Sustav za višestruku klasifikaciju i detekciju objekata

Lapov-Ugrina, Lucija

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

ZAVRŠNI RAD

Lucija Lapov-Ugrina

Zagreb, 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

ZAVRŠNI RAD

Mentor: Student:

Izv. prof. dr. sc. Tomislav Stipančić, dipl. ing. Lucija Lapov-Ugrina

Zagreb, 2024.

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

Zahvaljujem se mentoru izv. prof. dr. sc. Tomislavu Stipančiću na uloženom trudu i vremenu te pruženoj pomoći i savjetima tijekom izrade ovog rada.

Zahvaljujem se svojoj obitelji i prijateljima na podršci i razumijevanju te posebno zahvaljujem dečku Kristianu na neizmjernoj motivaciji kroz cijeli preddiplomski studij.

Lucija Lapov-Ugrina

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE
Središnje povjerenstvo za završne i diplomske ispite
Povjerenstvo za završne i diplomske ispite studija strojarstva za smjerove:
proizvodno inženjerstvo, računalno inženjerstvo, industrijsko inženjerstvo i menadžment, inženjerstvo materijala i mehatronika i robotika

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

ZAVRŠNI ZADATAK

Student: **Lucija Lapov-Ugrina** JMBAG: **0035233550**

Naslov rada na hrvatskom jeziku: **Sustav za višestruku klasifikaciju i detekciju objekata**

Naslov rada na engleskom jeziku: **System for multiple classification and object detection**

Opis zadatka:

Kod sustava strojnog vida primjena ponekad zahtjeva da su lagani, prenosivi te jednostavni za korištenje. Takvi sustavi često koriste računalne platforme namijenjene stvaranju, treniranju i implementaciji računalnih modela koji su optimizirani za uređaje na rubu mreže (engl. edge devices), kao što su mikroračunala ili IoT (engl. Internet of Things) uređaji. Inteligentni modeli mogu raditi lokalno na uređajima kako bi omogućili brze odgovore, smanjujući pritom potrebu za stalnom konekcijom na internet.

Cilj ovog rada je stvoriti lagan, ali učinkovit sustav za višestruku klasifikaciju objekata koji se može implementirati na mikroračunalu. To uključuje snimanje slika, njihovu obradu za identifikaciju objekata te izvođenje zaključivanja izravno na uređaju.

U radu je potrebno:

- odrediti hardverske i softverske komponente sustava
- prikupiti skup novih slika koje sadrže željene vrste objekata koje je potrebno detektirati
- definirati, trenirati te evaluirati model za višestruku klasifikaciju koji je sposoban identificirati više objekata unutar slike
- implementirati razvijeni model na mikroračunalu te dati kritički osvrt.

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

Zadatak zadan:	Datum predaje rada:	Predviđeni datumi obrane:
24. 4. 2024.	**2. rok (izvanredni):** 11. 7. 2024. **3. rok:** 19. i 20. 9. 2024.	**2. rok (izvanredni):** 15. 7. 2024. **3. rok:** 23. 9. – 27. 9. 2024.

Zadatak zadao:

izv. prof. dr. sc. Tomislav Stipančić

Predsjednik Povjerenstva:

prof. dr. sc. Damir Godec

SADRŽAJ

POPIS SLIKA

POPIS OZNAKA

Oznaka	Opis
F1	F1 rezultat
FN	Broj pozitivnih slučajeva pogrešno označenih kao negativni
FP	Broj negativnih slučajeva pogrešno označenih kao pozitivni
TP	Broj ispravno identificiranih pozitivnih slučajeva

SAŽETAK

Ovaj rad ima za cilj izgradnju sustava za višestruku klasifikaciju i detekciju objekata koristeći prilagođene algoritme strojnog učenja optimizirane za uređaje s ograničenim resursima poput mikrokontrolera. To znači da ovakav sustav istovremeno mora biti lagan, ali i učinkovit. U praktičnom dijelu rada koristi se mikrokontroler XIAO ESP32S3 za snimanje slika i testiranje modela te platforma Edge Impulse Studio za treniranje modela detekcije objekata. Primijenjen je FOMO model strojnog učenja, optimiziran za rad na uređajima s malom količinom memorije i niskom potrošnjom energije. Evaluacija modela obuhvaća analizu njegove točnosti i efikasnosti u uvjetima rada na uređajima s ograničenim memorijskim i procesorskim kapacitetima.

Ključne riječi: klasifikacija, detekcija objekata, strojno učenje, mikrokontroler, XIAO ESP32S3, FOMO

SUMMARY

This thesis aims to build a system for multi-object classification and detection using customized machine learning algorithms optimized for resource-constrained devices such as microcontrollers. This means that such a system must be both lightweight and efficient. In the practical part of the work, the XIAO ESP32S3 microcontroller is used for capturing images and testing the model, and the Edge Impulse Studio platform is employed for training the object detection model. The FOMO machine learning model, optimized for devices with limited memory and low energy consumption, is applied. The evaluation of the model includes an analysis of its accuracy and efficiency under conditions of limited memory and processing capacity.

Key words: classification, object detection, machine learning, microcontroller, XIAO ESP32S3, FOMO

1. UVOD

Sustavi za višestruku klasifikaciju i detekciju objekata postaju sve važniji u modernim aplikacijama, posebice u kontekstu brzorastućeg razvoja tehnologija poput interneta stvari (IoT) i računarstva na rubu mreže. Ovi sustavi omogućuju prepoznavanje i klasifikaciju više objekata unutar jedne slike, što je ključno za mnoge primjene, uključujući autonomna vozila, nadzorne sustave, industrijsku automatizaciju i pametne uređaje.

S obzirom na ograničene resurse uređaja na rubu mreže (engl. *edge devices*), kao što su mikroračunala i IoT uređaji, postoji potreba za razvojem učinkovitih algoritama za detekciju i klasifikaciju objekata koji mogu raditi lokalno na samom uređaju. Takvi sustavi moraju biti optimizirani kako bi pružili brze odgovore i smanjili potrebu za stalnom vezom s internetom ili moćnijim računalnim sustavima. To omogućuje ne samo veću autonomiju uređaja, već i smanjenje energetskih troškova i opterećenja mreže.

U ovom radu istražuje se kako izgraditi takav sustav koristeći modele strojnog učenja koji su prilagođeni za uređaje s ograničenim procesorskim kapacitetima i memorijskim resursima.

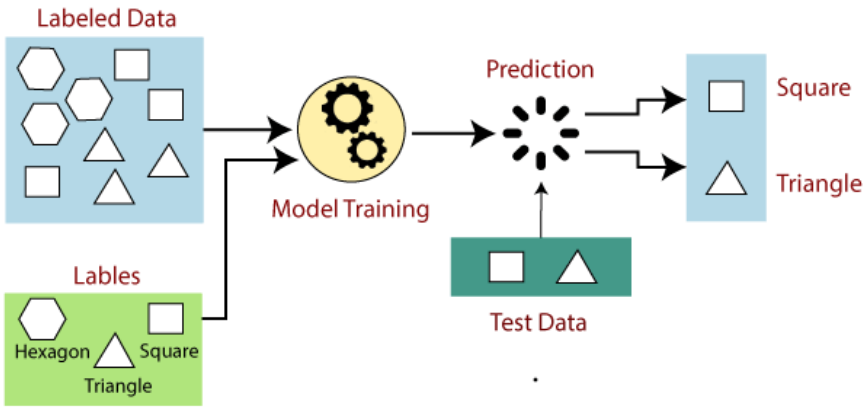
2. STROJNO UČENJE

Strojno učenje je grana umjetne inteligencije koja se bavi razvojem algoritama i modela koji omogućuju računalima da uče iz podataka i donose odluke bez eksplicitnog programiranja. [1] Algoritmi strojnog učenja koriste prethodne podatke kako bi poboljšali svoje performanse i preciznost u predviđanju novih izlaznih vrijednosti. Glavni cilj strojnog učenja je omogućiti računalima da samostalno poboljšavaju svoje rezultate i prilagođavaju se novim zadacima na temelju iskustava, što može biti prednost u odnosu na tradicionalne metode, ali također može otežati otkrivanje i ispravljanje grešaka ako rezultati nisu zadovoljavajući. Strojno učenje generalno se može podijeliti u četiri osnovne skupine:

- Nadzirano učenje
- Nenadzirano učenje
- Polunadgledano učenje
- Podržano učenje

2.1. Nadzirano učenje

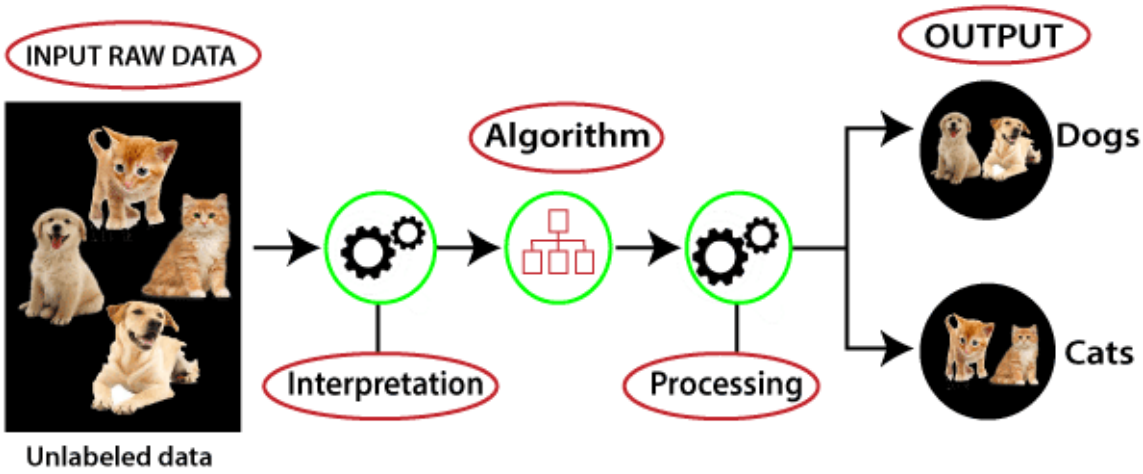
Nadzirano učenje (engl. *supervised learning*) je vrsta strojnog učenja u kojoj se algoritmu pružaju ulazni podaci zajedno s pripadajućim izlaznim oznakama. Ovaj pristup omogućuje algoritmu da razvije prediktivne modele ili donese odluke na temelju označenih podataka. Algoritam uči funkciju koja preslikava ulazne podatke na odgovarajuće izlaze, koristeći unaprijed definirane ulazno-izlazne parove. Nakon što je proces učenja završen, algoritam se testira na novim, neviđenim podacima kako bi se procijenila njegova preciznost. Primjeri nadziranog učenja uključuju klasifikaciju i regresiju, a koristi se u raznim stvarnim aplikacijama poput procjene rizika, klasifikacije slika, otkrivanja prijevara i filtriranja neželjene pošte. Ova metoda omogućuje predviđanje ispravnih izlaza na temelju označenih ulaznih podataka, slično kao što učenik uči pod nadzorom učitelja.



Slika 1. Prikaz nadgledanog strojnog učenja [2]

2.2. Nenadzirano učenje

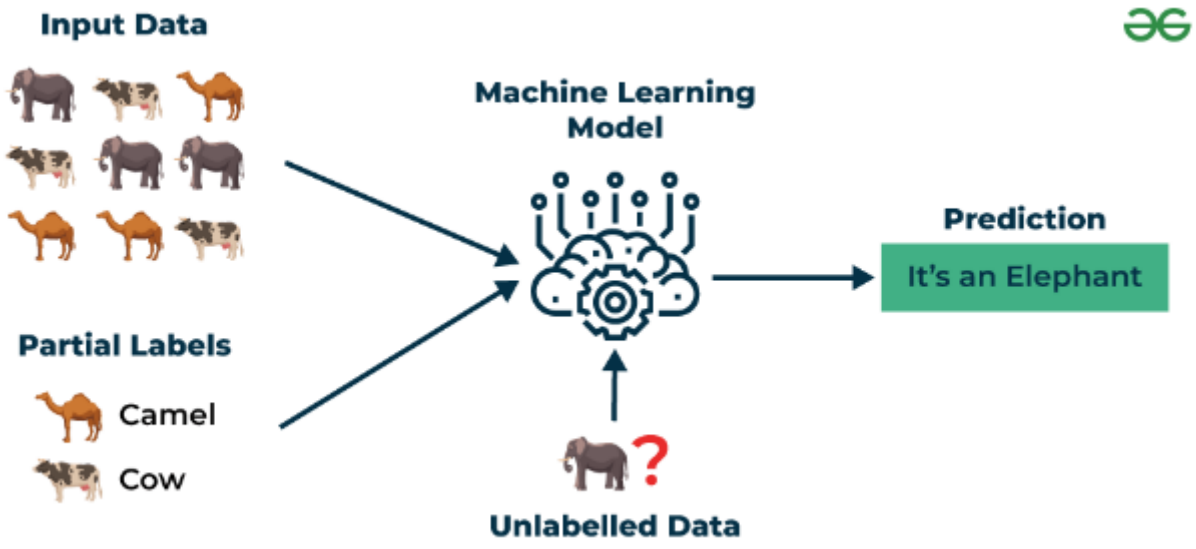
Nenadzirano učenje (engl. *unsupervised learning*) je vrsta strojnog učenja u kojoj algoritmi analiziraju neoznačene podatke kako bi otkrili skrivene uzorke, strukture i povezanosti bez prethodnih oznaka ili nadzora. Ovaj pristup omogućuje modelima da sami prepoznaju sličnosti i razlike u podacima, što je složenije od nadziranog učenja zbog nedostatka poznatih izlaznih podataka. Cilj nenadziranog učenja je identificirati unutarnju strukturu podataka, grupirati ih prema sličnostima, smanjiti dimenzionalnost te generirati nove uvide ili hipoteze. Ova metoda se često koristi za klasteriranje podataka, analizu asocijacija, otkrivanje anomalija i komprimiranje podataka. Algoritmi se oslanjaju na sposobnost prepoznavanja uzoraka i strukturalnih karakteristika u ulaznim podacima za generiranje korisnih rezultata.



Slika 2. Primjer nenadziranog strojnog učenja [3]

2.3. Polunadgledano učenje

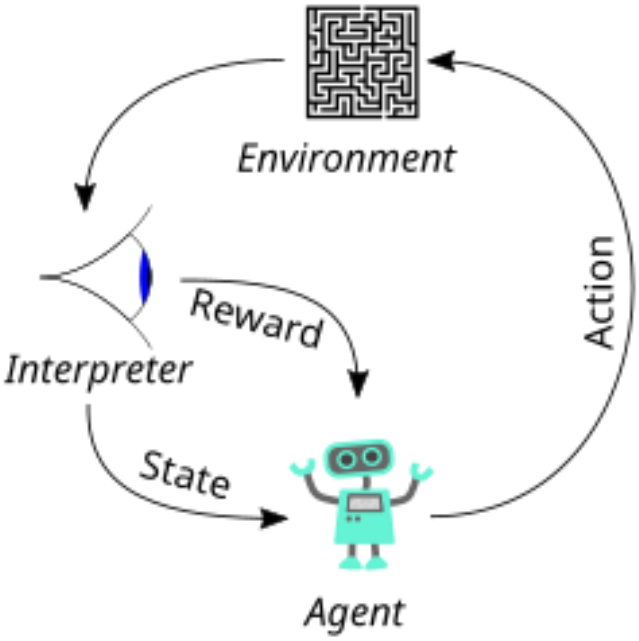
Polunadgledano učenje (engl. *semi-supervised learning*) je pristup strojnog učenja koji kombinira elemente nadziranog i nenadziranog učenja koristeći kako označene tako i neoznačene podatke. Ova metoda je posebno korisna kad je dobivanje označenih podataka skupo, vremenski zahtjevno ili zahtijeva značajne resurse. Označeni podaci koriste se za početnu obuku, dok neoznačeni podaci pomažu u poboljšanju modela istraživanjem dodatnih uzoraka i struktura u skupu podataka. Ovaj pristup omogućuje modelima da predviđaju oznake na temelju neoznačenih podataka i zatim koriste te oznake za daljnje učenje, što je posebno primjenjivo kod skupova podataka poput slika, gdje često nije svaka slika označena.



Slika 3. Primjer polunadgledanog strojnog učenja [5]

2.4. Podržano učenje

Podržano učenje (engl. *reinforcement learning*) je vrsta strojnog učenja koja se usredotočuje na obuku agenata da donose odluke s ciljem maksimiziranja ukupne nagrade u dinamičnom i potencijalno složenom okruženju. Za razliku od nadziranog učenja, koje koristi označene ulazno-izlazne podatke, podržano učenje temelji se na učenju kroz iskustvo. Agent uči kako ostvariti cilj obavljanjem radnji i primanjem povratnih informacija u obliku nagrada ili kazni. Ovaj pristup ne zahtijeva eksplicitne ispravke za loše radnje, već se fokusira na balansiranje između istraživanja novih mogućnosti i eksploatacije postojećeg znanja. Cilj je maksimizirati dugoročnu nagradu kroz interakciju s okolinom, a metoda se često primjenjuje u područjima poput teorije igara i optimizacije na temelju simulacija.



Slika 4. Primjer podržanog strojnog učenja [6]

2.5. TinyML

TinyML (*Tiny Machine Learning*) je grana strojnog učenja koja se fokusira na primjenu malih i efikasnih modela strojnog učenja na uređaje s ograničenim resursima, poput mikrokontrolera. Ovi uređaji često imaju vrlo malo procesorske snage i memorije pa je TinyML optimiziran za rad u takvim uvjetima, omogućujući obradu podataka i donošenje odluka lokalno, bez potrebe za slanjem podataka na moćnije servere. To rezultira brzim odgovorima u stvarnom vremenu i smanjenom potrebom za stalnim povezivanjem na internet.

3. FOMO

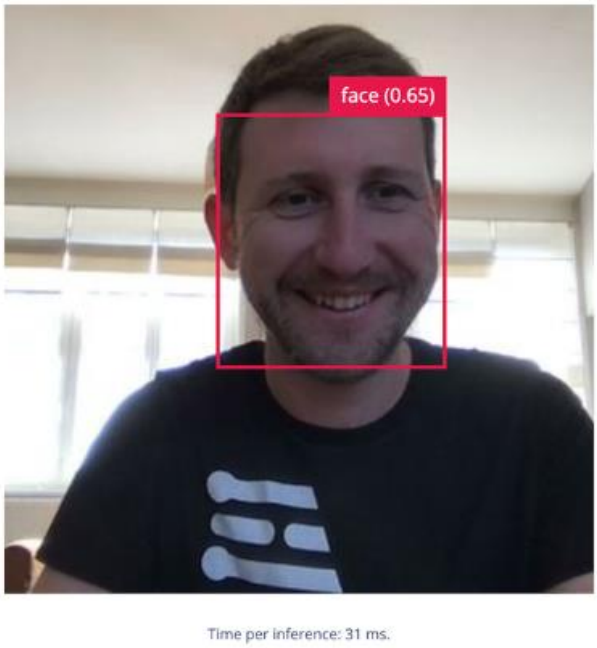
FOMO (*Faster Objects, More Objects*) je model strojnog učenja za detekciju, praćenje i brojanje objekata u stvarnom vremenu razvijen u inženjerskoj kompaniji Edge Impulse. Posebno je dizajniran za rad na uređajima s ograničenim resursima poput mikrokontrolera jer ovaj model je radi s manje od 200 KB RAM-a. Zbog svoje brzine i učinkovitosti u prepoznavanju i detekciji objekata, FOMO je posebno prikladan za TinyML okruženje, gdje su zahtjevi za procesorskom snagom i memorijom minimalni.

Dva najčešća problema u obradi slika su klasifikacija slika i detekcija objekata. Klasifikacija slika uzima sliku kao ulazni podatak te kao izlaz vraća informaciju o vrsti objekta koji se nalazi na slici. Ova tehnika pokazuje odlične rezultate, čak i na mikrokontrolerima, pod uvjetom da je potrebno detektirati samo jedan objekt na slici. Kao što je vidljivo na slici 4., pitanje na koje model pokušava odgovoriti je: "Postoji li lice na slici ili ne?" što je primjer klasifikacije slike. Ovaj jednostavniji pristup identificira prisutnost lica, vraćajući rezultat "lice" ili "nema lica".

Slika 5. Klasifikacija slike [7]

Detekcija objekata obrađuje sliku i vraća informacije o vrsti, broju objekata, njihovoj poziciji te veličini unutar slike. Budući da modeli za detekciju objekata donose složenije odluke od modela za klasifikaciju objekata, često su veći po broju parametara te zahtijevaju više podataka

za treniranje. Zbog toga se rijetko koriste na mikrokontrolerima. Na slici 5. vidljivo je da model pokušava detektirati objekt, to jest odgovoriti na pitanje koje glasi: "Ima li lica na slici, gdje se nalaze i koje su veličine?". Ova metoda je korisna u situacijama kada je potrebno prepoznati više objekata ili njihovu preciznu poziciju, ali zahtijeva puno više računalnih resursa i veći skup podataka.



Slika 6. Detekcija objekta [7]

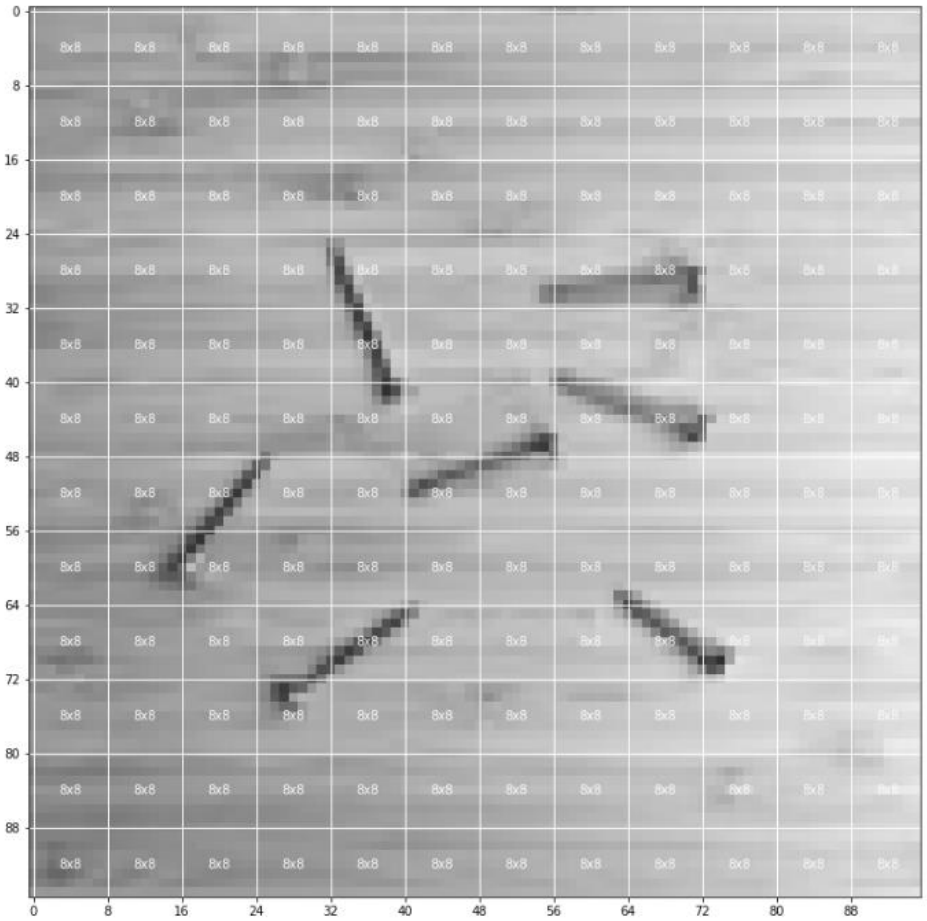
FOMO model nudi pojednostavljenu verziju detekcije objekata koja je prikladna za mnoge slučajeve upotrebe gdje je potrebno odrediti poziciju objekata na slici, ali gdje se ne može koristiti veliki ili složeni model zbog ograničenih resursa uređaja.

3.1. Princip rada

Princip rada ovog modela temelji se na ideji da mnogi problemi detekcije objekata ne zahtijevaju informacije o veličini objekata, već samo njihovu lokaciju u kadru. Kad je lokacija objekata poznata, mogu se postavljati daljnja pitanja poput: "Je li objekt iznad ili ispod drugog objekta?" ili "Koliko objekata je vidljivo?". Zbog ove pretpostavke, klasični *bounding box* okviri, pravokutni okviri koji se koristi u računalnom vidu za označavanje područja na slici u kojem se nalazi objekt kako bi se odredile njegove koordinate i veličina, više nisu potrebni. Umjesto toga, dovoljno je vršiti detekciju na temelju centroida objekata. No, radi zadržavanja

interoperabilnosti s drugim modelima, ulazne slike za treniranje i dalje koriste *bounding box* okvire.

FOMO funkcionira tako da mapira ulazne podatke, odnosno ulazne slike u nijansama sive u mrežu. Zadana veličina mreže je 8 x 8 piksela, što znači da će za sliku od 96 x 96 piksela izlaz biti 12 x 12, dok će za sliku od 320 x 320 piksela izlaz biti 40 x 40. [7] Zatim FOMO pokreće klasifikator kroz svaki blok piksela kako bi izračunao vjerojatnost nalazi li se unutar njega određeni objekt te određuje regije s najvećom vjerojatnošću prisutnosti objekta. Time mrežu za klasifikaciju slike pretvara u mapu vjerojatnosti klase po regiji. Ako blok nema objekte, klasificira se kao pozadina. Preklapanjem konačnih regija, FOMO izračunava koordinate centroida tih regija u odnosu na dimenzije slike, odnosno mapira ih na originalnu sliku prema omjeru skaliranja. Nakon dobivanja većeg broja koordinata koje izgledaju valjano, smatra se da su na tim mjestima objekti. Proces klasifikacije se paralelno izvršava za sve ćelije mreže neovisno.

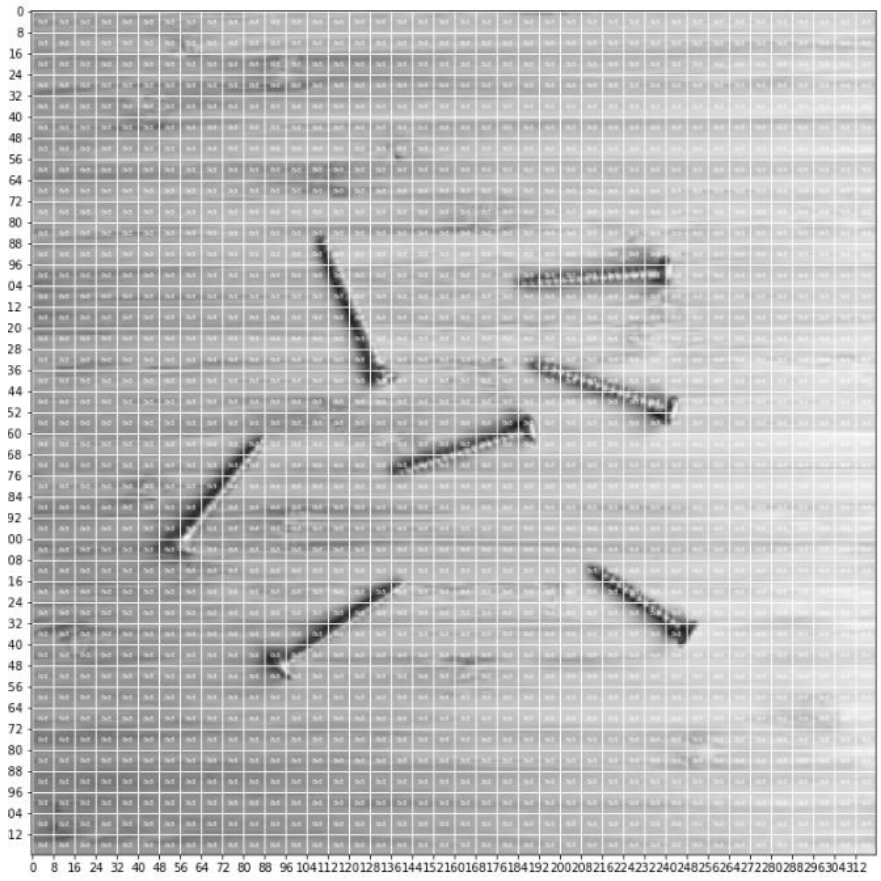


Slika 7. 96 x 96 slika podijeljena na 12 x 12 mrežu

Na slici 7. prikazano jek kako je slika na kojoj se nalaze sedam vijaka podijeljena na 12 x 12 mrežu pri čemu svaka ćelija označava je li nešto, u ovom slučaju, vijak ili nije.

FOMO je potpuno konvolucijski, što znači da radi s bilo kojom veličinom ulaznih podataka. Najmanja verzija FOMO modela (96 x 96, u nijansama sive) radi u manje od 100 KB RAM-a i postiže brzinu od približno 10 fps na Cortex-M4F procesoru pri 80MHz.

Cilj dizajna FOMO-a bio je kombinirati prednosti minimalnih računalnih zahtjeva jednostavne klasifikacije slika s dodatnim informacijama o lokaciji i broju objekata, što omogućuje detekciju objekata. Ovo omogućava FOMO-u rad na mikrokontrolerima s manje od 200 KB RAM-a. Dok drugi modeli koriste kvadrate za određivanje veličine objekta, FOMO se fokusira na informacije o lokaciji objekta putem koordinata njegova centroida, zanemarujući veličinu objekta.



Slika 8. 320 x 320 slika podijeljena na 40 x 40 mrežu

3.2. Ograničenja

FOMO se može smatrati modelom za detekciju objekata kod kojeg su sve ćelije kvadratnog oblika i zauzimaju 1/8 rezolucije ulazne slike što znači da pokazuje najbolje performanse kad

su svi objekti približno iste veličine. U mnogim slučajevima, poput onih s fiksnom pozicijom kamere, ovo nije problem. Objekti također ne bi trebali biti preblizu jedan drugome. Ako se detektiraju različite klase, poput "vijak" i "čavao", svaka ćelija može predstavljati samo jednu klasu: "vijak", "čavao", ili "pozadina", stoga nije moguće detektirati različite objekte ako se njihovi centri nalaze u istoj ćeliji.

4. IZRADA VLASTITOG SUSTAVA

Svi projekti strojnog učenja trebaju započeti s detaljno definiranim ciljem. Pretpostavka ovog rada je da za industrijski pogon poput pogona na slici 7 treba sortirati i prebrojati voće (naranče) i kukce (žabe). Drugim riječima, treba provesti višestruku klasifikaciju, gdje svaka slika može imati tri klase:

- Pozadina (nema objekata)
- Voće
- Kukac

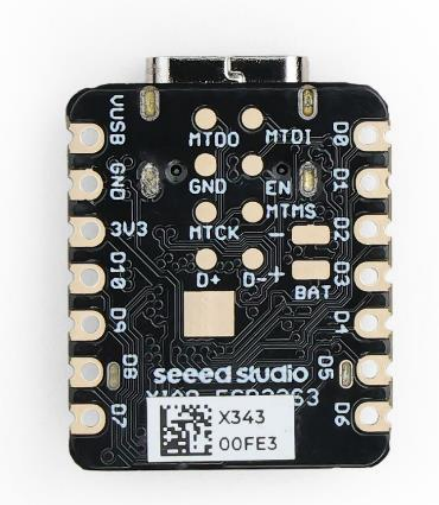
Razmatra se koji se objekt nalazi na slici, njegova lokacija (centroid) te koliko ih se može pronaći, to jest prebrojiti. Sustav je razvijen koristeći XIAO ESP32S3 za snimanje slika i provođenje inferencije modela. Projekt strojnog učenja razvijen je korištenjem Edge Impulse Studija.

Slika 9. Vizualni prikaz odabranog problema

4.1. Hardverske komponente sustava

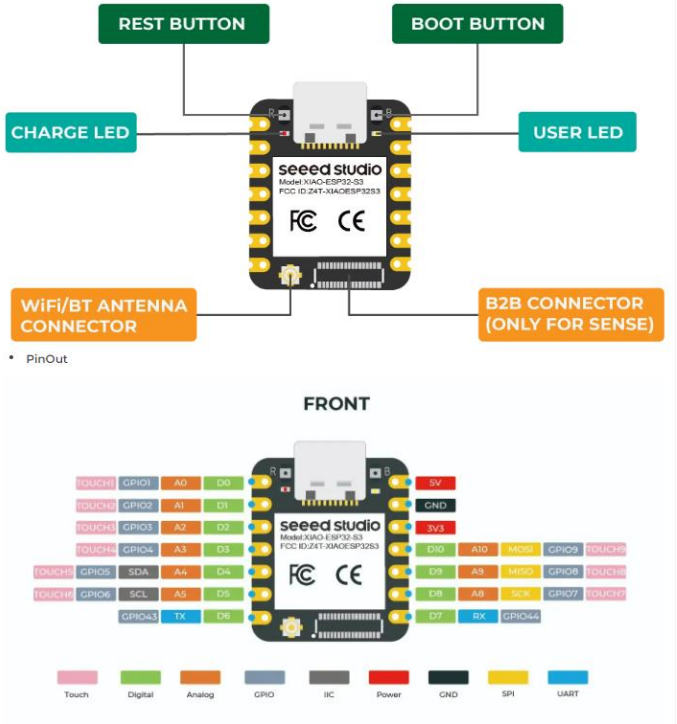
XIAO ESP32S3 Sense je kompaktan mikrokontroler optimiziran za primjene ugrađenog strojnog učenja. Opremljen je dvostrukim ESP32S3 čipom koji podržava Wi-Fi i Bluetooth

Low Energy bežičnu povezivost, s mogućnošću daljinske komunikacije na udaljenosti većoj od 100 metara uz antenu. Opremljen odvojivom kamerom koja omogućuje rezoluciju od 1600 x 1200 piksela i digitalni mikrofonom. S memorijskim kapacitetom od 8 MB PSRAM-a i 8 MB FLASH memorije, uz podršku za vanjsku SD karticu, XIAO ESP32S3 pruža dovoljno resursa za složenije aplikacije. Njegova kompaktna veličina, kombinirana s moćnim performansama, čini ga pogodnim za napredne projekte strojnog učenja u IoT okruženjima. Ovaj uređaj ima ugrađen čip za upravljanje napajanjem, što omogućuje da se napaja neovisno pomoću baterije ili da se baterija puni putem USB priključka kojim se spaja na računalo.



Slika 10. XIAO ESP32S3 mikrokontroler [9]

Slika 11. XIAO ESP32S3 mikrokontroler sa spojenom antenom i kamerom [9]



Slika 12. Raspored pinova

4.2. Softverske komponente sustava

4.2.1. Arduino IDE

Arduino IDE (Integrated Development Environment) je jednostavno i korisnički prilagođeno okruženje za razvoj koje omogućuje pisanje, uređivanje i učitavanje koda na Arduino ploče i kompatibilne mikrokontrolere. Podržava više programskih jezika, s posebnim naglaskom na C i C++. Ovaj softver omogućuje brzo prebacivanje koda na uređaj putem USB-a te dolazi s velikom bazom unaprijed definiranih biblioteka koje olakšavaju rad s raznim senzorima, modulima i komponentama.

Slika 13. Logo Arduino IDE-a [10]

4.2.2. Edge Impulse Studio

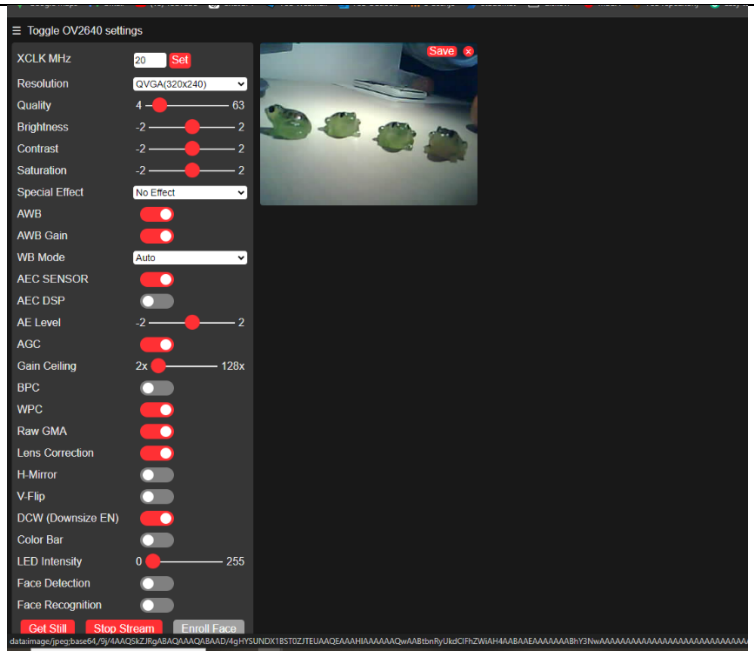
Edge Impulse Studio je razvojno okruženje namijenjeno stvaranju, treniranju i implementaciji modela strojnog učenja na uređajima na rubu mreže. Pruža intuitivno sučelje koje omogućuje korisnicima jednostavno prikupljanje podataka, njihovu obradu i treniranje modela bez potrebe za naprednim tehničkim znanjem. Studio podržava različite senzore i mikrokontrolere te nudi automatsko generiranje značajki i optimizaciju modela za uređaje s ograničenim resursima. Također, omogućuje implementaciju modela izravno na hardver.

4.3. Prikupljanje podataka

Potrebno je kreirati skup sirovih podataka (bez oznaka) sa slikama koje sadrže objekte koje želimo detektirati, a to su u ovom slučaju žabe i naranče. Prikupljanje podataka za ovaj projekt započinje postavljanjem i povezivanjem XIAO ESP32S3 mikrokontrolera s računalom putem Arduino IDE okruženja. Prvo je potrebno odabrati odgovarajuću razvojnu ploču XIAO ESP32S3 u Arduino IDE-u te potvrditi da je ispravan port odabran za uređaj. Zatim se odabire gotov primjer CameraWebServer koji omogućuje korištenje kamere na mikrokontroleru za prenošenje snimke u stvarnom vremenu. Snimka se prikazuje na web stranici koja za server koristi sami mikrokontroler čiji URL se generira u Arduino IDE-u kao što je prikazano na slici 14.

Slika 14. URL web stranice za povezivanje kamere

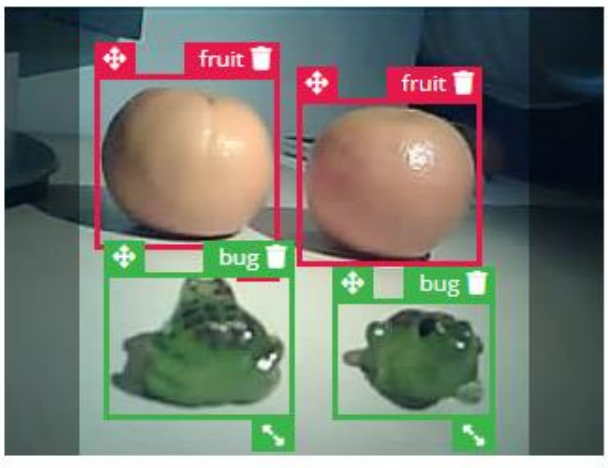
Kroz ovaj proces emitiranja snimke, snimljeno je 53 slike objekata koje su kasnije korištene za treniranje modela strojnog učenja.



Slika 15. Emitiranje snimke i snimanje slika

4.4. Označavanje podataka

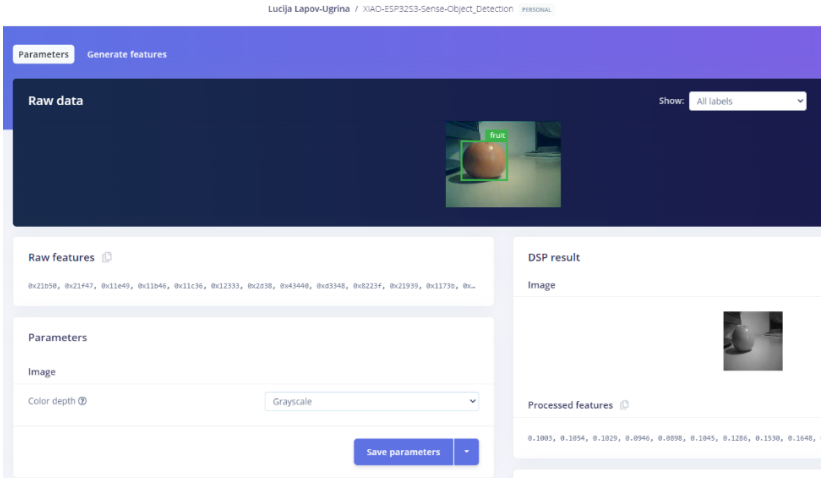
Nakon dovoljno prikupljenih podataka, snimljene slike prenose se u Edge Impulse Studio kao podaci za treniranje. Sve neoznačene slike potrebno je ispravno označiti prije nego što se mogu koristiti kao dio skupa podataka projekta. U tu svrhu, u studiju je dostupan alat za automatsko označavanje nazvan Praćenje objekata između kadrova. Ova opcija omogućava da se, nakon što je nacrtan *bounding box* i označeni objekti na jednoj slici, objekti automatski prate kroz naredne kadrove, pri čemu se novi objekti djelomično označavaju (iako nisu svi označeni ispravno). Nakon što su svi podaci označeni, potrebno je provjeriti točnost oznaka te ispraviti moguće pogreške.



Slika 16. Označavanje podataka u Edge Impulse Studiju

Dio slika potrebno je premjestiti u testni skup podataka kako bi se osigurala objektivna procjena učinkovitosti modela. Testni skup podataka služi za evaluaciju modela nakon što je treniran, kako bi se vidjelo koliko dobro model generalizira na nove, neviđene podatke. Ako bi se koristili samo podaci iz skupa za treniranje, model bi mogao postići visoku točnost na tim podacima, ali bi ostvarivao loše rezultate na stvarnim ili novim podacima. Odabrano je sedam slika, što predstavlja 13 % ukupnog skupa podataka.

Također, potrebno je u ovoj fazi primijeniti sive tonove (engl. *greyscale*) i promijeniti veličine pojedinih slika s 320 x 240 na 96 x 96 piksela, pri čemu se slike prilagođavaju u kvadratni oblik bez obrezivanja što je prikladno za korištenje s FOMO modelima te spremiti parametre.

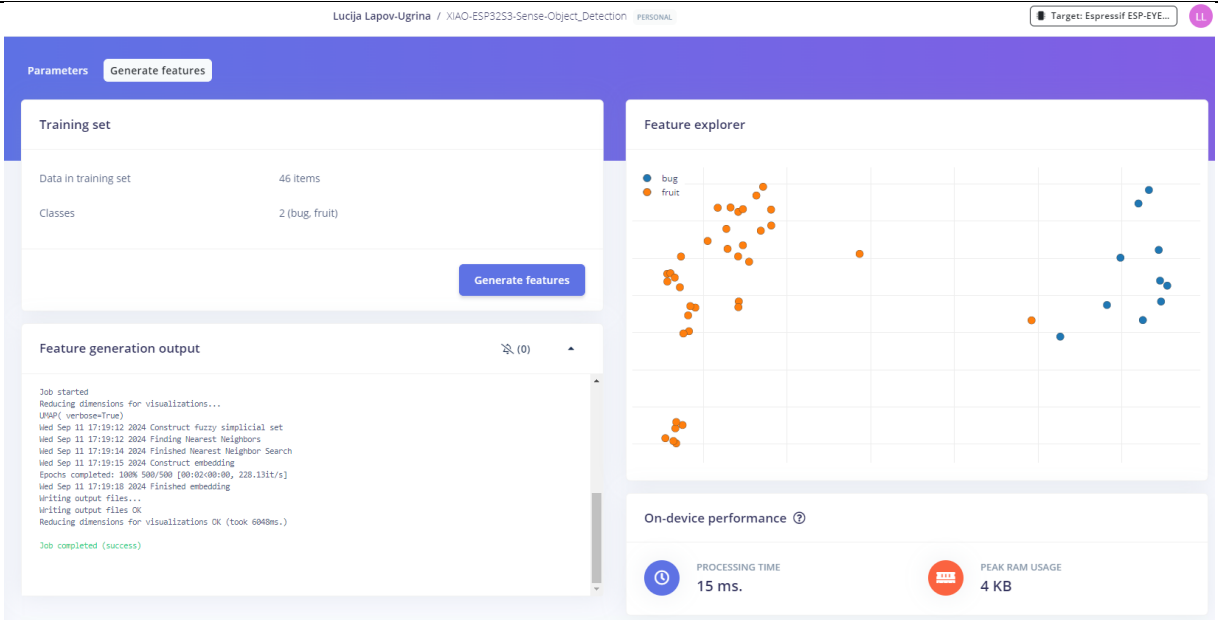


Slika 17. Postavljanje parametra boje

4.5. Treniranje modela

Za treniranje modela koristi se unaprijed istrenirani model FOMO. Ovaj model koristi oko 250 KB RAM-a i 80 KB ROM-a (*Flash*), što je pogodno za odabrani mikrokontroler.

Studio se nakon pohranjivanja parametara automatski prebacuje na sljedeću sekciju, generiranje značajki, gdje će svi uzorci biti prethodno obrađeni, rezultirajući skupom podataka s pojedinačnim slikama veličine 96 x 96 x 1 ili 9.216 značajki. To se odnosi na ukupan broj piksela koji se koristi za predstavljanje svih slika u skupu podataka nakon što su obrađene jer je svaka slika veličine 96 x 96 piksela te se slike su u sivoj skali, što znači da imaju samo jedan kanal (umjesto tri kanala za RGB slike).



Slika 18. Generirane značajke jasno odvojene

Na slici 18. vidljivo je da, nakon što su podaci obrađeni i pretvoreni u značajke, uzorci u skupu podataka su razdvojeni na način koji omogućuje modelu da učinkovito razlikuje između različitih klasa ili objekata. Svi uzorci pokazuju dobro razdvajanje nakon generiranja značajki što znači da su značajke koje su generirane jasno odvojene za različite klase ili objekte. To implicira da su značajke za različite klase (voće i kukci) različite u smislu svojih numeričkih vrijednosti, što omogućuje modelu da ih razlikuje. Neki uzorci izgledaju kao da su na pogrešnom mjestu, ali klikom na njih potvrđuje se da je označavanje ispravno.

4.5.1. Hiperparametri

Hiperparametri su varijable koje se postavljaju prije početka procesa učenja modela u strojnom učenju. Oni kontroliraju način na koji se model trenira i utječu na performanse modela. Za razliku od parametara modela, koji se uče tijekom treninga, hiperparametri se postavljaju unaprijed i nisu direktno optimizirani tijekom treninga.

Primjeri hiperparametara su:

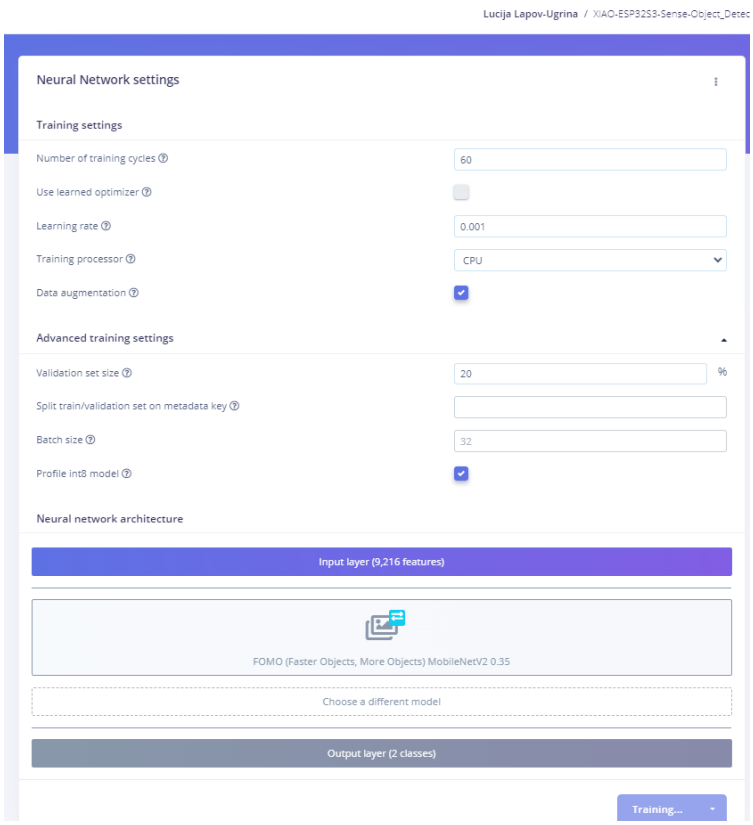
- Broj epoha - broj puta koliko će se cijeli skup podataka prolaziti kroz model tijekom treninga, više epoha može pomoći modelu da se bolje nauči, ali previše epoha može dovesti do prenaučenosti (engl. *overfitting*)
- Veličina serije - broj uzoraka koji se obrađuju u jednom prolazu kroz model tijekom treninga, manja veličina serije može omogućiti preciznije prilagodbe modela, ali

može produžiti vrijeme treniranja, dok veća veličina serije može ubrzati proces treniranja, no može rezultirati manje preciznim prilagodbama modela

- Brzina učenja - označava koliko brzo se model prilagođava tijekom treninga, manja brzina učenja omogućava preciznije fino podešavanje, ali može usporiti proces treniranja, dok prevelika brzina učenja može uzrokovati nestabilnost i neujednačeno treniranje.

Model u ovom radu treniran je sa sljedećim parametrima:

- Broj epoha = 60
- Veličina serije = 32
- Brzina učenja = 0.001



Slika 19. Postavljanje hiperparametara

4.5.2. Procjena rezultata

Za validaciju tijekom obuke, 20 % skupa podataka (*validation_dataset*) će biti sačuvano. Preostalih 80 % (*train_dataset*) će biti podvrgnuto proširenju podataka (*data augmentation*). Proširenje podataka odnosi se na tehnike koje se koriste za umjetno povećanje veličine skupa podataka za treniranje modela strojnog učenja. Ovo se postiže stvaranjem novih uzoraka iz

postojećih podataka putem različitih transformacija. Cilj proširenja podataka je poboljšati generalizaciju modela i spriječiti prenaučenost tako što se model izlaže raznolikijim primjerima. Dakle, 80 % podataka bit će nasumično okrenuti, drugačije veličine i svjetline slike s ciljem povećanja broj uzoraka u skupu podataka za obuku.

Kao rezultat, model završava s ukupnim F1 rezultatom od 95 %. F1 rezultat je metrički pokazatelj koji ocjenjuje prediktivne sposobnosti modela detaljnije analizirajući njegovu izvedbu po klasama, umjesto ukupne izvedbe kao što to čini točnost. F1 rezultat uzima u obzir dva bitna parametra – preciznost i odziv modela kako bi dao jedinstvenu mjeru uspješnosti modela. [13]

Preciznost se odnosi a omjer ispravno klasificiranih pozitivnih slučajeva u odnosu na sve slučajeve koji su klasificirani kao pozitivni te se računa prema sljedećem izrazu:

$$preciznost = \frac{TP}{TP + FP} \tag{1}$$

gdje su:

TP - broj ispravno identificiranih pozitivnih slučajeva,

FP - broj negativnih slučajeva pogrešno označenih kao pozitivni.

Odziv je omjer ispravno klasificiranih pozitivnih slučajeva u odnosu na sve stvarne pozitivne slučajeve. Formula za odziv je:

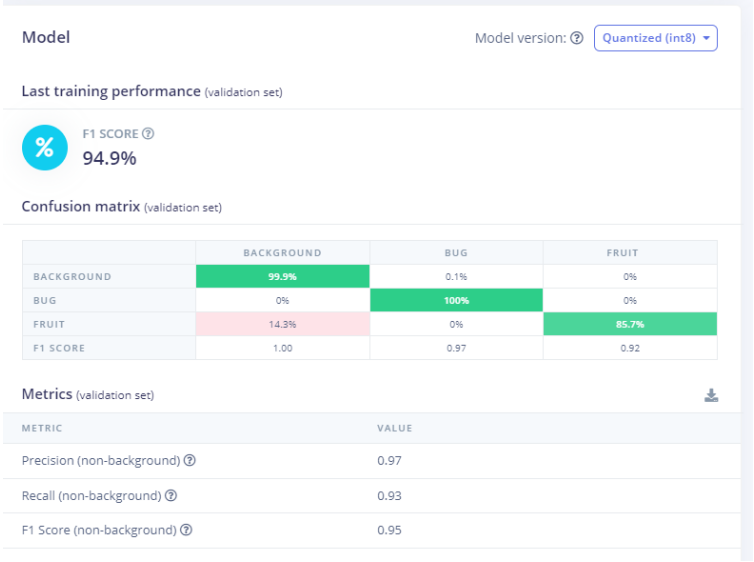
$$odziv = \frac{TP}{TP + FN} \tag{2}$$

gdje je:

FN - broj pozitivnih slučajeva pogrešno označenih kao negativni.

Konačno, F1 rezultat računa se:

$$F1 = 2 * \frac{preciznost * odziv}{preciznost + odziv} \tag{3}$$



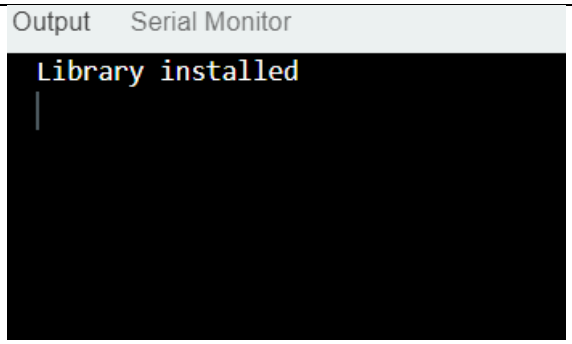
Slika 20. F1 rezultat modela

U zadacima detekcije objekata, točnost nije primarna metrika za evaluaciju modela. Detekcija objekata uključuje ne samo klasifikaciju objekata, već i određivanje njihovih položaja pomoću okvira, što je složeniji zadatak u odnosu na jednostavnu klasifikaciju. FOMO, koji se koristi u ovom slučaju, daje samo koordinate centroida objekata, umjesto kompletnih okvira. Stoga, korištenje točnosti kao metričke mjere može biti zbunjujuće i možda neće pružiti potpunu sliku učinkovitosti modela. Zbog toga se koristi F1 rezultat kao prikladniji pokazatelj.

F1 rezultat kombinira preciznost i odziv, pružajući uravnoteženu ocjenu performansi modela. U ovom slučaju, visoka preciznost od 0,97 ukazuje da model rijetko označava negativne slučajeve kao pozitivne. Visok odziv od 0,93 pokazuje da model uspješno prepoznaje većinu stvarnih pozitivnih slučajeva. Konačno, visok F1 rezultat od 0,95 potvrđuje da je model uspješan u identifikaciji pozitivnih slučajeva, uz minimalne pogreške u klasifikaciji. Ovi rezultati sugeriraju da model postiže izvrsne rezultate u prepoznavanju pozitivnih slučajeva, s minimalnim brojem pogrešaka.

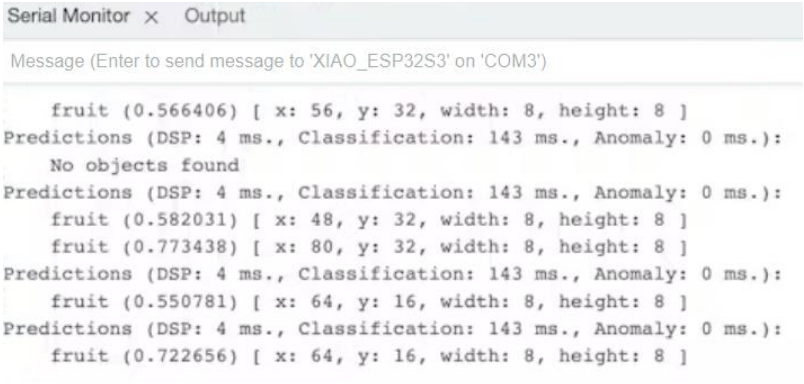
4.6. Implementacija

Kad je model razvijen i treniran te optimiziran za korištenje s mikrokontrolerom, može se implementirati na XIAO ESP32S3. Edge Impulse može kreirati Arduino biblioteku koja sadrži trenirani ML model te po završetku izrade, preglednik automatski počinje preuzimati generiranu biblioteku u obliku .ZIP datoteke. Ova datoteka potom se učita u Arduino IDE-u te se otvori izgenerirani projekt esp32_camera.



Slika 21. Učitavanje biblioteke u Arduino IDE

Program se pokreće na mikrokontroleru i testira postavljanjem objekata (žaba i naranči) ispred kamere. Na slici 21. prikazano je kako je program uspješno prepoznao naranče s određenom vjerojatnošću, uz dodatno pružanje koordinata položaja centroida objekata.



Slika 22. Rezultati programa - predviđanje voća

Na slici 22. prikazani su izlazni podaci za slučaj kad je pred kamerom žaba. Na obje slike vidljivo je da u trenutcima kad ni naranča ni žaba nije pred kamerom, model prepoznaje pozadinu te vraća povratnu informaciju u obliku poruke „No objects found“.

Slika 23. Rezultati programa - predviđanje kukaca

5. ZAKLJUČAK

Iako je detekcija objekata značajno zahtjevnija i složenija u odnosu na jednostavnu klasifikaciju slika, pokazano je da je moguće razviti sustav za višestruku klasifikaciju i detekciju objekata na mikrokontroleru. Korištenjem FOMO modela, uspješno su iskorištene prednosti oba pristupa: minimalne računalne zahtjeve koji su karakteristični za jednostavnu klasifikaciju slika, uz dodatne informacije o lokaciji i broju objekata, što omogućuje detekcija objekata. Cilj je bio stvoriti lagan, ali učinkovit sustav za višestruku klasifikaciju objekata, optimiziran za rad na uređajima s ograničenim resursima poput mikrokontrolera, što je uspješno postignuto te potvrđeno F1 rezultatom koji iznosi visokih 95 %. Ovaj sustav sposoban je postići izvrsne performanse u prepoznavanju i klasifikaciji objekata, čak i na hardverski ograničenim platformama poput mikrokontrolera čime se otvara mogućnost primjene u stvarnim scenarijima.

Međutim, u stvarnom svijetu takva implementacija susrela bi se s nekoliko izazova. Iako su FOMO modeli optimizirani za rad na uređajima s ograničenim resursima, industrijsko okruženje donosi specifične zahtjeve, poput veće brzine i velikog broja objekata koje je potrebno detektirati u kratkom vremenskom razdoblju. Model, s brzinom od 7 fps i F1 rezultatom od 95 %, pokazao je izvrsne rezultate u kontroliranim uvjetima, ali industrijski uvjeti, poput promjena osvjetljenja, položaja objekata te različitih veličina i boja, mogli bi smanjiti točnost prepoznavanja. Dodatni izazovi, poput prisutnosti smetnji u obliku prljavštine ili vlage, također bi mogli utjecati na performanse sustava. Iako je moguće koristiti ovaj model i mikrokontroler za detekciju žaba i naranči na pokretnoj traci, u industrijskom okruženju bi vjerojatno bilo potrebno skalirati rješenje i uvesti snažnije uređaje kako bi se osigurala veća pouzdanost, brzina i preciznost u stvarnim uvjetima, no ovaj sustav može poslužiti kao prototip ili pomoćni rješenje.

LITERATURA

[1] Machine learning, https://en.wikipedia.org/wiki/Machine_learning, pristupljeno: 12.9.2024.

[2] Supervised Machine Learning, https://www.javatpoint.com/supervised-machine-learning, pristupljeno: 12.9.2024.

[3] Unsupervised Machine Learning, https://www.javatpoint.com/unsupervised-machine-learning, pristupljeno: 12.9.2024.

[4] Types of Machine Learning, https://www.geeksforgeeks.org/types-of-machine-learning/, pristupljeno: 12.9.2024.

[5] Reinforcement learning, https://www.geeksforgeeks.org/what-is-reinforcement-learning/, pristupljeno: 12.9.2024.

[6] Reinforcement learning, https://en.wikipedia.org/wiki/Reinforcement_learning, pristupljeno: 12.9.2024.

[7] Announcing FOMO (Faster Objects, More Objects), https://www.edgeimpulse.com/blog/announcing-fomo-faster-objects-more-objects/, pristupljeno: 12.9.2024.

[8] TinyML Made Easy: Object Detection with XIAO ESP32S3 Sense, https://www.hackster.io/mjrobot/tinyml-made-easy-object-detection-with-xiao-esp32s3-sense-6be28d, pristupljeno: 13.9.2024.

[9] Seeed Studio XIAO ESP32S3 Sense, https://www.seeedstudio.com/XIAO-ESP32S3-Sense-p-5639.html?srsltid=AfmBOoprdx0s0eiAWHh_dMD8BeR5J9ymQIIv0vB-q0qgJdqXKwdoyQNx, pristupljeno: 13.9.2024.

[10] Arduino IDE, https://www.arduino.cc/en/software, pristupljeno: 13.9.2024.

[11] Edge Impulse, https://edgeimpulse.com/, pristupljeno: 13.9.2024.

[12] Hyperparameters in Machine Learning, https://www.javatpoint.com/hyperparameters-in-machine-learning, pristupljeno: 13.9.2024.

[13] F1 Score in Machine Learning: Intro & Calculation, https://www.v7labs.com/blog/f1-score-guide, pristupljeno: 13.9.2024.

PRILOZI

I. CD-R disc

II. CameraWebServer skripta

```
#include "esp_camera.h"
#include <WiFi.h>
#define CAMERA_MODEL_XIAO_ESP32S3 // Has PSRAM
#include "camera_pins.h"
const char* ssid = "A1_1131317356";
const char* password = "********";

void startCameraServer();
void setupLedFlash(int pin);

void setup() {
  Serial.begin(115200);
  Serial.setDebugOutput(true);
  Serial.println();

  camera_config_t config;
  config.ledc_channel = LEDC_CHANNEL_0;
  config.ledc_timer = LEDC_TIMER_0;
  config.pin_d0 = Y2_GPIO_NUM;
  config.pin_d1 = Y3_GPIO_NUM;
  config.pin_d2 = Y4_GPIO_NUM;
  config.pin_d3 = Y5_GPIO_NUM;
  config.pin_d4 = Y6_GPIO_NUM;
  config.pin_d5 = Y7_GPIO_NUM;
  config.pin_d6 = Y8_GPIO_NUM;
  config.pin_d7 = Y9_GPIO_NUM;
  config.pin_xclk = XCLK_GPIO_NUM;
  config.pin_pclk = PCLK_GPIO_NUM;
  config.pin_vsync = VSYNC_GPIO_NUM;
  config.pin_href = HREF_GPIO_NUM;
  config.pin_sccb_sda = SIOD_GPIO_NUM;
  config.pin_sccb_scl = SIOC_GPIO_NUM;
  config.pin_pwdn = PWDN_GPIO_NUM;
  config.pin_reset = RESET_GPIO_NUM;
  config.xclk_freq_hz = 20000000;
  config.frame_size = FRAMESIZE_UXGA;
  config.pixel_format = PIXFORMAT_JPEG; // for streaming
  //config.pixel_format = PIXFORMAT_RGB565; // for face detection/recognition
  config.grab_mode = CAMERA_GRAB_WHEN_EMPTY;
  config.fb_location = CAMERA_FB_IN_PSRAM;
  config.jpeg_quality = 12;
  config.fb_count = 1;
```

```
  // if PSRAM IC present, init with UXGA resolution and higher JPEG quality
  //                      for larger pre-allocated frame buffer.
  if(config.pixel_format == PIXFORMAT_JPEG){
    if(psramFound()){
      config.jpeg_quality = 10;
      config.fb_count = 2;
      config.grab_mode = CAMERA_GRAB_LATEST;
    } else {
      // Limit the frame size when PSRAM is not available
      config.frame_size = FRAMESIZE_SVGA;
      config.fb_location = CAMERA_FB_IN_DRAM;
    }
  } else {
    // Best option for face detection/recognition
    config.frame_size = FRAMESIZE_240X240;
#if CONFIG_IDF_TARGET_ESP32S3
    config.fb_count = 2;
#endif
  }

#if defined(CAMERA_MODEL_ESP_EYE)
  pinMode(13, INPUT_PULLUP);
  pinMode(14, INPUT_PULLUP);
#endif

  // camera init
  esp_err_t err = esp_camera_init(&config);
  if (err != ESP_OK) {
    Serial.printf("Camera init failed with error 0x%x", err);
    return;
  }

  sensor_t * s = esp_camera_sensor_get();
  // initial sensors are flipped vertically and colors are a bit saturated
  if (s->id.PID == OV3660_PID) {
    s->set_vflip(s, 1); // flip it back
    s->set_brightness(s, 1); // up the brightness just a bit
    s->set_saturation(s, -2); // lower the saturation
  }
  // drop down frame size for higher initial frame rate
  if(config.pixel_format == PIXFORMAT_JPEG){
    s->set_framesize(s, FRAMESIZE_QVGA);
  }

#if defined(CAMERA_MODEL_M5STACK_WIDE) ||
defined(CAMERA_MODEL_M5STACK_ESP32CAM)
  s->set_vflip(s, 1);
  s->set_hmirror(s, 1);
#endif
```

```
#if defined(CAMERA_MODEL_ESP32S3_EYE)
  s->set_vflip(s, 1);
#endif

// Setup LED FLash if LED pin is defined in camera_pins.h
#if defined(LED_GPIO_NUM)
  setupLedFlash(LED_GPIO_NUM);
#endif

  WiFi.begin(ssid, password);
  WiFi.setSleep(false);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");

  startCameraServer();

  Serial.print("Camera Ready! Use 'http://");
  Serial.print(WiFi.localIP());
  Serial.println("' to connect");
}

void loop() {
  // Do nothing. Everything is done in another task by the web server
  delay(10000);
}
```