SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Raspoznavanje uzoraka i strojno učenje

Detekcija i praćenje objekta

Dario Mihelčić

Diplomski studij računarstva, DRB

1. UVOD

Potreba za brzim i preciznim radom dovela je do razvoja računalnih programa čija se sposobnost razlučivanja informacija iz stvarnog svijeta sve više približava ljudskoj. Strojno učenje omogućilo je robotima i autonomnim vozilima bolje snalaženje u prostoru, u tolikoj mjeri da je intervencija čovjeka gotovo nepotrebna. Povećanjem broja proizvoda koje koriste postupke strojnog učenja, sve je veća potreba za brzim i jednostavnim načinima stvaranja modela sposobnih razlikovati različite predmete iz okoline. Danas su dostupni brojni alati izrađeni ciljano na olakšavanje stvaranja ovakvih modela.

Cilj ovog projektnog rada je na primjeru nekoliko premeta iz svakodnevnog života testirati alate za izradu modela dubokog učenja, odnosno modela za detekciju predmeta na slikama koji će se koristiti u svrhu praćenja objekta na video zapisu, te dobiveni model usporediti sa drugim rješenjima. Za izgradnju modela korišteno je preneseno učenje u Python programskom jeziku. Ulazni skup podatka je prikupljen iz *online* dostupnih baza podataka, te je nadopunjen novim skupovima podataka. Model za detekciju objekata izrađen je pomoću Tensorflow platforme korištenjem Google Colaboratory usluge. Kao baza detektora koristi se već treniran SSD MobileNet model, koji će se postupcima prenesenog učenja trenirati na novim klasama. Gotov model je testiran na video zapisima na kojima su označeni zadani objekti, a rezultati su uspoređeni sa modelom za praćenje objekata dostupnog u sklopu OpenCV programskog paketa.

U prvo dijelu rada opisani su postupci i alat potrebni za stvaranje modela za detekciju objekata. Zatim je opisan postupak prikupljanja i obrade ulaznih podataka, postavljanja radnog okruženja te treniranja modela. Na kraju su dani rezultati testiranja izrađenog modela te usporedbe sa modelima praćenja objekata.

2. PREGLED KORIŠTENIH POSTUPAKA I ALATA

Grana umjetne inteligencije zvana prepoznavanje objekata (eng. *object recognition*) je generalni izraz koji obuhvaća tehnologije za pronalaženje i identificiranje objekata na slikama ili na video zapisima. Klasifikacija objekata bavi se prepoznavanjem klase jednog objekta na cijeloj slici. Za jednu slika danu na ulazu, model za klasifikaciju objekata daje jednu klasu na izlazu. Lokalizacija objekta odnosi se na proces pronalaženja pozicije jednog ili više objekata na slici i pridruživanja graničnog okvira (eng. *bounding box, i.e. BB)* koji okružuje zadani objekt. Spoj ove dvije tehnike pronalazi se u detekciji objekata, metoda koja na zadanoj slici pronalazi objekte od interesa, lokalizira njihove položaje postavljanjem graničnog okvira te na kraju svakom lokaliziranom objektu pridružuje pripadajuću klasu.

2.1. DETEKCIJA OBJEKATA

U praksi, obično se koriste dva glavna tipa algoritama za detekciju objekata. Algoritmi poput R-CNN i Fast R-CNN koriste pristup u dva koraka. Prvo se pretpostavljaju polja za koja se očekuje da sadrže tražene objekte te se u svakom polju obavlja detekcija objekata. Suprotno tome, algoritmi poput YOLO (*You Only Look Once*) [1] i SSD (*Single-Shot Detector*) [2] koriste potpuno konvolucijski pristup kod kojeg model može pronaći sve objekte na slici u jednom prolazu kroz mrežu. Algoritmi sa pretpostavljenim poljima pretrage obično imaju bolju preciznost, ali su znatno sporiji od algoritmima s jednim prolazom.

SSD model sastoji se od dvije komponente: bazni model i SSD slojevi. Bazni model je prijašnje treniran klasifikator slika koji se koristi za izvlačenje značajki. Uzeti model je obično ResNet, VGG-16 ili MobileNet mreža kojim su zadnji slojevi zaslužni za klasifikaciju odvojeni. Na skupu dobivenih značajki pretpostavljaju se položaji nekoliko različitih graničnih okvira, a na svim danim okvirima će se obavljati klasifikacij. Na kraju su dodani konvolucijski slojevi čija se veličina smanjuje prema kraju mreže. Dodani konvolucijski slojevi odrađuju klasifikaciju na različitim graničnim okvirima, čime se postiže detekcija objekata u jednom prolazu.

2.2. TENSORFLOW

Tensorflow je platforma otvorenog koda za strojno učenje. TensorFlow programski paket je razvio tim istraživača i inženjera koji je radio na *Google Brain* projektu zajedno sa Googleovom *Machine Intelligence Research* organizacijom u svrhu provođenja istraživanja vezanog za strojno učenje i duboke neuronske mreže. Tensorflow ima razumljiv, fleksibilni ekosustav alata, biblioteka i javnih resursa što omogućuje konstantne napretke u strojnom učenju i daje programerima mogućnost jednostavne proizvodnje aplikacija koje koriste strojno učenje.

TensorFlow Object Detection API [3] je platforma građena na TensorFlow-u koja omogućuje jednostavno stvaranje, treniranje i razvoj modela za detekciju objekata. Ova platforma sadrži brojne alate koje ubrzavaju proces stvaranje modela. Platforma sadrži i kolekciju gotovih modela za detekciju treniranim na poznatim bazama podataka (COCO, Open Images, KITTI itd.).

Model name	Speed (ms)	COCO mAP[^1]	Outputs
ssd_mobilenet_v1_coco	30	21	Boxes
ssd_mobilenet_v1_0.75_depth_coco ☆	26	18	Boxes
ssd_mobilenet_v1_quantized_coco ☆	29	18	Boxes
faster_rcnn_inception_v2_coco	58	28	Boxes
faster_rcnn_resnet50_coco	89	30	Boxes
faster_rcnn_nas	1833	43	Boxes
faster_rcnn_nas_lowproposals_coco	540		Boxes
mask_rcnn_inception_resnet_v2_atrous_coco	771	36	Masks

Slika 2.1. Primjeri dostupnih modela (Izvor: [3])

TensorBoard je alat za vizualizaciju podataka u TensorFlow-u. TensorBard omogućuje jednostavno praćenje vrijednosti vrednovanja kao što su gubitak i preciznost, daje mogućnost vizualizacije grafa modela, pruža vizualizaciju rješenja tijekom treniranje i brojne druge prednosti. TenorBoard je koristan alat kojim se može na jednostavan način uočiti moguće pogreške u procesu testiranja. TensorBard pruža mogućnosti koje su se prije postojanja ovog programa odrađivale pisanjem dugačkih skripti za izvlačenje i prikaz informacija.

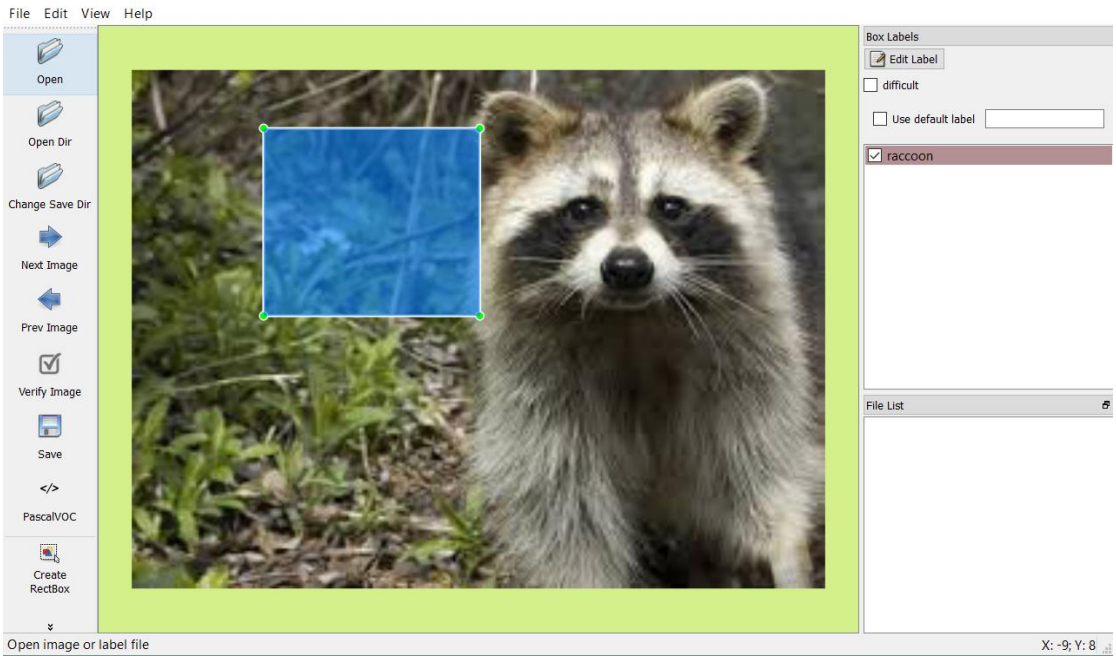
2.3. GOOGLE COLAB

Google Colab je usluga koja pruža korisniku okruženje za razvoj programa i aplikacija u oblaku. Pokreće se u web pregledniku korištenjem Goolge Cloud-a. Google Colab daje potporu za razvoj korištenjem Python 2.7 i 3.6 programskih jezika, svako radno okruženje dolazi s već instaliranim alatima za rad s obradom podataka i računalnog vida (NumPy, Pandas, Matplotlib, Sckit-Learn i drugi). Velika prednost Google Colab-a je mogućnost povezivanje s Google Drive repozitorija i besplatno korištenje Google Colab-ovih GPU-a. Ovo omogučuje potpuno i jednostavno treniranje modela umjetne inteligencije u oblaku. Od 2018. godine, Google Colab pruža besplatno korištenje 12GB NVDIA Tesla K80, a od 2019. godine GPU je unaprijeđen na Tesla T4 model.

Usluga se koristi putem Jupyter programskih skripti, gdje se dijelovi programa mogu obavljati zasebno. Trajanje jedne sjednice traje 12 sati, nakon čega će se obrisati sav napredak (osim napisanih skripti). Povezivanjem sa Google Drive-om omogućuje spremanje podataka za korištenje u sljedećoj sjednici.

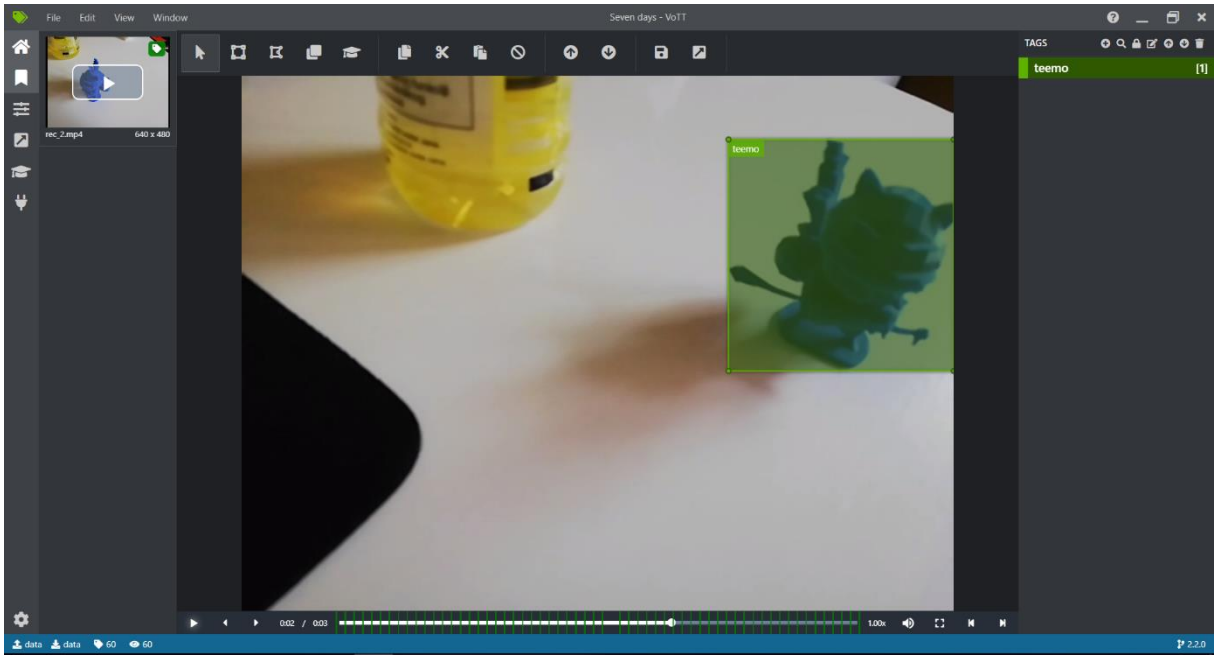
2.4. ALATI ZA ANOTIRANJE

LabelImg je vrlo jednostavan alat za anotiranje slika. Napisan je u Python programskom jeziku, koristi Qt za gradnju grafičkog sučelja. Anotacije se spremaju kao XML datoteke u PASCAL VOC formatu, a pruža potporu i za YOLO format.



Slika 2.2. Izgled LabelImg sučelja

Microsoft-ov VoTT (Visual Object Tagging Tool) je alat za anotiranje slika i video zapisa. VoTT spojem jednostavnog i čistog sučelja i dobro definiranih naredbi daje efikasno okruženje za anotiranje slika i video zapisa. Ima jasno definiran model za dobavljanje i slanje podataka sa lokalnih repozitorija ili repozitorija u oblaku. Program omogućuje spremanje podataka u brojnim oblicima pogodnih za specifične modele umjetne inteligencije.



Slika 2.3. Izgled VoTT sučelja

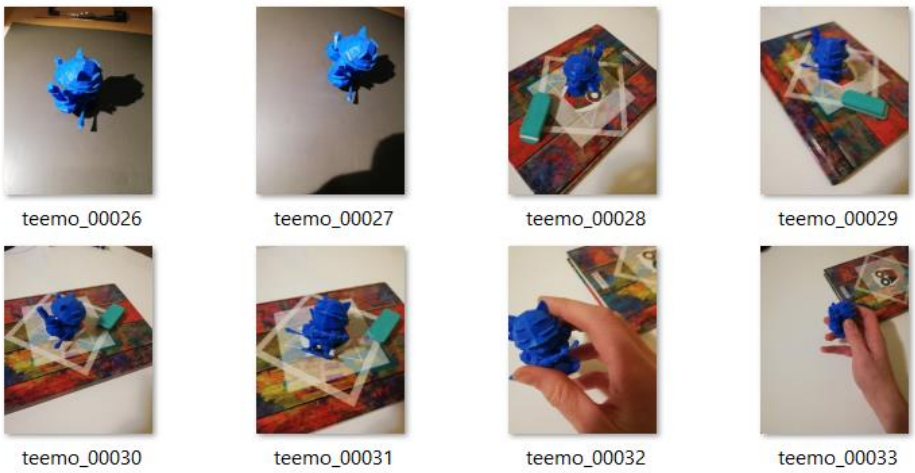
3. IZRADA MODELA PREPOZNAVANJA

Za izradu modela prepoznavanja izabrane su sedam različitih objekata: rakun, klokan, banana, jabuka, naranča, figurica i adapter. Prvih pet klasa preuzeto je iz *online* repozitorija. Skup podataka za klasa 'figurica' (u nastavku 'teemo') i 'adapter' ručno su izrađene korištenjem alata za označavanje slika. Za obradu podataka i treniranje modela detektora objekta postupkom prenesenog učenja napisana je Jupyter skripta koja se pokreće u oblaku, odnosno preko Google Colab-a.

3.1. SKUP PODATAKA

Za izradu modela prepoznavanja izabrane su sedam različitih objekata: rakun [4], klokan [5], banana, jabuka, naranča [6], figurica i adapter. Prvih pet klasa preuzeto je iz *online* repozitorija. Skup podataka za klasa 'figurica' (u nastavku 'teemo') i 'adapter' ručno su izrađene prikupljanjem slika kamerom mobitela te korištenjem LabelImg alata za označavanje slika. Skup podataka za jednu klasu sastoji se od niza slika i pripadajućih XML datoteka u kojima su definirane dimenzije graničnih okvira. Klase objekata se sastoje od različitih brojeva slika: klokan – 176, rakun – 200, jabuka – 95, banana – 94, naranča – 95 (slike miješanog voća 24), adapter – 90, teemo – 90. Ulazni skup podataka je podijeljen na trening skup i skup za testiranje tako da su se XML datoteke podijelile u odgovarajuće datoteke, a tijekom treniranja modela će se učitavati slike prema imenima slika zapisanih u XML datotekama. Nakon podjele, skup podataka je prebačen na Google Drive repozitoriji kako bi se koristio u daljnjoj obradi u oblaku.

Slika 3.1. Primjer slika klase 'klokan'

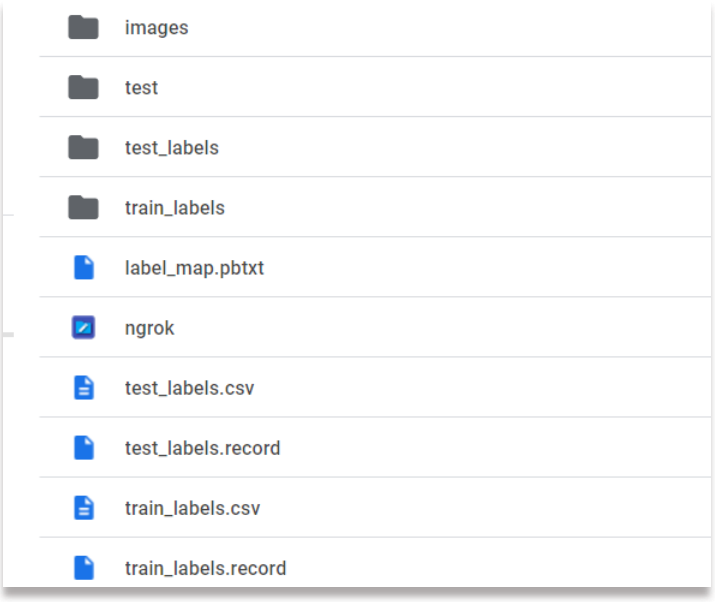


Slika 3.2. Primjer slika klase 'teemo'

3.2. TRENIRANJE MODELA

Treniranje modela obavljeno je pomoću Tensorflow programskog paketa, točnije TensorFlow Object Detection API-a u Google Colabu. Kako bi se koristila skripta za treniranje modela detekcije objekata potrebno je pripremiti ulazne podatke u odgovarajućem obliku.

Prvo, potrebno je pretvoriti ulazne podatke u odgovarajući oblik. Korištena skripta za treniranje prima podatke za treniranje i testiranje u .tfrecord obliku. Uz ulazni skup podataka, potrebno je definirati broj i nazive klasa u obliku .pbtxt datoteke. Za pretvorbu je korištena skripta u Jupyter bilježnici.



Slika 3.3. Struktura Google Drive repozitorija

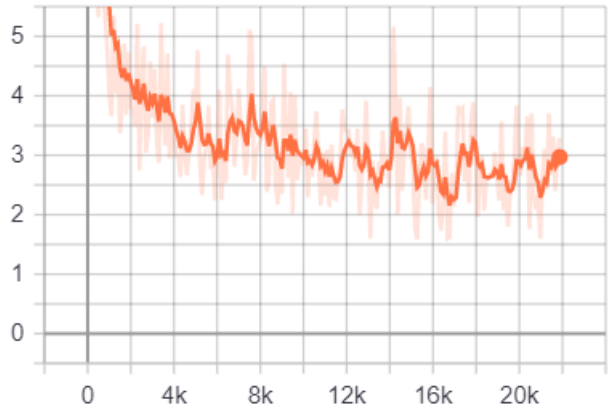
Drugo, potrebno je preuzeti željeni model a treniranje, dostupan na TensorFlow API repozitoriju, te izmijeniti odgovarajuću .config datoteka. Unutar ove datoteke je definirana struktura neuronske mreže, broj klasa, konfiguracija treniranja (augmentacija ulaznih podataka, veličina trening skupa, stopa učenja, funkcija pogreške), putanje do testnih i trening podataka, .pbtxt datoteke. Za ovaj projekt je odabran 'ssd_mobilenet_v2_coco' model. Ovaj model je jedan od najbržih modela za detekciju objekata dostupan na TensorFlow repozitoriju.

Datoteka 'ssd_mobilenet_v2_coco.config' izmijenjena je tako da su postavljene sedam klasa objekata, početna stopa učenja postavljena je 0.003, dodane su augmentacije ulaznog skupa podataka (nasumično podrezivanje slike, zrcaljenje slike, rotacije slike). Kako bi se spriječilo pretjerano učenje zbog malog broja slika ulaznog skupa, povećan je broj negativa koji se uzimaju sa slike. Negativi su zapravo nasumično postavljeni opsežni okviri za koje nije definirana klasa. Dodavanjem negativa u proces treniranja smanjuje se mogućnost pretjerane klasifikacije objekata. U .config datoteku su još dodane putanje do ulaznih podataka.

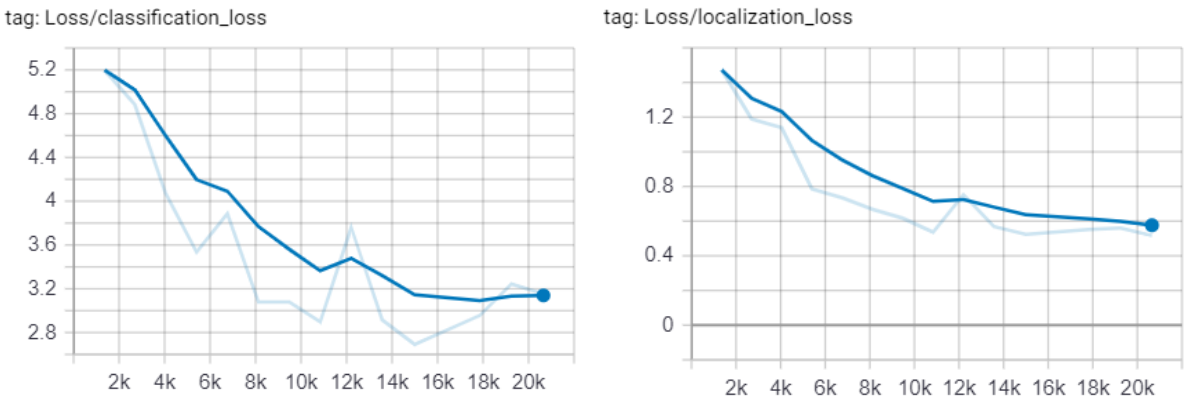
Za proces treniranja definirana je *checkpoint* datoteka u kojoj se periodično spremaju rezultati treniranja, kako bi se treniranje moglo nastaviti od zadnjeg spremljenog stanja u slučaju prekida rada programa. Treniranje se pokreće pozivom funkcije 'model_main.py'. Treniranje modela ovog projekta odvijalo se 21870 koraka. Kako je korišten GPU u Goolge Colab-u, 100 koraka treniranja potrebno je u prosjeku 42 sekunde. Nakon uspješnog treniranja, model koji se može koristiti za detektiranje objekata može se dobiti pozivom funkcije 'export_inference_graph.py'.

4. REZULTATI TESTIRANJE MODELA

Gubitak treniranja i validacije praćen je pomoću TensorBard-a. Na slici 4.1. prikazan je promjena vrijednosti funkcije gubitka ovisno o broju koraka treniranja. Nagli pad vrijednosti na početku te spori relativan pad nakon 4000. koraka ukazuje na dobru stopu učenja. Nakon 20000. koraka vrijednost gubitka ne pokazuje daljnji pad pa je treniranje zaustavljeno. Na slikama 4.2. i 4.3. prikazana je promjena vrijednosti funkcije gubitka klasifikacije i lokalizacije kod validacije. Gubitak klasifikacije i lokalizaciju pokazuju kontinuiran pad.



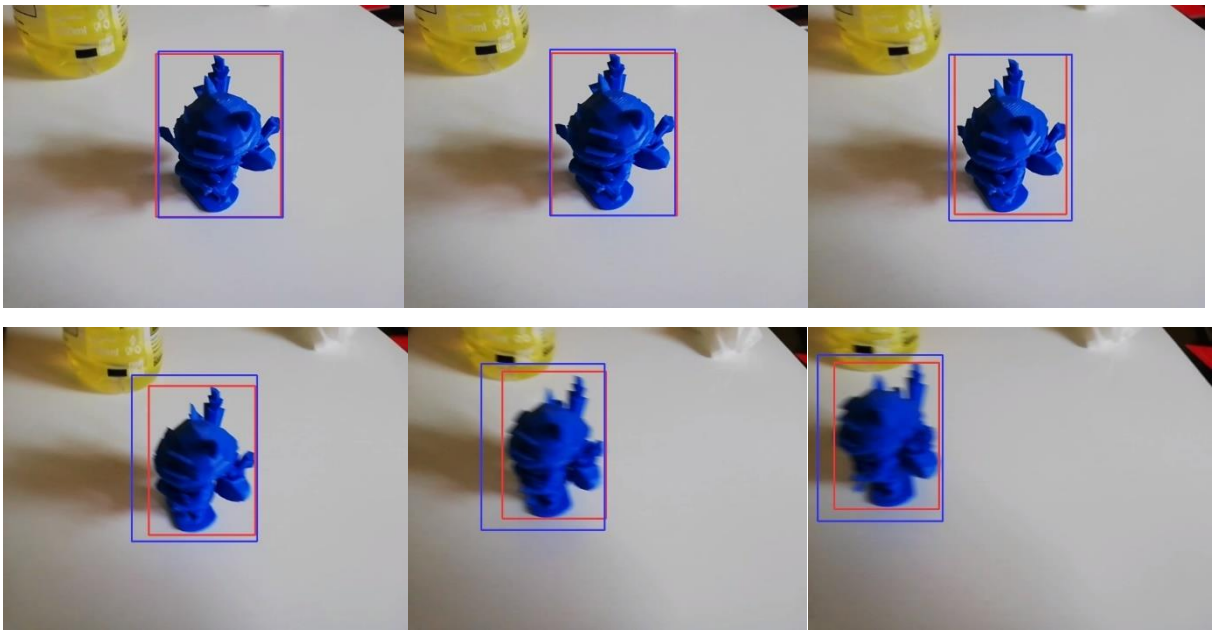
Slika 4.1. Graf gubitka treniranja



Slika 4.2. Graf gubitka klasifikacije (lijevo) i lokalizacije (desno) validacije

Treniran model detekcije objekata uspoređen je sa modelima praćenja objekata dostupnih u sklopu OpenCV programskog paketa. Korišteni modeli za praćenje objekata su: KCF (Kernelized Correlation Filters), CSRT (Discriminative Correlation Filter with Channel and Spatial Reliability) i MedianFlow. Model za praćenje objekata omogućuje praćenje bilo kojeg objekta koji je označen na prvom okviru video zapisa. Provedeno je dodatno testiranje gdje se trenirani model za detekciju objekata pokreće za svaki deseti okvir video zapisa, a na temelju dobivenih graničnih okvira dovija se praćenje modela za ostale okvire.

Model detektora objekata pokrenut za svaki okvir video zapisa imao je u prosjeku vrijednost od 7 FPS-a, a kada se detektor pokreće za svaki deseti okvir video zapisa, zajedno sa modelom praćenja objekata koji se odvija za ostale okvire, vrijednost FPS-a je 11. Model za praćenje objekata KCF se obavlja sa 15-55 FPS-a, CSRT 6-10 FPS-a, a MedianFlow 64-74 FPS-a. FPS vrijednost ovisi o veličini objekta na slici koji se mora pratiti, veći objekti uzrokuju duže vrijeme obrade podataka što smanjuje vrijednost FPS-a Ovi modeli su testirana na 7 video zapisa (prosjeku trajanja od 5 sekundi). Pokretanjem skripte 'run_tracker.py', 'run_detector.py' i 'detection_tracker_fusion.py' obavlja se praćenje objekata (samo praćenje, samo detekcija i spoj tim poretkom) na danom video zapisu, te se rezultati zapisuju u XML datotekama. Video zapisi su anotirani pomoću VoTT programa za 15 FPS, odnosno svaku drugu sliku video zapisa. Nakon što su svi podatci zapisani, testira se rad modela tako da se računa IoU (Intersection over Union) vrijednost za svaku anotiranu slika video zapisa. Rezultati su zbrojeni i podijeljeni sa brojem anotiranih slika. Rezultati testiranja modela praćenja dani su u tablici 4.1.



Slika 4.4. Primjer rezultata praćenja objekta (crveno – anotacija, plavo – metoda)

broj	opis	KCF	CSRT	MedianFlow
rec_1	teemo, izlaz iz okvira	46.73%	57.79%	**65.36%**
rec_2	teemo	71.63%	**79.25%**	76.34%
rec_3	banana	75.81%	78.83%	**85.48%**
rec_4	adapter	75.88%	**87.66%**	71.83%
rec_5	Naranča	**80.64%**	80.05%	80.28%
rec_6	teemo, test udaljenosti	61.30%	76.64%	**81.45%**
rec_7	teemo, rotacija, brzina	22.37%	**61.57%**	26.33%

Tablica 4.1. Rezultati modela praćenja (prosjek IoU)

KCF model daje najgore rezultate. CSRT i MedianFlow modeli imaju sličnu IoU vrijednost, no MedianFlow model radi znatno brže. Iz tog razloga je za za testiranje spoja modela praćenja objekata i detekcije objekata izabran MedianFlow. Spoj modela radi tako da se praćenje objekta ispravlja u svakom desetom okviru postavljanjem graničnog okvira dobivenog iz detektora.U tablici 4.2. dani su rezultati testiranja samog modela detekcije objekata, samog MedianFlow modela te njihov spoj.

broj	opis	detektor	MedianFlow	spoj modela
rec_1	teemo, izlaz iz okvira	65.89%	65.36%	**70.39%**
rec_2	teemo	**84.58%**	76.34%	82.52%
rec_3	banana	83.74%	85.48%	**86.02%**
rec_4	adapter	57.05%	71.83%	**78.41%**
rec_5	Naranča	82.70%	80.28%	**85.45%**
rec_6	teemo, test udaljenosti	80.87%	81.45%	**82.62%**
rec_7	teemo, rotacija, brzina	**67.79%**	26.33%	69.34%

Tablica 4.2. Rezultati usporedbe modela (prosjek IoU)

Spoj modela detekcije i praćenja objekta daje najbolje rezultate u većini slučajeva. Detektor objekata daje precizne granične okvire za velik broj okvira, no u nekim slučajevima detektor ne daje granične okvire te se točnost praćenja pogoršava. Model praćenja objekata radi kontinuirano, no točnost graničnih okvira se pogoršava s vremenom (zbog promjene izgleda objekta, zbrajanje grešaka postavljanja okvira, zbog velikih brzina gibanja objekta). Spajanjem modela kontinuiranog praćenja objekta i detekcije objekta za svaki deseti okvir ublažuju se nedostaci pojedinačnog modela te je konačno praćenje objekta znatno točnije.

5. ZAKLJUČAK

U ovom radu dan je pregled nekih od postupaka računalnog vida, način stvaranja modela koji primjenjuje te postupke te programskih paketa i alata koji pomažu u tom procesu. Izrađen je skup podataka od sedam različitih klasa, gdje su pet klasa preuzetih iz otvorenih repozitorija, a dvije klase su dodane. Skup podataka je korišten za treniranje detektora objekta primjenom postupka prenesenog učenja. Obrada podataka i treniranje je obavljeno u Google Colab-u, platformi za razvoj aplikacija i modela strojnog učenja koja pruža besplatne usluge za obradu podataka u oblaku. Trenirani model za detekciju objekata testiran je usporedbom sa postojećim modelom za praćenje objekata iz OpenCV biblioteke. Provedena je usporedba na sedam anotiranih video zapisa.

Treniranje modela se ubrzalo korištenjem usluga za izvođenje programa u oblaku. Kako bi se kompenziralo za mal broj slika u ulazno sklopu podataka, koristila se augmentacija trening podataka. Povećanjem broja negativa koji se koriste tijekom treniranja smanjuje se mogućnost pretjeranog treniranja na malom skupu podataka. Trenirani model za detekciju objekata radi precizne detekcije objekata na video zapisima, daje dobre rezultate IoU sa usporedbom sa anotiranim podatcima. No, model je spor u odnosu na već postojeće modele za praćenje objekata, te nije pogoda za korištenje u stvarnom vremenu. Spojem detektora objekata sa modelom za praćenje objekata umanjuju se nedostatci rada samih. Usporedbom rada samih modela detekcije i praćenja objekta sa spojem modela vidljivo je da su rješenja točnija za slučaj spojenih modela.

6. LITERATURA

[1] J. Redmon, S. Divvala, R. Girshick i A. Farhadi, »You Only Look Once: Unified, Real-Time Object Detection,« u *arXiv:1506.02640*, 2015.

[2] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed i C.-Y. Fu, »SSD: Single Shot MultiBox Detector,« u *arXiv:1512.02325*, 2016.

[3] [Mrežno]. Available: https://github.com/tensorflow/models/tree/master/research/object_detection. [Pokušaj pristupa Lipanj 2020].

[4] D. Tran, »Raccoon Detector Dataset,« [Mrežno]. Available: https://github.com/datitran/raccoon_dataset. [Pokušaj pristupa Lipanj 2020.].

[5] H. N. Anh, »kangaroo,« Lipanj 2020. [Mrežno]. Available: https://github.com/experiencor/kangaroo.

[6] M. Buyukkinaci, »Fruit Images for Object Detection,« Lipanj 2020. [Mrežno]. Available: https://www.kaggle.com/mbkinaci/fruit-images-for-object-detection.