Sveučilište u Zagrebu

Fakultet elektrotehnike i računarstva

Jura Hostić, Tvrtko Puškarić, Karlo Vrdoljak

Prava Vrećica – Reciklaža i razvrstavanje otpada primjenom računalnog vida

Zagreb, 2023.

Ovaj rad izrađen je na Zavodu za elektroniku, mikroelektroniku, računalne i inteligentne sustave Fakulteta elektrotehnike i računarstva Sveučilišta u Zagrebu pod vodstvom izv. prof. dr. sc. Dejana Škvorca i predan je na natječaj za dodjelu Rektorove nagrade u akademskoj godini 2022./2023.

Sadržaj

1. Uvod

Usprkos sve većoj ekološkoj osviještenosti u općoj populaciji i vrlo pozitivnom stavu prema odvajanju otpada i reciklaži, samo je 60 posto ispitanika ankete provedene u Sjedinjenim Američkim Državama 2022. godine odgovorilo kako recikliranje ne smatraju zahtjevnim [1]. Iako se taj broj može činiti visokim, nikako nije dostatan i sukladno tome postoji prostor za poboljšanje. Neki od razloga zbog kojeg broj ispitanika smatra recikliranje zahtjevnim su nedostatna informiranost o pravilnom načinu odlaganja i postojećoj infrastrukturi, nestandardiziranost pravila na različitim lokacijama, nedostatak vremena i nepraktičnost, što je već dulje vrijeme poznato jer se slični razlozi mogu pronaći i u istraživanju provedenom 1995. godine na Sveučilištu u Claremontu, u Kaliforniji [2]. Zaključci ankete nalažu kako bi proaktivnost u informiranju opće populacije imala pozitivan učinak na percepciju zahtjevnosti reciklaže kao i na broj ljudi koji su sigurni u svoje odabire pri reciklaži otpada. Jedan od koraka koji se može poduzeti u tu svrhu je razvoj novih alata za pomoć pri reciklaži i razvrstavanju otpada.

2. Motivacija i ciljevi rada

Mobilna aplikacija *Prava Vrećica* naš je pokušaj da se pozabavimo problemima navedenima u uvodnom poglavlju, prvenstveno u našoj okolini, no moguće i šire, na jedan nov i inovativan način. Motivaciju za početak rada činili su slični problemi koje smo mogli zamijetiti u svojoj okolini, gradu Zagrebu. Okosnicu projekta čini razvoj algoritma pomoću modela umjetne inteligencije kojem je cilj pravilno raspoznati vrste otpada na određenoj fotografiji te ih dodatno kategorizirati sukladno lokalnim pravilima odlaganja. Modeliranje i implementacija modela u svrhu računalnog vida te proučavanje dobivenih rezultata su glavne teme ovog rada. Aplikacija je rješenje kojim se pokušava dobiti opširan uvid u mogućnosti i realnu primjenu ove tehnologije, a zamišljena je kao novi alat za pomoć pri reciklaži koji se temelji na razvijenom algoritmu.

Slika 2.1 Rad aplikacije na dvije lokacije s različitim pravilima odlaganja otpada

Glavni problemi koje aplikacija pokušava riješiti su nesigurnost i neinformiranost korisnika pri reciklaži otpada. Ukratko, rad aplikacije temelji se na primjeni računalnog vida na fotografije koje korisnik unese kako bi se prikazale informacije o mjestu i načinu odlaganja na korisničkoj lokaciji, dodatne informacije o fotografiranim predmetima, kao što je na primjer mogućnost povratka ambalaže za novac, popratni naputci, kao na primjer savjet

da se kutije ili boce isprazne prije odlaganja, ili preporuka da se odjevni predmeti poklone u dobrotvorne svrhe.

Korisnik također ima mogućnost dati povratne informacije o prepoznatim predmetima čime se skupljanju podaci za poboljšanje modela umjetne inteligencije, što je jedan od bitnih značajki koje se moraju sagledati pri implementaciji takvih tehnologija.

Aplikacija sadrži i neke popratne mogućnosti kojima se pokušava razmotriti upotrebljivost danih tehnologija. Dugotrajnom uporabom aplikacije sprema se korisnička statistika koja se može proučiti na jednostavan način kroz sučelje aplikacije. Statistika služi korisniku kao orijentir u vlastitim naporima recikliranja i informativni alat iz kojeg može dobiti uvid u svoje navike ili mogućnosti uštede pri reciklaži. Podaci iz kojih se gradi statistika nusprodukt su uporabe modela umjetne inteligencije.

3. Plan i izvedba rada

Ovo poglavlje pruža uvid u strukturu i plan izvedbe projekta kroz listu primijenjenih tehnologija. U fokusu je način izvedbe i upotrebe umjetne inteligencije zbog čega je razrađen detaljan opis svih značajki te tehnologije, što podrazumijeva: računalni vid u svrhu detekcije objekata, moguće modele i implementacije, strukturu i algoritme iza odabranog modela te sve popratne korake koje je bitno razmotriti kod takvih tehnologija. Uz modeliranje i prikupljanje podataka te način osposobljavanja vlastitih modela uz pomoć pogodne strukture podataka opisani su i svi upotrebljeni pomoćni alati. Također je dan plan izvedbe aplikacije i baze podataka kao okvira u kojima se promatra rad izabranog modela, što uključuje alate poput razvojnih okvira, programskih jezika i radne okoline.

3.1. Model umjetne inteligencije

Sljedeća potpoglavlja predstavljaju strukturirani pregled koraka provedenih pri upotrebi umjetne inteligencije. Opisano je područje računalnog vida, način rada tehnologije, istraženi modeli i konkretne implementacije, osmišljena primjena te svi dodatni međukoraci.

3.1.1. Računalni vid u svrhu raspoznavanja objekata

Računalni vid je grana umjetne inteligencije koja se bavi interakcijom umjetne inteligencije s vizualnim podacima. Cilj je omogućiti računalima samostalno razumijevanje i interpretaciju vizualnih podataka s fokusom na raznovrsnoj primjenjivosti. Jedna od osnovnih i najčešćih zadaća računalnog vida je raspoznavanje objekata.

U zadnjih nekoliko godina, raspoznavanje objekata je postalo široko zastupljeno područje istraživanja zbog svojih raznovrsnih primjena, poput omogućavanja autonomne vožnje vozila ili podizanja ispravnosti dijagnoza u okvirima medicine [3], zbog čega postoji velik broj implementacija modela za prepoznavanje objekata. Neke od njih su: *YOLO (You Only Look Once)* [4], *SSD (Single Shot MultiBox Detector)* [5], *Faster R-CNN* [6] te izabrani model, *EfficientDet* [7]. Model umjetne inteligencije je sustav algoritama i matematičkih funkcija koji samostalno dolazi do određenog zaključka na temelju ulaznih podataka. Posjeduje ugrađene vrijednosti koje se mogu mijenjati treniranjem, čime se specijalizira za drukčije vrste zadataka. U našem slučaju model za prepoznavanje objekata specijaliziramo na prepoznavanje različitih vrsta otpada.

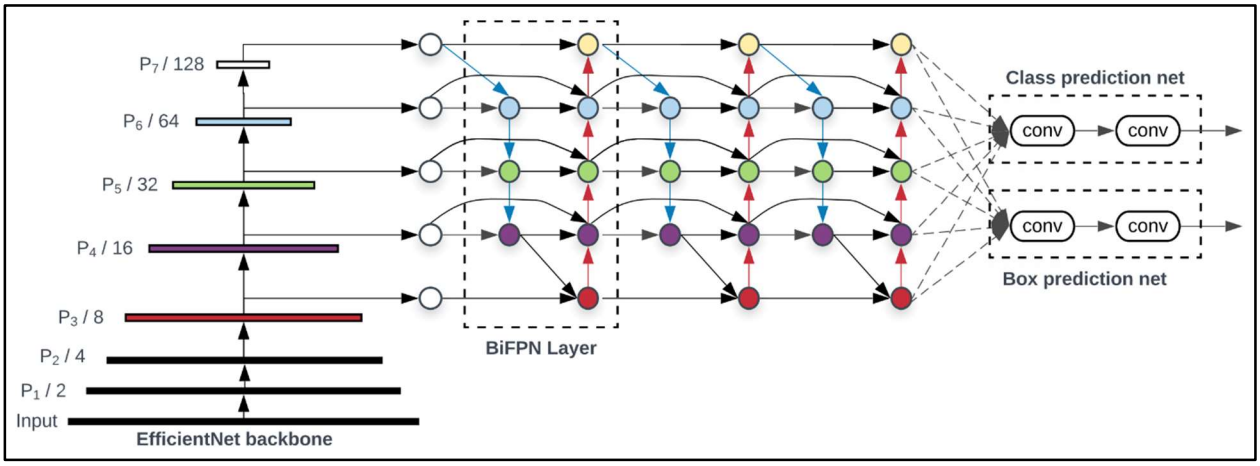
Raspoznavanje objekata je složen proces koji se sastoji od više koraka poput prepoznavanja (prepoznavanje dijela slike kao potencijalnog objekta), lokalizacije (određivanje granica i pozicije tog objekta) i klasifikacije (određivanje specifične klase za taj objekt). Prilikom raspoznavanja koriste se razne, opširne tehnike poput dubokog učenja [8] i ekstrakcije značajki, koje olakšavaju pojedine korake. Ekstrakcija značajki podrazumijeva definiranje i pronalazak bitnih karakteristika određene slike analizom, kako bi se sadržaj mogao iskoristiti u nekom računalnom procesu. Duboko učenje je grana strojnog učenja koja iskorištava neuronske mreže u svrhu rješavanja zadataka. Svaka pojedina neuronska mreža je sastavljena od međusobno povezanih čvorova, odnosno neurona, koji su raspoređeni u slojeve. Svaki od neurona prima ulaze iz

prijašnjih neurona pri čemu se preko težinske sume veza (svaka veza posjeduje težinu) računa izlaz. Pomoću toga duboko učenje može modelirati kompleksne veze između ulaza i izlaza, što omogućava rješavanje naprednih ili zahtjevnih problema poput raspoznavanja objekata.

3.1.2. Ideja i izabrani model

Naša promatrana okolina primjene računalnog vida su mobilni uređaji, zbog čega nam je bio potreban brz i računalno nezahtjevan model, što podrazumijeva kratko vrijeme raspoznavanja, nisko zaposjedanje resursa i relativno malu veličinu. Također je nužno moći implementirani i pokrenuti model u zadanoj okolini. Nakon razmatranja broja modela poput *YOLOv5* i *YOLOv7*, odlučili smo se za model *EfficientDet-lite2*. Model je dostatno brz i lak, a presudna značajka u odabiru bila je jednostavnost implementacije u našem odabranom okviru aplikacije.

Model spada u skupinu *EfficientDet* modela koje je razvio tim istraživača kompanije *Google*. Može se zamijetiti visoka stopa zastupljenosti među najčešće korištenim modelima, zbog njihove efikasnosti i skalabilnosti, odnosno mogućnosti efikasne primjene u sustavima proizvoljnih veličina, što potvrđuje činjenica kako postižu visoku preciznost, iako su znatno, neki čak do 9 puta, manji od nekih drugih vodećih modela [9]. Njihovu strukturu čine 3 glavna dijela: *EfficientNet backbone*, *BiFPN* sloj te *Box network*.



Slika 3.1 Struktura EfficientDet modela [9]

EfficientNet backbone je prvi dio modela koji se sastoji od *EfficientNet* mreže. Ona izvršava konvolucijske operacije na ulaznoj slici i time izvlači značajke niske razine.

Primjer tih značajki su obojenost i intenzitet piksela koji omogućuju prepoznavanje boja i sjena na slici kao i lokalni binarni uzorci (eng. *Local Binary Patterns, LBP*) koji se koriste za detekciju tekstura i oblika. Prednosti *EfficientNet* mreža su, što se tiče veličine ulaza i izlaza, visoka prilagodljivost i učinkovitost.

BiFPN sloj (eng. *Bi-directional Feature Pyramid Network*) je drugi dio modela. To je mreža koja spaja značajke niske i visoke razine. Unutar sloja, značajke prolaze kroz niz *BiFPN* blokova koji spajaju i usavršavaju značajke. U blokovima značajke prolaze kroz tri koraka:

1. dvosmjerno spajanje značajki iz prošlog i sljedećeg sloja
2. težinsko spajanje značajki preko naučenih težina
3. usavršavanje značajki gdje se provode kroz konvolucijske slojeve u svrhu smanjenja šuma i poboljšanja prikaza

Box network je treći dio modela koji se sastoji od mreže za predviđanje okvira te klase objekata. Ona uz pomoć konvolucijskih slojeva stvara niz prijedloga za okvire i klase objekata iz kojih se uklanjaju duplikati i izabiru samo najsigurniji prijedlozi.

EfficientDet-lite2 je lakša inačica *EfficientDet* modela, optimizirana za rad na mobilnim uređajima. Zbog svoje veličine, može vrlo brzo izvršavati prepoznavanje, no istodobno gubi na preciznosti.

3.1.3. Prikupljanje podataka

U svrhu prepoznavanja odabranih objekata, model je prvo potrebno istrenirati na većem skupu vjerodostojnih podataka, što su u našem slučaju pravilno označene fotografije odabranih objekata. Kako bismo kroz aplikaciju na smislen način mogli proučiti i pokazati primjenjivost dane tehnologije izabrali smo deset objekata koje naš model mora biti u stanju prepoznati:

- Plastične boce (svi oblici boca kao plastične ambalaže, mogu biti prozirni kao boce za sok ili vodu, ili neprozirni kao bočice za mliječne proizvode, ne uključuje ambalažu i čašice za jogurt, puding i slično)
- Staklene boce (svi oblici boca kao staklene ambalaže za pohranjivanje tekućina, alkoholnih i bezalkoholnih pića, ne uključuje staklenke i druge staklene posude za pohranjivanje hrane)

- Tetrapak (ambalaža u obliku kvadra za pohranjivanje tekućina, mliječnih proizvoda i slično, podrazumijeva se da je izvana kartonska, a iznutra obložena drugim materijalom, npr. plastikom)
- Baterije (svi oblici kućanskih baterija; A, AA, AAA, D, C...)
- Limenke (ambalažu raznih alkoholnih i bezalkoholnih pića u obliku limenke, ne uključuje konzerve)
- Žarulje (sve vrste kućanskih žarulja)
- Kartonske kutije (svi oblici kartonskih kutija i kartonske ambalaže, prvenstveno u obliku kvadra, bitno je da nisu plastificirane, te ako se vidi unutrašnjost da su prazne)
- Ostaci voća i povrća (neobrađeni ostaci voća i povrća; kora od naranče, kora od banane, ostaci salate, mrkve, rajčice...)
- Obuća (uključuje svu vrstu zatvorene obuće; tenisice, čizme, cipele...)
- Elektronički proizvodi (elektronički proizvodi poput laptopa, mobitela, televizora, monitora, računalnih komponenti i periferije, i slično, ne uključuje bijelu tehniku poput frižidera, klima i perilica)

Bit pokazne aplikacije je informirati korisnika o pravilima razvrstavanja i postojećoj infrastrukturi stoga je bilo bitno odabrati objekte kojima će se pokriti različite kategorije otpada i mogućnosti reciklaže. Objekti su također morali pokriti niz drugih svojstava koja su objašnjena u nastavku rada (vidi poglavlje 3.3. Strukture za primjenu podataka).

3.1.3.1. Zahtjevi fotografiranja

Kako bi točnost prilikom prepoznavanja bila dovoljno visoka, za svaki je objekt bilo potrebno skupiti pedesetak fotografija iz kojih će model naučiti prepoznavati određenu vrstu objekta. Fotografije su također morale zadovoljavati broj uvjeta kojima se osigurava kvaliteta slika i primjenjivost pri treniranju.

Slika 3.2 Primjeri mogućih fotografija pojedinog objekta

Određeni objekt bilo je potrebno fotografirati pri dostatnom osvjetljenju kako bi se osigurala mogućnost prepoznavanja. Nadalje, objekt je morao biti jasno vidljiv unutar okvira fotografije. Kako bismo dodatno povećali skup podataka i dodatno poboljšali prepoznavanje, objekte smo fotografirali iz različitih kutova i u različitim pozicijama, što je vidljivo iz priloženih slika.

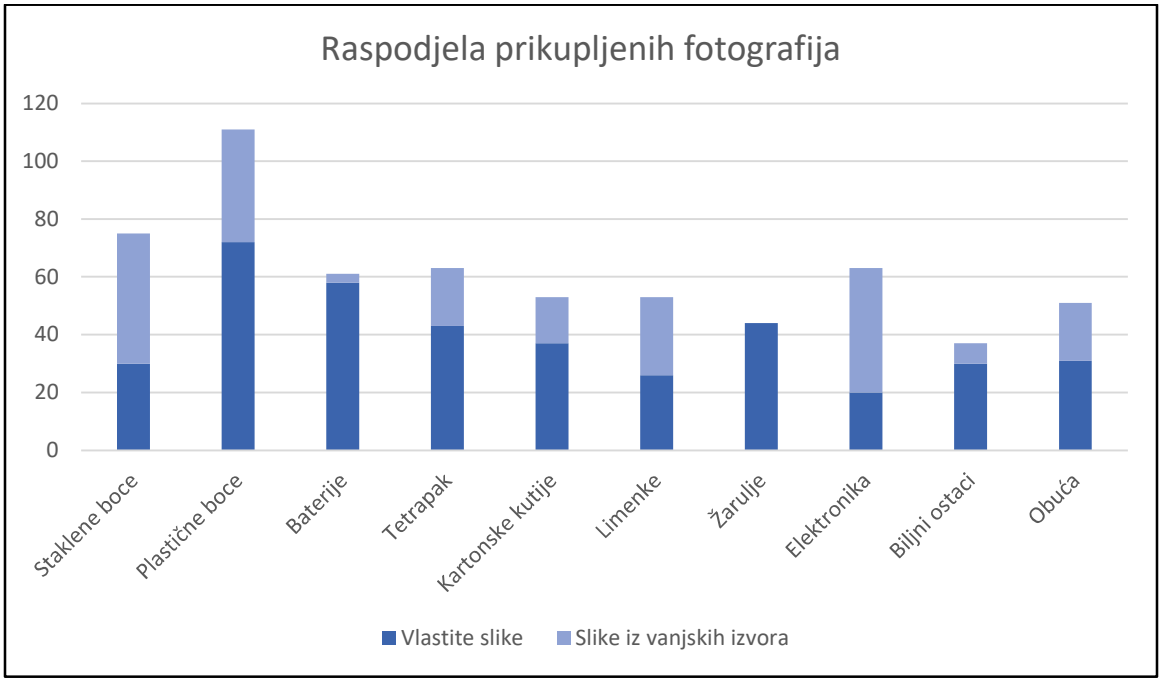
Slika 3.3 Primjeri fotografija različitih grupacija objekata

Objekte smo, osim toga, pokušavali fotografirati i u grupama, čime se modelu pružaju primjeri za prepoznavanje raznovrsnih konfiguracija, različitih količina objekata. Može se napomenuti kako smo objekte težili fotografirati u što više različitih slučajeva kako bi se

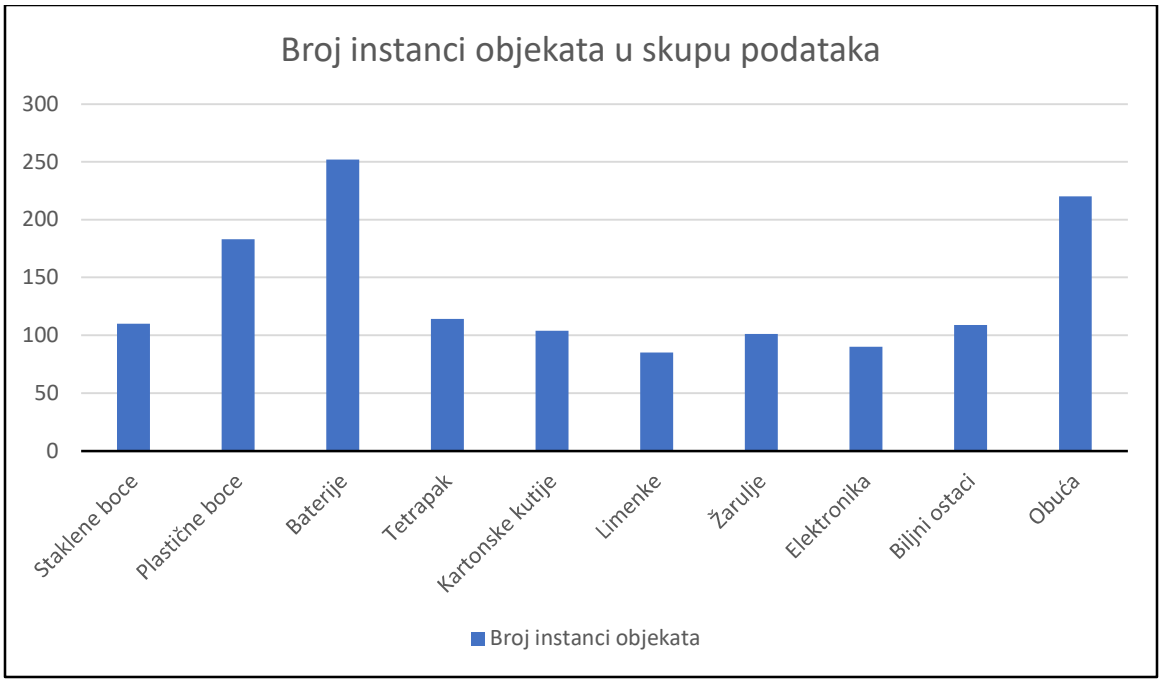
ograničilo promicanje pojedinih značajki pri prepoznavanju. Primjeri grupa dani su u priloženim fotografijama.

3.1.3.2. Prikupljanje fotografija pomoću zajednice

Prvobitni izvor slika činili smo mi sami, fotografirali smo objekte u različitim prilikama, ponajviše u vlastitim kućanstvima. Uspjeli smo prikupiti dostatan broj slika za stvaranje početnog modela kojeg smo koristili pri razvoju aplikacije, no za dostizanje opisanog cilja morali smo pronaći druge metode prikupljanja podataka. Odlučili smo zamoliti naše kolege, poznanike i prijatelje za pomoć. Kako bismo osigurali kvalitetu fotografija iz vanjskih izvora, stvorili smo dokument s već objašnjenim pravilima i naputcima koji smo im zatim proslijedili. Uz njihovu pomoć uspjeli smo sakupiti dostatan broj slika za razvoj modela do postavljenog cilja. Važno je napomenuti kako smo svakog pitali za dopuštenje za upotrebu fotografija prije čega smo objasnili za što će se sve točno koristiti. Na kraju smo sve primljene fotografije ponovno razmotrili kako bismo osigurali kvalitetu i spriječili pojavljivanje osobnih ili osjetljivih informacija.



Slika 3.4 Raspodjela prikupljenih fotografija

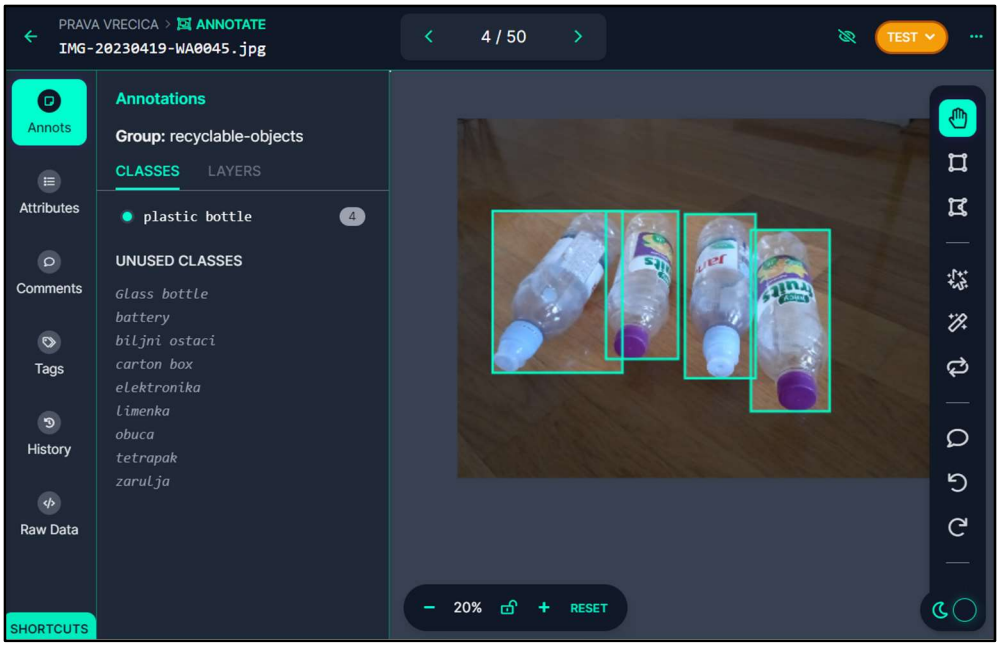


Slika 3.5 Broj instanci objekata u skupu podataka

Iako se na prvom priloženom dijagramu može primijetiti kako količina fotografija za određene objekte nije premašila 50, drugi dijagram ipak daje uvid u pravu količinu prikupljenih podataka jer pojedine fotografije sadrže više instanci pojedinog objekta.

3.1.4. Treniranje modela

Na svakoj je fotografiji bilo potrebno okvirima označiti pojedine predmete i klasifikaciju kojoj pripadaju, kako bi bile korisne pri treniranju. Ti su koraci izvedeni uz pomoć alata *Roboflow*, koji omogućuje jednostavnu organizaciju, označavanje i upotrebu podataka. Priložena slika objašnjava označavanje niza objekata. Osim za označavanje objekata, alat smo koristili i za pretprocesiranje podataka, što podrazumijeva svođenje na isti format i veličinu, povećavanje količine podataka uz pomoć rotacije i zamućivanja, što podrazumijeva stvaranje novih fotografija iz postojećih promjenom spomenutih značajki, te za laku podjelu podataka na setove za treniranje, validaciju i testiranje.



Slika 3.6 Primjer uporabe alata Roboflow

Set za treniranje je skup podataka na kojem se trenira model. Preko validacijskog seta se na kraju svake epohe provjerava točnost modela, nakon čega se u slučaju nedostatnog poboljšanja omogućuje završetak procesa kako bi se spriječio takozvani *overfitting*, odnosno preveliki utjecaj danih podataka na prepoznavanje, čime se smanjuje mogućnost generalne uporabe. Zadnji je set za testiranje na kojem se pri završetku treniranja provjerava točnost modela. Bitna napomena je kako setovi nemaju presjeka, odnosno ne postoji fotografija objekta koja je u dva seta odjednom. Razlog tome je što se točnost mora isprobati na modelu prethodno neviđenim podacima, kako bi se provjerila

generalna točnost rada, odnosno točnost u prepoznavanju objekata izvan seta za treniranje.

```
model = object_detector.create(train_data, model_spec=spec, batch_size=8,
        train whole model=True, validation data=validation data)
```

Slika 3.7 Isječak za treniranje modela

Proces treniranja proveli smo pomoću *TensorFlow Lite Model Maker* biblioteke, programskog jezika *Python*, a pokretanje treniranja vidljivo je unutar priloženog isječka koda. Biblioteka daje mogućnosti za učitavanje početnog modela, u našem slučaju *EfficientDet-lite2*, te za treniranje na zadanim podacima. Samo treniranje se odvija kroz određeni broj epoha, od kojih u svakoj model prođe kroz cijeli set podataka i ovisno o tome mijenja svoje vrijednosti kako bi mogao prepoznati željene objekte. Nakon što model prođe kroz set podataka za treniranje, prolazi kroz validacijski set gdje se provjerava uspješnost.

```
Epoch 1/50 48/48 [==============================] - 85s 597ms/step -
det_loss: 1.6627 - cls_loss: 1.1043 - box_loss: 0.0112 - reg_l2_loss: 0.0762
- loss: 1.7390 - learning_rate: 0.0090 - gradient_norm: 1.8265 -
val_det_loss: 1.2100 - val_cls_loss: 0.7986 - val_box_loss: 0.0082 -
val_reg_l2_loss: 0.0762 - val_loss: 1.2862

Epoch 2/50
48/48 [==============================] - 27s 564ms/step - det_loss: 1.1643 -
cls_loss: 0.7669 - box_loss: 0.0079 - reg_l2_loss: 0.0763 - loss: 1.2406 -
learning_rate: 0.0100 - gradient_norm: 4.4127 - val_det_loss: 0.9883 -
val_cls_loss: 0.6119 - val_box_loss: 0.0075 - val_reg_l2_loss: 0.0764 -
val_loss: 1.0647

Epoch 3/50
48/48 [==============================] - 26s 547ms/step - det_loss: 0.8693 -
cls_loss: 0.5557 - box_loss: 0.0063 - reg_l2_loss: 0.0765 - loss: 0.9458 -
learning_rate: 0.0099 - gradient_norm: 4.7472 - val_det_loss: 0.8736 -
val_cls_loss: 0.5291 - val_box_loss: 0.0069 - val_reg_l2_loss: 0.0766 -
val_loss: 0.9502
…
```

Slika 3.8 Primjer ispisa za vrijeme treniranja

Nakon svake epohe, mogu se primijetiti razne dostupne metrike pomoću kojih se prati napredak treniranja, što je vidljivo iz priloženog isječka ispisa. Vidljivo je o kojoj se epohi radi te niz vrijednosti:

- vrijednost *det_loss (detection loss)* koja predstavlja točnost modela pri lokalizaciji i detekciji objekata na fotografijama
- vrijednost *cls_loss (class loss)* koja predstavlja grešku pri klasificiranju objekata
- vrijednost *box_loss* koja predstavlja grešku pri lokalizaciji objekata
- vrijednost *reg_l2_loss (L2 regularization loss)* koja se pridodaje drugim gubicima u svrhu generalizacije modela, proporcionalna je težini veza unutar modela, što su veće težine nekih veza, model se više usredotočuje na neke značajke što nije poželjno ponašanje, ova vrijednost kažnjava takvo ponašanje i time sprječava specijalizaciju na neke značajke
- vrijednost *loss* je generalna metrika koja se računa uz pomoć svih drugih *loss* metrika, odnosno jedan broj koji predstavlja točnost modela
- vrijednosti *val_det_loss*, *val_cls_loss*, *val_box_loss*, *val_reg_l2_loss* te *val_loss* su analogne vrijednosti prije navedenima, jedina razlika je što su one nad validacijskim setom podatka, a ne nad setom za treniranje
- vrijednost *learning_rate* koja predstavlja utjecaj rezultata specifične epohe na težine unutar modela, što je veća, to će se veće promjene dogoditi na modelu
- vrijednost *gradient_norm* predstavlja duljinu vektora gradijenta gubitaka, model uz pomoć vlastitih težina i vrijednosti gubitaka, može stvoriti vektor gradijenta gubitaka te u svrhu smanjivanja greške, vrijednosti težina rasporediti u suprotnom poretku

Nakon završenog treniranja, model se provjerava nad testnim setom podatka kako bi se procijenila točnost modela.

```
{'AP': 0.4054174, 'AP50': 0.74998087, 'AP75': 0.39733866, 'APs':
0.15171348, 'APm': 0.40062436, 'APl': 0.576063, 'ARmax1': 0.26189485,
'ARmax10': 0.4613269, 'ARmax100': 0.5221528, 'ARs': 0.19, 'ARm':
0.54897726, 'ARl': 0.6479167, 'AP_/Glass bottle': 0.4731644,
'AP_/battery': 0.18169308, 'AP_/plastic bottle': 0.66795146,
'AP_/tetrapak': 0.2988607}
```

Slika 3.9 Primjer ispisa evaluacije nad testnim podacima

Nakon procjene, ponovno su prisutne razne metrike, koje se mogu proučiti u danom isječku koda. Najbitnije metrike su:

- *AP (Average precision)* mjeri prosječnu preciznost kroz omjere točnih pozitivnih detekcija, broja detekcija
- *AP50* i *AP75* također mjere prosječnu preciznost, ali samo za detekcije s prosjekom od 50 ili 75 posto između predviđenih i točnih granica objekata označenih na slikama
- *APs (s - small)*, *APm (m - medium)*, *APl (l - large)* su također prosječna preciznost ali za objekte određenih veličina
- *AR (Average recall)* je prosječan odziv koji predstavlja prosječni omjer točnih pozitivnih detekcija i točnih na slikama označenih objekata, ostale *AR* vrijednosti su analogne onima za *AP*
- *AP_/objekt* predstavlja prosječnu preciznost za svaki od objekata, specifično na ovom primjeru se može vidjeti vrlo niska vrijednost za baterije, iz čega se može zaključiti kako primjerni model nije precizan u detekciji baterija zbog čega je potrebna promjena parametara učenja ili, vjerojatnije, promjena seta podataka s baterijama

Nakon završetka evaluacije, ukoliko su rezultati zadovoljavajući, utoliko se model izvozi u *tflite* format koji se može ugraditi u aplikaciju. Nakon izvoza poželjno je ponovno evaluirati model na testnom setu podataka jer pri promjeni formata može doći do neželjene promjene performansi.

3.1.5. Posluživanje modela

Kako bismo osposobili model unutar aplikacije, koristimo *tflite_flutter* i *tflite_flutter_helper* biblioteke koje omogućavaju pristup implementacijama *TensorFlow Lite* na *iOS* i *Android* uređajima unutar razvojnog okruženja *Flutter*. *TensorFlow Lite* je jednostavnija i efikasnija verzija *TensorFlow* okvira, jednog od najraširenijih okvira namijenjenih strojnom učenju, koja je prilagođena radu na mobilnim uređajima. Takvi okviri nude širok pojas alata koji pomažu u svakom koraku stvaranja i implementacije različitih modela umjetne inteligencije [10]. *TensorFlow Lite* smo odabrali zbog jednostavnosti treniranja modela, široke dostupnosti i podrške te izvrsnim performansama na mobilnim uređajima.

Kako bi se omogućilo prepoznavanje otpada pomoću aplikacije potrebno je nekoliko koraka:

1. stvoriti instancu *Interpeter* koja stvara okruženje u kojem se može pokretati model, a stvara se iz datoteke koja sadrži naš model

```
Final interpreter = await Interpreter.fromAsset("model_v1.tflite", options:
InterpreterOptions()..threads = 4);
```

Slika 3.11 Isječak koda za inicijalizaciju Interpretera

```
TensorImage getProcessedImage(TensorImage inputImage) {
        var padSize = max(inputImage.height, inputImage.width);
        var imageProcessor = ImageProcessorBuilder()
                .add(ResizeWithCropOrPadOp(padSize, padSize))
                .add(ResizeOp(inputSize, inputSize, ResizeMethod.BILINEAR))
                .build();
        inputImage = imageProcessor.process(inputImage);
        return inputImage;
}
```

Slika 3.10 Procesiranje slike prije predaje modelu

2. slika na kojoj se prepoznaju objekti procesira se kako bi bila prikladna za ulaz u model, ona se svodi na kvadratni oblik veličine 448x448 piksela

3. potrebno je definirati ulaze i izlaze, *TensorBuffer* varijable se spremaju u mapu *outputs* prije pokretanja modela, ali isječak je izostavljen zbog duljine

```
inputImage = getProcessedImage(inputImage);


TensorBuffer outputScores = TensorBufferFloat(_outputShapes[0]);
TensorBuffer outputLocations = TensorBufferFloat(_outputShapes[1]);
TensorBuffer numLocations = TensorBufferFloat(_outputShapes[2]);
TensorBuffer outputClasses = TensorBufferFloat(_outputShapes[3]);


List<Object> inputs = [inputImage.buffer];
```

Slika 3.12 Isječak za definiranje ulaza i izlaza

4. pokretanje predviđanja

```
_interpreter.runForMultipleInputs(inputs, outputs);
```

Slika 3.13 Pokretanje modela sa slikom kao ulazom

5. kod za obrađivanje izlaza izostavljen je zbog duljine, nakon završenog predviđanja, u varijabli tipa *Map<int, Object> outputs* se nalaze objekti koje je model prepoznao na slici, njihove lokacije, labele i sigurnost, ostaju samo objekti s dovoljno visokom sigurnošću te se njihove granice okvira transformiraju kako bi odgovarale lokacijama na originalnoj slici

3.2. Strukture za primjenu podataka

Osnovne informacije koje nam pruža analiza fotografije računalnim vidom same po sebi, najčešće nisu primjenjive kao krajnji produkt. Tako u našem slučaju informacije o vrstama prepoznatog otpada čine samo temelj na kojem se gradi detaljniji sažetak naputaka od kojeg korisnici imaju koristi. Kako je već navedeno, aplikacija je alat koju pokušava olakšati reciklažu otpada pružanjem korisnih informacija o pravilima i regulacijama te infrastrukturi za odlaganje. Kako bi se u tu svrhu upogonila pomoć umjetne inteligencije potrebno je bilo osmisliti način vezivanja podataka dobivenih računalnim vidom i prikazivih, korisnih podataka.

Takvo vezivanje ostvarili smo upotrebom dviju zasebnih *JSON (JavaScript Object Notation)* datoteka koje sadrže sve potrebne, lako izmjenjive podatke. *JSON* datoteke su tekstualne datoteke koje služe za prijenos i zapisivanje podataka u *JSON* formatu, skupu zapisa vezanih principom ključ – vrijednost. Format se vrlo često koristi jer je vrlo pogodan za svoju namjenu, vrlo se lako stvara i čita te ne zauzima mnogo prostora [11].

Svaka od zasebnih *JSON* datoteka sadrži opis jedne potrebne strukture s njezinim popratnim informacijama. Prva je popis prepoznatljivih objekata u kojem smo odlučili pohranjivati opis svih dostupnih vrsti otpada na pojedinom jeziku te dodatnih značajki koje su uz njih vezane, što uključuje labelu (način na koji je otpad označen u modelu umjetne inteligencije), ime (naziv koji će se prikazivati), opis (kratki opis objekta koji će se prikazivati) i atribute (značajke koje su vezane uz pojedinu vrstu otpada, poput informacija kako je prije odlaganja objekt potrebno isprazniti, kako je opasan ili kako ga je moguće donirati). U tom zapisu su pohranjene sve lokalizirane vrijednosti koje su potrebne za prikaz informacija o pojedinoj vrsti otpada kao takvoj.

```
{
  "id": 1,
  "language": "hr",
  "objects": [

      "label": "plastic bottle",
      "name": "Plastična boca",
      "desc": "Plastična ambalaža raznih vrsta tekućina, u obliku boce.",
      "attributes": {
        "container": true
      }
    },
    ...
```

Slika 3.14 Primjer isječka JSON datoteke za pohranjivanje objekata

Druga *JSON* datoteka sadrži opis strukture i pravila recikliranja za pojedini grad ili područje. Struktura sadrži popis vlastitih značajki (jezik, ime, autor), popis korisnih informacija o autoru pravila i dodatnih informacija, te liste kategorija i specijalnih kategorija. Svaka od kategorija dodatno sadrži vlastite informacije (ime, vezana boja, dodatne informacije, mogućnosti odlaganja) te popis objekata koji spadaju u tu kategoriju, s dodatnim atributima za svaki od njih, što se pridodaje atributima koji su za objekt već navedeni u drugoj *JSON* datoteci.

```
{
  "id": 1,
  "language": "HR/hr",
  "name": "Pravila odlaganja grada Zagreba",
  "author": "Zagrebački Holding - Podružnica Čistoća",
  "author-info": {
    ...
    "Opis" : {
      "final": true,
      "info": "Podružnica Zagrebačkog Holdinga zadužena za gospodarenje otpadom."
    },
    ...
  },
  "info-list": [
     ...
  ],
  ...
}
```

Slika 3.15 Primjer isječka JSON datoteke s informacijama o strukturi pravila za grad Zagreb

```
{
  ...
 "categories-list": [
   {
     "name": "Plastična i metalna ambalaža",
     "color": "0xFFFFEB3B",
     "info": {},
     "where": {
       ...
       "0": {
         "final": true,
         "data": "Vrećice za plastiku i metalnu ambalažu u kućanstvu"
       },
       "1": {
         "final": true,
         "data": "Žuti spremnici"
       },
       ...
     },
     "objects": [
       {
         "label": "plastic bottle",
         "attributes": {
           "returnable": {
             "money": "10c",
             "info": "Moguće je da plastičnu ambalažu možete vratiti."
           }
         }
       },
       ...
```


Slika 3.16 Primjer isječka JSON datoteke s listom kategorija za grad Zagreb

```
{
  ...
  "special": [
    {
      "label": "battery",
      "color": "0xFF000000",
      "info": {},
      "where": {
        ...
        "0": {
          "final": true,
          "data": "Odlagališta otpada"
        }
      }
    },
    ...
```

Slika 3.17 Primjer isječka JSON datoteke s listom specijalnih vrsta otpada za grad Zagreb

Iz ove dvije *JSON* datoteke moguće je na temelju labele povezati sve informacije o pojedinom objektu i načinu na koji se odlaže, specifično za neko područje te lokalizirano na određen jezik. Tako je, na primjer, u priloženim odsječcima zapisano kako u gradu Zagrebu postoji kategorija *Plastična i metalna ambalaža*, a otpad koji u nju pripada može se odložiti pomoću žutih spremnika ili vrećica u kućanstvu. Dodatno u tu kategoriju pripadaju plastične boce koje se uvijek moraju isprazniti prije odlaganja, a u gradu Zagrebu moguće ih je kao ambalažu vratiti za određenu količinu novca. Baterije ne spadaju ni pod jednu kategoriju, već je posebno zapisao kako ih se može odložiti na odlagalištima otpada.

Na kraju se može napomenuti kako je još jedna prednost *JSON* datoteka njihova formatiranost i laka izmjenjivost, što dopušta brzu i laku prilagodbu u slučaju izmjene pravila odlaganja. Nadalje, na temelju istog formata vrlo jednostavno se mogu izraditi pravila za različita područja i jezike, što dozvoljava široku upotrebu ovakvog alata. Pri

izradi pokaznih pravila konzultirali smo se pravilima i naputcima dostupnim na stranicama *Zagrebačkog Holdinga* i *podružnice Čistoća*.

3.3. Izvedba mobilne aplikacije

Aplikacija kao okvir za proučavanje mogućnosti našeg modela umjetne inteligencije morala je podržavati sve dosad navedene zahtjeve, od udomljavanja modela, njegovog pokretanja i analize dobivenih rezultata, do komunikacije s bazom podataka. Korisniku je zatim bilo potrebno pružiti intuitivno, jednostavno sučelje kroz koje može dobiti uvid u sve što ga zanima te iskoristiti mogućnosti koje se nude.

Kako je za uporabu modela nužna kamera, logičan izbor sustava za koje se takva aplikacija može razviti čine mobilni uređaji, najčešće mobiteli i tableti s operacijskim sustavima *iOS* i *Android*. Razvojni okvir za koji smo se odlučili, prema ovom zahtjevu, pri razvoju aplikacije jest *Flutter*, sustav za stvaranje aplikacija na više platformi kojeg je razvio *Google*. Izvedba aplikacija unutar okvira temelji se na programskog jeziku *Dart* koji se ne oslanja na nikakav pomoćni *markup* jezik za izgradnju sučelja i strukture, već u tu svrhu koristi vlastite klase [12]. Pogodan je za izradu raznih aplikacija za različite sustave jer nudi podršku za različite platforme kroz jednu kodnu bazu, što znači da nije potrebno pisati zaseban kod za recimo *iOS* i *Android* uređaje, jer se izvršni programi automatski izvode iz onog napisanog pomoću jezika *Dart*. Kao takav, okvir je vrlo moderno i pogodno rješenje za razvoj mobilnih aplikacija [13]. Pregled razvijene pokazne aplikacije dan je u nastavku rada (vidi poglavlje 4.1. Pregled sučelja aplikacije).

3.4. Baza podataka i sučelje za pristup

Baze podataka omogućuju efikasno upravljanje velikom količinom podataka. Aplikacije često imaju potrebu za takvim alatom jer se podaci kojima barataju, poput korisničkih zapisa, poruka, slika, datoteka i drugih informacija, često mijenjaju. Bez baze podataka, aplikacija ne bi mogla pohraniti, organizirati i upravljati tim podacima na efikasan način, što bi dovelo do narušenog načina rada ili bi značilo kako pak svaki put kad se aplikacija pokrene, korisnik započinje rad s čistom stranicom, bez ikakvih prethodno spremljenih promjena.

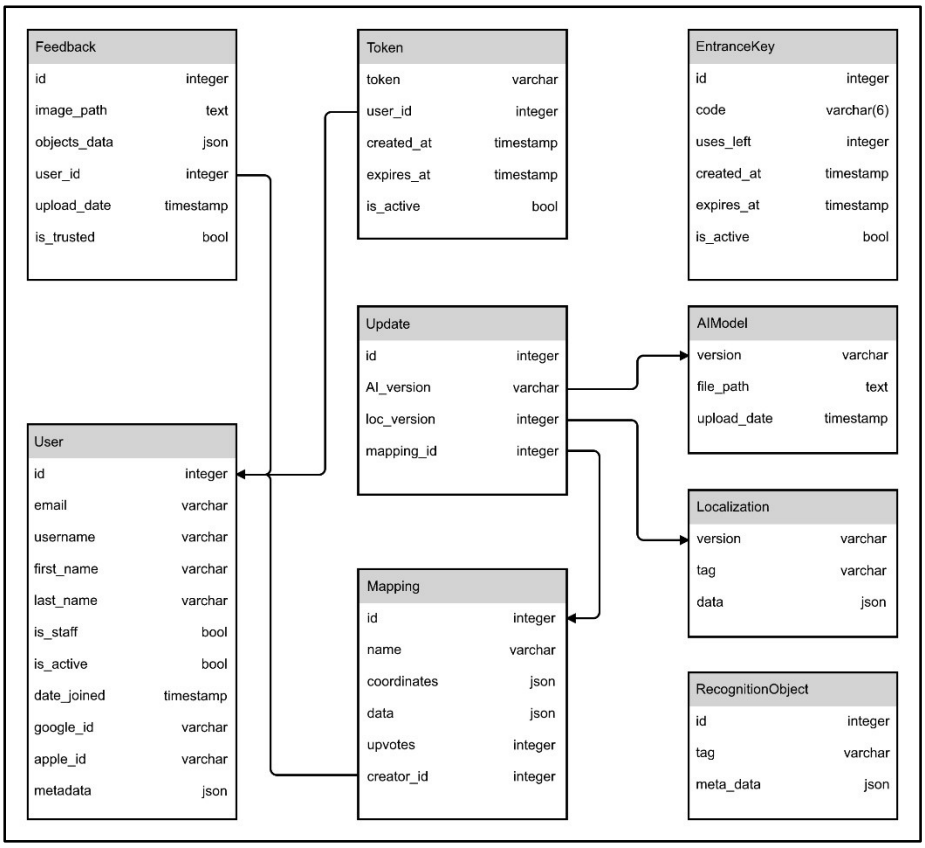
3.4.1. Struktura baze podataka

Za zamišljeni rad aplikacije bilo bi potrebno spremati sljedeće podatke:

- podatke o korisniku
    - osnovni podaci kao što su identifikacijski broj korisnika, ime, prezime, korisničko ime te autentikacijsku varijablu poput broja telefona ili adrese elektroničke pošte
    - opcionalne vanjske poveznice za autentikaciju, poput onih koje pružaju *Google* ili *Apple*
    - *metadata*, što odgovara dosadašnjim rezultatima i statistikama recikliranja otpada korisnika
    - jednosmjerno kriptirana zaporka (standardna praksa koja osigurava autentikaciju pri kojoj jedino korisnik zna pravu zaporku)
- povratne informacije korisnika (eng. *feedback*)
    - fotografija na kojoj se prepoznaju objekti
    - podaci o objektima koje je prepoznala umjetna inteligencija (vrsta objekta i pozicija unutar slike)
    - identifikacijski broj korisnika koji je poslao povratnu informaciju
- upotrebljive modele umjetne inteligencije
- lokalizaciju (datoteke s prijevodom teksta aplikacije na druge jezike)
- pravila recikliranja za pojedinačne lokacije (*mapping*)

Fotografije i modeli generalno zauzimaju previše prostora za bazu podataka zbog čega ih se sprema u poseban direktorij za pohranu, a u bazu podataka se upisuje priležeća

adresa tog dokumenta. S obzirom na to da veličine datoteka koje sadrže neke modele mogu biti poprilično velike, spremat će se samo neki konačan broj modela, pri čemu se najstariji odbacuje.



Slika 3.18 Shema zamišljene baze podataka

3.4.2. Rješenje sučelja za pristup

Izvedba internetske baze podataka moguća je upotrebom servera. Za pristup serveru koristi se *API (Application Programming Interface)* mehanizam *REST (Representational State Transfer)* arhitekture, koji nama omogućuje upravljanje, a korisnicima interakciju sa serverom, odnosno postavljanje ili preuzimanje određenih podataka. *API* mehanizam je skup definiranih pravila i protokola, odnosno sučelje, koje omogućava komunikaciju između raznih aplikacija i servisa. *REST* arhitektura je jedan način pristupa izradi takvog alata koji podrazumijeva uporabu *HTTP (Hyper Text Transfer Protocol)* zahtjeva (*GET* – dohvati, *POST* – postavi, *PUT* – izmijeni, *DELETE* - izbriši) i zasebnih adresa za svaki definirani zahtjev [14]. Kao takva, *REST* arhitektura je vrlo pristupačna i stoga vrlo raširena u razvoju ovakvih rješenja.

Konkretna implementacija izvedena je pomoću *Django* okvira kao jednim od danas najraširenijih platformi za izradu *REST* sučelja pomoću programskog jezika *Python*. Nudi mnoge pogodnosti poput integracije sa standardnim *Django* funkcijama, podrške za autentikaciju i autorizaciju, automatske dokumentacije i mnogo drugih značajki koje olakšavaju izradu i održavanje. Također iza sebe ima veliku zajednicu korisnika koji pridonose lakom rješavanju problema s kojima se pojedinac pri implementaciji može susresti [15].

Izvedeno sučelje se po potrebi pokreće na platformi *PythonAnywhere* koja omogućuje programerima stvaranje, provjeru i izvršavanje *Python* aplikacija u oblaku. Odabrali smo *PythonAnywhere* zbog jednostavnosti i mogućnosti primjene za različite veličine projekata.

3.4.3. Autentikacija

Pristup serveru, zbog zaštite podataka i naših modela, te validacije zaprimljenih povratnih informacija, ne smiju imati neovlaštene osobe, zbog čega se pomoću tokena provodi autentikacija korisnika. Tokeni su važeći ključevi koji se stvaraju pri valjanim uvjetima, na primjer, pri valjanoj registraciji korisnika. Pri svakom sljedećem pristupu serveru, potrebno je u zaglavlje zahtjeva staviti odgovarajući token kako bi se konkretni korisnik mogao identificirati.

Korisnik se može registrirati na dva načina:

- uporabom autentikacijskih usluga koje pružaju *Google* i *Apple*
- upisivanjem ulaznog ključa

Autentikacijske usluge koje pružaju *Google* i *Apple* nude veću razinu sigurnosti u odnosu na uobičajenu registraciju pomoću ključa i lozinke.

Ulazni ključ služi kao alternativna, sigurna metoda registracije. Ulazni ključ se može stvoriti u trenucima kad je potrebno registrirati veći broj korisnika za koje se može jamčiti bez dodatne validacije. Jedan primjer je predstavljanje aplikacije pred publikom, gdje se ključ može podijeliti sudionicima događaja kako bi se u kratkom vremenskom roku registrirali i isprobali aplikaciju.

Prilikom kreiranja ključeva, mogu se zadati i dva dodatna parametra:

- datum i vrijeme isteka ključa
- maksimalan broj korisnika

Istekom vremena ili registracijom maksimalnog broja korisnika koristeći određeni ključ, on se gasi, zbog čega se njime više ne mogu stvarati novi korisnici. Za takav sustav smo se odlučili kako bi se osiguralo stvaranje samo očekivanih korisnika.

3.4.4. Krajnje točke

Mobilna aplikacija može upravljati podacima pohranjenim na serveru putem takozvanih krajnjih točaka, odnosno adresa oblikovanih dodatkom pojedinih argumenata na kraj korijena, ovisno o vrsti poslanog zahtjeva.

Korijen adrese sučelja automatski se dodjeljuje prilikom pokretanja na platformi *PythonAnywhere*, a sastoji se od imena korisničkog računa i standardnog dijela.

$$< ime >.pythonanywhere.com$$

Neke od bitnijih krajnjih točaka koje se koriste pri općem radu aplikacije dane su u tablici.

Tablica 3.1 Bitne krajnje točke

Tip zahtjeva	Adresa	Objašnjenje
GET	*korijen/user*	dohvaćanje podataka korisnika
POST	*korijen/user*	stvaranje novog korisnika
PUT	*korijen/user*	promjena korisničkih podataka
GET	*korijen/updates*	dohvaćanje novog modela, objekata i prijevoda
POST	*korijen/feedback*	slanje povratnih informacija

Postoje i krajnje točke za održavanje sustava koje trenutno možemo koristiti jedino mi kao autori. One omogućuju postavljanje novih, ažuriranih tehnologija na server kako bi postale dostupne korisnicima. U to spadaju novi modeli, kao i nove popratne lokalizacije i pravila odlaganja. Uz to, postoje i krajnje točke za druge značajke kojima se koristimo za održavanje, poput stvaranja ulaznih ključeva.

4. Rezultati

Unutar sljedećih poglavlja opisani su rezultati projekta što podrazumijeva izvedenu pokaznu aplikaciju i model, analizu rada modela te upute za instalaciju i pokretanje aplikacije za znatiželjne čitatelje.

4.1. Pregled sučelja aplikacije

Ovo poglavlje pruža uvid u rad izrađene pokazne aplikacije, opis korisničkog sučelja i izvedbe zasebnih komponenti. U nastavku se također mogu pronaći upute za preuzimanje i instalaciju pokazne aplikacije na vlastitim mobilnim uređajima.

4.1.1. Zaslon dobrodošlice

Zaslon dobrodošlice je vrlo jednostavna uvodna komponenta koja se sastoji od logotipa aplikacije. Logo se sastoji od obrisa zelene kvačice i plave vrećice, temeljene na zagrebačkim plavim vrećicama za mješoviti otpad po kojima je aplikacija dobila ime. Obrisi zajedno čine oblik slova *V* što insinuira na riječ *vrećica*, a kvačica na pridjev *prava* što zajedno daje naziv aplikacije. Zaslon dobrodošlice u okvirima aplikacije služi kao osigurač za vrijeme učitavanja bitnih komponenti i stiliziran prijelaz u glavno sučelje. Za stvaranje zaslona dobrodošlice prilagođenih za različite uređaje koristi se paket *flutter_native_splash* koji daje pristup naredbama za automatizirano stvaranje takvih zaslona. Za dodatno postavljanje prilagođene vanjske ikone na uređaju upotrebljen je paket *flutter_launcher_icons*.

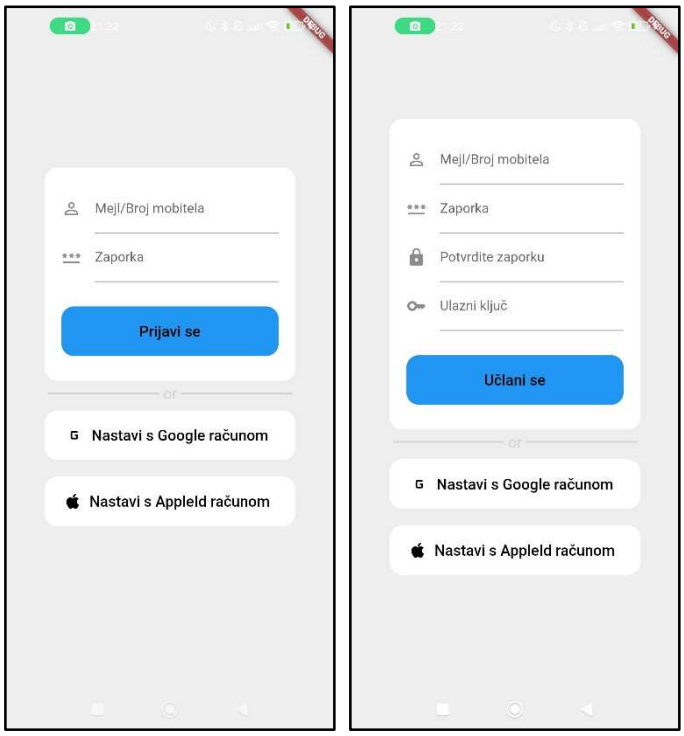
Slika 4.1 Zaslon dobrodošlice

4.1.2. Registracija, prijava i navigacija

Nakon završetka učitavanja potrebnih podataka i komponenti aplikacija izlazi iz zaslona dobrodošlice u zaslon za odabir korisničkog računa. Ovdje se mogu vidjeti tri gumba, koji nude različite mogućnosti prijave. Prvi vodi na zaslon za stvaranje novog računa, drugi na zaslon za prijavu, a treći dopušta uporabu aplikacije bez prethodne prijave, što se može vidjeti i iz popratnog teksta. S obzirom na to da se ne može osigurati rad internetske baze podataka na dugi vremenski rok zbog ograničenih resursa, pokazna inačica aplikacije namijenjena je radu bez korisničkog računa, te se preporučuje odabir tog gumba prilikom pokretanja aplikacije.

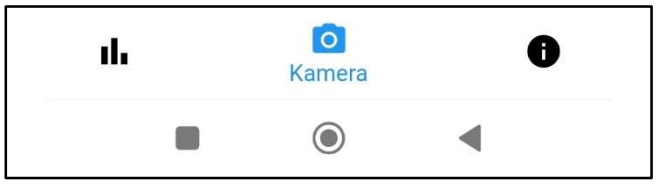
Slika 4.2 Zaslon za odabir korisničkog računa

Zasloni za stvaranje računa i prijavu izvedeni su u obliku formulara s potrebnim poljima, s dodatnom mogućnošću nastavljanja s *Apple* ili *Google* računom. Kako je već navedeno ovi zasloni prisutni su isključivo zbog kompletnosti aplikacije te ne pružaju osmišljenu uslugu u pokaznoj inačici.



Slika 4.3 Zasloni za stvaranje računa i prijavu

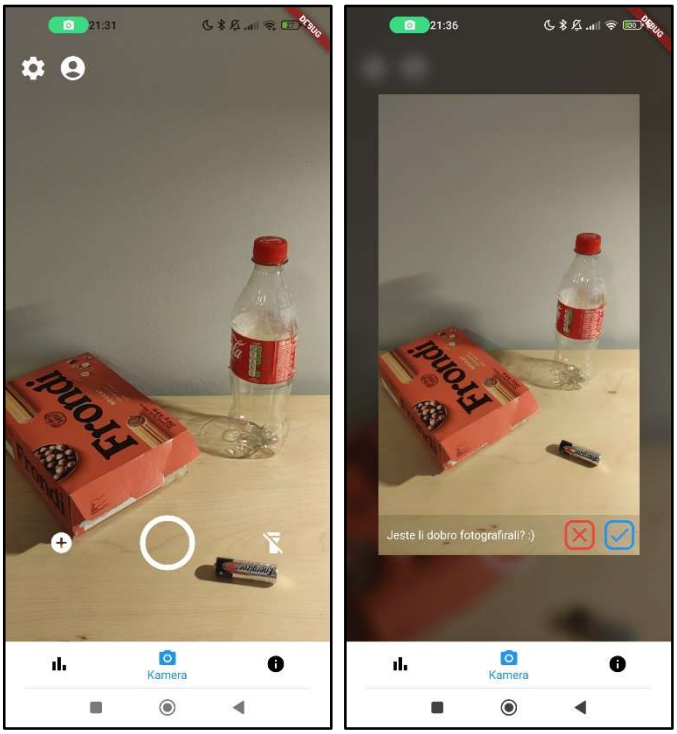
Nakon odabira načina rada i korisničkog računa aplikacija prelazi u glavno sučelje. Izvedeno je u obliku niza zaslona koji dijele zajednički okvir s prikazanom trakom za navigaciju. Unutar trake vidljive su ikone koje predstavljaju zasebne zaslone pri čemu je vidljiv naziv trenutno odabranog zaslona, koji je zajedno s ikonom pobojan u plavo radi bolje vidljivosti. Pritiskom na neku drugu ikonu mijenja se odabrani zaslon što je popraćeno animacijom. Animacije su izvedene pomoću paketa *animations*.



Slika 4.4 Traka za navigaciju

4.1.3. Kamera i upotreba modela

Početno odabran zaslon unutar okvira glavnog sučelja standardno je zaslon s kamerom, koji omogućava korisniku fotografiranje željenog otpada. Standardni paket za implementaciju funkcija kamere jest *camera* kojeg smo upotpunili alatima koje pruža paket *provider*. Navedeni paket implementira strukture koje omogućuju pristup određenim komponentama u bilo kojem dijelu aplikacije. Za pružanje usluga kamere smo tako definirali *camera provider* koji širi pristup kameri. U sklopu aplikacije koriste se brojni primjerci strukture *provider*, na primjer za baratanje spremljenim podacima, korisničkim postavkama, prijevodom ili modelom umjetne inteligencije.



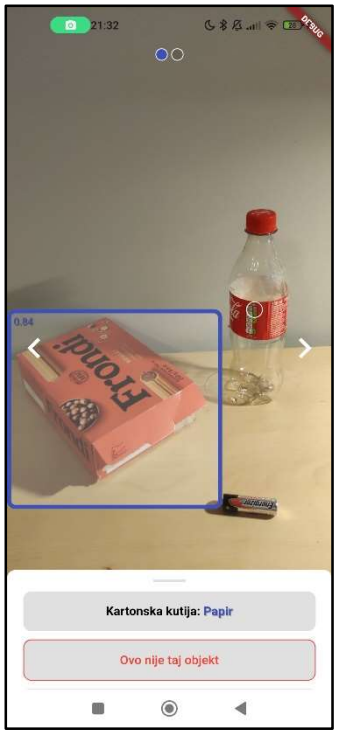
Slika 4.5 Zaslon kamere i pretpregled

U centru zaslona nalazi se glavni gumb za fotografiranje, pritiskom na kojeg se pokreće snimanje fotografije. Ako je snimanje uspješno prikazuje se pretpregled fotografije koji omogućuje korisniku pregled fotografije prije pokretanja modela za prepoznavanje otpada. Korisnik također ima mogućnost odbaciti fotografiju ako njome nije zadovoljan. Ukoliko je model uspješan u pronalasku objekata, utoliko se prelazi na zaslon za prikaz rezultata.

Dodatno se na zaslonu mogu naći gumb za odabir postojeće fotografije iz memorije uređaja te gumb za paljenje, odnosno gašenje, svjetiljke uređaja. U gornjem lijevom kutu nalaze se gumbi za prelazak na generalne postavke ili zaslon korisničkog računa. Prijelazi su popraćeni transformacijskim animacijama.

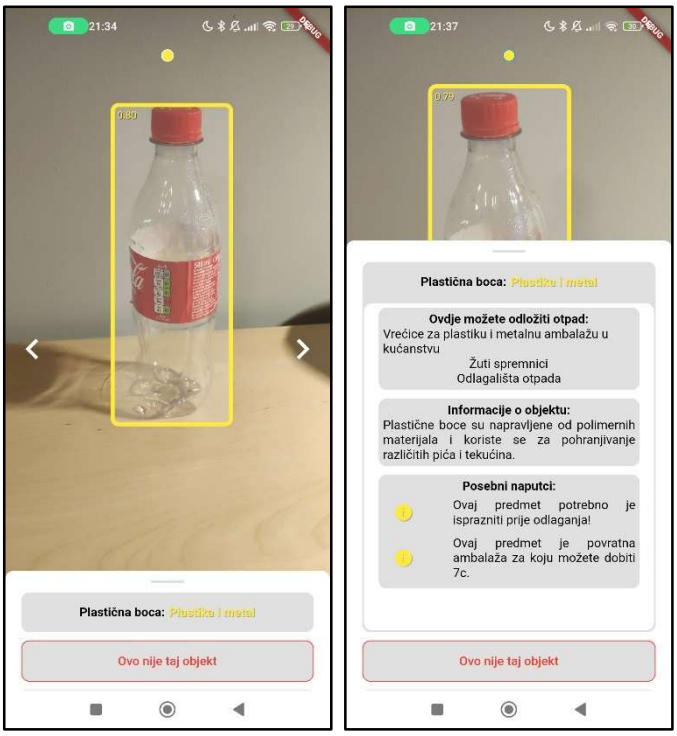
4.1.3.1. Prikaz rezultata

Glavni dio zaslona za prikaz rezultata čini pregled odabrane fotografije s prepoznatim objektima, na kojoj je u svakom trenutku odabran isključivo jedan prepoznati objekt. Broj prepoznatih objekata jednim je dijelom ograničen s postavljenom preciznošću prepoznavanja, koja dopušta prikaz isključivo predmeta u čije je prepoznavanje model dovoljno siguran. Granica je standardno 0.5, odnosno 50 posto. Trenutno odabrani prepoznati objekt može se raspoznati po osvjetljenju i obojenom okviru, čija boja odgovara kategoriji otpada u koju objekt pripada. Dodatno, u gornjem lijevom kutu okvira može se vidjeti decimalni zapis sigurnosti modela u prepoznavanju trenutnog objekta. Uz sredinu gornjeg ruba ekrana može se pronaći niz uokvirenih krugova, svaki od kojih odgovara jednom prepoznatom objektu. Onaj koji odgovara trenutno odabranom objektu po boji se poklapa s okvirom objekta. Pritiskom na određeni krug trenutno odabrani objekt postaje onaj kojem odgovara krug. Uz centar lijevog i desnog ruba dodatno se mogu pronaći strelice za cikličnu navigaciju na prethodni ili sljedeći objekt.



Slika 4.6 Zaslon prikaza rezultata

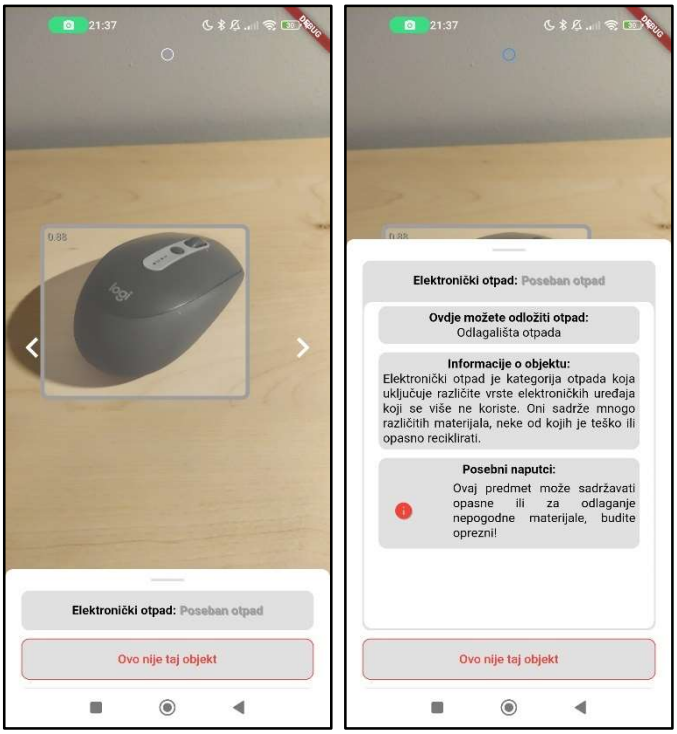
Uz donji rub zaslona nalazi se fleksibilna traka koja pomoću dva gumba pruža korisniku mogućnost interakcije s informacijama o trenutno odabranom objektu. Gornji prikazuje naziv prepoznatog objekta kao i obojenu kategoriju otpada koja mu odgovara. Pritiskom na gumb otvara se komponenta s dodatnim informacijama.



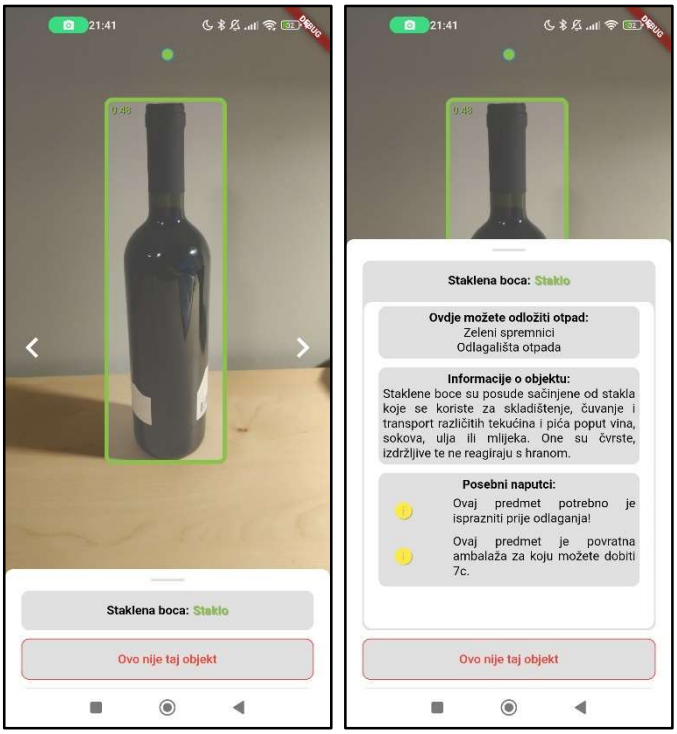
Slika 4.7 Primjer prikaza rezultata za plastičnu bocu

Unutar komponente mogu se pronaći svi poznati podaci o objektu, koji se pronalaze u već opisanim strukturama za spremanje pravila odlaganja za određenu lokaciju i vrstu otpada. Pokazna inačica nudi uvid u mogućnosti odlaganja otpada, generalne informacije te popis dodatnih naputaka koji opisuju posebne značajke objekta, poput mogućnosti povrata ambalaže ili potrebe za pražnjenjem prije odlaganja.

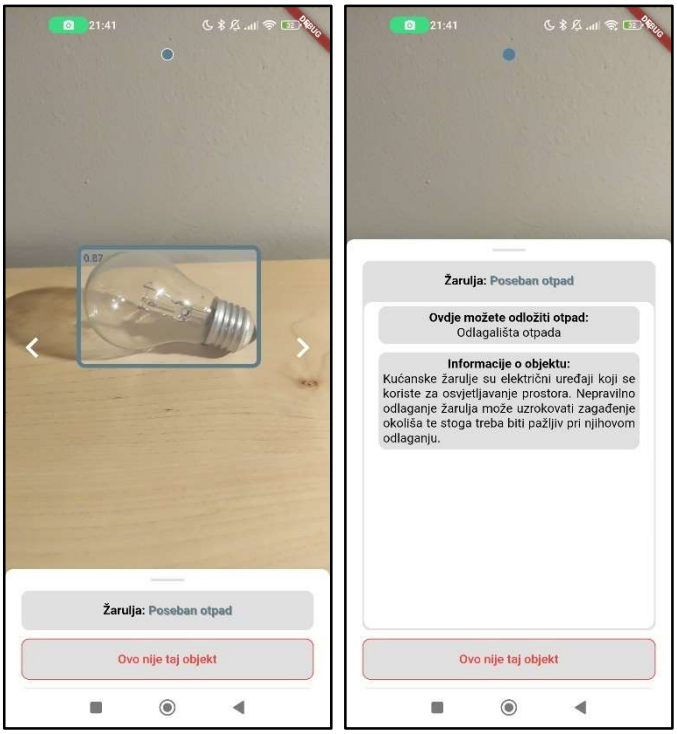
U nastavku su dani primjeri različitih prepoznatih objekata te podataka koje aplikacija o njima može pružiti.



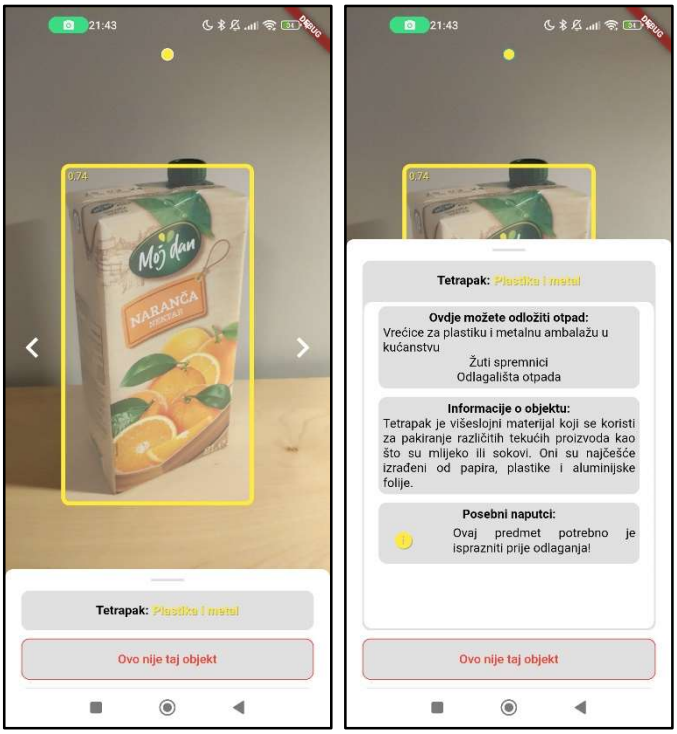
Slika 4.8 Primjer prikaza rezultata za elektronički otpad



Slika 4.9 Primjer prikaza rezultata za staklenu bocu



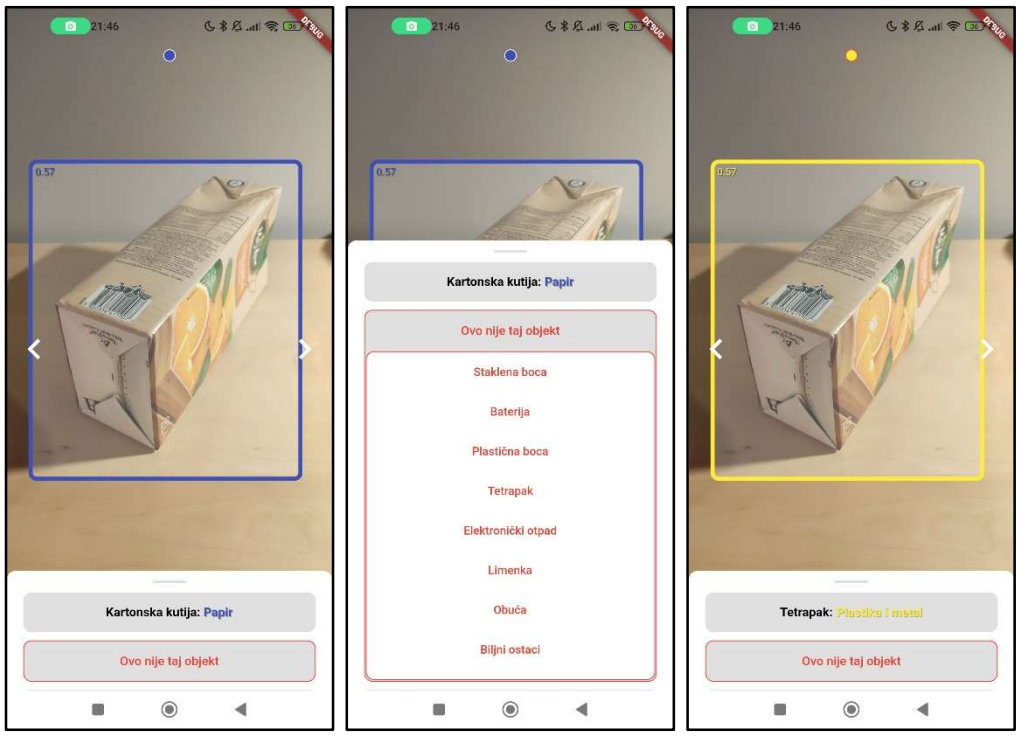
Slika 4.10 Primjer prikaza rezultata za žarulju



Slika 4.11 Primjer prikaza rezultata za tetrapak

4.1.3.2. Povratne informacije

Donji gumb na fleksibilnoj traci korisniku pruža mogućnost ispravka krivo prepoznatog objekta, čime se stvara povratna informacija o uspješnosti modela. Pritiskom na gumb otvara se komponenta za odabir ispravnog objekta. Pritiskom na neki od ponuđenih alternativa stanje trenutno prepoznatog objekta se mijenja te se prikazuju informacije za onu kategoriju otpada kojoj pripada novoodabrani objekt.



Slika 4.12 Primjer stvaranja povratne informacije

Interakcijom s bilo kojim od dva gumba bilježi se povratna informacija o uspješnosti modela što se može prepoznati po boji kruga uz centar gornjeg ruba zaslona koji odgovara trenutno odabranom objektu. Ako je krug bijele boje smatra se kako korisnik nije dao sud o ispravnosti prepoznavanja. Ako je krug crven korisnik je zamijenio prepoznati objekt nekim drugim, zbog čega se smatra kako je prepoznavanje neispravno. Na kraju, ako je korisnik pregledao informacije o objektu, krug je plave boje te se smatra kako je prepoznavanje ispravno. Pri zamišljenom radu aplikacije sve povratne informacije se lokalno bilježe i prvom mogućom prilikom šalju na server.

4.1.4. Modularni zasloni

Modularni ekrani sastoje se od niza izmjenjivih komponenti, odnosno modula, svaki od kojih služi zasebnoj svrsi. Takvim pristupom moguće je vrlo jednostavno prilagoditi određeni zaslon novim potrebama stvaranjem novih modula.

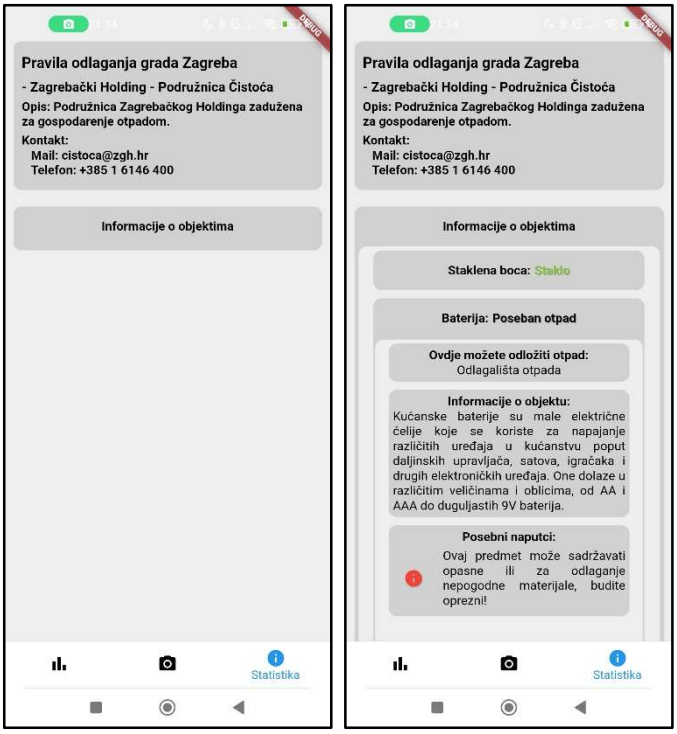
4.1.4.1. Statistika

Zaslon za statistiku zamišljen je kao niz modula za uvid u korisne podatke koji se prikupljaju upotrebom aplikacije. Svaki modul računao bi zasebnu metriku koja se može izračunati iz zapisanih podataka. Pokazna inačica aplikacije posjeduje modul za prikaz količine recikliranog otpada u svakoj prepoznatoj kategoriji, u obliku stupčastog dijagrama. Vrijednosti su masa izračunata zbrajanjem predodređenih, srednjih masi svakog prepoznatog objekta unutar pojedine kategorije. Koristan paket koji omogućuje stvaranje raznih vrsta dijagrama i drugih prikaza zove se *fl_chart*.

Slika 4.13 Zaslon za statistiku

4.1.4.2. Dodatne informacije

Zaslon za dodatne informacije sastoji se od niza modula čija je namjena prikaz svih dostupnih informacija, po potrebi bez fotografiranja objekta. Trenutno su implementirana dva modula, jedan koji prikazuje generalne informacije o autoru pravila reciklaže koja se trenutačno koriste te drugi koji se pritiskom širi u prikaz informacija o svim dostupnim objektima, koje su prikazane na sličan način kao u traci na zaslonu za prikaz rezultata.



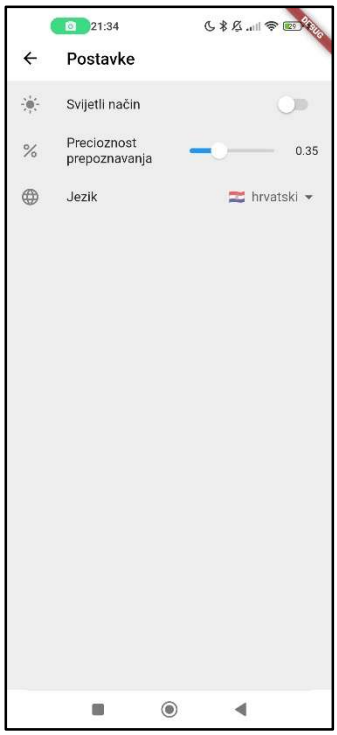
Slika 4.14 Zaslon za dodatne informacije

4.1.5. Pomoćni zasloni

Pomoćni ekrani korisniku pružaju pristup dodatnim značajkama kojima nije mjesto u sklopu glavnom sučelja.

4.1.5.1. Postavke

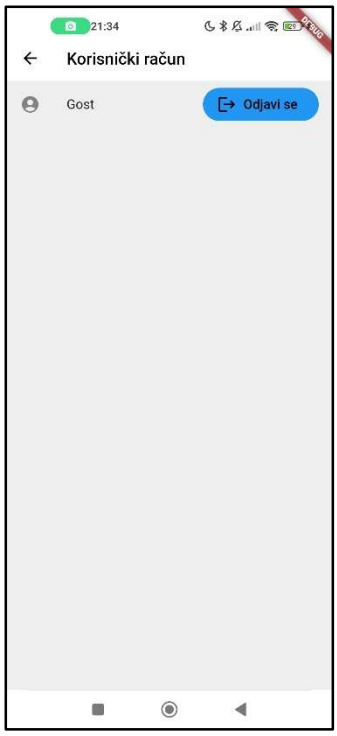
Zaslon s postavkama sadrži listu mjenjača za sve dostupne promjenjive vrijednosti. Korisnik ima mogućnost postaviti svjetlosni način rada aplikacije, preciznost prepoznavanja i prikazni jezik. Kako bi se osigurala dugotrajnost postavki, odnosno kako se ne bi promijenile prilikom ponovnog pokretanja aplikacije, koristi se paket *shared_preferences* koji implementira spremnik male veličine za spremanje podataka oblika ključ – vrijednost. Nadalje, u svrhu podrške više jezika upotrebljeni su paketi *flutter_localizations* i *intl*, standardni paketi za lokalizaciju koji omogućavaju zapisivanje prijevoda u *JSON* formatu.



Slika 4.15 Zaslon s postavkama

4.1.5.2. Korisnički račun

Zaslon korisniku daje uvid u informacije i postavke računa. Kako je pokazna inačica aplikacije namijenjena radu bez korisničkog računa sučelje pruža samo mogućnost izlaska u zaslon za odabir korisničkog računa čime se brišu spremljene postavke.



Slika 4.16 Zaslon korisničkog računa

4.2. Točnost modela umjetne inteligencije

Iako smo se trudili istrenirati što točniji model, u svrhu razvoja aplikacija točnost nije bila presudna. Zbog mogućnosti koje aplikacija nudi, korisnik može ispraviti eventualne greške modela ako mu se ponudi dovoljno dobar okvir prepoznavanja u kojem trenutku model s nižom točnošću također postaje zadovoljavajuć, sve dok može prepoznati objekte u većini slučajeva. Za procjenu točnosti modela dostupne su razne metrike, objašnjenja kojih su dana u sklopu poglavlja 3.2.4.

```
{'AP': 0.49347013, 'AP50': 0.87950885, 'AP75': 0.5165194, 'APs':
0.08593609, 'APm': 0.3356199, 'APl': 0.52431226, 'ARmax1': 0.30946988,
'ARmax10': 0.57430166, 'ARmax100': 0.604058, 'ARs': 0.13333334, 'ARm':
0.42556253, 'ARl': 0.67657936, 'AP_/Glass bottle': 0.5917348,
'AP_/battery': 0.4299941, 'AP_/plastic bottle': 0.6206493,
'AP_/tetrapak': 0.419264, 'AP_/limenka': 0.40069044, 'AP_/zarulja':
0.56184113, 'AP_/elektronika': 0.37503603, 'AP_/carton box': 0.6327981,
'AP_/biljni ostaci': 0.5197902, 'AP_/obuca': 0.38290328}
```

Slika 4.17 Rezultat 1; Metrike za TensorFlow model prije prebacivanja formata

```
{'AP': 0.47889203, 'AP50': 0.8604406, 'AP75': 0.5009307, 'APs':
0.07067784, 'APm': 0.32837573, 'APl': 0.5127623, 'ARmax1': 0.29425383,
'ARmax10': 0.5332125, 'ARmax100': 0.5424429, 'ARs': 0.11666667, 'ARm':
0.40018314, 'ARl': 0.5982976, 'AP_/Glass bottle': 0.5793274,
'AP_/battery': 0.39227057, 'AP_/plastic bottle': 0.57554454,
'AP_/tetrapak': 0.4084113, 'AP_/limenka': 0.39599088, 'AP_/zarulja':
0.57755774, 'AP_/elektronika': 0.38260752, 'AP_/carton box': 0.59420085,
'AP_/biljni ostaci': 0.50132483, 'AP_/obuca': 0.3816845}
```

Slika 4.18 Rezultat 2; Metrike za TensorFlow Lite model nakon prebacivanja formata

Metrike navedene pod *Rezultat 1* predstavljaju vrijednosti za trenirani model formata *TensorFlow*, koji predstavlja nužan korak u stvaranju *TensorFlow Lite* modela transformacijom formata. Zbog procesa kvantizacije (postupak kojim se smanjuje preciznost brojeva koji predstavljaju parametre modela u svrhu ubrzanja i smanjenja prostora) te smanjenja broja maksimalnih detekcija i promjene načina odstranjivanja preklapajućih detekcija, dolazi do gubitka preciznosti pri promjeni formata. Iz ovih vrijednosti možemo doći do nekoliko zaključaka:

- preciznost modela je malo niža od 50 posto, što je prihvatljiva vrijednost zbog okruženja u kojem će se izvršavati detekcija i mogućnosti ispravljanja grešaka od strane korisnika
- model je vrlo neprecizan nad malim objektima, odnosno preciznost modela je proporcionalna veličini objekta, isti trend se može vidjeti i za odziv
- najzahtjevniji objekti za prepoznavanje su baterija, tetrapak, elektronika, limenka te obuća, iz čega zaključujemo kako bismo trebali promijeniti skup podataka tih objekata kako bi se povećala preciznost
    - problem prilikom prepoznavanja baterija bi mogao biti što su najčešće relativno male, a model ima problema pri prepoznavanju malih objekata, uz što sliče limenkama
    - tetrapak bi mogao biti problematičan zbog sličnosti s kartonskim kutijama i manjeg broja podataka u našem skupu podataka
    - elektronika je možda pre opširna kategorija, tako da bez puno većeg seta podataka nećemo moći doći do mnogo boljih rezultata, također, ovi rezultati nisu jako reprezentativni, jer je naš skup podataka ograničen, zbog čega će model teško prepoznati dosad neviđeni elektronički otpad
    - za obuću je vjerojatno potrebno prilagoditi skup podataka kako bi se dodatno poboljšala preciznost, specifično, jedna od testnih slika sadrži mnogo objekata koji su teže prepoznatljivi, što također utječe na preciznost
    - prepoznavanje limenke je najvjerojatnije problematično zbog manjeg skupa podataka u odnosu na druge kategorije te dijeljenju raznih osnovnih značajka poput oblika i boja s baterijama
- može se primijetiti kako se većina metrika pogoršalo pri prelasku na *TensorFlow Lite* format, što nije zabrinjavajuće jer je većina promjena dovoljno mala, a unutar nekih kategorija je čak došlo do poboljšanja

Iz pregledanih metrika i dodatnog testiranja zaključili smo kako je izvedeni model zadovoljavajuć za naše potrebe i primjenjiv u sklopu aplikacije, no u budućnosti bi se skup podataka trebao poboljšati kako bi se postigao efikasniji rad.

4.3. Upute za upotrebu

Iako je razvojni okvir *Flutter* pogodan za razvoj mobilnih aplikacija na različitim uređajima, razvoj *iOS* aplikacija zahtjeva posjedovanje raznih resursa kojima mi nemamo pristup, zbog čega je aplikaciju trenutno moguće koristiti isključivo na mobilnim uređajima operacijskog sustava *Android*. Paket za instalaciju (*PravaVrecica.apk*) se može besplatno preuzeti s naše *Google drive* mape navedene u korisnim poveznicama. Korisnik treba preuzeti dotičnu datoteku na svoj mobilni uređaj te je spremiti na uređaju vidljivo mjesto. Datoteku je zatim potrebno pronaći i pokretanjem započeti instalaciju.

Sustav *Android* standardno sprječava instalaciju aplikacija putem paketa za instalaciju, zbog čega je moguće kako će korisnik biti upozoren prilikom pokušaja instalacije. U toj obavijesti često je moguće pronaći vezu koja vodi do relevantnih postavki, gdje je moguće omogućiti instalaciju. Nakon instalacije, zbog sigurnosti uređaja, preporučujemo promijenjene postavke vratiti na prvobitno stanje.

Ako je aplikacija uspješno instalirana korisnik je može koristiti sukladno načinu opisanom u pregledu sučelja.

5. Zaključak

Mobilna aplikacija *Prava Vrećica* sama po sebi ne predstavlja konačno rješenje problema i nedoumica s kojima se ljudi susreću pri odlaganju otpada, no vjerujemo kako može poslužiti kao informativni alat i pomoć pri pravilnom odlaganju otpada. Smatramo kako aplikacija može poslužiti u podizanju svijesti o pravilima reciklaže i odlaganja, kako među ljudima koji su već upoznati s problemom, tako i među onima koji nisu, jer pristupa rješavanju tog problema na novi i inovativan način. Iako ne možemo sa sigurnošću reći kako će većina reagirati na aplikaciju ovog tipa te koliko će značiti za pojedinačnog korisnika, vjerujemo kako može biti primjer za razvoj sličnih alata kojima se može podići informiranost građana, posebno u gradu Zagrebu gdje je problem razvrstavanja otpada često u javnom fokusu.

Nadalje, smatramo kako je aplikacija dobar okvir kroz koji se može proučiti opravdanost primjene umjetne inteligencije za rješavanje konkretnih problema. Također smatramo kako prepoznavanje otpada i njegova kategorizacija nisu korisni isključivo pojedincima već se mogu primijeniti u različitim područjima, poput obrazovanja ili industrijskog razvrstavanja otpada. Na kraju, zadovoljni smo postignutim rezultatima i razvijenim modelom te smatramo kako su opravdali stvaranje ovakvog rada.

6. Literatura, razvojni paketi i korisne poveznice

6.1. Literatura

[1] The Recycling Partnership, »Recycling Confidence Index,« The Recycling Partnership, August 2022. [Mrežno]. Available: https://recyclingpartnership.org/first-of-its-kind-recycling-confidence-index-not-surprisingly-confidence-is-heavily-impacted-by-communication-and-support/. [Pokušaj pristupa 4 2023].

[2] P. W. Schultz, A field experiment on interventions to improve curbside recycling, The Claremont Graduate University, 1995.

[3] M. D. S. W. X. H. P. Z. a. Z. Z. Zhuoling Li, »CLU-CNN: Object detection for medical images,« u *Neurocomputing*, 2019, pp. 53-59.

[4] S. D. R. G. A. F. Joseph Redmon, *You Only Look Once: Unified, Real-Time Object Detection,* 2015.

[5] D. A. D. E. C. S. S. R. C.-Y. F. A. C. B. Wei Liu, *SSD: Single Shot MultiBox Detector,* 2015.

[6] J. D. T. D. J. M. Ross Girshick, *Rich feature hierarchies for accurate object detection and semantic segmentation,* 2013.

[7] R. P. Q. V. L. Mingxing Tan, *EfficientDet: Scalable and Efficient Object Detection,* 2019.

[8] N. D. A. D. a. E. P. Athanasios Voulodimos, »Deep Learning for Computer Vision: A Brief Review,« *Computational Intelligence and Neuroscience,* svez. 2018, 2018.

[9] A. Y. Mingxing Tan, *EfficientDet: Towards Scalable and Efficient Object Detection,* AI Google Blog, 2020.

[10] B. Pang, E. Nijkamp i . Y. N. Wu, »Deep learning with tensorflow: A review,« *Journal of Educational and Behavioral Statistics,* svez. 45, br. 2, pp. 227-248, 2020.

[11] K. Maeda, »Performance evaluation of object serialization libraries in XML, JSON and binary formats,« u *2012 Second International Conference on Digital Information and Communication Technology and it's Applications (DICTAP)*, 2012.

[12] G. Bracha, The Dart programming language, Addison-Wesley Professional, 2015.

[13] A. Tashildar, N. Shah, R. Gala, T. Giri i P. Chavhan, »Application development using flutter,« *International Research Journal of Modernization in Engineering Technology and Science,* svez. 2, br. 8, pp. 1262-1266, 2020.

[14] M. Masse, REST API design rulebook: designing consistent RESTful web service interfaces, O'Reilly Media, Inc., 2011.

[15] S. Dauzon, A. Bendoraitis i A. Ravindran, Django: web development with Python, Packt Publishing Ltd, 2016.

6.2. Razvojni paketi

Paket *tflite_flutter* – https://pub.dev/packages/tflite_flutter

Paket *tflite_flutter_helper* – https://pub.dev/packages/tflite_flutter_helper

Paket *provider* – https://pub.dev/packages/provider

Paket *shared_preferences* – https://pub.dev/packages/shared_preferences

Paket *camera* – https://pub.dev/packages/camera

Paket *animations* – https://pub.dev/packages/animations

Paket *fl_chart* – https://pub.dev/packages/fl_chart

Paket *flutter_localization* i paket *intl* –

https://docs.flutter.dev/development/accessibility-and-localization/internationalization

Paket *flutter_native_splash* – https://pub.dev/packages/flutter_native_splash

Paket *flutter_launcher_icons* – https://pub.dev/packages/flutter_launcher_icons

6.3. Korisne poveznice

Poveznica na alat *Roboflow* – https://roboflow.com/

Poveznica na alat *PythonAnywhere* – https://www.pythonanywhere.com/

Poveznice na internetske stranice *podružnice Čistoće* – https://www.cistoca.hr/

Poveznica na *Google drive* mapu gdje se može pronaći pokazna inačica aplikacije – https://drive.google.com/drive/folders/1p8Pvjge1W7SV6kHZ2elfOrdJzkzZ13w2

7. Sažetak na hrvatskom i engleskom jeziku

Unutar sljedeća dva poglavlja nalaze se sažetci na hrvatskome i engleskom jeziku.

7.1. Sažetak (Hrvatski)

Prava Vrećica - Reciklaža i razvrstavanje otpada primjenom računalnog vida

Autori: Jura Hostić, Tvrtko Puškarić, Karlo Vrdoljak

U sklopu rada razvijen je algoritam za prepoznavanje i razvrstavanje otpada prema zadanim pravilima o recikliranju u lokalnoj sredini zasnovan na umjetnoj inteligenciji, odnosno računalnom vidu. Rad algoritma se temelji na prepoznavanju različitih objekata pomoću računalnog vida, modelom umjetne inteligencije *EfficientDet*. Prepoznati otpad kategorizira se na temelju službenih pravila reciklaže i odlaganja otpada koja pružaju zadužene organizacije. Dodatno je izvedena mobilna aplikacija *Prava Vrećica* kao novi alat za pomoć pri reciklaži i okvir za proučavanje korisnosti razvijenog modela. U sklopu aplikacije korisnik do svih bitnih informacija može doći u par lakih koraka, nakon čega ima mogućnost pružiti povratne informacije o uspješnosti prepoznavanja na temelju kojih se gradi skup podataka za daljnji razvoj modela. Aplikacija uz to također nudi uvid u određene korisne statistike koje se izvode na temelju podataka prikupljenih dugotrajnom upotrebom. Aplikacija je izrađena unutar *Flutter* okvira za razvoj mobilnih aplikacija.

Ključne riječi: *Flutter* mobilna aplikacija, prepoznavanje otpada, pravila odlaganja, računalni vid, *EfficientDet*

7.2. Summary (English)

Prava Vrećica – Recycling and waste sorting using computer vision

Authors: Jura Hostić, Tvrtko Puškarić, Karlo Vrdoljak

The main focus of the project has been the development of an algorithm for the recognition and sorting of waste according to the established recycling rules in the local community, based on artificial intelligence and computer vision. The algorithm is based on the recognition of different objects using computer vision, with the “EfficientDet“ artificial intelligence model. The recognized waste is categorized based on the official recycling and waste disposal rules provided by the responsible organizations. Additionally, the mobile application "Prava Vrećica" has been developed as a new tool to assist with recycling and as a framework for studying the usefulness of the developed model. Within the application, users can access all important information in a few easy steps, after which they have the option to provide feedback on the recognition performance, which is used to build a dataset for further model development. The application also provides insights into certain useful statistics derived from data collected through long-term use. The application was developed using the „Flutter“ framework for mobile application development.

Key words: Flutter mobile application, waste recognition, waste disposal rules, computer vision, EfficientDet