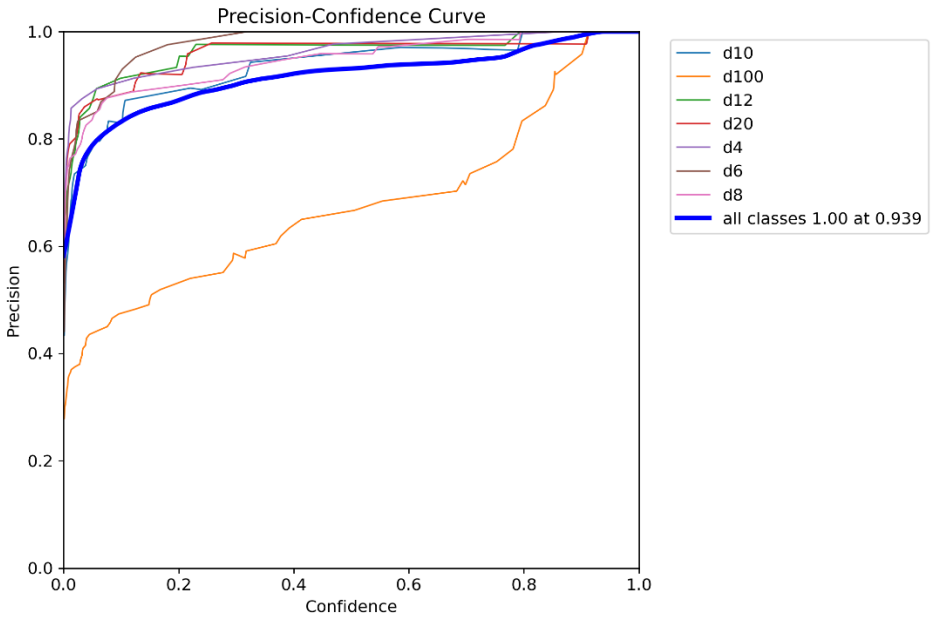
Grafički prikaz instanci modela neuronske mreže bi trebao biti identičan kao za slučaj 3., no vidljive su male promjene u broju instanci modela. U ovom slučaju je neuronske mreža detaljnije pregledavala svaku sliku i imala veću preciznost i točnost pri određivanju klasa. Razlika u broju instanci između slučaja 4. i slučaja 3. je zanemariv i ne utječe previše na poboljšanje rada neuronske mreže.



Slika a) Slika b)

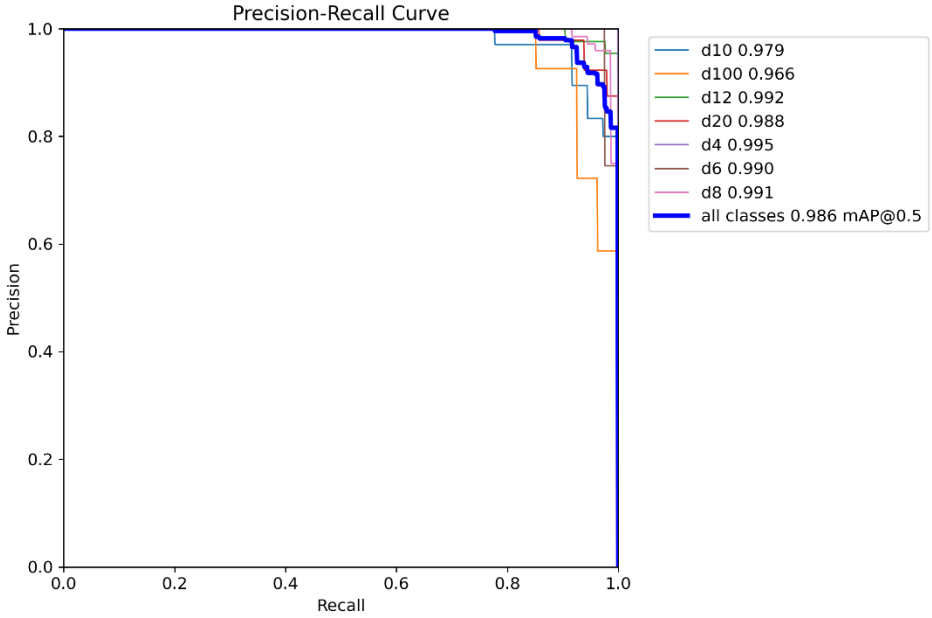
Slika 40. Slika a) (prikaz slika iz validacijskog skupa podataka) i slika b) (predikcija modela)

Slika a) prikazuje stvarnu sliku prije treniranja sa svim potrebnim graničnim okvirima i klasa, slika b) je predikcija tijekom procesa treniranja. Na slici a) u donjem lijevom kutu i na slici b) u donjem desnom kutu je vidljivo kako je model neuronske mreže zamijenio klase D10 i D100 i za tu predikciju dao sigurnost od 90%. Kada su objekti s klasama individualno na slikama, točnost neuronske mreže je 100%, dolazi do krivih predikcija kada su objekti u grupama i nije dobra preglednost slike. Slučaj 4. u usporedbi sa slučajem 3. daje bolje rezultate, ali ako se u obzir uzme vrijeme učenja neuronske mreže i malog poboljšanja, slučaj 3. bi se mogao unaprijediti s nekoliko dodatnih epoha.



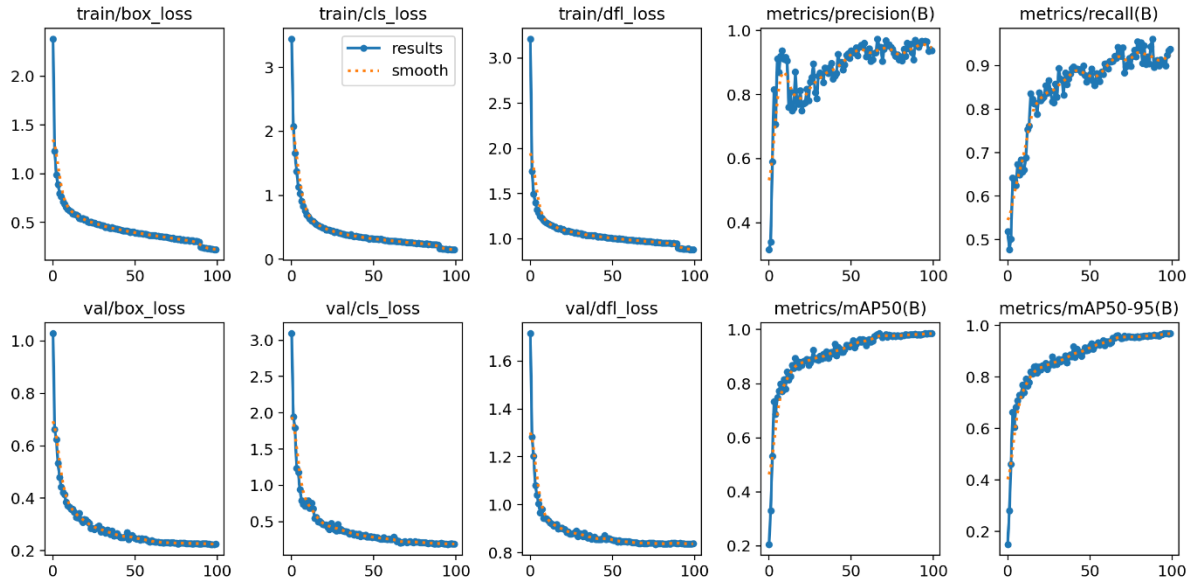
Slika 41. Grafički prikaz preciznost-sigurnost za slučaj 4.

Iz grafičkog prikaza preciznost-sigurnost za slučaj 4. se vidi kako model neuronske mreže nije siguran u klasu D100 kao u ostale klase, što se vidjelo iz matrica konfuzije.



Slika 42. Grafički prikaz preciznost-odziv za slučaj 4.

Preciznost i odziv su nakon 100 epoha imaju približno jednake vrijednosti, na što ukazuje plava krivulja, koja daje prosječnu vrijednost između odziva i preciznosti iznačunatu iz formule (3) 98.6%.



Slika 43. Grafički prikaz metričkih vrijednosti modela u procesu treniranja i validacije

Krivulje na grafičkim prikazima za validaciju i treniranje imaju silaznu putanju, što znači kako je neuronske mreža tijekom epoha minimizirala gubitke vezane uz pojedinu komponentu. Neuronska mreža već nakon 50 epoha prikazuje zavidne rezultate te između 50-te i 100-te epohe je razlika u minimizaciji gubitka zanemariva, što ukazuje da je neuronska mreža bez potrebe obavila još 50 epoha. Ista situacija se važi i za grafove na kojim se nalazi krivulja za preciznost i odziv, krivulje već nakon 50 epoha prikazuju veliku preciznost i odziv za model neuronske mreže, što se vidi u grafu mAP50, gdje se krivulja za prosječnu vrijednosti između odziva i preciznosti asimptotski približava broju 1.

4.1.5. Kritički osvrt na navedene slučajeve.

Slučaj 4. daje najbolje rezultate za prepoznavanje klasa i graničnih okvira, ali provedena je na velikom skupu podataka i sa 100 epoha, očekivale su se najbolje performanse neuronske mreže, ali ako se uzme u obzir vrijeme trajanja učenja modela u odnosu na slučaj 3., gdje je neuronska mreža trenirala na istom skupu podataka kao za slučaj 4. s 20 epoha i proces treniranja je trajao nekoliko dana manje nego za slučaj 3., a rezultati su i dalje na zadovoljavajućoj razini. Što govori o tome, da za ispravan rad neuronske mreže treba posjedovati kvalitetan skup podataka i imati optimalan broj epoha, nakon kojeg neuronska mreža ima najmanje vrijeme treniranja, a daje najbolje rezultate. Za ovaj skup podataka i usporedbu rezultata za slučaj 3. i slučaj 4., dolazi se do zaključka, kako optimalan broj epoha bi bio između 40 i 60.

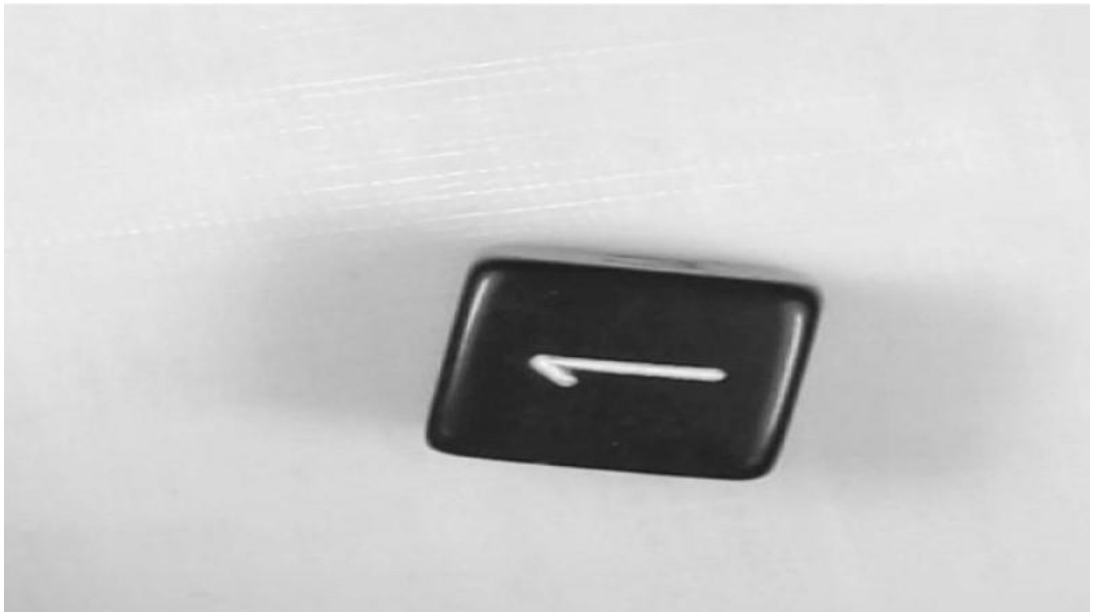
4.2. Dodatak modelu neuronske mreže

S obzirom da se u društvenoj igri D&D potrebna destincija ne samo tipa kockice, nego i broj na koji će se ta kockica zaustaviti. Ideja je izvući podatke iz same kockice pomoću obrade slike te na taj način uz pomoć alata kao što su OpenCV i Pytesseract. Pytesseract je pythonov programski paket koji se bazira na pretvaranju oblika u tekst ili broj. OpenCV je pythonov programski paket koji služi za manipulaciju i obradu slika te izdvajanja ključnih podataka iz slika.

Tok procesa je u prvom planu prepoznavanja tipa kockice te prepoznavanje broja s idejom prijavljivanja automatskog rezultata prilikom bacanja kockica. Proces izdvajanja broja se prvo vrši prebacivanje originalne slike iz RGB (RedGreenBlue) formata u BGR (BlueGreenRed) format, što je standardni postupak kod analize slike pomoću računalnog vida. Nakon toga iz BGR formata u Grey (sivi) format. Pretpostavka je, da je broj najsvjetljiji element u slici, što također uključuje tamnu pozadinu. Napisani kod bi u tom slučaju izdvojio najveći svijetli element (u ovom slučaju broj) te bi se ta slika proslijedila Pytesserect-u koji bi sa svojim integriranim funkcijama pretvorio sliku u string. Vrijedno je spomena, kako se koristeći Pytesserect-a prepoznavanje može ograničiti na skup željenih znakova te je u ovom slučaju učinjeno na rasponu numeričkih znakova od nula do devet.

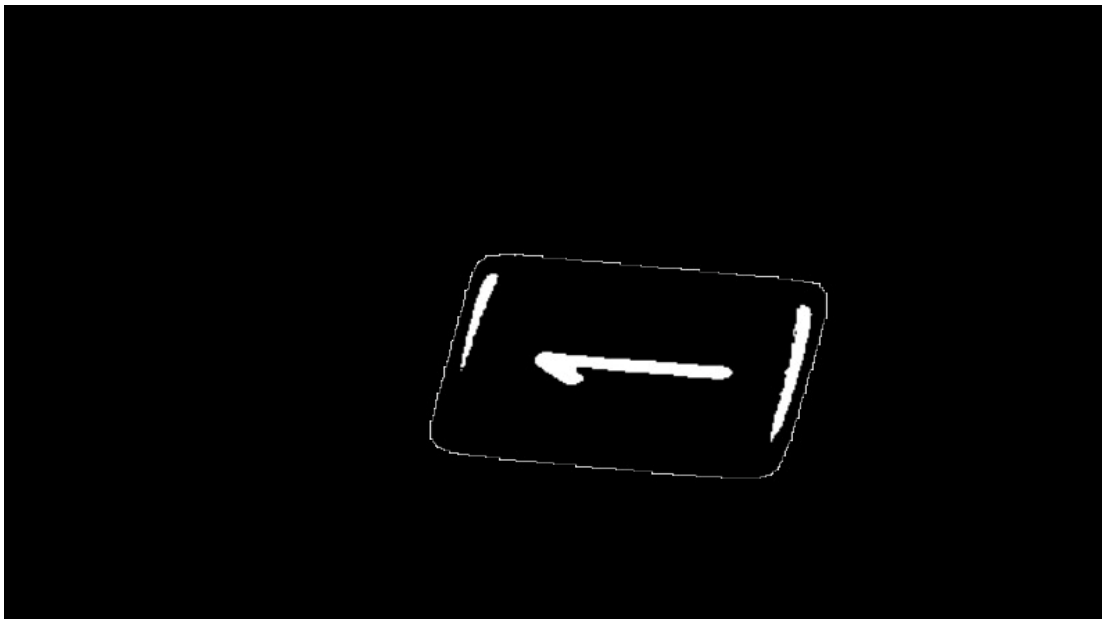
Slika 44. Originalna slika u RGB formatu

Na slici 44. je prikazana slika u RGB (BGR s obzirom na konverziju koju OpenCV zahtjeva) formatu na kojoj se vrši dodatna analiza.



Slika 45. Slika u sivom (grey) formatu

Slika se prebacuje u sivi format s pretpostavkom da je broj najsvjetljiji element te se na sliku primjenjuje maska s donjom i gornjom granicom, koja je u rasponu, koji odgovara bijeloj boji visokog inteziteta.



Slika 46. Primjena maske na slici sivog formata

Primjenom maske na sliku dobiva se krajnji rezultat kao na slici 46. na kojoj je broj vidljiv. Pomoću lokalizacije provedene nad svim elementima koji ulaze u donju granicu intenziteta, nalazi se najveći element, koji bi prema pretpostavci bio broj na kockici. Taj izdvojeni element (broj) je ulaz u Pytesserect, čija je funkcija prebacivanje slike u string, dodatna preciznost je postignuta s ograničenjem prepoznavanja brojeva isključivo u intervalu od 0 do 9.

4.2.1. Nedostaci

Prvi nedostatak je loše prepoznavanje i konverzija Pytesserect-a pa tako i u onim slučajevima, koji se naizgled čine kao jednostavni. Nije rijetko da Pytesserect prepozna krivi broj pa je tako konzistencija najveći problem Pytesserect-a. Pytesserect je svoju bazu podataka trenirao na tekstualnim datotekama, stoga nije neobično da u okruženju kao što je kockica teško prepoznaje broj s obzirom na nekonzistentnoj orijentaciji, veličini, fontu broja.

Drugi nedostatak ove pretpostavke je pozadina koja u određenim slučajevima može imati sličan ili isti intezitet kao i sam broj te u tom slučaju lokalizacija uzima pozadinu kao element za prepoznavanje.

Treći i posljednji značajni nedostatak se odnosi na boju kockice i broja, koja nije dosljedna na svim setovima kockica te to čini ovaj program još manje robusnim.

ZAKLJUČAK

U ovom diplomskom radu je bilo potrebno napraviti neuronsku mrežu za prepoznavanje geometrijskih tijela pomoću YOLOv8. Prepoznavanje geometrijskih tijela se pokazalo kao kompleksna radnja, čak i za naprednu i modernu metodu kao što je YOLOv8. Treniranje je provedeno na četiri različita slučaja, slučaj 1 i slučaj 2 imaju mali broj podataka te su se pokazali kao nepouzdani i neprecizni. Za slučajeve 3 i 4 je učenje provedeno na identičnom skupu podataka s različitim brojem epoha. Model proveden na 20 epoha daje zadovoljavajuće rezultate, koji mogu biti poboljšani korištenjem dodatnog broja epoha. Kod modela na kojem je provedeno 100 epoha treniranja, došlo je do pretreniranosti, što znači da za skup podataka az treniranje model daje perfektne rezultate, a za modele testiranja zadovoljavajuće. Model je na grafičkim prikazima za gubitak graničnog okvira i klasifikacije, već se nakon 50 epoha asimptotski približavati apscisi i u drugih 50 epoha je taj napredak zanemariv. Za prepoznavanje brojeva potrebna je bolja metoda od Pytesserect-a koji je treniran na tekstualnim podacima. Za prepoznavanje brojeva potreban je poseban model, koji bi bio treniran na bazi podataka sa samih kockica te bi se tako eliminiralo nedostatke u vidu boje, oblika i fonta.

Da bi se ovaj model pustio u komercijalnu proizvodnju potrebno ga je učiniti još robusnijim te ga ukomponirati s modelom prepoznavanja broja, kako bi korisnik imao cjelokupan paket.

LITERATURA:

[1] *ResearchGate*. URL: https://www.researchgate.net/figure/ReLU-activation-function_fig3_319235847 (Pristupljeno 2024-6-20)

[2] *NPR in Kansas City*. URL: https://www.kcur.org/arts-life/2023-05-20/kansas-city-dungeons-dragons-tabletop-rpg-campaign-games (Pristupljeno 2024-6-20)

[3] *Anigota.hr*. URL: https://www.anigota.hr/canon-digitalni-fotoaparat-powershot-sx740-hs-bk/5863/product/ (Pristupljeno 2024-6-20)

[4] *Roboflow*. URL: https://blog.roboflow.com/upload-model-weights-yolov8/ (Pristupljeno 2024-6-22)

[5] *PyTorch*. URL: https://pytorch.org/get-started/locally/ (Pristupljeno 2024-6-22)

[6] *Pyimagesearch*. URL: https://pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/ (Pristupljeno 2024-6-22)

[7] *Ultralytics*. URL: https://docs.ultralytics.com/quickstart/#install-ultralytics (Pristupljeno 2024-6-22)

[8] *Biologija 8*. URL: https://edutorij-admin-api.carnet.hr/storage/extracted/3b8a4b4e-84b0-4580-aa6f-e38efe028ed9/biologija-8/m03/j01/index.html (Pristupljeno 2024-6-20)

[9] *Ultralytics*: URL: https://docs.ultralytics.com/ (Pristupljeno 2024-6-22)

[10] *ResearchGate*. URL:https://www.researchgate.net/figure/The-structure-of-an-artificial-neuron-consists-of-a-set-of-inputs-weights-biases-and_fig3_342801342 (Pristupljeno 2024-6-20)

[11] Builtin: URL: https://builtin.com/machine-learning/relu-activation-function (Pristupljeno 2024-6-22)

[12] *Kili-techology*. URL: https://kili-technology.com/data-labeling/machine-learning/mean-average-precision-map-a-complete-guide (Pristupljeno 2024-6-22)

[13] Dilberović, I. (2020) *Prepoznavanje slika pomoću konvolucijske neuronske mreže*. Završni rad. Zagreb: Algebra. (Pristupljeno 2024-6-20)

PRILOG:

Skripta treniranje:

```
from ultralytics import YOLO
import torch

if __name__ == '__main__':

    #device = 'cuda' if torch.cuda.is_available() else 'cpu'
    model = YOLO('yolov8l.pt')  #.to(device)
    #torch.backends.cudnn.benchmark = True

    model.train(data="C:\\Users\\Domagoj\\Desktop\\Zadro
Dipl\\Data\\data.yaml", epochs=20, imgsz=640, batch=4)#,
device=device)

    metrics = model.val()

    model.save('best_model.pt')
```

Skripta testiranje:

```
From ultralytics import YOLO
import cv2

path =
f"C:\\Users\\Domagoj\\runs\\detect\\train5\\weights\\best.pt"
big_dataset_path = "C:\\Users\\Domagoj\\Desktop\\Zadro
Dipl\\detect\\train2\\weights\\best.pt"


model = YOLO(big_dataset_path)
source = f"C:\\Users\\Domagoj\\Desktop\\Zadro Dipl\\Real Life
data\\Train_slika.png"

resized_path = 'C:\\Users\\Domagoj\\Desktop\\Zadro
Dipl\\Testna_slika.jpg'
image = cv2.imread(source)
resized_image = cv2.resize(image, (957, 641))

cv2.imwrite(resized_path, resized_image)

#model.predict(source, stream=  True, conf=0.3, save = True)
```

```
result = model.predict(resized_image ,show = True, conf=0.3,
save = True)

print('gotovo')
```

Skripta prepoznavanje brojeva:

```
import cv2
import pytesseract
import numpy as np

# Read the image
put_do_slike = "C:\\Users\\Domagoj\\Desktop\\Zadro
Dipl\\Data\\test\\images\\d10_wood0584_jpg.rf.27d117cf1e8a150f
01e476fbee15c7d4.jpg"


image = cv2.imread(put_do_slike)
rgb_image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)
kopija = image.copy()
print(np.shape(kopija))
ratio = 640*2252//4000
kopija = cv2.resize(kopija, (640, ratio))

# Convert the image to grayscale
gray = cv2.cvtColor(kopija, cv2.COLOR_BGR2GRAY)

lower_white = np.array([0,0,168])
upper_white = np.array([172,111,255])

mask = cv2.inRange(gray, 220, 255)

result = cv2.bitwise_and(kopija, kopija, mask=mask)

contours, _ = cv2.findContours(mask, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)


contours = sorted(contours, key=cv2.contourArea, reverse=True)

x, y, w, h = cv2.boundingRect(contours[0])
number_region = gray[y:y+h, x:x+w]
invertirano = cv2.threshold(number_region, 100, 255,
cv2.THRESH_BINARY)[1]
```

```
number = pytesseract.image_to_string(invertirano, config='--
psm 6 --oem 3 -c tessedit_char_whitelist=0123456789')

print("Extracted Number:", number.strip())



# Display the masked image
cv2.imshow('Masked Image', invertirano)
cv2.imshow('Orignial Image', kopija)
cv2.imshow('Gray scaled Image', gray)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Skripta prikaz videa:

```
import cv2
from ultralytics import YOLO

pretrained = f"C:\\Users\\Domagoj\\Desktop\\Zadro
Dipl\\detect\\train2\\weights\\best.pt"

# Load the YOLOv8 model
model = YOLO(pretrained)

# Open the video file
video_path = f"C:\\Users\\Domagoj\\Desktop\\Zadro Dipl\\Real
Life data\\20240628_111908.mp4"
cap = cv2.VideoCapture(video_path)

# Loop through the video frames
while cap.isOpened():
    # Read a frame from the video
    display_width = 1920
    display_height = 1080
    success, frame = cap.read()
    height, width, _ = frame.shape
    aspect_ratio = width / height
    if aspect_ratio > display_width / display_height:
            new_width = display_width
            new_height = int(display_width / aspect_ratio)
    else:
            new_width = int(display_height * aspect_ratio)
            new_height = display_height
```

```
    if success:

        resized_frame = cv2.resize(frame, (new_width,
new_height))
        # Run YOLOv8 inference on the frame
        results = model(resized_frame, save = True)

        # Visualize the results on the frame
        annotated_frame = results[0].plot()


        # Display the annotated frame
        cv2.imshow("YOLOv8 Inference", annotated_frame)

        # Break the loop if 'q' is pressed
        if cv2.waitKey(1) & 0xFF == ord("q"):
            break
    else:
        # Break the loop if the end of the video is reached
        break

# Release the video capture object and close the display
window
cap.release()
cv2.destroyAllWindows()
```