UNIVERZITET U ZENICI

POLITEHNIČKI FAKULTET

SEMINARSKI RAD IZ PREDMETA:
Prepoznavanje oblika i obrada slike

Tema rada:	**Detekcija i klasifikacija saobraćajnih znakova**

Predmetni nastavnik:	van. prof. dr. Nermin Goran

Student:	Ajdin Bukvić
Broj indeksa:	II-89
Usmjerenje:	Softversko inženjerstvo
Godina studija:	1. godina, 2. ciklus
Rezultat rada:	Primjena različitih AI/ML algoritama za detekciju i klasifikaciju saobraćajnih znakova na posebno pripremljenom skupu podataka, uz osvrt i analizu na rezultate implementacije korištenjem testnih slika i videa, radi evaluacije performansi.

Datum: 05.02.2024.

SADRŽAJ

1. UVOD

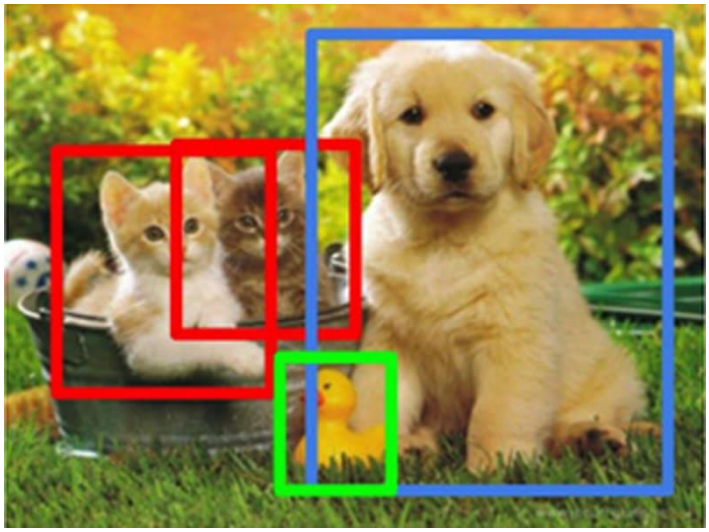
Razvojem tehnologije zadnjih nekoliko godina došlo je do velikog porasta za uključivanje tehnologije u razne poslovne procese i zadatke koji se mogu na vrlo brz i efikasan način riješiti primjenom različitih tehnologija. Područje mašinskog učenja i vještačke inteligencije je znatno napredovalo i postalo je neophodan alat za izvršavanje raznih zadataka. Pošto se svakodnevno pojavljuju novi podaci na internetu generisani iz raznih izvora, ovi algoritmi procesom treniranja modela su sve uspješniji i precizniji. Poseban dio vještačke inteligencije odnosi se na područje kompjuterske vizije (eng. computer vision). Upotreba naprednih algoritama omogućava izvršavanje različitih operacija na slikama ili videu, koji se tretiraju kao poseban tip podataka, za razliku od standardnih tekstualnih podataka.

Kroz ovaj seminarski rad će biti prikazani procesi detekcije i klasifikacije na primjeru saobraćajnih znakova. Za odvijanje ovih procesa prethodi nekoliko operacija nad slikama (ili videu). Ove operacije mogu uključivati obradu, procesiranje, segmentaciju i analizu slike. Digitalna slika je predstavljena skupom piksela, te je prilikom rada sa slikama potrebno uzeti u obzir različite osobine slike, kao što su boje i veličina. Naknadni zadaci obrade su otkrivanje ivica, oblika i objekata. Praktična primjena detekcije i klasifikacije obuhvata razne segmente, koji olakšavaju svakodnevne poslove, a na konkretnom primjeru saobraćajnih znakova može se koristiti u vozilima.

2. DETEKCIJA

Detekcija slike je proces koji obrađuje sliku i otkriva objekte u njoj. Detekcija obično prethodi procesu klasifikacije, jer je prvo potrebno odrediti (izdvojiti) tražene objekte sa slike, kako bi se oni nakon toga mogli klasifikovati, odnosno dodijeliti jednoj od klasa (grupa, kategorija). Procesom detekcije moguće je utvrditi tačnu lokaciju nekog objekta na slici ili saznati tačan broj objekata na slici. Objekat na slici može biti bilo šta iz prirode, kao naprimjer čovjek, životinja, vozilo, drvo, kuća i mnogi drugi. Detekcijski modeli rade na principu učenja neophodnih karakteristika, kako bi znali odrediti da li je nešto na slici traženi objekat ili nije. Određivanje objekata na slici je aktivnost koja je jednostavna za ljudski mozak, ali za sam računar predstavlja znatan izazov. Za detekciju objekata algoritmu se prvo moraju „dostaviti“ slike koje sadržavaju traženi objekat, odnosno da već imaju unaprijed određene lokacije (koordinate) objekta. [1]

Detekcijom je potrebno utvrditi da detektovani objekat najviše odgovara traženom objektu. Pošto na slici može biti više različitih objekata, algoritmi se trude da odrede veliki broj karakteristika koje su specifične za jedan traženi objekat, kao što su boja, oblik, veličina. Efikasnost detekcije se poboljšava kada se koristi veliki broj slika na kojima se traženi objekat nalazi na različitim pozicijama, te da se kvalitet slika znatno razlikuje (od slika visoke kvalitete do slika vrlo niske kvalitete). Također, mogu se uključiti i razni parametri koji mogu utjecati na rezultate detekcije kao što su: ugao slikanja, vrijeme i doba dana, blizina (daljina) objekta na slici, broj objekata i drugi. Napredni zadaci detekcije se mogu iskoristiti u aplikacijama za praćenje objekata, odnosno primjena detekcije na videu. Posebno su aktuelni primjeri detekcije pješaka, vozila, lica, teksta, poze.

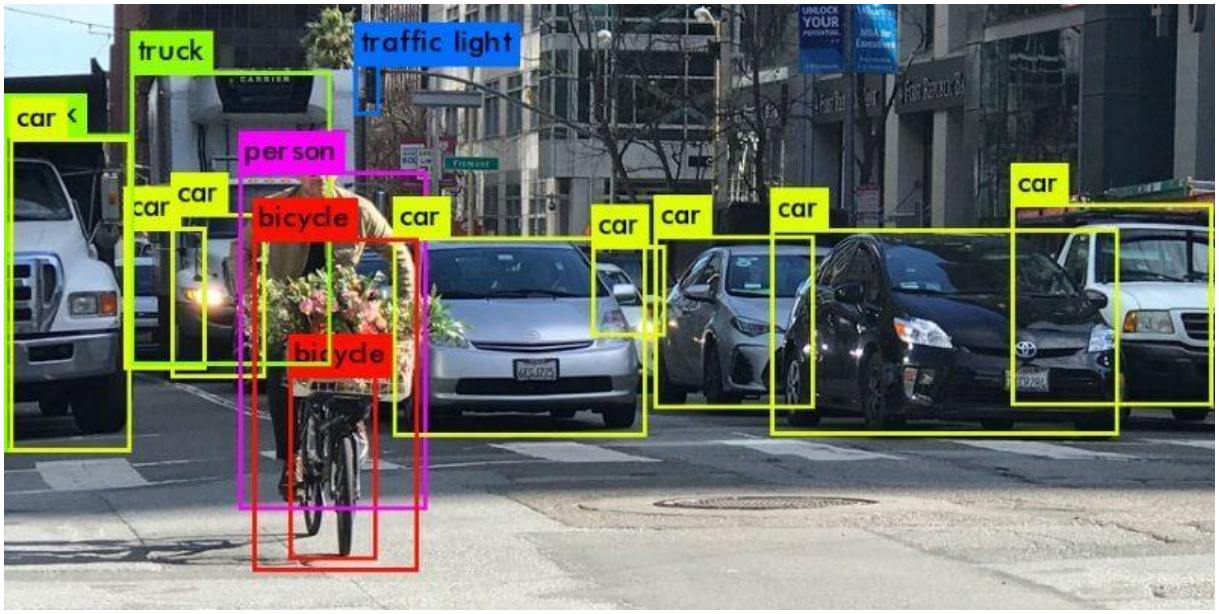


Slika 1 – Detekcija objekata [2]

3. KLASIFIKACIJA

Klasifikacija je proces označavanja objekata na slici. Ovim pristupom se očekuje da su objekti na slici prethodno detektovani, te ih je sada potrebno pridružiti određenoj klasi. Pošto klasifikacija dolazi nakon detekcije, to znači da se detekcijom samo izuzimaju objekti, a klasifikacijom se određuje koji je objekat u pitanju. Klasifikacija daje rezultat koji govori da li je objekat prisutan na slici ili nije, odnosno kada algoritam primi sliku on izračunava vjerovatnoću da li je to objekat A ili objekat B (za primjer s dvije klase). Najjednostavniji je upravo ovaj binarni klasifikator, dok su napredniji modeli multiklasni klasifikatori (više od dvije klase). Na primjeru binarne klasifikacije za dvije klase A i B, te učitanu sliku, klasifikator radi na način da prvo mora izdvojiti neke karakteristike (obilježja) koje međusobno razlikuju ove dvije klase. Izdvajanjem karakteristika se određuje kojoj klasi objekat više pripada, odnosno s kojom ima najviše sličnosti. [3]

Kada se kreće s procesom treniranja klasifikatora, neophodno je da se koristi označeni skup podataka, koji pored podataka (slika) sadrži i informacije o klasi kojoj pripada objekat na slici. Istrenirani model nakon toga će biti u stanju neku novu sliku biti u stanju dodijeliti unaprijed definisanoj klasi. Problemi kod klasifikacije mogu nastati kada se objekat koji pripada klasi A greškom dodijeli klasi B, što ukazuje na nedostatke istreniranog modela. Također, važno je napomenuti da se objekat može klasifikovati kao objekat A ili objekat B, ali ne i jedno i drugo istovremeno. Pored toga, najveći problem jeste dodijeljivanje kategorije A ili B objektu koji ne pripada ni jednoj od kategorija.



Slika 2 – Klasifikacija objekata [4]

4. PRAKTIČNI DIO

Kroz praktični dio implementirat će se algoritmi (modeli) za detekciju i klasifikaciju saobraćajnih znakova. Odabrani domen koji se odnosi na saobraćajne znakove, može se iskoristiti kod autonomnih ili samovozećih vozila. Proces implementacije može se primijeniti u raznim domenima, na vrlo sličan način, ali je prvo potrebno odrediti područje primjene i nabaviti odgovarajući skup podataka.

4.1. Alati, biblioteke i radno okruženje

Praktični dio je implementiran pomoću Python programskog jezika (verzija 3.10.0) [5]. Za zadatke koji obuhvataju analizu podataka i treniranje raznih modela mašinskog učenja i vještačke inteligencije Python je idealan kandidat. Napredne mogućnosti koje pruža Python uključuju vrlo jednostavno i efikasno rukovanje podacima i njihovu vizualizaciju. Razlog zbog kojeg je Python posebno postao popularan u ovakvim projektima jeste njegova fleksibilnost, široka podrška zajednice, platformska nezavisnot, te veliki ekosistem biblioteka koje pružaju mnoge gotove metode za rad s velikim skupovima podataka i njihovo procesiranje. Također, korištenje algoritama je vrlo jednostavno s unaprijed definisanim parametrima, kako bi se u samo nekoliko linija koda mogli postići očekivani rezultati. Biblioteke koje su korištene u projektu su:

- numpy
- pandas
- mathplotlib
- scikit-learn
- scikit-image
- os
- tensorflow/keras
- cv2
- ultralytics

Kod je pisan unutar Visual Studio Code [6] okruženja, uz korištenje posebnog formata (ekstenzije) Jupyter Notebook (.ipynb), koja je prilagođena pisanju Python skripti. Jupyter Notebook pruža podršku za kreiranje i dijeljenje dokumenata koji sadrže „živi“ (izvršeni) kod, zajedno s vizualnim prikazom (grafovi, slike), te formatirani opisni dio teksta. Implementacija se sastoji od analize skupa podataka, detekcije korištenjem YOLO algoritma, klasifikacije korištenjem CNN i SVM modela, te testiranje modela na novim slikama i videu, uz pregled

performansi. Što se tiče same strukture projekta, organizovan je u nekoliko foldera i fajlova. Projekat se sastoji od 6 Jupyter Notebook (.ipynb) fajlova s implementiranim algoritmima. Također, postoje odvojeni folderi za podatke za detekciju i klasifikaciju, kao i folderi za output (izlazne slike nakon detekcije i klasifikacije), te modeli, za spremanje istreniranih modela. O samim podacima za treniranje i testiranje će biti više govora u nastavku.

4.2. Skup podataka (dataset)

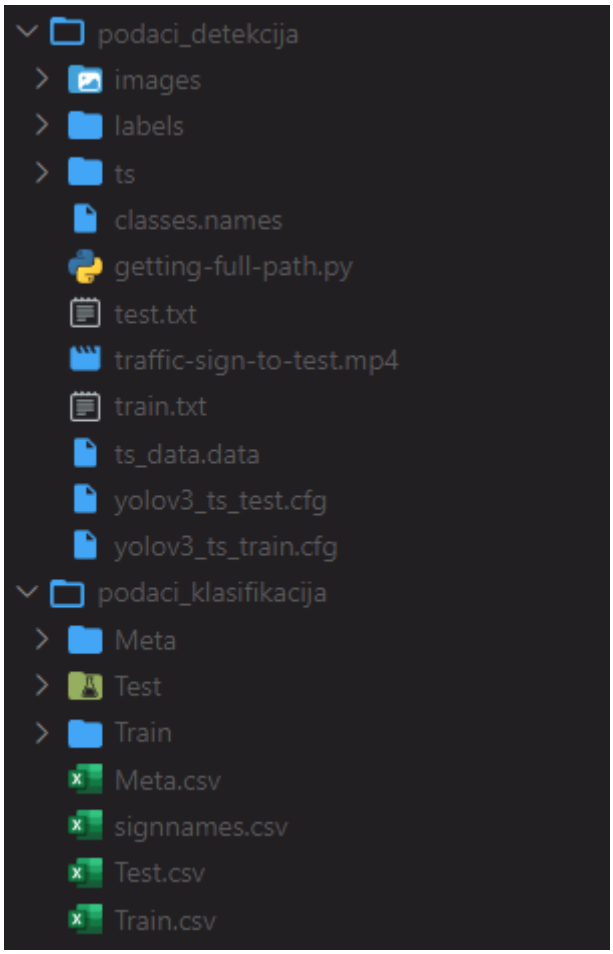
Za implementaciju praktičnog dijela korištena su dva nezavisna skupa podataka, jedan za detekciju, a drugi za klasifikaciju:

- GTSDB (German Traffic Sign Detection Benchmark) [7]
- GTSRB (German Traffic Sign Recognition Benchmark) [8]

Skupovi podataka su korišteni na takmičenju održanom na Međunarodnoj zajedničkoj konferenciji o neuronskim mrežama (IJCNN) 2011. godine, a kreirani su od strane samog instituta (https://benchmark.ini.rub.de/). Nakon održanog takmičenja, objavljeni su podaci o najboljim rezultatima, te su nakon toga skupovi postali dostupni svima na korištenje, radi vlastitog istraživanja. Skupovi su preuzeti s web stranice Kaggle (https://www.kaggle.com), koja omogućava korisnicima da pronađu različite skupove podataka, te da vrše svoja istraživanja u implementaciji i izgradnji modela vještačke inteligencije. Ideja koja stoji iza ove stranice je kolaboracija s drugim naučnicima, te dijeljenje vlastitih projekata i rezultata. Također, redovno se organizuju razna takmičenja i izazovi koji se zasnivaju na rješavanju probleme "nauke o podacima" (data science), a sama stranica nudi i mogućnost pokretanja kodova na njihovim serverima.

Prvi skup podataka GTSDB se koristi za detekciju saobraćajnih znakova i sastoji se od 741 slike, rezolucije 1360 x 800 piksela u .jpg formatu. Na slikama se nalaze različite scene na kojima su prikazani saobraćajni znakovi, zajedno s drugim objektima. Uz svaku sliku, pridružen je prateći .txt fajl koji sadrži informacije o x i y koordinatama znaka na slici (za treniranje), širini i visini, te broju znakova na slici. Skup podataka je prilagođen YOLO algoritmu (koji će biti objašnjen u nastavku). Također, u skupu se nalaze i gotove konfiguracije za YOLOv3 algoritam (spremljene u .cfg fajlovima), koje će poslužiti za kreiranje vlastitog YOLO modela. Pošto se ovaj algoritam može koristiti i za klasifikaciju, radi jednostavnosti sadrži 4 osnovne klase saobraćajnih znakova (generalizovano): zabrana (prohibitory), opasnost (danger), obavezni (mandatory), ostali (other). Ove klase služe samo radi izdvajanja saobraćajnih znakova od ostalih objekata, a za detaljniju klasifikaciju će se koristiti drugi skup podataka, odnosno GTSRB.

Drugi skup podataka GTSRB služi za klasifikaciju, te sadrži oko 50.000 slika podijeljenih u 43 klase. Klase su opisane unutar signnames.csv fajla koji sadrži pune nazive svake klase (na engleskom jeziku). Pored toga, slike su podijeljene u 2 foldera Train i Test (otprilike podijeljenih na oko 40.000 trening slika i 10.000 test slika). Train folder se sastoji od 43 podfoldera (folder za svaku od klasa). Test folder sadrži slike koje će se koristiti za procjenu preciznosti modela nakon treniranja. Također, u posebnom Meta folderu su izvorne slike za svaku od 43 klase (koje služe za detaljnije razumijevanje klasa). Za svaki folder, postoji i prateći .csv fajl Train.csv, Test.csv, Meta.csv, koji sadrži sljedeće kolone: width (širina), height (visina), ROI.x1, ROI.y1, ROI.x2, ROI.y2, ClassId (jedinstveni identifikator klase), Path (putanja do slke). ROI ili Region of Interest su x i y koordinate koje određuju znak na slici, odnosno ROI (područje interesovanja) predstavlja lokaciju na kojoj se nalazi znak u odnosu na cijelu sliku. Sve slike su u .png formatu.



Slika 3 – Struktura foldera

4.3. Priprema i analiza podataka

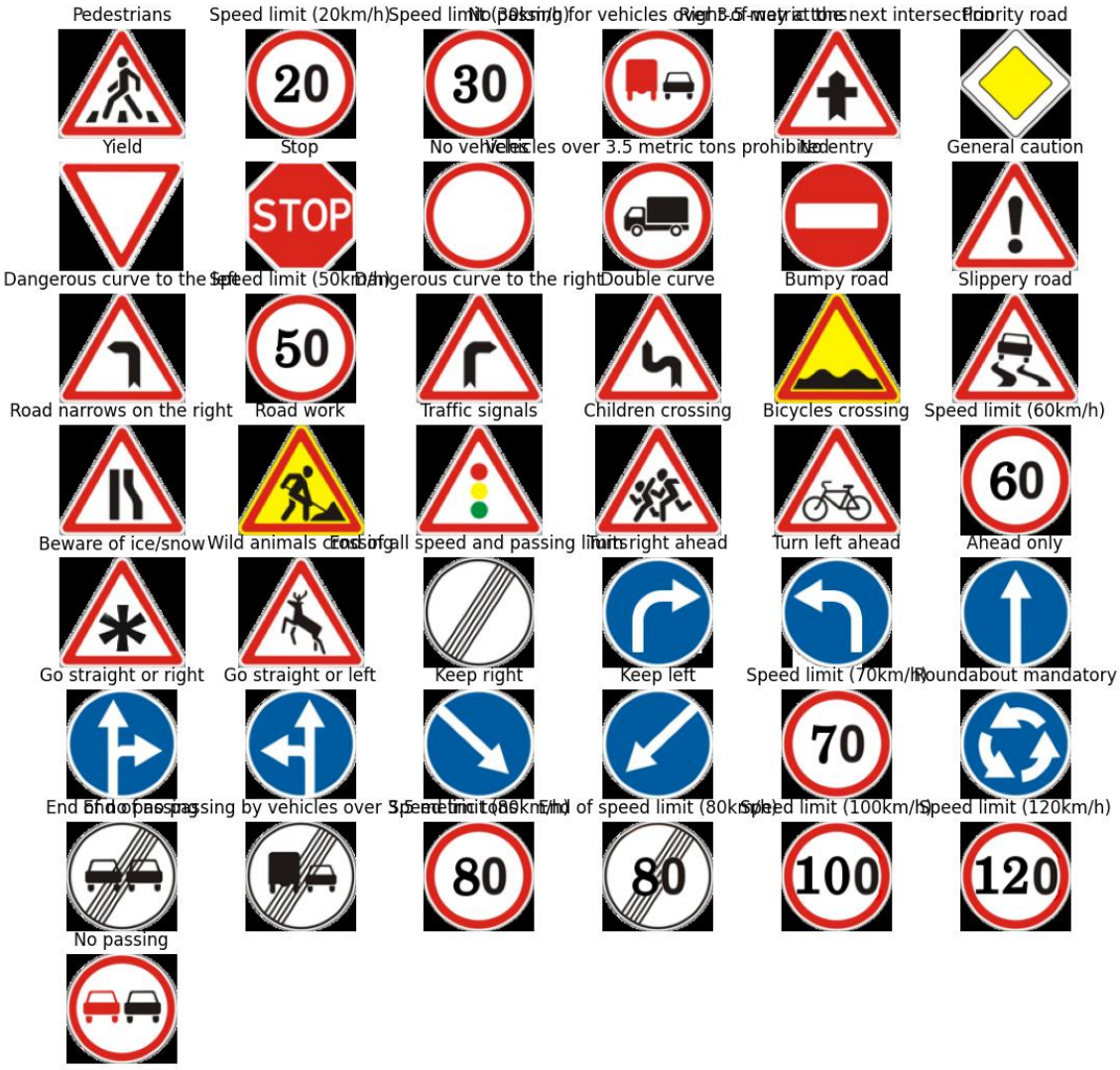
Nakon preuzimanja skupova podataka, neophodno ih je pripremiti za aktivnosti detekcije i klasifikacije. Kada je u pitanju skup za detekciju, on dolazi s 741 slikom, te ih je prvo potrebno podijeliti u tri odvojena foldera (Train, Test, Valid). Kada su folderi kreirani, potrebno je odrediti omjer kojim će podaci (slike) biti podijeljeni po folderima. Odabrani omjer je 70% trening, 20 % validacija i 10 % test. Da bi se postigli što bolji rezultati slike se prije podjele nasumično „miješaju“, korištenjem biblioteke random i metode shuffle. Rezultati koji se nakon toga dobijaju su: 518 trening slika, 148 validacijskih slika i 75 test slika spremljenih u zasebne foldere.

Slika 4 – Primjer slike iz skupa za detekciju

Istraživanje skupa za klasifikaciju sastoji se od određivanja ključnih karakteristika za svaku sliku kao što su minimalna i maksimalna visina slika, čime je utvrđeno da sve slike u prosjeku imaju širinu između 25 – 243, te visinu između 25 – 225. Može se vidjeti da su slike prilično malih dimenzija, jer za razliku od slika za detekciju sadrže samo znak na slici, bez bilo kakvih drugih objekata ili pozadine. Sljedeći korak je vizualizacija nekoliko slučajno odabranih slika radi utvrđivanja kvalitete slika. Ovim korakom je utvrđeno da većina slika ima vrlo nisku kvalitetu, pa je na nekim slikama teško čak i ljudskom oku procijenititi pravu klasu slike. Također, slike su uslikane u različitim dobima dana, imaju različito osvjetljenje i zamućenje. Na slici ispod se mogu vidjeti kako izgledaju 3 različite klase znakova iz trening, test i meta foldera (s lijeva na desno).

Slika 5 – Primjer trening, test i meta slike

Kada je u pitanju razumijevanje klasa znakova i njihova distribucija na trening i test skupu podataka, utvrđeno je da neke klase dominiraju nad ostalim po broju slika. Većina klasa ima oko 500 slika, dok nekoliko klasa ima i do 2000 slika. Kao što se može vidjeti neke od klasa su ograničenja brzine, dozvoljeni smjerovi, STOP i ostali.



Slika 6 – Klase saobraćajnih znakova

5. IMPLEMENTACIJA

Kada su podaci (slike) spremni za daljnju obradu može se krenuti s konkretnom implementacijom algoritama za detekciju i klasifikaciju. Nakon detaljne analize po pitanju kvalitete, veličine i formata slika ove slike se mogu koristiti za treniranje modela, a zatim i za evaluaciju njegovih performansi, odnosno tačnosti i preciznosti. Cilj je prvo izvršiti neophodno pretprocesiranje slike, te slike kao takve proslijediti modelu, koji će na kraju biti u stanju detektovati ili klasifikovati novu sliku koja mu se proslijedi.

5.1. YOLO algoritam

YOLO (You Look Only Once) je jedan od najpopularnijih algoritama za detekciju objekata, koji radi u realnom vremenu, te se koristi u mnogim komercijalnim proizvodima od strane najvećih tehnoloških kompanije, koje koriste kompjuterski vid. YOLO algoritam je prvi put originalno objavljen 2016. godine i već tada je ostvario znatno veće brzine od ostalih algoritama. Nakon toga objavljeno je nekoliko unaprijeđenih verzija i varijanti YOLO algoritma, gdje svaka nova verzija pruža bolje performanse i efikasnost. Verzija koja je korištena u ovom projektu je trenutno najnoviji YOLOv8 iz 2023. godine, koji sa sobom donosi inovacije kao što su poboljšanje mozaika podataka, samo-suparnička obuka i normalizacija mini serija. YOLOv8 razvio je Ultralytics (https://docs.ultralytics.com/), te je dizajniran da bude brz, precizan i jednostavan za korištenje. YOLO algoritam koristi karakteristike naučene od strane duboke konvolucionalne neuronske mreže (CNN). [9]

YOLO radi tako što model za detekciju pregleda cijelu sliku u vremenu testiranja, što znači da globalni kontekst na slici daje informacije o predviđanjima. Algoritam prvo dijeli sliku u mrežu (grid). Svaka ćelija mreže služi za predviđanje određenog broja graničnih okvira (frame-ova) oko objekata koji imaju visoke ocjene s unaprijed definisanim klasama. U svakom graničnom okviru identifikuje samo jedan objekat. Nakon toga se generišu granične „kutije“ (boxes) tako što se grupiraju dimenzije osnovnih okvira kako bi se pronašli najčešći oblici i veličine. Granične kutije mogu imati različite omjere širine i visine, koji se određuju prije treninga korištenjem nekog od algoritama grupisanja, kao naprimjer K-means algoritam. Naredni korak jeste poređenje koliko se odabrana kutija poklapa s osnovnom kutijom (odnosno unaprijed definisanom kutijom koja predstavlja izdvojeni objekat – ROI). Na kraju algoritam vraća oznake kao što su x i y koordinate, klasu i sigurnost predviđanja (confidence score). Ono što posebno karakteriše YOLO jeste što može napraviti duple detekcije za isti objekat, pa se iz rog razloga primjenjuje nemaksimalno potiskivanje, kako bi se uklonilo dupliciranje s nižim sigurnostima (povjerenjem), a zadržava se samo jedna s najvećim rezultatom. [10]

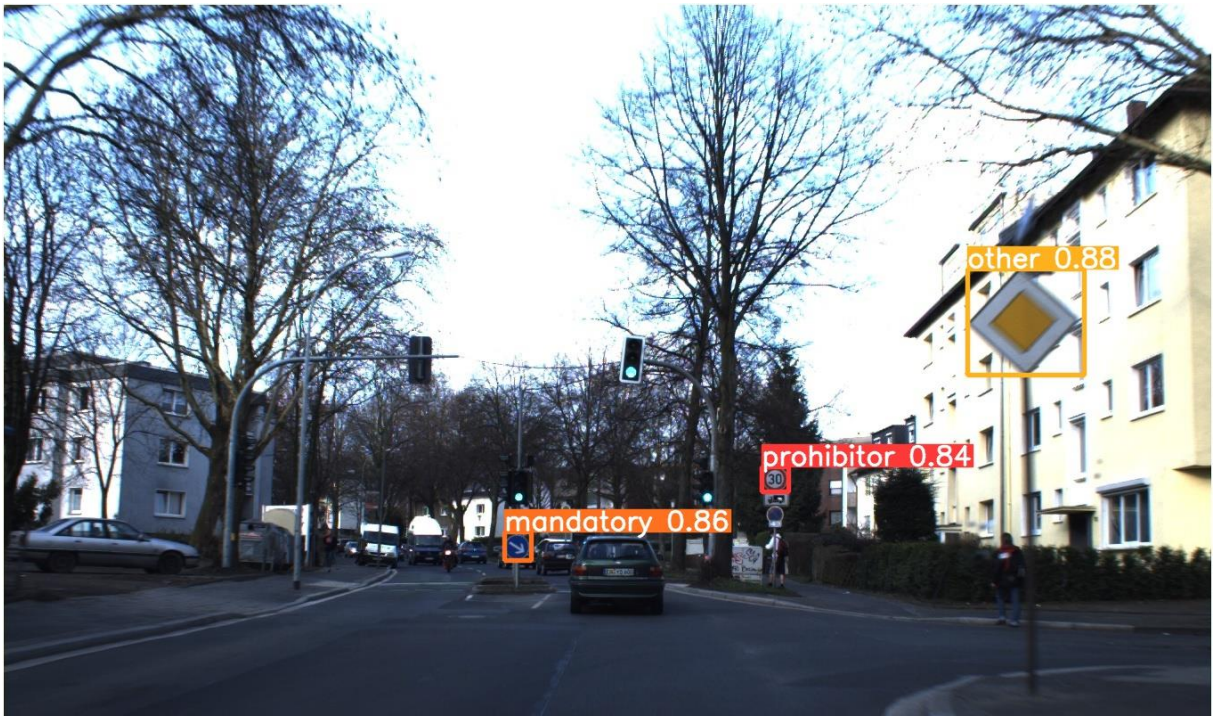
Za konkretni primjer prvo je instalirana biblioteka „ultralytics“ kako bi se mogle koristiti pogodnosti najnovijeg YOLOv8 algoritma. Pošto su slike za detekciju već podijeljenje u 3 foldera (Train, Valid, Test) slijedi korak kreiranja .yaml konfiguracijskog fajla, kako bi YOLO algoritam mogao znati kako će odrediti putanje do slike, te koje su specifične klase. Poslije toga potrebno je kreirati YOLO model, koji će automatski preuzeti istrenirane težinske vrijednosti samog modela (yolov8x.pt), koji je već osposobljen za detekciju nekih vrsta objekata (naprimjer ljudi i vozila). Sada je cilj koristiti već naučena znanja, kako bi se model prilagodio posebnoj vrsti podataka (slika), odnosno slika na kojima se nalaze saobraćajni znakovi. Proces treniranja modela je jednostavan pozivom metode train na samom modelu, uz prosljeđivanje parametara, kao što su broj epoha, preferiranu veličinu slike koja se prosljeđuje algoritmu, te prethodno generisani .yaml konfiguracijski fajl. Treniranje za 10 epoha je trajalo približno 4 i pol sata, što može znatno varirati u odnosu na specifikacije računara. Istrenirani model se može sačuvati u posebnom .onnx formatu, kako bi se mogao koristiti u nastavku za daljnja unaprijeđenja i testiranja.

```python
from ultralytics import YOLO
import yaml

data_yaml = dict(
    train = 'train',
    val = 'valid',
    test = 'test',
    nc = 4,
    names =['prohibitor','danger','mandatory','other']
)

with open('data.yaml', 'w') as outfile:
    yaml.dump(data_yaml, outfile, default_flow_style=True)

model = YOLO("yolov8x.pt")

results = model.train(data='./data.yaml', epochs=10, imgsz=480)

metrics_val = model.val()

metrics_test = model.val(split="test")

success = model.export(format="onnx")

results = model.predict("./podaci_detekcija/images/test/", save=True, conf=0.5)
```

Slika 7 – Izvorni kod kreiranja YOLO modela

5.2. Analiza rezultata detekcije

Pri završetku procesa treniranja, automatski se kreira poseban folder runs u kojem se čuvaju podaci o detekciji i treningu, koji su grafički predstavljeni i spremljeni u formatu slike (.png i .jpg). Neki od najvažnijih grafičkih prikaza uključuju: konfuzijsku matricu, F1 confidence krivulju, oznake, korelogram oznaka, P krivu (Precision-Confidence), PR krivu (Precision-Recall), R krivu (Recall-Confidence). Istrenirani model se onda može iskoristiti za kreiranje predikcija, tako što se metodi predict proslijedi putanja do slike (ili u ovom slučaju cijeli folder testnih slika, koje nisu korištenje prilikom treniranja). Neki od dodatnih parametara koji se mogu uključiti su da li će se slika s detektovanim saobraćajnim znakovima spremiti na disk ili ne, te koji će biti prag sigurnosti predviđanja (decimalni broj između 0-1). Proces detekcije na testnim slikama u ovom slučaju sprema nove slike na disk, a također i daje informacije o broju detektovanih znakova po slici. Još jedna od prednosti korištenja YOLO algoritma je to što se na tim slikama oko objekata (znakova) automatski iscrtava pravougaonik koji označava ROI, zajedno s nazivom klase i povjerenju predikcije. Kada je u pitanju evaluacija modela, vrlo jednostavno se određuje pozivom metode val na samom modelu, koji daje preciznost za trening i validacijske slike, a ako mu se proslijedi parametar „test“, dobijaju se rezultati preciznosti za testne slike. Ova metrika je izražena kao mAP (mean average precision). [11] U konkretnom primjeru dobijaju se vrijednosti 0.66 za validacijski skup i 0.70 sa testni skup (obe zaokružene na dvije decimale).



Slika 8 – Rezultat YOLO detekcije

5.3. CNN model

CNN (Convolutional Neural Network) ili konvolucijska neuralna mreža je mreža dubokog učenja koja se najčešće koristi za klasifikaciju slika. Radi na principu izdvajanja karakteristika iz matričnog skupa podataka u obliku mreže, tako što se s ulazne slike otkrivaju linije, gradijente, oblici i granice. Nakon toga, konvolucijski slojevi pomažu u obradi izlaza. CNN sadrži mnogo konvolucijskih slojeva sastavljenih jedan na drugom, pri čemu je svaki od njih kompetentan za prepoznavanje sofisticiranijih oblika. Arhitektura CNN sastavljena je od sljedećih slojeva input (ulazni), konvolucijski (convolutional), sloj za spajanje (pooling) i potpuno povezani sloj (fully connected). Ulazni sloj je sloj u kojem se daje ulaz (slika) modelu. Konvolucijski sloj primjenjuje filtere na ulaznu sliku kako bi izdvojio karakteristike, pa sloj za spajanje smanjuje uzorkovanje slike kako bi smanjio izračunavanje, te potpuno povezani sloj daje konačno predviđanje. Mreža uči optimalne filtere putem propagacije unazad i gradijenta. Između konvolucijskog i sloja za spajanje može se dodati i aktivacijski sloj, koji dodaje aktivacijsku funkciju na izlaz konvolucijskog sloja, što dodaje nelinearnost mreži. Također, između sloja za spajanje i potpuno povezanog sloja, može se dodati sloj za izjednačavanje, koji služi da se mape karakteristika koje su izlaz iz sloja za spajanje pretvore u jednodimenzionalni vektor, koji se zatim prenose u potpuno povezani sloj radi klasifikacije. [12]

Za kreiranje modela u konkretnom primjeru prvo je potrebno učitati sve trening slike u Python listu radi lakšeg pristupa. Pošto je već od ranije poznato, nakon procesa analize, gdje se su i kako smještene slike, te koliko ima klasa, taj dio neće biti opisan u ovom potpoglavlju. Za učitavanje svih slika koristi se cv2 (OpenCV verzija 2) biblioteka. Slikama se nakon učitavanja dodatno mijenjaju dimenzije u 30 piksela širine i 30 piksela visine. Pored učitanih slika, učitavaju se i oznake (labele) klase kojoj svaka slika pripada. Učitane slike se zatim pretvaraju u niz korištenjem numpy biblioteke. Naredni korak je podjela podataka na trening i test, korištenjem ugrađene train_test_split metode u omjeru 80% trening i 20% test. Još jedan dodatni korak je podjela dobijenih trening podataka na trening i validacijske u istom omjeru. Konačni rezultat podjele je 60% trening, 20% validacija i 20% test. Prije kreiranja samog modela, slike je potrebno prethodno procesirati kako bi se prilagodile, te da bi model mogao ostvariti bolje rezultate. Najjednostavniji vid procesiranja je pretvaranje slike u grayscale (sivi) format boje i izjednačavanje histograma. Kada je slika u sivom formatu boje, smanjuje se vrijeme izračunavanja, dok se izjednačavanjem histograma povećava kontrast slike, tako što se inteziteti piksela izjednačavaju normalizacijom s njihovim najbližim pikselima. Pikseli se normalizuju tako što se dobijene vrijednosti dijele s 255, kako bi se dobile nove vrijednosti između 0 i 1. Još jedan dodatni korak je transformacija oznaka klasa u kategoričke vrijednosti.

```python
model = Sequential()

model.add(Conv2D(filters=16, kernel_size=(3,3), activation="relu", input_shape=(30,30,1)))
model.add(Conv2D(filters=32, kernel_size=(3,3), activation="relu"))
model.add(MaxPool2D(pool_size=(2,2)))
model.add(Dropout(rate=0.25))
model.add(Conv2D(filters=64, kernel_size=(3,3), activation="relu"))
model.add(Conv2D(filters=128, kernel_size=(3,3), activation="relu"))
model.add(MaxPool2D(pool_size=(2,2)))
model.add(Dropout(rate=0.25))
model.add(Flatten())
model.add(Dense(512, activation="relu"))
model.add(Dropout(rate=0.5))
model.add(Dense(43, activation="softmax"))

model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])

model.summary()

history = model.fit(aug.flow(X_train,y_train,batch_size=64),
                    epochs=20,
                    validation_data=(X_validation,y_validation),
                    shuffle=1)

model.save('./modeli/model_1.h5')
model.save('./modeli/model_1.keras')
```

Slika 9 – Izvorni kod kreiranja CNN modela

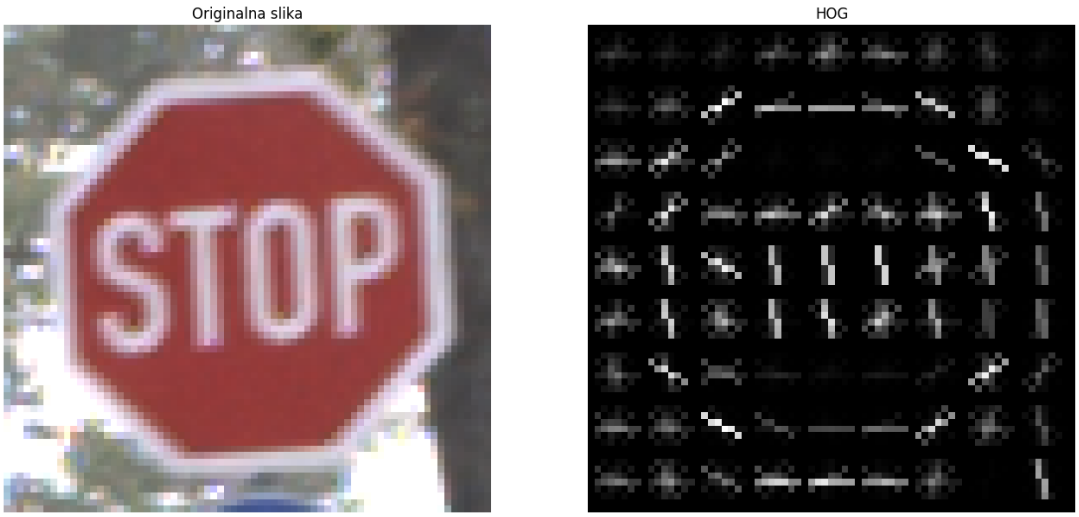
Kao što je prikazano na slici iznad, kreira se model koji se sastoji se od sljedećih slojeva:

- ulazni sloj – definira ulazne dimenzije slike (30, 30, 1 – širina, visina, broj kanala)
- konvolucijski slojevi (Conv2D) – koriste relu aktivacijsku funkciju i izdvajaju različite karakteristike iz slike, filteri 16 i 32 određuje koliko se filtera primjenjuje na ulaznu sliku, kernel_size (3,3) određuje veličinu matrice filtera
- slojevi spajanja (MaxPool2D) – smanjuju dimenzionalnost izlaza konvolucijskih slojeva, pool_size (2,2) označava da se koristi sažimanje maksimalnim vrijednostima veličine 2x2
- slojevi odbacivanja (Dropout) – služe za regularizaciju mreže i zaštitu od prenaučenosti modela (overfitting), gdje se odbaciju slučajno odabrane jedinice, rate 0.25 i 0.5 označava proporciju odbacivanja
- potpuno povezani slojevi (Flatten, Dense) – dense s 512 neurona služi za učenje složenih karakteristika iz izlaza prethodnih slojeva, koji koriste relu i softmax aktivacijsku funkciju, koja daje vrijednost pripadnosti jednoj od ukupno 43 klase

Tako kreirani model se zatim kompajluje, koristeći Adam optimizaciju, categorical_crossentropy kao funkciju gubitka, te kao metriku koristi preciznost. Zatim se model trenira s prethodno podijeljenim trening podacima (ulazne slike i oznake klasa). Proces treniranja je podijeljen u 20 epoha, te se nakon toga dobijene izračunate težinske vrijednosti modela mogu spremiti na disk, kako bi se mogle naknadno koristiti. Analiza rezultata će biti prikazana u jednom od narednih potpoglavlja.

5.4. SVM algoritam

SVM (Support Vector Machine) je jedan od algoritama nadziranog učenja koji se koristi za klasifikaciju i regresiju, ali većinom za probleme klasifikacije. SVM model predstavlja različite klase u hiperravni (hyperplane). Hiperravan je ravan ili prostor, koji dijeli podatke između skupova objekata različitim klasama. Cilj SVM-a je odajanje skupa podataka u klase, tako što se uzima maksimalna margina između hiperravni. Nakon pronalaska maksimalne margine SVM generiše hiperravan iterativno, a svakom iteracijom se klase dijele na bolji način. Na kraju se odabire hiperravan koja ispravno dijeli klase. Za poboljšanje SVM modela koriste se neki kerneli koji mogu uključivati linearno, polinomno ili jezgro radijalne baze. [13] Prije kreiranja samog modela potrebno je izdvojiti karakteristike iz slike. Za treniranje efikasnog klasfikatora izdvajaju se samo korisne informacije iz slike procesom poznatim pod nazivom deskriptor karakteristika, a metoda koja se koristi za ovo naziva se HOG (Histogram of Oriented Gradients). HOG je deskriptor karakteristika koji se koristi za karakterizaciju objekata na osnovu njihovih oblika. Ovom metodom se izračunava histogram (pojavljivanje) svake orijentacije gradijenta u određenom dijelu slike. Orijentacija gradijenta predstavlja promjenu smjera osobina kao što su boje i intenzitet boje u slici. [14]



Slika 10 – Određivanje HOG-a na slici

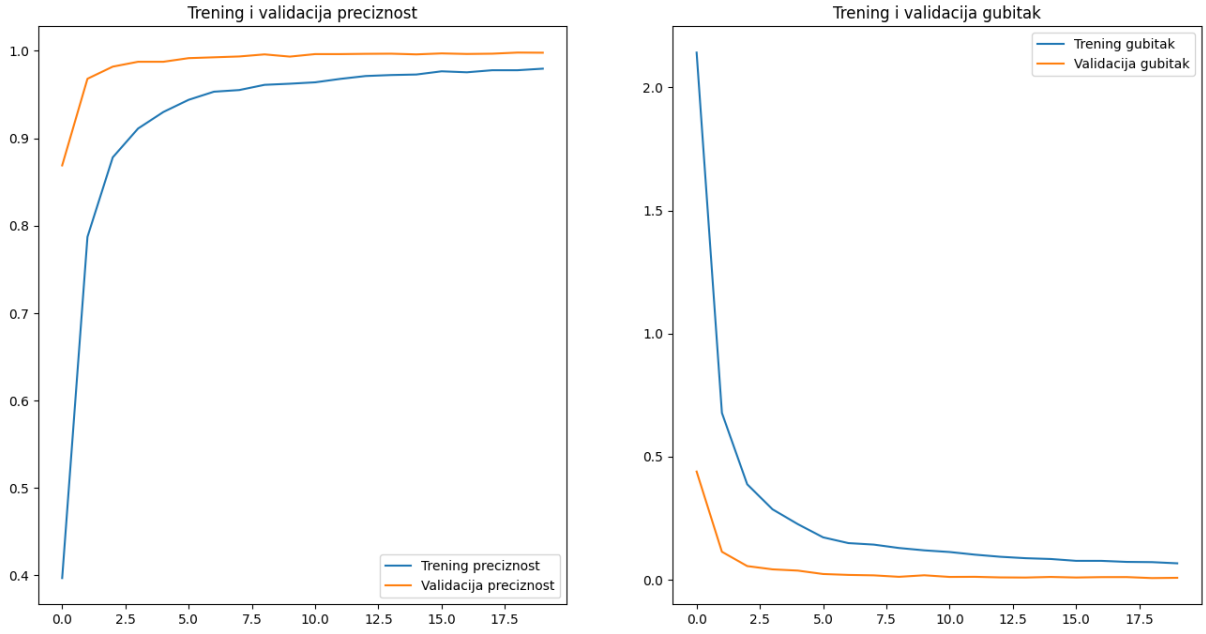
Za implementaciju u konkretnom slučaju prvo se učitavaju trening slike i oznake klasa, kao što je već opisano u prethodnom potpoglavlju. Kada se slika učita, vrši se njeno procesiranje i to pretvaranje u grayscale (sivi) format boja, nakon čega slijedi primjena metode tresholda i blur (zamućivanje/zamagljivanje), te se na kraju pronalaze konture (oblici) oko objekta. Zatim slijedi prilagođavanje širine i visine, te normalizacija piksela (dijeljenjem s 255). Glavni korak je izdvajanje karakteristika iz svake slike pojedinačno korištenjem hog metode. Zatim se slike pretvaraju u niz pomoću numpy biblioteke. Sljedeći korak je podjela podataka na trening i test, u omjeru 80% trening i 20% test. SVM model se kreira pozivom metode, kojoj se kao parametri mogu proslijediti vrsta kernela, u ovom slučaju „linearni“ i gamma parametar postavljen na „scale“ što znači da će SVM automatski određivati gamma vrijednosti, tako da budu obrnuto proporcionalne varijansi ulaznih vrijednosti. Zatim se model trenira i koristi za predikciju klasa. Analiza rezultata će biti prikazana u narednom potpoglavlju.

```python
def preprocess_image(image):
    gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    gray_image = np.uint8(gray_image)
    _, binary_image = cv2.threshold(gray_image, 128, 255, cv2.THRESH_BINARY)
    blurred_image = cv2.GaussianBlur(binary_image, (5, 5), 0)
    contours, _ = cv2.findContours(blurred_image, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
    cv2.drawContours(image, contours, -1, (0, 255, 0), 2)
    return image

images = []
labels = []

for index, row in train_df.iterrows():
    img_path = root_path + row['Path']
    class_id = row['ClassId']
    img = cv2.imread(img_path)
    preprocessed_img = preprocess_image(img)
    img_resized = cv2.resize(preprocessed_img, (32, 32))
    img_rescaled = img_resized / 255.0
    images.append(img_rescaled)
    labels.append(class_id)

def extract_hog_features(images):
    hog_features = []
    for image in images:
        features = hog(image, orientations=9, pixels_per_cell=(8, 8),
                       cells_per_block=(2, 2), transform_sqrt=True, block_norm='L2-Hys', channel_axis=2)
        hog_features.append(features)
    return hog_features

images_features = extract_hog_features(images)

X_train, X_test, y_train, y_test = train_test_split(images_features, labels, test_size=0.2, random_state=42)

svm_model = SVC(kernel='linear', gamma='scale')

svm_model.fit(X_train, y_train)

y_pred = svm_model.predict(X_test)
```

Slika 11 – Izvorni kod kreiranja SVM modela

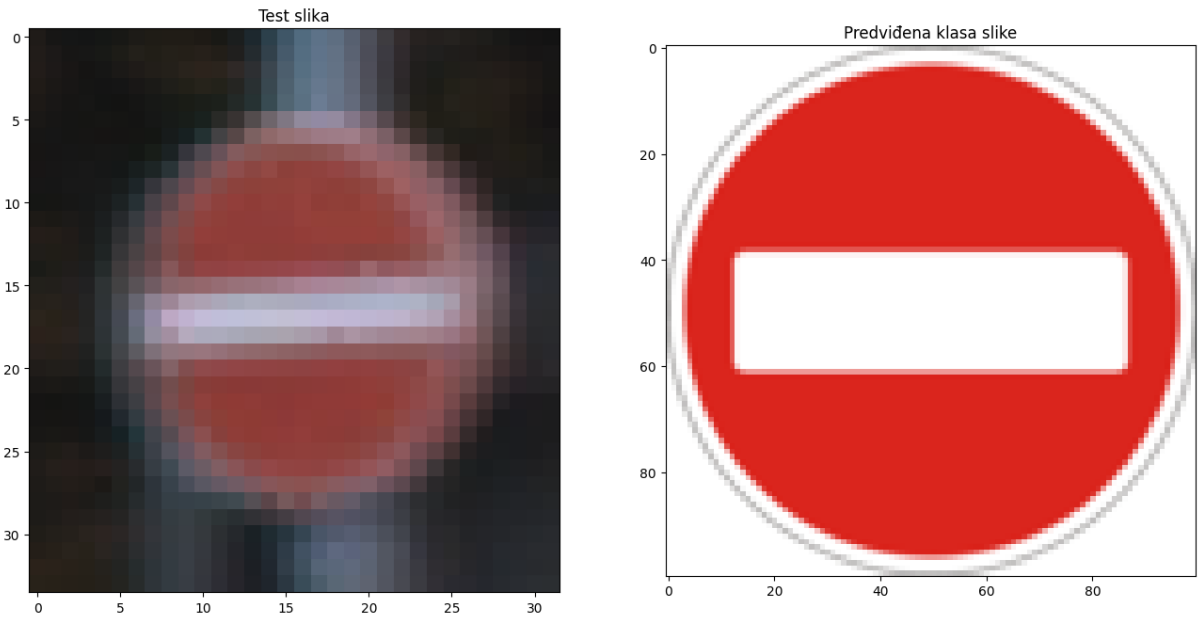
5.5. Poređenje rezultata testiranja klasifikacije

Za dva prethodno kreirana modela za klasifikaciju, CNN i SVM može se odrediti evaluacija modela. Mogu se koristiti različite metode i tehnike, od numeričkog ili grafičkog prikaza dobijenih rezultata. Jedan od načina analize performansi modela je kreiranje konfuzijske matrice (confusion matrix). Konfuzijska matrica je matrica koja prikazuje stvarne i predviđene klase za klasifikacijski model. Druga metoda je generisanje izvještaja o klasifikaciji (classification report) koji daje detaljniji pregled performansi modela, tako što sadrži informacije o preciznosti pojedinačne klase. Također, jedan od načina grafičkog prikaza jeste pregled kako se odvijao proces treniranja, prateći parametre kao što su preciznost i gubitak.



Slika 12 – Prikaz preciznosti i gubitka CNN modela

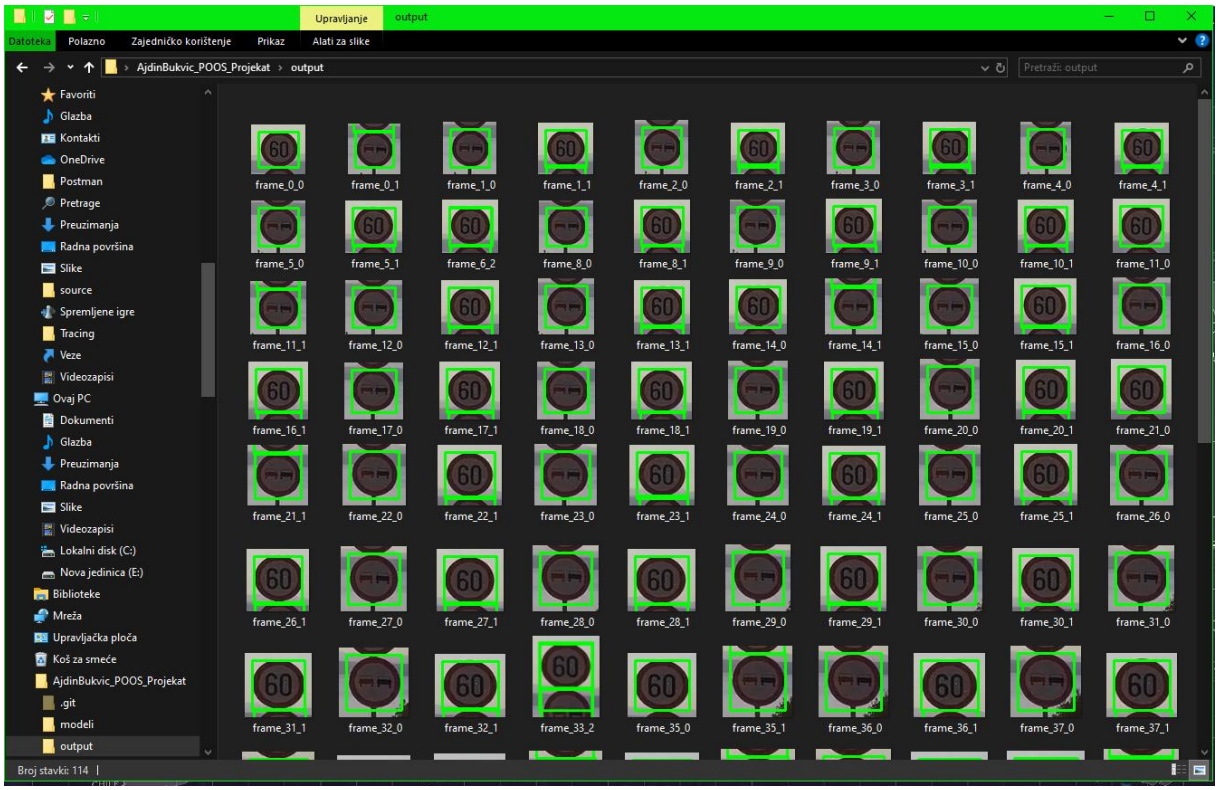
Kada su u pitanju preciznost i gubitak treninga CNN modela, dobija se vrlo visoka vrijednost za preciznost od 99%, te gubitak od 0.007. S druge strane preciznost treniranja SVM modela je 79%. Sada je potrebno utvrditi preciznost modela na testnom skupu podataka, odnosno novim slikama, koje model nije do sada vidio (prilikom treniranja). Testne slike se učitavaju na isti način kao i već prethodno opisani način učitavanja trening slika, ali bez potrebe za direktnim učitavanjem oznaka klasa, već se dobijene klase porede naknadno poslije klasifikacije sa stvarnim klasama, jer su i testne slike iz skupa podataka također označene klasama. Predikcijom klasa za testne slike, dobija se rezultat od 96% preciznosti za CNN model, te 66% za SVM model. Na isti način se po potrebi mogu formirati konfuzijska matrica i izvještaj o klasifikaciji. Pregledom ovih rezultata može se zaključiti da CNN model daje bolje rezultate.



Slika 13 – Rezultat CNN klasifikacije

5.6. Detekcija i klasifikacija na videu

Završni dio implementacije je testiranje dva istrenirana (i spremljena) modela za detekciju i klasifikaciju, ali na videu. Iako detekcijski YOLO model ujedno određuje i generalizovane klase znakova, ovdje je korišten pristup po kojem će se detekcijski model primijeniti da samo detektuje znakove na videu, a da ih zatim CNN klasifikacijski model klasifikuje u jednu od 43 dostupne klase. Testni video koji se koristio je video koji je dostupan unutar foldera za detekciju u .mp4 formatu. Video traje nekoliko sekundi i visokog je kvaliteta, a na njemu se nalaze 2 različita znaka. Snimak je nastao iz vozila prilikom vožnje. Za početak se prvo učitavaju detekcijski i klasifikacijski modeli, a zatim se testni video učitava korištenjem cv2 biblioteke, a istovremeno se pokreće i snimanje, kako bi se rezultat detekcije mogao spremiti u posebnom videu. Kada se video učita, otvara se prozor za prikaz videa, na kojem se za svaki frame (okvir) automatski (u stvarnom vremenu) detektuju znakovi, te se oko njih prikazuje pravougaonik, generalizovana klasa i vrijednost sigurnosti klasifikacije. Detekcijski model za svaki od znakova određuje njihove x i y koordinate (ROI). Pomoću ovih podataka lokacija se ručno crta pravougaonik oko znakova, te se frame (okvir) kao takav sprema na disk kao .png slika u poseban output folder. Ovaj postupak se iterativno ponavlja sve dok se video ne završi. Na videu se jasno mogu utvrditi 2 znaka i to Speed limit 60 km/h (ograničenje brzine 60 km/h) i No passing (nema prolaza). Učitavanjem generisanih slika iz output foldera i primjenom klasifikacijskog modela se kreiraju predikcije. Pokretanjem videa se u ovom slučaju generiše ukupno 114 slika, a poređenjem predviđenih i stvarnih klasa za ova 2 znaka preciznost klasifikacije iznosi 74%, što je prilično manje od performansi CNN modela na običnim slikama.



Slika 14 – Folder s detektovanim znakovima na videu

6. ZAKLJUČAK

Korištenje detekcije i klasifikacije u okviru kompjuterskog vida su trenutno jedni od aktuelnih tehnika koje se svakodnevno istražuju i unaprijeđuju. U širokom sprektru primjena, ali i povećanju trendova primjene u budućnosti kao što su sigurnost i nadzor, medicinska dijagnostika, samovozeća vozila i automatizacija industrije ovo područje je doživjelo veliku ekspanziju. Ciljevi su kreiranje što naprednijih algoritama koji će biti u stanju obrađivati sliku i video u stvarnom vremenu, te davati što preciznije rezultate uz minimalne greške. U odabranom primjeru u kojem su implementirane detekcija upotrebom YOLO algoritma, te klasifikacija pomoću CNN i SVM modela na skupu podataka saobraćajnih znakova analizirani su rezultati koji se dobiju kako treningom, tako i posebnim testnim slikama, koje nisu bile dostupne prilikom treninga. Modeli su istrenirani da daju što bolje rezultate na slikama koje se međusobno razlikuju po veličini, kvaliteti, te različitim brojem objekata i scenama na slici. Za poboljšanje rezultata korištene su razne tehnike pretprocesiranja slike, prije nego što su poslane u model za predikciju. CNN model je pokazao veću preciznost u odnosu na SVM na ovom primjeru s ovim skupom podataka i prosljeđenim parametrima.

7. LITERATURA

[1] https://viso.ai/deep-learning/object-detection/

[2] https://ambolt.io/wp-content/uploads/classification.jpg

[3] https://labelyourdata.com/articles/object-detection-vs-image-classification

[4] https://azati.ai/wp-content/uploads/2020/04/object-detection-800x400-1.jpg

[5] https://www.python.org/downloads/release/python-3100/

[6] https://code.visualstudio.com/

[7] https://www.kaggle.com/datasets/valentynsichkar/traffic-signs-dataset-in-yolo-format

[8] https://www.kaggle.com/datasets/meowmeowmeowmeowmeow/gtsrb-german-traffic-sign/data

[9] https://keylabs.ai/blog/under-the-hood-yolov8-architecture-explained/

[10] https://www.labellerr.com/blog/understanding-yolov8-architecture-applications-features/

[11] https://medium.com/axinc-ai/map-evaluation-metric-of-object-detection-model-dd20e2dc2472

[12] https://www.geeksforgeeks.org/introduction-convolution-neural-network/

[13] https://www.analyticsvidhya.com/blog/2021/06/build-an-image-classifier-with-svm/

[14] https://towardsdatascience.com/hog-histogram-of-oriented-gradients-67ecd887675f