SVEUČILIŠTE U SPLITU  
FAKULTET ELEKTROTEHNIKE, STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

IZRADA ANDROID APLIKACIJE ZA KLASIFIKACIJU BILJAKA I DRVEĆA NA TEMELJU SLIKA

Goran Bogetić

Split, kolovoz 2025.

|  |  |  |
| --- | --- | --- |
|  | SVEUČILIŠTE U SPLITU  FAKULTET ELEKTROTEHNIKE, STROJARSTVA I BRODOGRADNJE |  |

Sveučilišni diplomski studij: **Naziv studija**

Smjer/Usmjerenje: **Naziv smjera/usmjerenja**

Oznaka programa: 210

Akademska godina: 2023./2024.

Ime i prezime: **IME PREZIME**

JMBAG: xxxxxxxxxx

**ZADATAK DIPLOMSKOG RADA**

Naslov: **NASLOV RADA**

Zadatak: Prilikom prijave diplomskog rada mentor definira temu i radni naslov, a točan naslov rada utvrđuje se po završetku rada. Diplomski se rad, potpisan od strane mentora, predaje Odboru za diplomski rad u tri tiskana primjerka s tvrdim uvezom te u elektroničkoj formi. Izradu diplomskog rada ocjenjuje mentor, a predsjednik Odbora za diplomski rad potvrđuje zadatak i konačni naslov rada. Ukoliko je predsjednik Odbora za diplomski rad ujedno i mentor, diplomski rad uz mentora potpisuje i jedan od članova Odbora. **Ovu stranicu piše mentor prije predaje rada nakon što je utvđen konačni naslov rada.**

Rad predan:

|  |  |  |
| --- | --- | --- |
| Predsjednik  Odbora za diplomski rad: |  | Mentor: |
|  |  |  |
| prof. dr. sc. Ime Prezime |  | izv. prof. dr. sc. Ime Prezime |

**IZJAVA**

Ovom izjavom potvrđujem da sam diplomski rad s naslovom IZRADA ANDROID APLIKACIJE ZA KLASIFIKACIJU BILJAKA I DRVEĆA NA TEMELJU SLIKA pod mentorstvom dr. sc. Josipa Vasilja pisao samostalno, primijenivši znanja i vještine stečene tijekom studiranja na Fakultetu elektrotehnike, strojarstva i brodogradnje, kao i metodologiju znanstveno-istraživačkog rada, te uz korištenje literature koja je navedena u radu. Spoznaje, stavove, zaključke, teorije i zakonitosti drugih autora koje sam izravno ili parafrazirajući naveo/la u diplomskom radu citirao/la sam i povezao/la s korištenim bibliografskim jedinicama.

Student/ica

Goran Bogetić

**SADRŽAJ**

[1 UVOD 1](#_Toc207539454)

[2 TEORIJSKA POZADINA I POVEZANA ISTRŽIVANJA 2](#_Toc207539455)

[2.1 Struktura diplomskog rada – obvezni dijelovi radnje 2](#_Toc207539456)

[2.1.1 Osnovni koncepti 2](#_Toc207539457)

[2.1.2 Usporedba koncepata 5](#_Toc207539458)

[2.2 Neuronske mreže i duboko učenje 6](#_Toc207539459)

[2.2.1 Konvolucijske neuronske mreže 7](#_Toc207539460)

[2.2.2 Transfer learning i fine-tuning 8](#_Toc207539461)

[2.3 Računalni vid 10](#_Toc207539462)

[2.4 Povezna istraživanja i postojeći sustavi 12](#_Toc207539463)

[3 IZRADA BAZE PODATAKA 13](#_Toc207539464)

[3.1 Ciljevi i dizajn baze podataka 13](#_Toc207539465)

[3.2 Izvori podataka 15](#_Toc207539466)

[3.3 Tehnička izvedba prikupljanja slika 17](#_Toc207539467)

[3.4 Organizacija i podjela podataka 23](#_Toc207539468)

[3.5 Rezultati izrade baze 26](#_Toc207539469)

[3.6 Rekonstrukcija baze podataka 27](#_Toc207539470)

[4 TRENIRANJE I EVALUACIJA MODELA 31](#_Toc207539471)

[4.1 Ciljevi treniranja i metodologija 31](#_Toc207539472)

[4.2 Priprema podataka 32](#_Toc207539473)

[4.3 Arhitektura modela 32](#_Toc207539474)

[4.4 Strategije učenja i postupak treniranja 33](#_Toc207539475)

[4.5 Protokol evaluacije i izvještavanje 34](#_Toc207539476)

[4.6 Rezultati treniranja 36](#_Toc207539477)

[4.7 Izvoz modela i pripreme za Android 37](#_Toc207539478)

[4.8 Ograničenja i potencijalna poboljšanja 39](#_Toc207539479)

[5 ANDROID APLIKACIJA 41](#_Toc207539480)

[5.1 Pregled 41](#_Toc207539481)

[5.2 Postavke projekta 41](#_Toc207539482)

[5.3 Artefakti modela 42](#_Toc207539483)

[5.4 Dohvaćanje slika preko kamere i galerije 44](#_Toc207539484)

[5.5 Pretprocesiranje na uređaju 44](#_Toc207539485)

[5.6 Inferencija i mapiranje rezultata 45](#_Toc207539486)

[5.7 Prikaz rezultata 46](#_Toc207539487)

[5.8 UX povratne informacije i animacije 48](#_Toc207539488)

[5.9 Operativni aspekti rada na uređaju 50](#_Toc207539489)

[6 ZAKLJUČAK 54](#_Toc207539490)

[LITERATURA 56](#_Toc207539491)

[PRILOZI 58](#_Toc207539492)

[Kazalo slika i tablica 58](#_Toc207539493)

[Kazalo slika 58](#_Toc207539494)

[Kazalo tablica 59](#_Toc207539495)

[Popis oznaka i kratica 59](#_Toc207539496)

[SAŽETAK/ABSTRACT I KLJUČNE RIJEČI/KEYWORDS 60](#_Toc207539497)

# UVOD

Umjetna inteligencija postala je jedna od najznačajnijih tehnologija 21. stoljeća, s utjecajem na gotovo svaki aspekt društva – od financija i zdravstva do obrazovanja i poljoprivrede. Najveću primjenu pronašla je u analizi velikih količina podataka, automatiziranom donošenju odluka te inteligentnom rješavanju složenih problema iz stvarnog svijeta. Posebno su strojno učenje i duboko učenje omogućili značajan napredak u obradi prirodnog jezika i računalnoj obradi slika, čime je otvoren put sustavima koji mogu obavljati zadatke do tada rezervirane za stručnjake, uz sve veću preciznost koja se kontinuirano poboljšava.

Za ovaj rad posebno je važan računalni vid, odnosno sposobnost automatskog prepoznavanja i klasifikacije objekata na slikama. On nalazi široku primjenu, od medicinske dijagnostike i autonomne vožnje do praćenja okoliša i sigurnosnih sustava. Kao rezultat toga, računalni vid postao je moćan alat u situacijama gdje su ručno promatranje i klasifikacija vremenski zahtjevni, skloni pogreškama ili nepraktični u velikim razmjerima. Točna identifikacija biljnih vrsta ključna je u poljoprivredi, šumarstvu i ekologiji, no često zahtijeva stručno botaničko znanje koje nije lako dostupno široj javnosti. S razvojem aplikacija temeljenih na umjetnoj inteligenciji, danas je moguće premostiti te prepreke pružanjem automatiziranih i jednostavnih alata koji identifikaciju biljaka čine dostupnijom i bržom.

U ovom radu prikazat će se postupak izrade Android aplikacije za automatsku klasifikaciju biljaka i drveća, u slučaju kada je potrebno razviti vlastitu bazu podataka te na temelju nje istrenirati model za prepoznavanje. U drugom poglavlju bit će objašnjena teorijska osnova strojnog učenja i princip rada algoritama temeljenih na neuronskim mrežama. Treće poglavlje bavit će se postupkom izrade baze podataka od početka, dok će četvrto prikazati treniranje modela. Peto poglavlje posvećeno je izradi Android aplikacije koja implementira automatsku klasifikaciju na temelju slika korištenjem prethodno istreniranog modela.

# TEORIJSKA POZADINA I POVEZANA ISTRŽIVANJA

Ovo poglavlje pružiti će teorijsku pozadinu i osnovne koncepte potrebne za razumijevanje razvoja predložene Android aplikacije za automatsku klasifikaciju biljaka i drveća. Uvesti će se temeljni principi umjetne inteligencije i strojnog učenja, s posebnim naglaskom na nadzirano strojno učenje koje je vrlo važno u klasifikaciji slika.

Nakon toga prelazi se na neuronske mreže i duboko učenje uz naglasak na arhitekture koje se najčešće koriste za vizualno prepoznavanje, poput konvolucijskih neuronskih mreža. Zatim će biti pokazano područje računalnog vida, gdje će biti pokriveni osnovni procesi prepoznavanja slika i njegove raznolike primjene. Naposljetku, biti će pregled povezanih istraživanja i postojećih mobilnih aplikacija za identifikaciju biljaka, trenutnim ograničenjima i motivaciju za ovaj rad.

## Struktura diplomskog rada – obvezni dijelovi radnje

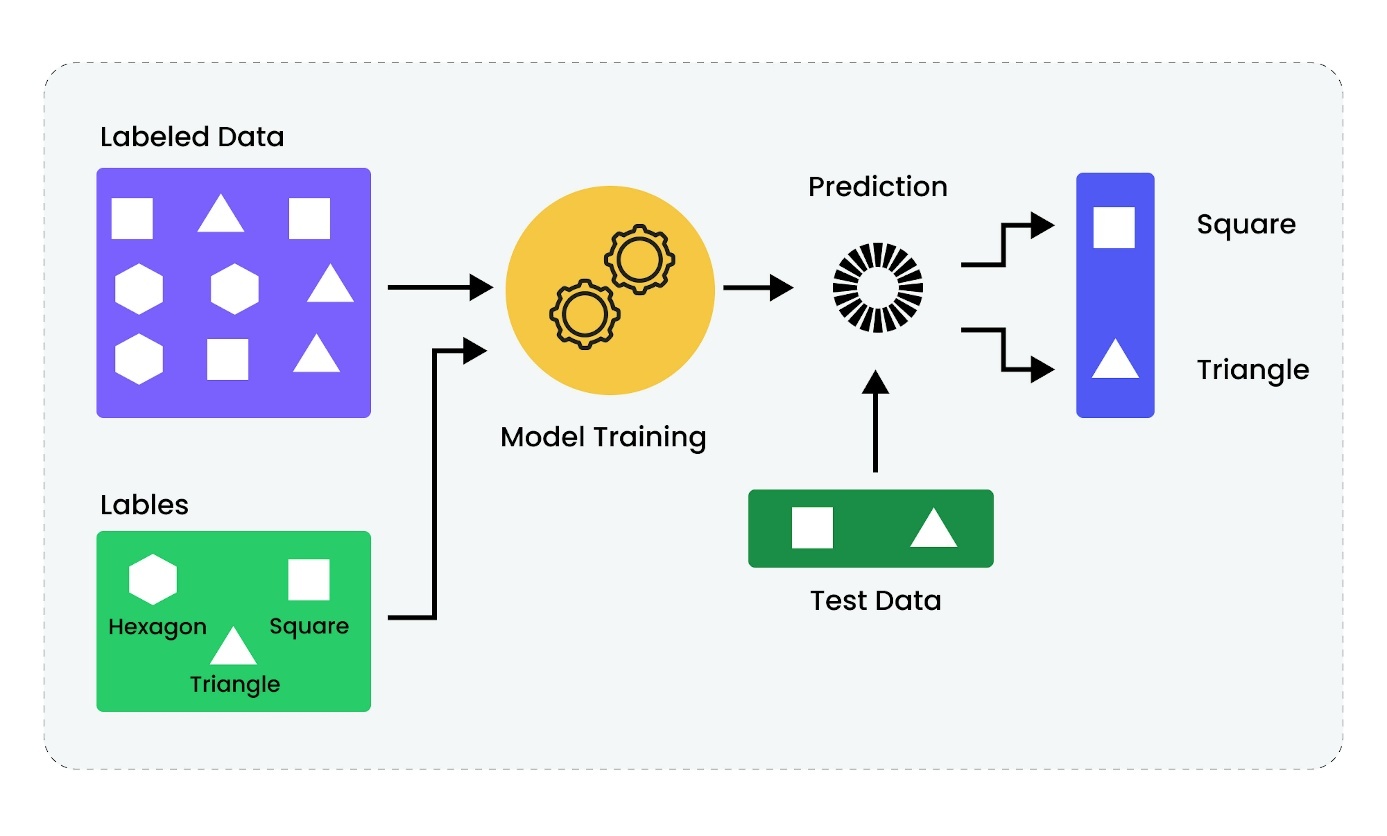
Umjetna inteligencija odnosi se na sposobnost strojeva za izvršavanje radnji obično povezanih sa ljudskom inteligencijom, poput rasuđivanja, percepcije, učenja te donošenja odluka. Unutar ovog vrlo širokog područja, strojno učenja omogućava tim sustavima poboljšanje performansi u zadatcima učenjem iz podataka umjesto da se eksplicitno programiraju pravila određene domene [1].

Osnovni cilj strojnog učenja jest generalizacija, odnosno sposobnost modela da nakon učenja na nekom skupu podataka ispravno prepoznaje obrasce i donosi točne zaključke o novim obrascima koje nikada prije nije vidio. Proces učenja odvija se kroz treniranje modela, pri čemu se parametri modela postupno prilagođavaju u cilju minimiziranja pogrešaka.

### Osnovni koncepti

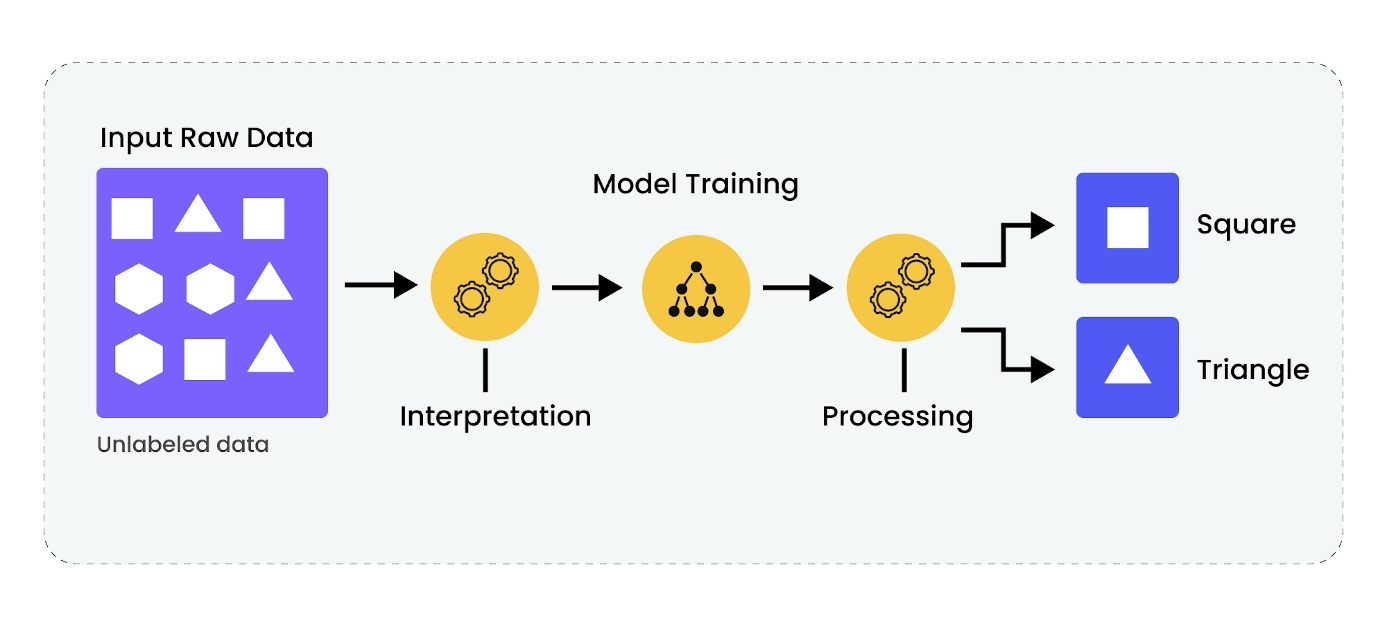
Strojno učenja ima više pristupa, pri čemu su sljedeća najvažnija, nadzirano i nenadzirano strojno učenje, dok se u suvremenim primjenama sve češće koristi i podržano strojno učenje.

Nadzirano strojno učenja podrazumijeva korištenje označenih podataka prilikom treniranja, odnosno korištenje podataka u kojima su uz ulazne podatke poznati i željeni izlazni podatci. Model uči povezati ulaze s pripadajućim izlazima, kao što je vidljivo na slici Slika 2‑1 te se najčešće koristi u zadatcima klasifikacije, npr. prepoznavanje vrste stabla prema slici lista, i regresije, npr. procjena veličine lista. Prednost nadziranog strojnog učenja jest velika preciznost ukoliko se podatci kvalitetno i točno označeni, dok je glavni nedostatak velik trošak i vrijeme potrebno za ručno označavanje velikih skupova podataka [2].



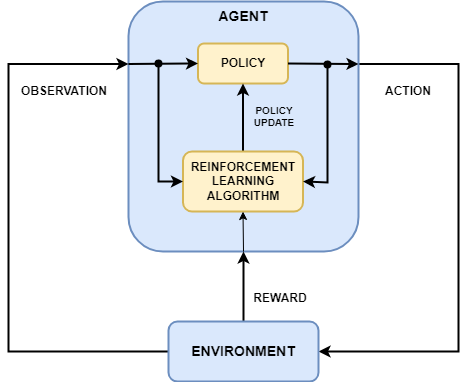
Slika ‑ Prikaz kako radi nadzirano strojno učenje. (preuzeto s <https://maddevs.io/blog/semi-supervised-learning-explained/> )

Nenadzirano strojno učenje, za razliku od nadziranog učenja, ne koristi označene podatke, već algoritmi sami otkrivaju skrivene strukture i obrasce u podatcima, a prikaz kako radi nenadzirano strojno učenje može se vidjeti na slici Slika 2‑2. Najčešće se klasteriranja, što je grupiranje sličnih primjera, i smanjenje dimenzionalnosti, gdje se veliki broj varijabli u podacima svodi na manji broj, pri čemu se nastoji zadržati što je moguće više informacija. Glavna prednost ovog pristupa je mogućnost pronalaska nepoznatih odnosa u podacima, ali evaluacija i interpretacija rezultata često su složenije nego kod nadziranog učenja [3].



Slika ‑ Prikaz kako radi nenadzirano strojno učenje (preuzeto s <https://maddevs.io/blog/semi-supervised-learning-explained/>)

Podržano strojno učenje predstavlja dodatan koncept , gdje sustav uči putem interakcije s okolinom, primajući nagrade ili kazne na temelju svojih akcija, kao što se može vidjeti na slici Slika 2‑3. Iako se rjeđe koristi u zadacima klasifikacije slika, važan je u područjima poput robotike i autonomnih vozila [4].



Slika ‑ Prikaz podržanog strojnog učenja (preuzeto s <https://www.mathworks.com/help/reinforcement-learning/ug/what-is-reinforcement-learning.html>)

### Usporedba koncepata

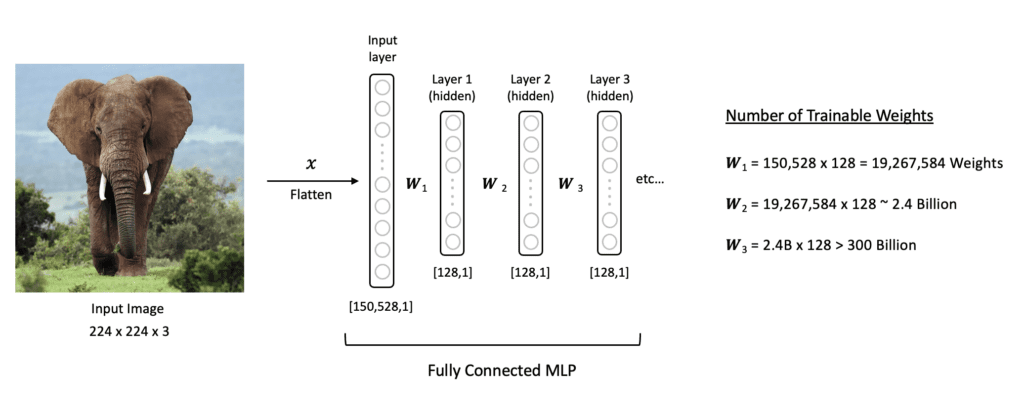
U tablici Tablica 1 može se vidjeti direktna usporedba tri prethodno spomenuta koncepta.

Tablica Usporedba između nadziranog, nenadziranog i podržanog strojnog učenja

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Nadzirano strojno učenj** | **Nenadzirano strojno učenje** | **Podržano strojno učenje** |
| **Ulazni podaci** | Označeni (ulaz i izlaz) | Neoznačeni (samo ulaz) | Stanja i akcije, bez oznaka |
| **Cilj** | Predviđanje poznatih izlaza | Otkrivanje strukture i uzoraka | Maksimizacija kumulativne nagrade |
| **Tipični zadaci** | Klasifikacija, regresija | Klasteriranje, smanjenje dimenzionalnosti | Upravljanje, donošenje odluka |
| **Evaluacija** | Točnost, preciznost, odziv, F1-mjera | Teže mjerljiva, često subjektivna | Vrijednost nagrade kroz epizode |
| **Prednost** | Visoka preciznost uz kvalitetne oznake | Otkiva nepoznate obrasce | Uči iz iskustva i interakcije |
| **Nedostatak** | Skupo i zahtjevno označavanje podataka | Rezultati teško interpretabilni | Velika složenost i potreba za simulacijom |

## Neuronske mreže i duboko učenje

Neuronske mreže predstavljaju temeljnu klasu algoritama strojnog učenja inspiriranih radom ljudskog mozga. Osnovni element svake neuronske mreže jest umjetni neuron, koji prima ulazne podatke, postavlja im težine, primjenjuje funkciju aktivacije te generira izlaz. Perceptron bio je najraniji oblik neuronske mreže sposoban za rješavanje jednostavnih problema linearne separabilnosti [5]. Kasnije su uvedeni višeslojni perceptroni, tzv. MLP, koji preko skrivenih slojeva omogućuju modeliranje složenijih nelinearnih odnosa, a primjer takvog MLP-a može se vidjeti na slici Slika 2‑4.



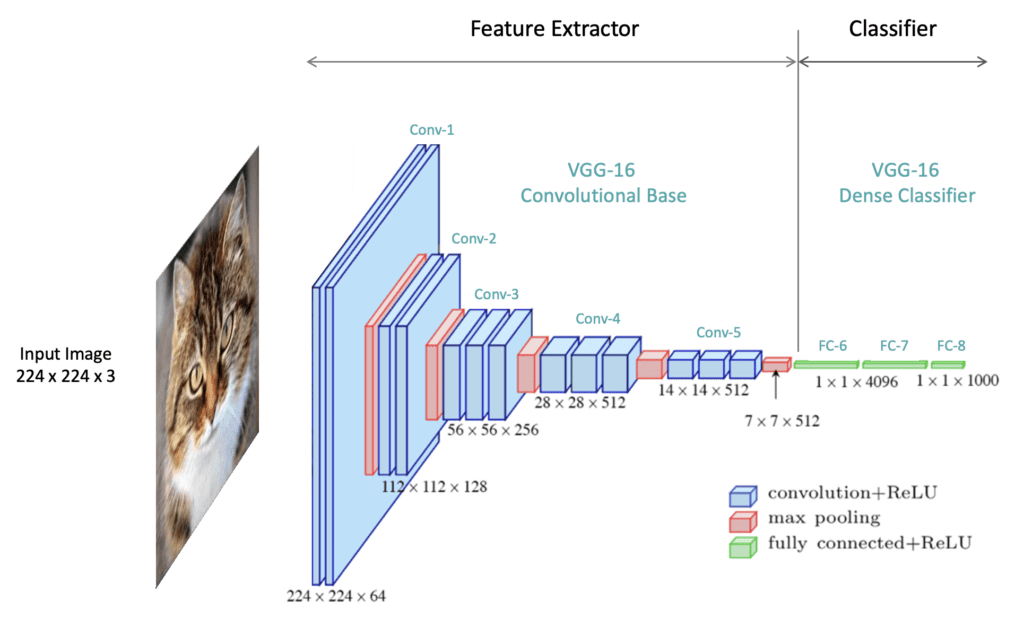
Slika ‑ Primjer izgleda višeslojnog perceptrona, tzv. MLP-a (preuzeto s <https://learnopencv.com/understanding-convolutional-neural-networks-cnn/>)

Razvoj računalne snage i dostupnost velike količine podataka doveli su do pojave dubokog učenja, što je temeljeno na dubokim neuronskim mrežama koje sadrže veliki broj slojeva. Duboke mreže sposobne su automatski učiti složene reprezentacije podataka, što ih čini posebno pogodnima za obradu slike, zvuka i prirodnog jezika [6].

### Konvolucijske neuronske mreže

U računalnom vidu posebno su značajne konvolucijske neuronske mreže (engl. *Convolutional Neural Networks*, CNN). Njihova ključna prednost u odnosu na prethodno spomenite MLP arhitekture jest u konvolucijskim slojevima, koji omogućuju automatsko izdvajanje značajki iz slika, npr. rubova, oblika, tekstura, kroz primjenu filtara. Tipična struktura CNN-a sastoji se od niza međusobno povezanih slojeva. Konvolucijski slojevi imaju ulogu automatskog izdvajanja značajki iz ulazne slike, poput rubova, oblika ili tekstura. Nakon njih slijede pooling slojevi, koji smanjuju dimenzionalnost podataka i povećavaju robusnost modela na translacije ili male promjene u slici. Na kraju mreže nalaze se potpuno povezani slojevi, koji na temelju prethodno izdvojenih značajki provode završnu klasifikaciju.

Razvoj CNN arhitektura značajno je unaprijedio računalni vid. Počevši od *AlexNet*-a [7], koji je donio proboj u prepoznavanju slika, preko *VGGNet*-a i *ResNet*-a koji su omogućili dublje i učinkovitije mreže, pa sve do mobilno optimiziranih modela poput *MobileNet*-a, CNN-ovi su postali temelj modernih sustava računalnog vida.

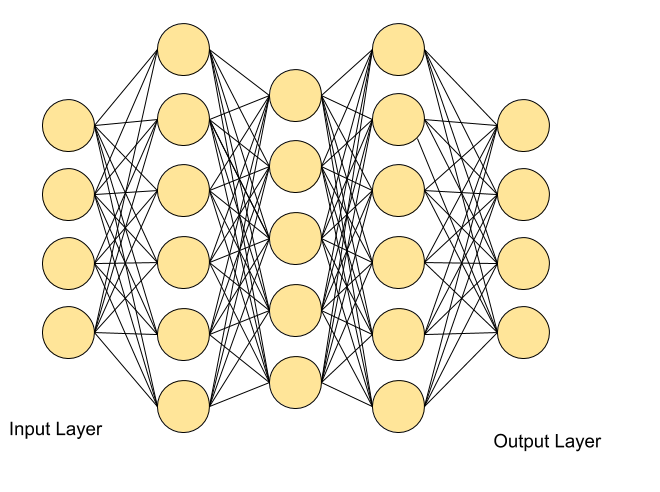


Slika ‑ Izgled konvolucijskih slojeva u VGG-16 konvolucijskoj neuralnoj mreži (preuzeto s <https://learnopencv.com/understanding-convolutional-neural-networks-cnn/>)

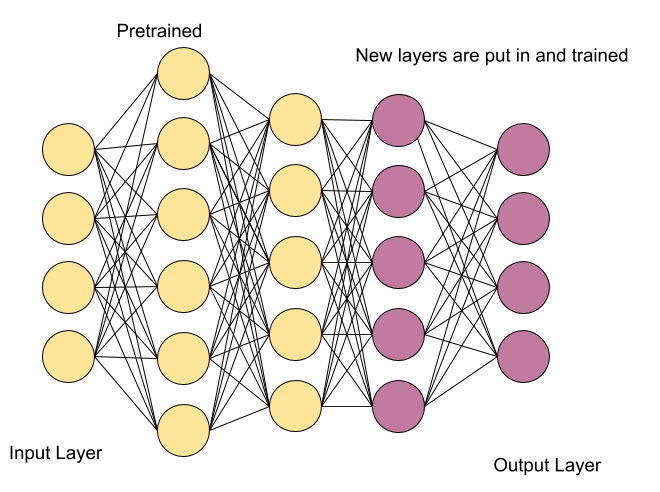
### Transfer learning i fine-tuning

Treniranje dubokih neuronskih mreža od početka zahtijeva velike količine podataka i značajne računalne resurse, što u mnogim slučajevima nije izvedivo. Zbog toga se u praksi često koristi tzv. transfer learning, tehnika kojom se unaprijed istrenirani modeli, najčešće na velikim skupovima podataka poput *ImageNet*-a, prilagođavaju novim zadacima. Uobičajeni pristup sastoji se od zamrzavanja ranijih slojeva, koji sadrže opće značajke poput rubova i osnovnih oblika, dok se treniraju samo završni slojevi. Alternativno, moguće je dodatno fino podešavati i dublje slojeve mreže, čime se postiže bolja specijalizacija za konkretni problem. Usporedba prikaze neuronske mreže trenirane od početka, slika Slika 2‑6, i napravljene tehnikom *transfer learning*-a, slika Slika 2‑7.

Na ovaj način moguće je izgraditi kvalitetne modele i kada su dostupni manji skupovi podataka, primjerice u klasifikaciji biljnih vrsta ili detekciji bolesti na lišću. *Transfer learning* danas predstavlja jedan od ključnih pristupa u računalnom vidu te se široko primjenjuje i u akademskim istraživanjima i u industrijskim projektima.



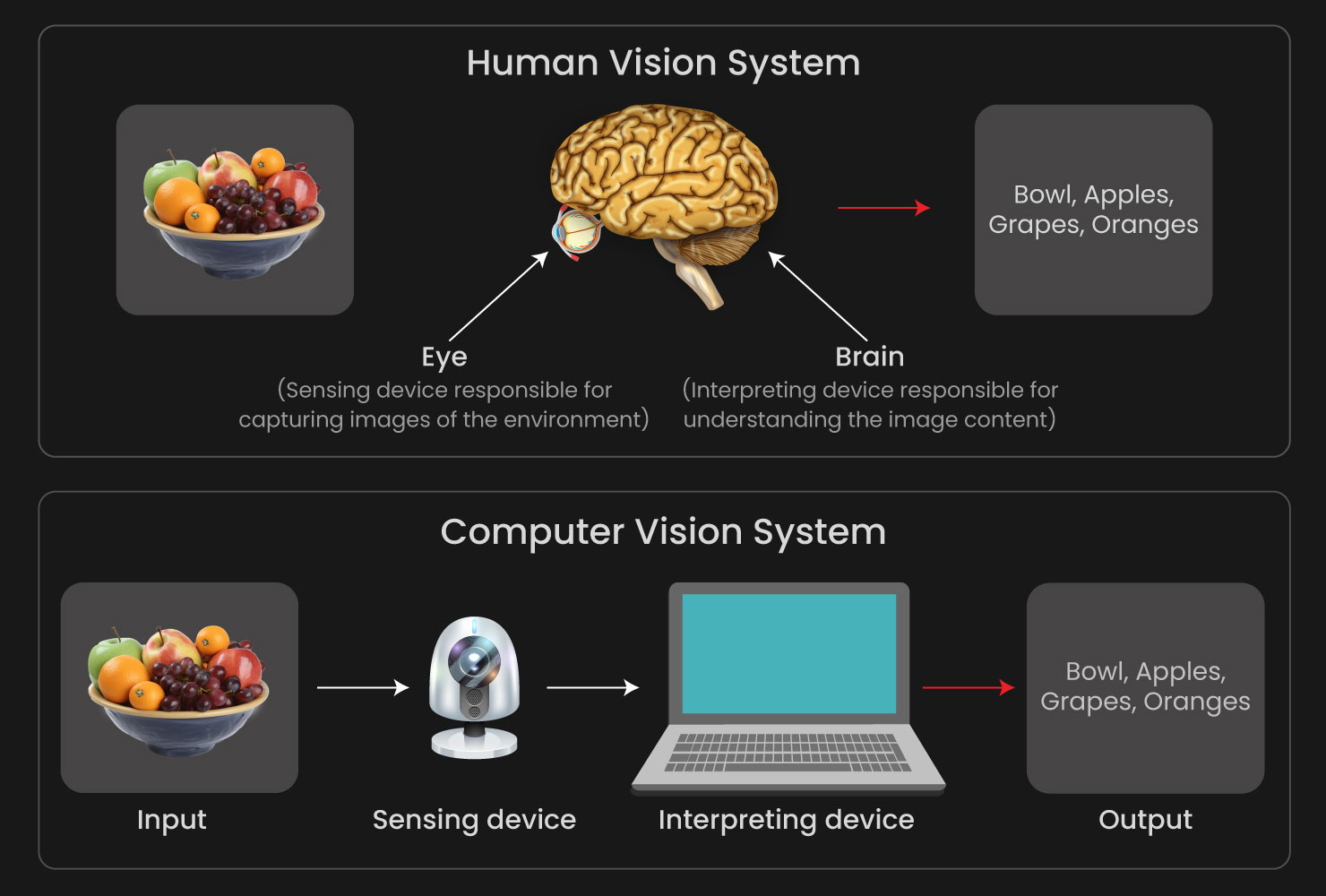
Slika ‑ Neuronska mreža trenirana od počeka (preuzeto s <https://www.danrose.ai/blog/transfer-learning-from-a-business-perspective>)

****

Slika ‑ Neuronska mreža nakon upotrebe tranfer learning tehnike (preuzeto s <https://www.danrose.ai/blog/transfer-learning-from-a-business-perspective>)

## Računalni vid

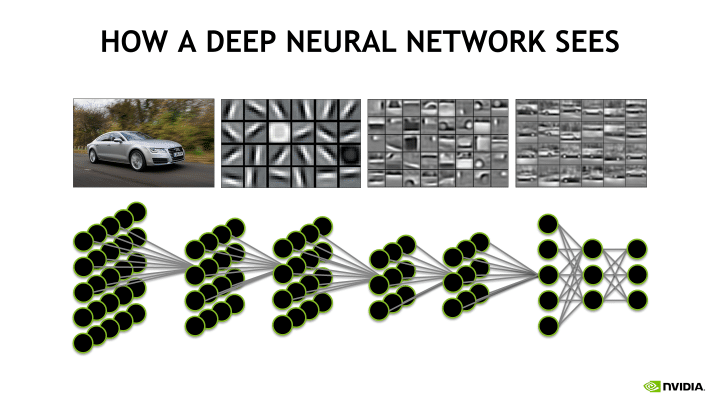
Računalni vid (engl. *Computer Vision*) predstavlja područje umjetne inteligencije koje se bavi omogućavanjem računalima da interpretiraju i razumiju vizualne informacije iz okoline. Glavni cilj računalnog vida jest omogućiti strojevima da, na temelju slika ili videozapisa, prepoznaju objekte, scene i obrasce te da na temelju tih informacija donose odluke ili izvode akcije. U osnovi, računalni vid nastoji replicirati ljudsku sposobnost vizualne percepcije, ali u digitalnom obliku, pri čemu se koristi snaga algoritama i procesorske arhitekture [8]. Ispod, na slici Slika 2‑8, se nalazi ilustracija koja prikazuje usporedbu između ljudskog vida i računalnog vida.



Slika ‑ Usporedba ljudskog vida i računalnog vida (preuzeto s <https://imagevision.ai/blog/understanding-computer-vision-a-technical-overview/>)

Proces računalnog vida obično započinje prikupljanjem vizualnih podataka, najčešće putem digitalnih kamera ili senzora. Slike se zatim podvrgavaju različitim tehnikama predobrade, koje uključuju prilagodbu veličine, normalizaciju boja, smanjivanje šuma ili povećanje kontrasta, s ciljem dobivanja kvalitetnih i konzistentnih ulaznih podataka [9]. Nakon toga slijedi faza ekstrakcije značajki, u kojoj algoritmi izdvajaju vizualne obrasce poput rubova, oblika ili tekstura, a u suvremenim pristupima te značajke uče se automatski korištenjem dubokih neuronskih mreža. Na slici Slika 2‑9 prikazano kako neuronska mreža prepoznaje značajke sa slika. Konačna faza jest klasifikacija ili detekcija objekata, u kojoj model donosi odluku kojoj klasi pripada određeni uzorak ili gdje se u slici nalazi željeni objekt.

Razvoj računalnog vida značajno je unaprijeđen pojavom dubokog učenja, a posebno konvolucijskih neuronskih mreža. One su omogućile automatsko učenje hijerarhijskih značajki iz podataka, čime je smanjena potreba za ručnim dizajnom algoritama i omogućeno postizanje rezultata koji u mnogim zadacima nadmašuju ljudsku razinu prepoznavanja. Osim toga, tehnike poput proširivanja podataka dodatno poboljšavaju učinkovitost modela stvaranjem varijacija iz postojećih slika, čime se povećava robusnost mreže i smanjuje rizik od pretreniranja.



Slika ‑ Prikaz kako neuronska mreža vidi različite značajke sa slike (preuzeto s <https://www.nvidia.com/en-gb/glossary/computer-vision/>)

Primjene računalnog vida vrlo su široke i obuhvaćaju područja od medicinske dijagnostike, gdje se koristi za otkrivanje tumora ili analizu radioloških snimaka, do autonomnih vozila koja zahtijevaju pouzdano prepoznavanje objekata u stvarnom vremenu. U poljoprivredi računalni vid omogućuje klasifikaciju biljnih vrsta, otkrivanje bolesti ili nadzor usjeva, dok je u ekologiji i šumarstvu važan alat za identifikaciju drveća, praćenje bioraznolikosti i nadzor okoliša [10]. Upravo zbog tih mogućnosti računalni vid predstavlja ključnu tehnologiju za razvoj sustava koji čine prepoznavanje i klasifikaciju biljaka dostupnijima i učinkovitijima široj javnosti.

## Povezna istraživanja i postojeći sustavi

Razvoj dubokog učenja i računalnog vida u posljednjem desetljeću potaknuo je niz istraživanja usmjerenih na automatsku identifikaciju biljnih vrsta. Brojna znanstvena djela pokazala su da konvolucijske neuronske mreže postižu vrlo dobre rezultate u zadacima klasifikacije biljaka i drveća, nadmašujući klasične metode koje su se oslanjale na ručno dizajnirane značajke poput oblika lista ili boje [11]. Na primjer, Hughes i Salathé (2015) koristili su CNN modele za detekciju biljnih bolesti na slikama lišća te postigli visoku razinu točnosti čak i u slučajevima kada su vizualne razlike između zdravih i oboljelih listova bile minimalne [12]. Slično tome, Lee i sur. (2017) demonstrirali su primjenjivost dubokog učenja na prepoznavanje različitih vrsta drveća na temelju fotografija krošnji, čime su potvrdili potencijal ove tehnologije u šumarstvu [13].

U području poljoprivrede i ekologije razvijene su i specijalizirane baze podataka poput *PlantCLEF* i *PlantVillage*, koje su postale standard za treniranje i evaluaciju modela u ovom području [14]. Ove baze sadrže stotine tisuća označenih slika biljaka i drveća, omogućujući istraživačima da testiraju različite arhitekture neuronskih mreža i uspoređuju njihove performanse.

Paralelno s akademskim istraživanjima, razvijen je i niz komercijalnih i otvorenih mobilnih aplikacija za identifikaciju biljaka. Među najpoznatijima ističu se *PlantNet*, *PlantSnap* i *Google Lens*. *PlantNet* se temelji na suradničkom prikupljanju podataka te omogućuje korisnicima da fotografijama doprinose rastu bazi slika i poboljšanju sustava [15]. *PlantSnap* koristi unaprijed istrenirane modele dubokog učenja za prepoznavanje više od 600.000 biljnih vrsta, dok *Google Lens* integrira prepoznavanje biljaka u širi sustav računalnog vida za različite objekte[16].

# IZRADA BAZE PODATAKA

## Ciljevi i dizajn baze podataka

Izrada pouzdane i reprezentativne baze podataka jedan je od ključnih koraka u razvoju modela strojnog učenja za klasifikaciju biljaka. Kvaliteta, raznolikost i veličina skupa podataka izravno utječu na točnost i sposobnost generalizacije istreniranog modela. Budući da ne postoji javno dostupan skup podataka koji bi obuhvatio ciljane biljne vrste relevantne za ovo istraživanje, bilo je nužno konstruirati vlastitu bazu posebno prilagođenu zadatku klasifikacije.

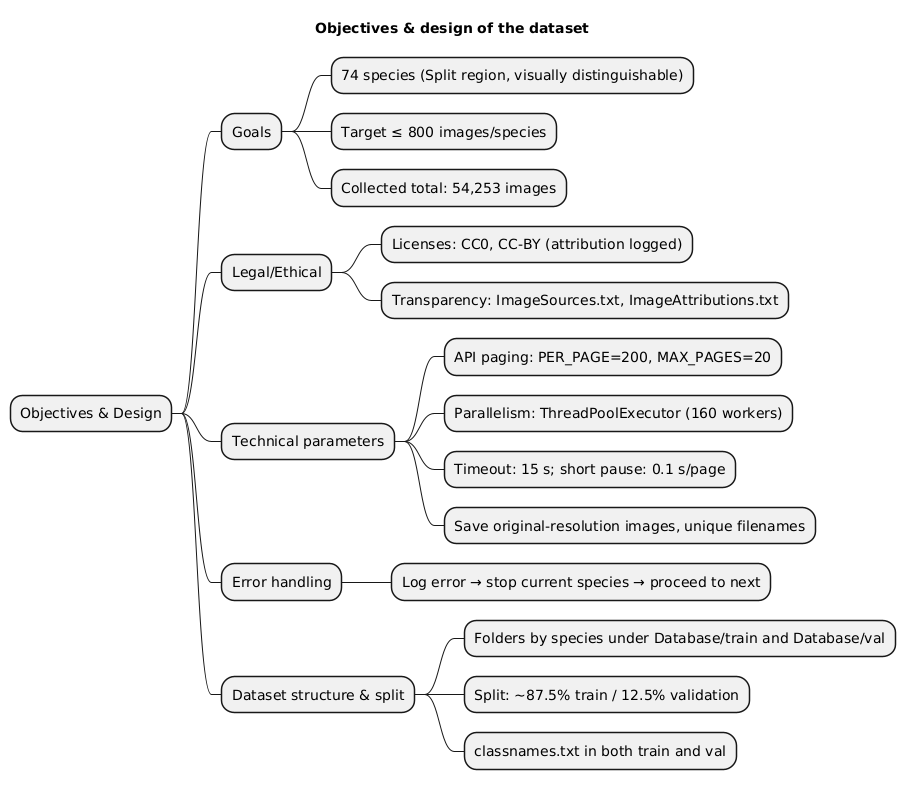
Baza izrađena u sklopu ovog rada obuhvaća 74 vrste biljaka i drveća, većinom onih koje se često nalaze na području Splita i koje su međusobno jasno vizualno prepoznatljive. Kriterij odabira vrsta bio je njihova lokalna rasprostranjenost te praktična upotrebljivost za kasniju automatiziranu identifikaciju. Cilj je bio prikupiti do 800 slika po vrsti, kako bi se obuhvatila dovoljna varijabilnost u osvjetljenju, kutu snimanja, godišnjem dobu i pozadini. Nakon provedbe čitavog postupka prikupljanja i filtriranja, konačna baza sadržava ukupno 54253 slike, što čini dovoljno velik i raznolik skup podataka za treniranje i validaciju modela dubokog učenja.

Fotografije su preuzete putem *iNaturalist* API-ja, jedne od najvećih globalnih platformi za dokumentiranje bioraznolikosti. Kako bi se osigurala etička i pravna ispravnost, korištene su isključivo slike pod licencama CC0 i CC-BY. Kod fotografija pod licencom CC-BY sve zasluge, ime autora i poveznica na izvor, zabilježene su u posebnu datoteku ImageAttributions.txt, dok je u datoteci ImageSources.txt zabilježen popis svih URL-ova, uz oznaku pripadaju li skupu za treniranje ili skupu za validaciju. Na taj je način zajamčena transparentnost i ponovljivost čitavog procesa.

S tehničke strane, sustav je dizajniran s naglaskom na učinkovitost i otpornost na greške. API se dohvaćao stranica po stranica, s maksimalno 200 rezultata po stranici i do 20 stranica po vrsti. Proces preuzimanja slika bio je paraleliziran korištenjem 160 niti (engl. *Threads*), što je omogućilo brzo i učinkovito prikupljanje velikog broja slika. Svaka slika spremljena je u izvornoj rezoluciji, pod jedinstvenim imenom u odgovarajući direktorij vrste. Mehanizmi za obradu pogrešaka osigurali su da se eventualni problemi poput nedostupni server ili istek vremena zahtjeva pravilno zabilježe, nakon čega se preuzimanje za tu vrstu prekida i prelazi na sljedeću. Time je spriječeno zadržavanje cijelog procesa na jednoj problematičnoj vrsti.

Prikupljene slike organizirane su u strukturirane direktorije, a baza je kasnije podijeljena na dva skupa, jedan za treniranje i jedan za validaciju. Otprilike 87,5 % slika dodijeljeno je za treniranje, dok je 12,5 % izdvojeno za validaciju. U obje skupine automatski je generirana datoteka classnames.txt s popisom svih 74 vrste, čime se osigurava konzistentnost između baze i modela tijekom faze treniranja.

Ovaj odjeljak definira ciljeve, obuhvat i smjernice dizajna baze podataka. U sljedećim odjeljcima detaljnije će se prikazati izvori podataka, tehnička implementacija sustava za preuzimanje slika, organizaciju i strukturu baze, te konačni rezultati njezine izrade, a na slici Slika 3‑1 vidljiva je umna mapa svega spomenutog u ovom odjeljku.

[](//www.plantuml.com/plantuml/png/LLF1RjD04BtdA-P2LQAqIKf3AoEaXZPKbOg8I0zmGkFrr5vOxvgTsT1UE29nvHluidu9utLenSNoZjzxyzxC-em5exJMLors8x7Y2DwNNyY8tHB37bJ4jlOGRa4QWWe5cMGq6l-FkiYeqNWCbm4TwyiOJew1Ep9M0VkRpbc1IBKDVW9RomcTkuVAibXV9yiDbewUPzuDnfe47dxzWTFP36oBDV5q9vKHvy4vxKqLI11q1IoE9qUB5ncQ8N1DDRhfIXfhq6NEjJNacRY0y_FPH1y7Rpx1Fef4MoPHM-12NLEryn3HSuUHlBaluAhNtOGK3V6XtCbag2oVoBayweba6jyt1MLZIq9nyBnSNscfrhG5h5SVFg-NbwlNHpErycxvCPysUipGjJ8reBFS5d3JHC9g7O9RtP594YBipr_Eu7k8NrLxPzUs598KC5y0lm9kGXJjbfWAc1tEWQVQcZ9qWrk24AqwGNSGYOFB2NR3cq3ozbiYkBMEFBR3E5SnQjy6VUKqm33FK0FbyiFFty0IEZ0fHlBokFN-GnU3eRmey3ICw6AuGyg9oKYA17jASLQoxjlWAeeCvVsZJl9QoRHIYLE9Q3sebwVIThVaVCqA-75wShXu1WDm2lEZ_gGOMs4VDKEDGsODE6mK55W6QU19lDSy8r_z-pV-0W00)

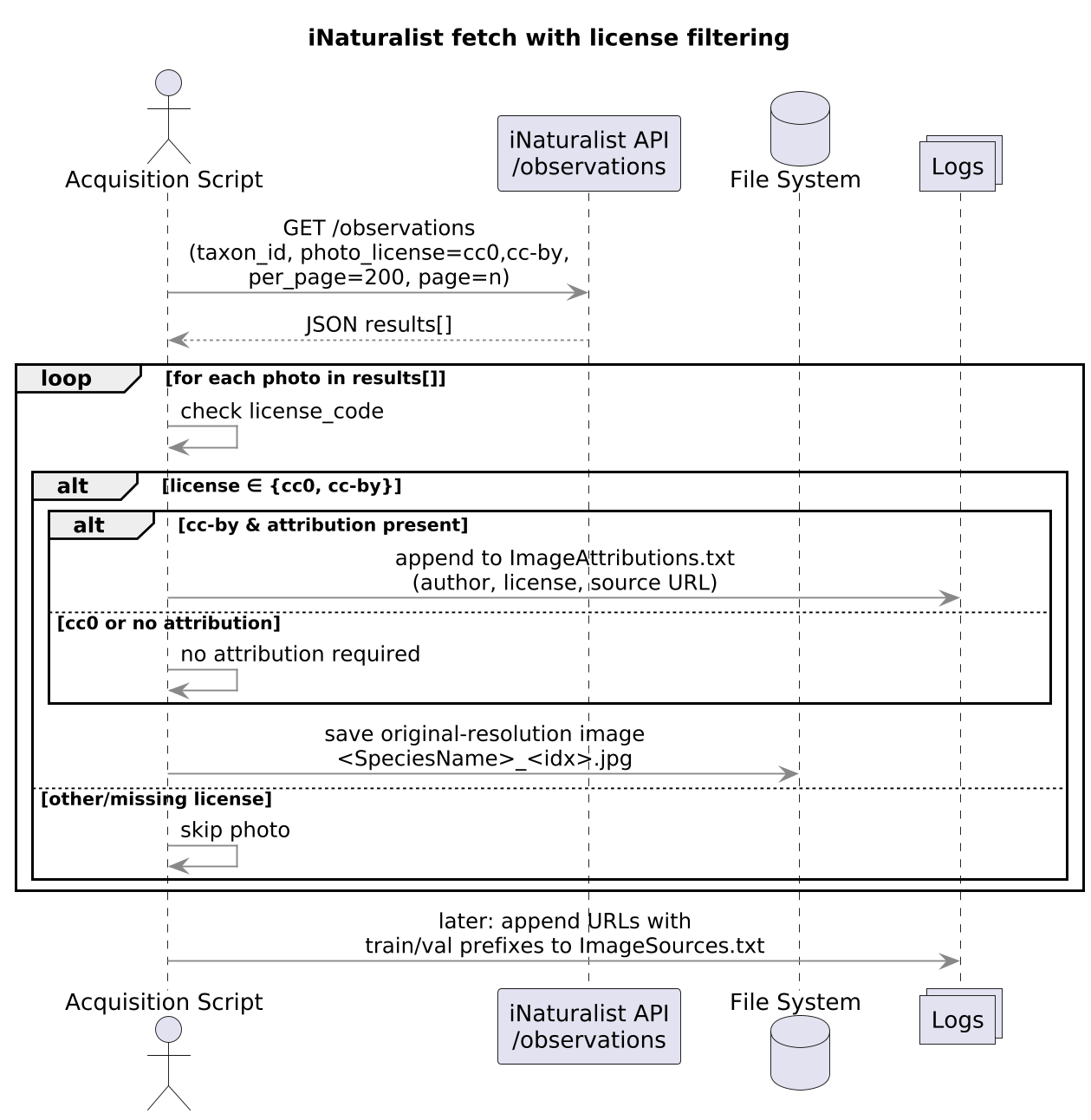
Slika ‑ Sažetak ciljeva, pravnih okvira, tehničkih parametara i planirane strukture baze podataka (izrađeno u web editoru <https://plantuml.com/>)

## Izvori podataka

Primarni izvor slika u ovom radu jest platforma *iNaturalist*, dostupna putem javnog *REST API*-ja koji omogućuje dohvat opažanja biljnih i životinjskih vrsta zajedno s pripadajućim metapodacima. Odabir *iNaturalista* motiviran je opsegom i kvalitetom dostupng sadržaja te postojanjem strukturiranih polja koja su ključna za istraživačku upotrebu, uključujući naziv vrste, poveznice na fotografije u više rezolucija, kod licence i podatke o autorima. Takva organizacija podataka omogućuje transparentnu i ponovljivu izgradnju skupa, uz precizno praćenje podrijetla svake pojedinačne slike.

Pristup podacima ostvaren je pozivima na krajnju točku, *observations*, uz korištenje parametara koji su definirani u sklopu skripte za akviziciju. Za svaku vrstu iz unaprijed pripremljenog popisa, pohranjenog u datoteci *SpeciesList.py*, dohvaćaju se opažanja filtrirana po taksonomskom identifikatoru i ograničena na fotografije s prihvatljivim licencama. Paginacija je namještena tako da se po stranici preuzima do dvjesto zapisa, dok je ukupan broj stranica po vrsti ograničen na dvadeset, čime se postiže razumna gornja granica preuzetih slika i kontrolira opterećenje udaljene usluge. Kako bi se smanjio rizik od prekoračenja ograničenja poslužitelja i ublažile kratkotrajne nestabilnosti mreže, između uzastopnih zahtjeva uvodi se vrlo kratka odgoda, a svaki zahtjev ima podešen vremenski prag nakon kojega se smatra neuspjelim. U slučaju pogreške zapisuje se opis problema i dohvat se za tu vrstu prekida, pri čemu proces odmah nastavlja s idućom vrstom, što omogućuje robusno i učinkovito prikupljanje bez zastoja.

Korištene su isključivo fotografije pod licencama CC0 i CC-BY. Licenca CC0 podrazumijeva odricanje nositelja prava i stavljanje djela u javno vlasništvo, što omogućuje slobodno korištenje i prerade bez dodatnih ograničenja. Licenca CC-BY dopušta široku upotrebu, ali uz obvezno navođenje autorstva, iz tog je razloga svaka slika s ovom licencom, kada su podaci dostupni, popraćena atribucijom koja uključuje ime autora i izvorni URL. Ove su informacije sustavno zapisane u datoteku ImageAttributions.txt, dok se potpuni popis svih URL-ova koristi u datoteci ImageSources.txt, u kojoj je za svaki zapis naznačeno pripada li trenirajućem ili validacijskom skupu. Izbor upravo ovih licenci postavlja jasne pravne okvire i uklanja neizvjesnosti oko daljnje znanstvene i obrazovne upotrebe, restriktivnije licence koje uvjetuju, primjerice, nekomercijalnu upotrebu ili zabranu prerada nisu obuhvaćene, kako bi se osigurala potpuna kompatibilnost s postupkom treniranja, evaluacije i kasnijeg prikaza rezultata. Na slici Slika 3‑2 se vidi kako skripta dohvaća i filtrira slike po licencama.



Slika ‑ Tijek dohvaćanja i filtriranja po licenci (izrađeno u web editoru <https://plantuml.com/>)

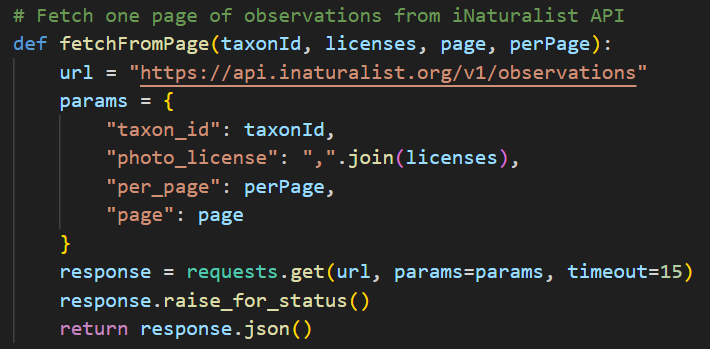
S obzirom na kvalitetu ulaznih podataka, slike se preuzimaju u izvornoj rezoluciji kada je to moguće, pri čemu se na temelju metapodataka formira izravna poveznica na izvornu varijantu fotografije. Svaka se datoteka pohranjuje pod jednoznačnim nazivom koji uključuje ime vrste i indeks, u pripadajući direktorij te vrste, čime se osigurava konzistentnost i lako pretraživanje. Na kraju postupka prikupljanja, cjelokupan popis izvora i pripadne atribucije omogućuju potpunu sljedivost skupa te olakšavaju naknadnu rekonstrukciju baze na temelju istih URL-ova, što je detaljnije opisano u posebnom pododjeljku o rekonstrukciji.

Opisani pristup spaja tehničku učinkovitost s pravnom i metodološkom urednošću. *iNaturalist* API pruža standardiziran i bogat skup metapodataka, čime se omogućuje dosljedno filtriranje po licenci i vrsti, kontrolirano preuzimanje velikog broja slika uz paginaciju i pauze između zahtjeva, te sustavno bilježenje izvora i atribucija. Rezultat je pouzdan i transparentan temelj na kojem se u sljedećim pododjeljcima gradi organizacija direktorija, nasumična podjela na trenirajući i validacijski dio te cjelokupna reproducibilnost izrade baze.

## Tehnička izvedba prikupljanja slika

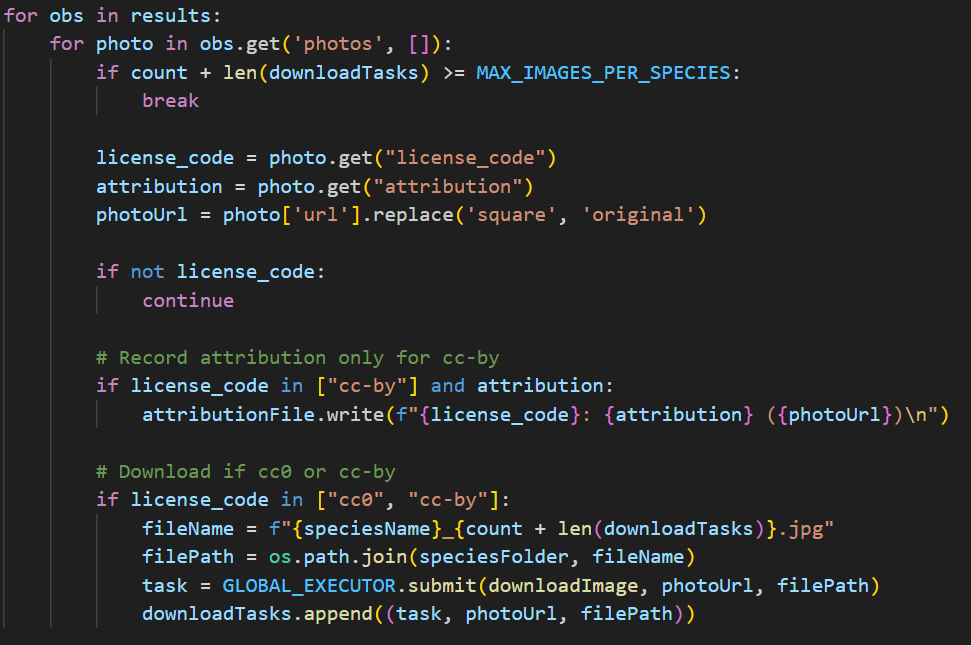
Postupak akvizicije provodi se skriptom u programskom jeziku *Python* koja nad *iNaturalist* API-jem izvršava seriju paginiranih upita te asinkrono preuzima fotografije prema unaprijed zadanim ograničenjima. Ulaz u proces jest popis ciljnih vrsta zapisan s njihovim taksonomskim identifikatorima unutar *SpeciesList.py* datoteke, dok su izlaz strukturirani direktoriji s datotekama slika i popratnom dokumentacijom izvora i atribucija. Fokus implementacije je na učinkovitosti preuzimanja i reproducibilnosti, uz jasna pravila filtriranja licenci i determinističko imenovanje datoteka.

Komunikacija s API-jem ostvaruje se putem krajnje točke observations. Za svaku vrstu generira se upit s parametrima *taxon\_id*, *photo\_license*, *per\_page* i *page*, pri čemu *photo\_license* eksplicitno ograničava rezultate na CC0 i CC-BY. Vrijednosti paginacije odabrane su tako da se preuzme do 200 zapisa po stranici, uz najviše 20 stranica po vrsti. Svaki zahtjev ima vremensko ograničenje od 15 sekundi, a između uzastopnih stranica uvodi se kratka odgoda kako bi se smanjilo opterećenje udaljene usluge i izbjegla neželjena ograničenja na strani poslužitelja. Slika ispod pokazuje način formiranja i provjere odgovora.



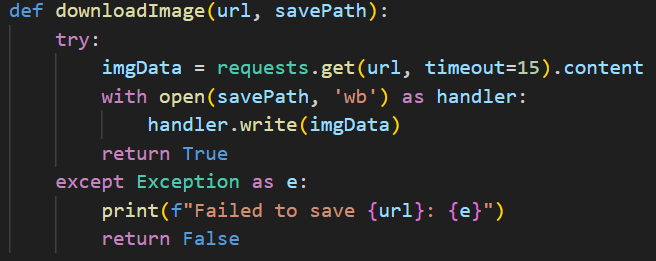
Slika ‑ Formiranja i provjere odgovora

Akvizicija je organizirana po vrstama, za svaku vrstu otvara se ciljna mapa na putanji *Database/train/<NazivVrste>* i pokreće iteracija kroz stranice i opažanja sve dok se ne dostigne gornja granica od najviše 800 slika po vrsti, ne iscrpe rezultati ili ne nastupi greška. Iz svakog opažanja čitaju se metapodaci fotografija. Ako fotografija nema odgovarajući kod licence, preskače se, u suprotnom se priprema URL izvorne rezolucije zamjenom segmenta *square* s *original*, čime se izbjegavaju umanjene inačice. Fotografije s licencom CC-BY dodatno prate atribucije autora, koje se zapisuju u *Sources/ImageAttributions.txt* zajedno s URL-om kako bi kasnije citiranje bilo jednoznačno i provjerljivo. Na slici Slika 3‑4 prikazan je sažetak logike obrade jednog opažanja.



Slika ‑ Sažetak programske logike za jedno opažanje

S ciljem ubrzanja procesa koristi se globalni spremnik niti, *ThreadPoolExecutor*, s najviše 160 paralelnih zadataka kako bi se u potpunosti iskoristila brzina mreže od 1gigabita po sekundi. Preuzimanje je granulirano na razinu datoteke i svaka se slika zapisuje pod jedinstvenim imenom oblika *<NazivVrste>\_<indeks>.jpg*. Ovako formirani nazivi omogućuju determinističko adresiranje primjera te olakšavaju kasnije rekonstruiranje i provjeru. Nakon što se grupa zadataka preuzimanja preda izvršitelju, glavna nit čeka dovršetak tih zadataka prije prelaska na sljedeću stranicu ili zaključivanja vrste, čime se postiže kontrolirano korištenje mrežnih i diskovnih resursa. Funkcija za preuzimanje slika koja provodi osnovno rukovanje pogreškama i vraća uspješnost vidljiva je na slici Slika 3‑5.



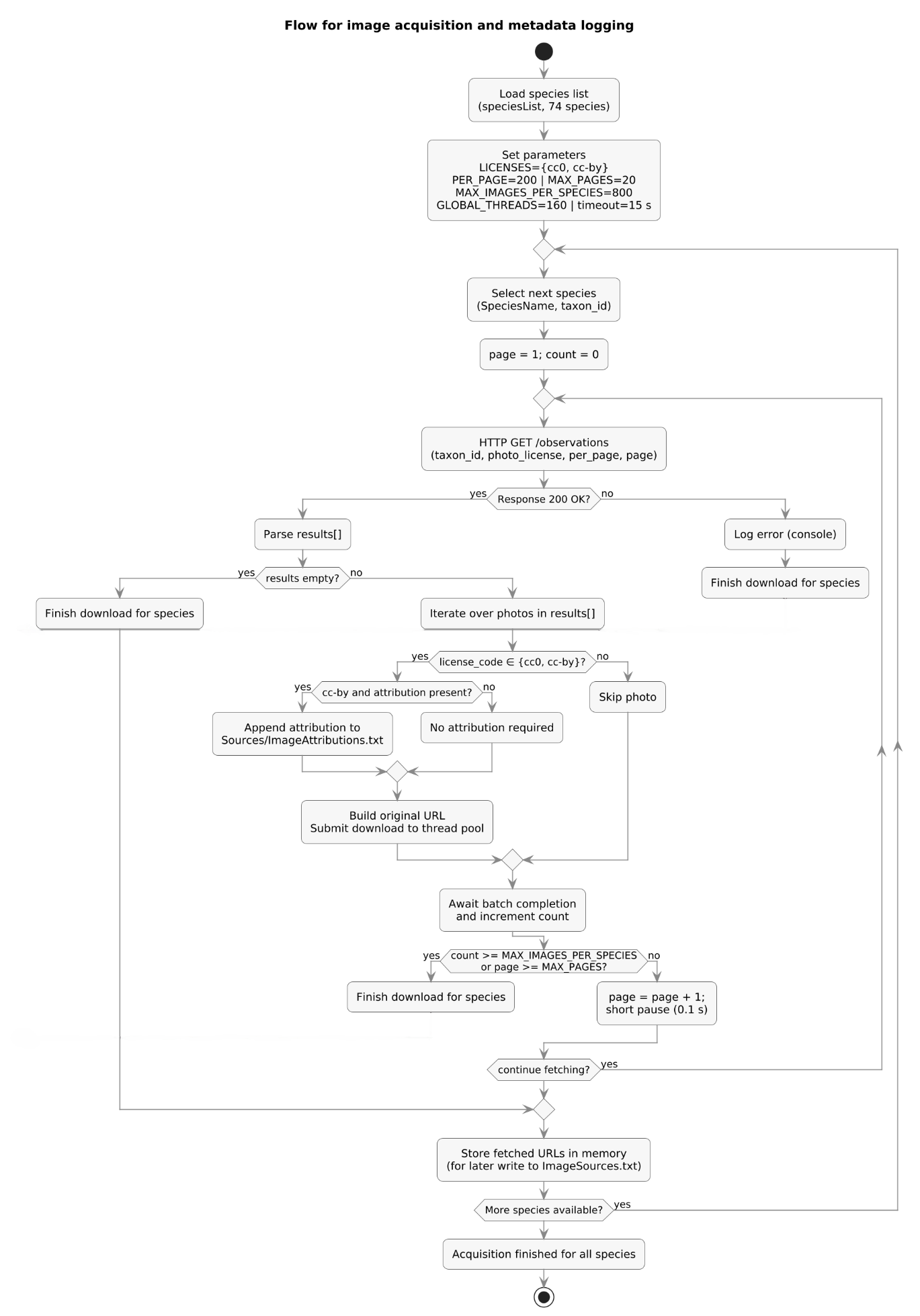
Slika ‑ Funkcija za preuzimanje slika

Rukovanje pogreškama oslanja se na uzorak „zabilježi i nastavi s idućom jedinicom posla“. U slučaju iznimke pri dohvaćanju stranice, skripta zapisuje opis greške i prekida akviziciju za tu vrstu, odmah prelazeći na sljedeću. Takav kompromis izbjegava zaglavljenja i nepotrebne ponovne pokušaje u uvjetima povećane nestabilnosti. Istodobno, budući da se u *Sources/ImageSources.txt* naknadno zapisuje točan popis URL-ova po vrsti, omogućeno je kasnije ciljano ponavljanje neuspješnih preuzimanja u zasebnoj fazi, kao i potpuna rekonstrukcija baze na temelju logova. Završetkom obrade jedne vrste, sve uspješno preuzete slike ostaju u njezinoj mapi unutar *train*, a metapodaci su spremni za sljedeći korak nasumične podjele na trenirajući i validacijski skup, o čemu je detaljnije riječ u odjeljku 3.4.

Posebna pažnja posvećena je transparentnosti i ponovljivosti. Svaki URL koji je doista korišten zapisuje se s oznakom pripada li trenirajućem ili validacijskom skupu, a atribucije za CC-BY bilježe se odvojeno. Time se postiže sljedivost od modela natrag do izvora slike i autora, što je važno i u akademskom i u produkcijskom okruženju. Imenovanje datoteka i struktura direktorija pojednostavljuju ručne inspekcije i skriptirano pretraživanje, a ograničenja poput maksimalnog broja slika po vrsti i maksimalnog broja stranica štite od neželjenog rasta baze ili pretjeranog opterećenja udaljene usluge.

U pogledu optimizacija, trenutna verzija akvizicije koristi *requests.get(...).content* za preuzimanje, što je praktično i dovoljno robusno za prosječne veličine slika; alternativno strujanje (*stream=True*) i zapis u komadima, kakvo se primjenjuje u skripti za rekonstrukciju, može se uvesti i ovdje kada se očekuju iznadprosječno velike datoteke ili sporiji diskovni sustavi. Dodatna poboljšanja mogla bi uključivati heuristike za deduplikaciju na temelju sažetaka, filtar minimalne rezolucije prije uvrštavanja u skup te ograničenja po hostu radi ravnomjernije potrošnje mrežnih resursa; takve su nadogradnje kompatibilne s postojećom arhitekturom i mogu se dodati bez promjene protokola s API-jem.

Na kraju ovog koraka, baza podataka postoji u „prolaznom“ obliku u kojem su sve slike pohranjene unutar *Database/train/<NazivVrste>,* a logovi izvora i atribucija su aktualni. Sljedeći odjeljak obrađuje formalnu podjelu na trenirajući i validacijski skup, fizičko premještanje odabranih primjera u *Database/val/<NazivVrste>* i generiranje pomoćnih datoteka za rad s modelom. Dijagram toka skripte može se vidjeti na slici Slika 3‑6.

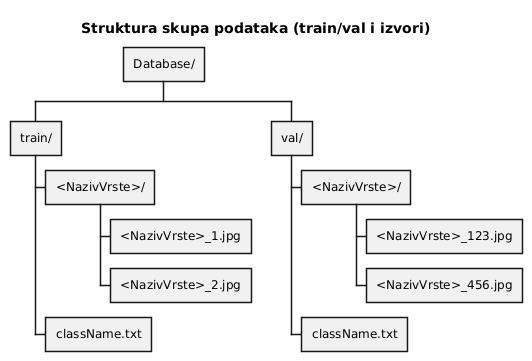


Slika ‑ Dijagram toka za izradu baze podataka (izrađeno u web editoru <https://plantuml.com/>)

## Organizacija i podjela podataka

Organizacija skupa podataka izvedena je tako da neposredno podržava tipičan tijek treniranja modela računalnog vida i da ostane maksimalno transparentna i ponovljiva. Svi uzorci razvrstani su po klasama na razini vrste te fizički razdvojeni u dio za treniranje i dio za validaciju. Time se izbjegava nenamjerno „curenje“ podataka između faza treniranja i provjere, a istodobno se zadržava jasna veza između svake datoteke i pripadne klase.

Struktura direktorija slijedi jednostavan, ali u praksi provjeren raspored. Za svaku je vrstu izrađen poddirektorij unutar mape *Database/train*, a nakon podjele i odgovarajući poddirektorij unutar Database/val. Datoteke slika imenuju se deterministički obrascem *<NazivVrste>\_<indeks>.jpg*, što olakšava ručnu inspekciju i skriptirano pretraživanje te sprječava kolizije imena. U korijenu i trenirajućeg i validacijskog dijela automatski se generira datoteka classnames.txt s popisom svih sedamdeset i četiri klase. Ova lista služi kao jedini izvor istine za mapiranje redoslijeda klasa na indekse etiketa tijekom treniranja i evaluacije te omogućuje potpunu dosljednost između skupa podataka i modela. Struktura direktorija prikazana je na slici Slika 3‑7.



Slika ‑ Struktura skupa podataka (izrađeno u web editoru <https://plantuml.com/>)

Podjela na trenirajući i validacijski dio provedena je nasumično, zasebno za svaku vrstu, kako bi se očuvala proporcionalnost klasa i izbjegla sistemska pristranost. Odabrani omjer iznosi približno 87,5 % za treniranje i 12,5 % za validaciju, pri čemu se vodi računa da svaka klasa ima barem jedan primjer u validacijskom skupu, čak i kada je broj dostupnih slika malen. Implementacijski, odabir validacijskih uzoraka radi se nad popisom već preuzetih datoteka za pojedinu vrstu, a zatim se odabrane datoteke fizički premještaju iz trenirajuće mape u odgovarajuću validacijsku mapu. Takva fizička separacija onemogućuje slučajno korištenje istih datoteka u obje faze te pojednostavljuje konfiguraciju alata za treniranje koji očekuju konvencionalan raspored mapa po klasi. Funkcija za podjelu na dio za treniranje i dio za validaciju vidljiv je na slici Slika 3‑8.



Slika ‑ Funkcija za raspodjelu slika u na set za treniranje i set za validaciju

Kako bi se očuvala sljedivost i omogućila rekonstrukcija, sve iskorištene poveznice na izvorne datoteke bilježe se u *Sources/ImageSources.txt* uz prefiks koji naznačuje pripadnost trenirajućem ili validacijskom dijelu. Ovaj manifest nastaje nakon podjele i predstavlja točan presjek skupa podataka u trenutku treniranja. U zasebnoj datoteci *Sources/ImageAttributions.txt* evidentiraju se i atribucije za fotografije pod licencom CC-BY, čime se zadovoljavaju pravni i etički zahtjevi te omogućuje naknadna revizija izvora. U slučaju ponovne izgradnje skupa na drugom računalu ili nakon gubitka podataka, ovi zapisi služe kao jedini potrebni ulaz, što je u potpunosti opisano u poglavlju o rekonstrukciji.

Posebna je pozornost posvećena integritetu i izbjegavanju curenja podataka. Budući da se validacijski uzorci biraju iz već formiranog skupa datoteka za pojedinu vrstu, ista se slika ne može pojaviti u obje particije. Ipak, kada isti izvor sadrži više vizualno vrlo sličnih kadrova, preporučljivo je u budućim iteracijama uvesti dodatne provjere, primjerice deduplikaciju na temelju sažetaka (*hash*) ili prag minimalne rezolucije, kako bi se otkrile i isključile degradirane ili duplicirane datoteke. Isto tako, iako trenutna podjela ne provodi eksplicitno balansiranje klasa, neravnoteža se u praksi može ublažiti tehnikama obogaćivanja podataka u fazi treniranja ili ponderiranjem gubitka po klasi, svjesno je odlučeno da se balansiranje ne uvodi u fazi organizacije skupa podataka kako bi se zadržala vjernost realnoj raspodjeli dostupnih primjera.

U implementaciji se vodi računa o idempotentnosti i otpornosti na prekide. Kreiranje odredišnih direktorija tolerantno je na postojeće mape, a premještanje datoteka obavlja se atomarno gdje god je to moguće. U slučaju prekida procesa, ponovno pokretanje neće duplicirati već premještene datoteke niti će promijeniti već donesene odluke o pripadnosti trenirajućem ili validacijskom dijelu, budući da se one odražavaju u manifestu URL-ova. Ovakav pristup pojednostavljuje verzioniranje skupa i omogućuje usporedivost eksperimenata, jer različite konfiguracije treniranja mogu referencirati isti, stabilizirani presjek podataka.

Opisani raspored direktorija, determinističko imenovanje datoteka, eksplicitna evidencija izvora i nasumična, po-klasna podjela s minimalnim brojem validacijskih primjera čine cjelinu koja je odmah upotrebljiva u standardnim trenirnim okruženjima i lako prenosiva između računala. U sljedećem poglavlju o treniranju modela ova će struktura biti izravno korištena za izgradnju i procjenu klasifikatora.

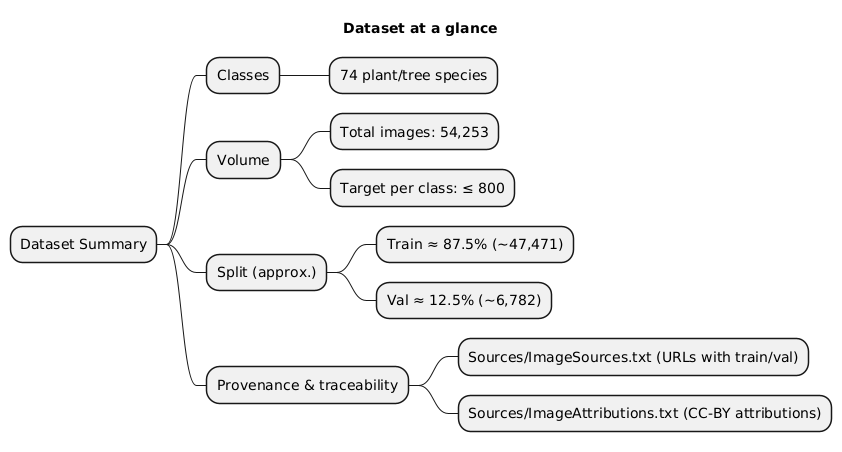
## Rezultati izrade baze

Provedenim postupkom akvizicije i organizacije oblikovana je baza podataka prilagođena zadatku klasifikacije biljaka i drveća u lokalnom kontekstu. Baza obuhvaća ukupno 74 vrste, za koje je prikupljeno 54 253 fotografija. Ciljani prag od najviše 800 slika po vrsti nije uvijek dosegnut zbog ograničene dostupnosti ili licencnih uvjeta pojedinih zapisa, no ostvareni ukupni obujam i raznolikost vizualnih uvjeta, različita osvjetljenja, pozadine, kutovi snimanja i godišnja doba, čine skup dostatno bogatim za treniranje i procjenu modernih konvolucijskih arhitektura.

Podaci su podijeljeni na trenirajući i validacijski dio sukladno omjeru određenom u skripti. U trenirajući skup raspoređeno je približno 47 471 slika (oko 87,5 %), a u validacijski 6 782 slike (oko 12,5 %). Promatrano prosječno po klasi, to odgovara približno 642 primjera po vrsti u trenirajućem skupu i 92 primjera po vrsti u validacijskom skupu, uz napomenu da je raspodjela po klasama neujednačena jer odražava realnu dostupnost izvornog materijala. Nasumična podjela provedena je zasebno za svaku vrstu, pri čemu je osigurano da svaka klasa ima barem jedan primjer u validacijskom skupu te da ne dolazi do preklapanja istih datoteka između particija.

Cjelokupna izrada baze popraćena je pomoćnim artefaktima koji osiguravaju sljedivost i ponovljivost. Datoteka *Sources/ImageSources.txt* sadrži popis svih URL-ova korištenih slika s oznakom pripadnosti trenirajućem ili validacijskom skupu, dok *Sources/ImageAttributions.txt* evidentira potrebne atribucije za zapise pod licencom CC-BY. Ovakav manifest omogućuje reviziju izvora i, prema potrebi, potpunu rekonstrukciju baze na drugom sustavu bez promjene sastava podataka. Ujedno, determinističko imenovanje datoteka i dosljedna struktura direktorija olakšavaju ručnu inspekciju i automatizirano korištenje u treniranju. Pregled baze podataka može se vidjeti na slici Slika 3‑9.

Ako su tijekom akvizicije prisutne uobičajene tehničke teškoće kao što su povremeni prekidi veze ili nedostupni resursi, mehanizmi za bilježenje pogrešaka i tolerantno ponašanje skripte sprječavaju zastoj postupka i minimaliziraju gubitak podataka. Potencijalna buduća poboljšanja obuhvaćaju deduplikaciju na razini sažetaka, pragove minimalne rezolucije i jednostruko balansiranje klasa prije treniranja; ta su unaprjeđenja kompatibilna sa sadašnjom arhitekturom i ne mijenjaju zaključak da je izrađena baza dovoljno velika, raznolika i dobro dokumentirana za daljnju eksperimentalnu evaluaciju modela u sljedećem poglavlju.

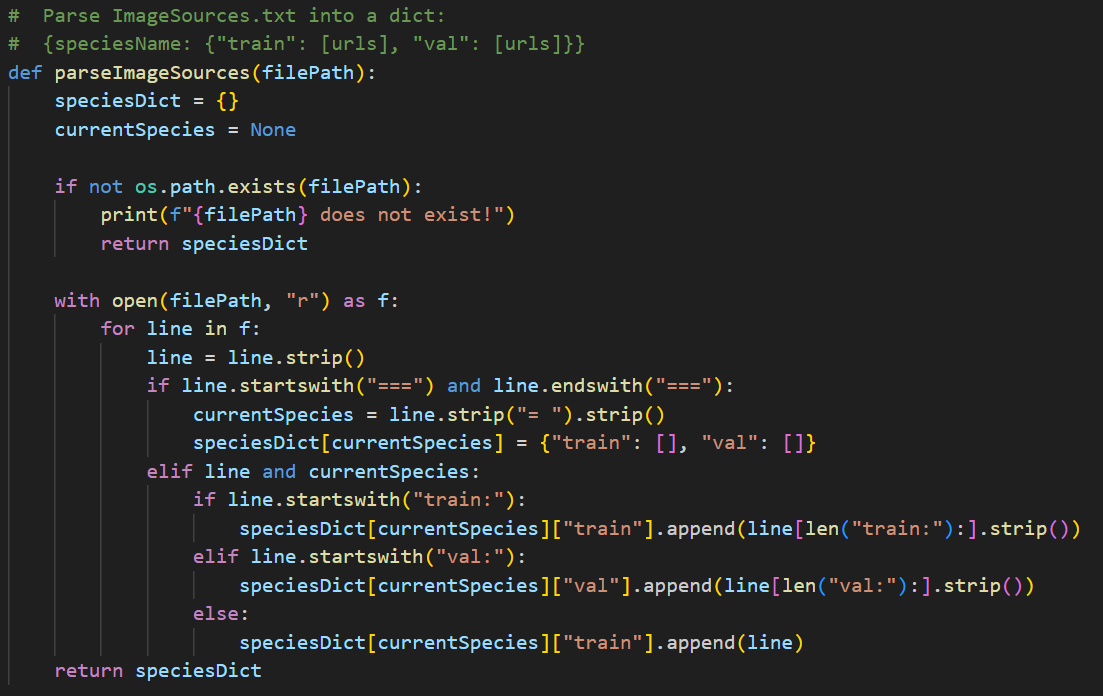


Slika ‑ Pregled baze podataka (izrađeno u web editoru <https://plantuml.com/>)

## Rekonstrukcija baze podataka

Rekonstrukcija baze podataka osigurava ponovljivost, prenosivost i dugoročnu održivost rezultata ovog rada. Polazišna pretpostavka jest da su tijekom akvizicije i organizacije podataka, kako je opisano u odjeljcima 3.3 i 3.4, generirani manifesti izvora i atribucija, *ImageSources.txt* i *ImageAttributions.txt*. Na temelju tih zapisa, bez ponovnog pozivanja *iNaturalist* API-ja, moguće je na novom sustavu ili nakon gubitka podataka ponovno izgraditi doslovno isti raspored direktorija uz isti presjek skupova, čime se jamči da će kasniji eksperimenti nad rekonstruiranom bazom dati usporedive rezultate.

Skripta za rekonstrukciju polazi od datoteke ImageSources.txt, koja je strukturirana po vrstama i particijama. Naslove odjeljaka označene linijama u obliku „=== NazivVrste ===“ skripta interpretira kao početak bloka za pojedinu vrstu, dok se u nastavku nalaze redci s prefiksima „train:“ i „val:“ koji sadrže izvorne URL-ove slika dodijeljenih trenirajućem i validacijskom skupu. Parsiranjem ove datoteke gradi se u memoriji preslikavanje vrste na dva popisa poveznica, čime se očuva ista podjela koja je korištena pri treniranju modela. Budući da je ImageAttributions.txt već nastao u fazi akvizicije, njegova se uloga u rekonstrukciji svodi na dokumentiranje pravnih i etičkih metapodataka. Rekonstrukcija ne dohvaća atribucije iz izvora nego samo ponovno preuzima slikovne sadržaje. Funkcija za parsiranje datoteke s URL-ovima prikazana je na slici Slika 3‑10.

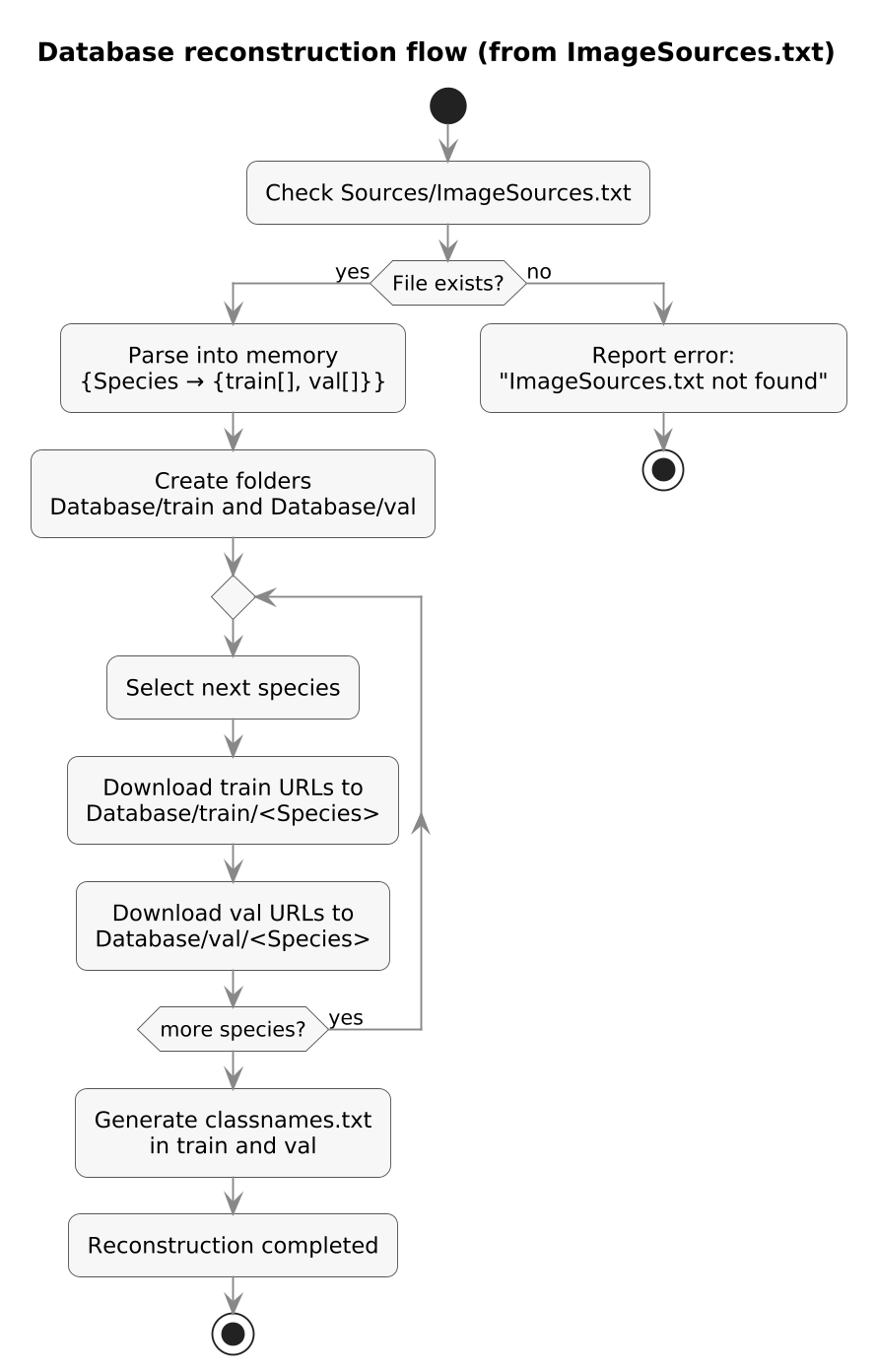


Slika ‑ Funkcija za parsiranje datoteke sa izvorima slika

Preuzimanje se provodi paralelno, uz korištenje spremnika niti s visokim stupnjem usporednosti, kako bi se proces značajno ubrzao i učinkovitije iskoristili mrežni i diskovni resursi. Za razliku od inicijalne akvizicije, gdje su slike preuzimane preko API-ja, ovdje se radi o izravnom dohvaćanju već poznatih URL-ova, uz strujni zapis na disk i fragmentiranje u manje blokove podataka. Takav način zapisa smanjuje potrošnju memorije i povećava robusnost u slučaju sporijih veza ili većih datoteka. Svaka je slika deterministički imenovana prema uzorku *<NazivVrste>\_<indeks>.jpg* i smješta se u odgovarajući direktorij ovisno o pripadnosti trenirajućem ili validacijskom skupu, čime se dosljedno reproducira struktura *Database/train/<NazivVrste>* i *Database/val/<NazivVrste>*.

Skripta je osmišljena tako da bude idempotentna i nastaviva. Ako datoteka s očekivanim nazivom već postoji, preskače se ponovni download, što omogućuje sigurno ponavljanje procesa nakon prekida ili djelomične rekonstrukcije. Neuspjeli pokušaji preuzimanja evidentiraju se u standardni izlaz s opisom uzroka, poput isteknuće vremena, nedostupan resurs, pogreška poslužitelja, no takvi pojedinačni padovi ne zaustavljaju cjelokupan proces. Rekonstrukcija za tu sliku se preskače, a izvođenje se nastavlja za preostale URL-ove i vrste. Na kraju procesa automatski se ponovno generiraju datoteke *classnames.txt* u korijenu train i val direktorija na temelju popisa vrsta iz *SpeciesList.py*, čime se osigurava usklađenost između rekonstruirane baze i trening skripti koje očekuju te pomoćne datoteke Dijagram toka skripte za rekonstrukciju vidljiv je na slici Slika 3‑11.

Ograničenja ove metode proizlaze iz ovisnosti o vanjskim izvorima: s vremenom pojedini URL-ovi mogu zastarjeti ili postati nedostupni. To ne narušava strukturu niti integritet cjelokupnog skupa, ali može smanjiti broj dostupnih primjera za pojedine vrste. Kao razumna nadogradnja preporučuje se uvesti strategiju ponovnih pokušaja s postupnim čekanjem, provjeru tipa sadržaja i minimalne rezolucije prije spremanja te izračun kriptografskih sažetaka radi detekcije dupliciranih ili oštećenih datoteka. Unatoč tim potencijalnim poboljšanjima, opisani postupak već ispunjava osnovni cilj, pouzdano i doslovno ponovno izgraditi skup podataka na temelju manifest-datoteka, bez ovisnosti o trenutačnom stanju API-ja ili promjenama na izvoru, te time omogućiti potpunu ponovljivost treninga i evaluacije u sljedećim poglavljima.



Slika ‑ Dijagram toka za rekonstrukciju baze podataka (izrađeno u web editoru <https://plantuml.com/>)

# TRENIRANJE I EVALUACIJA MODELA

## Ciljevi treniranja i metodologija

Cilj ovoga poglavlja jest izgraditi i procijeniti model za automatsku klasifikaciju 74 biljnih vrsta na skupu opisanom u poglavlju 3, koji ukupno sadrži 54 253 slika. Primarni kriterij uspjeha je točnost na validacijskom skupu, uz stabilne krivulje učenja bez znakova pretreniranosti. Dodatni, praktični cilj jest da model bude upotrebljiv na mobilnom uređaju, u ovom slučaju Android, pa se tijekom dizajna vodi računa o omjeru točnosti i računalne učinkovitosti.

Metodologija se temelji na nadziranom učenju s već pripremljenom podjelom na train i val unutar strukture baze podataka. Ulazne slike prolaze kroz standardiziran pipeline, u treniranju se primjenjuju prostorne i fotometrijske transformacije koje uvode realistične varijacije, slučajni izrez do ciljnih 300×300 piksela, horizontalna i vertikalna zrcaljenja, rotacije do 30°, blage afine i perspektivne transformacije, prilagodbe svjetline, kontrasta, zasićenja, nijanse te *RandomErasing*, dok je validacijski tok determinističan, promjena veličine, središnji izrez na 300×300, normalizacija. Sve slike normalizirane su na *ImageNet* statistiku, u skladu s inicijalizacijom modela.

Arhitektura klasifikatora je *EfficientNet-B3* inicijalizirana unaprijed naučenim težinama, a završni klasifikacijski sloj zamijenjen je slojem čija dimenzija odgovara broju klasa, 74. Treniranje se provodi križno-entropijskim gubitkom i optimizatorom *AdamW* s korakom učenja 1×10⁻⁴. Maksimalno trajanje iznosi 30 epoha, uz rano zaustavljanje s odgodom od 4 epohe, tj. ako se validacijska točnost ne poboljša kroz četiri uzastopne epohe, proces se prekida. Tijekom svake epohe računa se gubitak u treniranju i validaciji te točnost na validacijskom skupu, zadržavaju se najbolje težine prema validacijskoj točnosti, a po završetku model se vraća na taj optimum. Trening se izvršava na *NVIDIA RTX 4060 (laptop)* grafičkoj kartici uz veličinu paketa 8 i 4 radnika za učitavanje podataka, što je pragmatičan kompromis s obzirom na memorijska ograničenja.

Evaluacija u ovom koraku usredotočena je na vrijednosti gubitka i točnosti po epohi te pripadajuće artefakte koji se automatski generiraju, graf krivulja *TrainingPlot.png* i zapis točnosti po epohi , *ValAccuracyPerEpoch.txt*. Ovi rezultati služe za kvalitativnu provjeru stabilnosti učenja i usporedbu različitih pokretanja. Najbolji model potom se serijalizira i priprema za izvoz u mobilno-prikladan format, što je tema završnih odjeljaka ovoga poglavlja.

## Priprema podataka

Skup podataka učitava se iz direktorijske strukture opisane u poglavlju 3, pri čemu *Database/train/<NazivVrste>* i *Database/val/<NazivVrste>* definiraju klase. Učitavanje je realizirano preko *torchvision.datasets.ImageFolder*, a nazivi klasa dobivaju se iz *trainingDataset.classes* i dalje koriste za oblikovanje klasifikacijske glave i generiranje labels.txt za aplikaciju.

Tijekom treniranja primjenjuje se skup transformacija koje istodobno standardiziraju ulaz i obogaćuju podatke realističnim varijacijama. Slike se stohastički kadriraju na 300×300 piksela preko *RandomResizedCrop(300, scale=(0.8, 1.0))*, zatim slijede horizontalno i vertikalno zrcaljenje te rotacije do 30°. Fotometrijske promjene uvodi *ColorJitter*, svjetlina i kontrast do 0.4, zasićenje 0.3, nijansa 0.1, a geometrijske varijacije *RandomAffine*, translacija do 15 %, skala 0.9–1.1 i *RandomPerspective*, faktor 0.2, vjerojatnost 0.5. Nakon pretvorbe u tenzor (*ToTensor*) provodi se normalizacija na *ImageNet* statistiku, *mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]), a RandomErasing(p=0.3, scale=(0.02, 0.15)*) dodatno simulira parcijalne okluzije i smanjuje prenaučenost na lokalne uzorke.

## Arhitektura modela

Klasifikator je zasnovan na *EfficientNet-B3* iz *torchvision*-a. Inicijalizacija se provodi funkcijom *initializeModel(numClasses, device)*, koja učitava unaprijed istrenirane težine, *EfficientNet\_B3\_Weights.DEFAULT*, te zamjenjuje završni linearni sloj u *model.classifier[1]* novim slojem dimenzije *numClasses*. Time se broj izlaza usklađuje s brojem klasa (74) detektiranim iz *classnames.txt*. Ostatak mreže, konvolucijski backbone i optimizacije *EfficientNeta*, ostaje neizmijenjen kako bi se iskoristile generičke vizualne reprezentacije naučene na velikom skupu, npr. *ImageNet*-u. Model se nakon izmjene premješta na odabrani uređaj, CPU ili GPU, i spreman je za treniranje prema protokolu iz sljedećeg odjeljka. Ulazna rezolucija od 300×300 iz *DataUtility* usklađena je s kapacitetom B3 varijante i omogućuje dobar omjer točnosti i računalne učinkovitosti za kasniju mobilnu primjenu.

U odabiru *EfficientNet-B3* vodilo se računa o ravnoteži između točnosti i računalne učinkovitosti u mobilnom okruženju. Arhitektura koristi ujednačeno skaliranje dubine, širine i rezolucije, što omogućuje da se na ulazu dimenzije 300×300 postigne dobar omjer kapaciteta i troška izvođenja. Prethodno naučene reprezentacije, pretrained težine, služe kao opći vizualni izvor znanja, dok se završna klasifikacijska glava u potpunosti prilagođava ciljnom problemu zamjenom linearnog sloja s 74 izlaza. Time se zadržava snaga backbone-a za ekstrakciju značajki, a prediktor postaje specifičan za biljne vrste u bazi.

## Strategije učenja i postupak treniranja

Strategija učenja u ovom radu temelji se na *end-to-end* finom podešavanju unaprijed istreniranog *EfficientNet-B3* modela na ciljnom skupu od 74 klase. Polazišna točka su težine naučene na općem domenskom skupu , dok se tijekom treniranja ažuriraju svi parametri mreže, uključujući novo dodani završni klasifikacijsku sloj. Takav pristup omogućuje da se postojeće generičke vizualne reprezentacije adaptiraju na specifičnosti biljnih vrsta, a da se pritom zadrži brzina konvergencije karakteristična za transfer učenje.

Ulazni podaci dolaze iz dva tokova, trenirajućeg i validacijskog, konstruiranih u *DataUtility*. U trenirajućem toku primjenjuju se prostorne i fotometrijske transformacije, nasumični izrez na 300×300, zrcaljenja, rotacije, perspektivne deformacije, promjene svjetline, kontrasta, zasićenja, nijanse te *RandomErasing*, dok je validacijski tok determinističan, promjena veličine, središnji izrez na 300×300, normalizacija. U obje faze slike se normaliziraju na *ImageNet* statistiku, što je usklađeno s inicijalizacijom *EfficientNeta*.

Funkcionalno, treniranje je organizirano po epohama, a svaka epoha ima dvije faze, train i val. U fazi treniranja model je u načinu *train()*, izračunava se križno-entropijski gubitak nad izlaznim podatcima, provodi se povratno širenje pogreške i ažuriranje težina optimizatorom *AdamW* s korakom učenja 1×10⁻⁴. U validacijskoj fazi model je u načinu *eval()*, gradijenti su isključeni, a mjeri se gubitak i točnost na zasebnom skupu. Veličina paketa postavljena je na 8, a broj radnika za učitavanje na 4, što je učinjeno na takav način da se maksimalno iskoristi hardver, s obzirom na korištenje *NVIDIA RTX 4060 (laptop)* grafičke kartice i ciljani kompromis između propusnosti i potrošnje memorije.

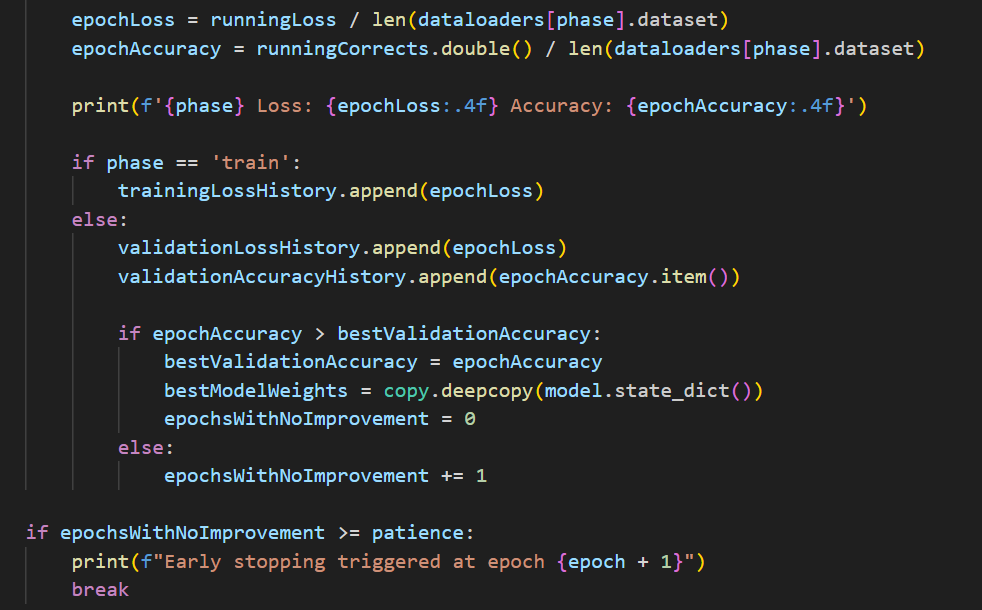
Tijekom svake epohe akumulira se prosječni gubitak po uzorku te broj točnih predikcija iz kojih se računa točnost epohe. U isto vrijeme vodi se povijest metrika, nizovi trenirajućeg i validacijskog gubitka te validacijske točnosti po epohama. Ključan element strategije je rano zaustavljanje s odgodom od četiri epohe, tj. ako se validacijska točnost ne poboljša kroz četiri uzastopne epohe, proces se prekida kako bi se izbjeglo pretreniranje i nepotrebno trošenje računalnih resursa. Paralelno se čuva najbolja instanca modela kada god validacijska točnost ostvari novi maksimum, pohranjuje se duboka kopija trenutnih težina. Po završetku treniranja model se vraća upravo na taj skup težina i kao takav serijalizira za daljnju uporabu.

Opisani postupak rezultira skupom artefakata za praćenje učenja: graf *TrainingPlot.png*, krivulje trenirajućeg/validacijskog gubitka i validacijske točnosti po epohama, te tekstualni zapis *ValAccuracyPerEpoch.txt*. Ti se zapisi koriste u idućem odjeljku za formalno izvještavanje i kvalitativnu interpretaciju tijeka učenja, a najbolja pohranjena verzija modela predstavlja polazište za izvoz u mobilni format.

## Protokol evaluacije i izvještavanje

Evaluacija se provodi na validacijskom skupu definiranom u poglavlju 3, pri čemu se koristi determinističan tok obrade slika, promjena veličine, središnji izrez na 300×300, normalizacija na *ImageNet* statistiku, bez dodatnih augmentacija. Time se osigurava da su mjerenja usporediva kroz epohe i različita pokretanja. Primarna metrika je točnost, računata na razini epohe kao omjer ispravno klasificiranih uzoraka i ukupnog broja uzoraka u validacijskom skupu. Paralelno se bilježi i gubitak na validaciji, što omogućuje uvid u dinamiku učenja i pojavu mogućeg pretreniranja.

Tijek treniranja strukturiran je u cikluse „train potom val“, a nakon svake validacijske faze ažurira se interni maksimum točnosti. Ako je postignuta nova najbolja vrijednost, model se smatra „najboljim do sada“ i njegove se težine čuvaju u memoriji, u suprotnom se uvećava brojač epoha bez poboljšanja. Rano zaustavljanje aktivira se kada broj uzastopnih epoha bez rasta točnosti dosegne zadani prag, čime se treniranje prekida i model vraća na najbolji zabilježeni skup težina, a isječak koda koji implementira rano zaustavljanje nalazi se na slici Slika 4‑1. Ovakav protokol sprječava nepotrebnu potrošnju računalnih resursa i smanjuje rizik regresije performansi pri kraju treniranja.



Slika ‑ Implementacija ranog zaustavljanja

Rezultati se sustavno dokumentiraju artefaktima u direktoriju *TrainingFeedback*. Grafički prikaz krivulja učenja, sadrži trenirajući i validacijski gubitak te validacijsku točnost po epohama, što vizualno prikazuje konvergenciju, potencijalno razilaženje gubitaka i trenutak aktivacije ranog zaustavljanja. Uz to se generira tekstualni zapis točnosti po epohama, *ValAccuracyPerEpoch.txt*, koji omogućuje preciznu identifikaciju epohe s najvećom vrijednošću i lakše usporedbe između različitih pokretanja. Težine konačnog modela serijaliziraju se u *Models/ModelWeights/plant\_classifier\_efficientnetb3.pth*, a taj se model u idućim odjeljcima koristi kao polazište za izvoz u mobilni format.

S obzirom na neravnomjernu zastupljenost klasa u skupu, uz ukupnu točnost je korisno po potrebi dopuniti izvještavanje metrikama osjetljivim na balans klasa, npr. makro-F1, i matricom zabune za kvalitativnu analizu tipičnih pogrešaka. Te se dopune, kada su izračunate, prikazuju u odjeljku o rezultatima.

## Rezultati treniranja

Tijek učenja zabilježen u *ValAccuracyPerEpoch.txt* pokazuje stabilan rast validacijske točnosti u ranim epohama, od 0,6814 (ep. 1) do 0,8302 (ep. 8), nakon čega slijedi faza finog podešavanja s manjim poboljšanjima. Najveća vrijednost postiže se u epohi 10 s 0,8465. U sljedećim epohama bilježe se oscilacije bez nadmašivanja maksimuma, 0,8370 u ep. 11; 0,8310 u ep. 12; 0,8344 u ep. 13; 0,8391 u ep. 14. S obzirom na postavljeni *patience = 4*, rano zaustavljanje aktivira se nakon epohe 14, četiri uzastopne epohe bez poboljšanja u odnosu na ep. 10, a model se vraća na najbolje težine iz epohe 10 koje se serijaliziraju za daljnju uporabu. Tijek promjene vrijednosti prametara može se vidjeti na slici Slika 4‑2.



Slika ‑ Krivulje učenja: trenirajući i validacijski gubitak te validacijska točnost po epohama.

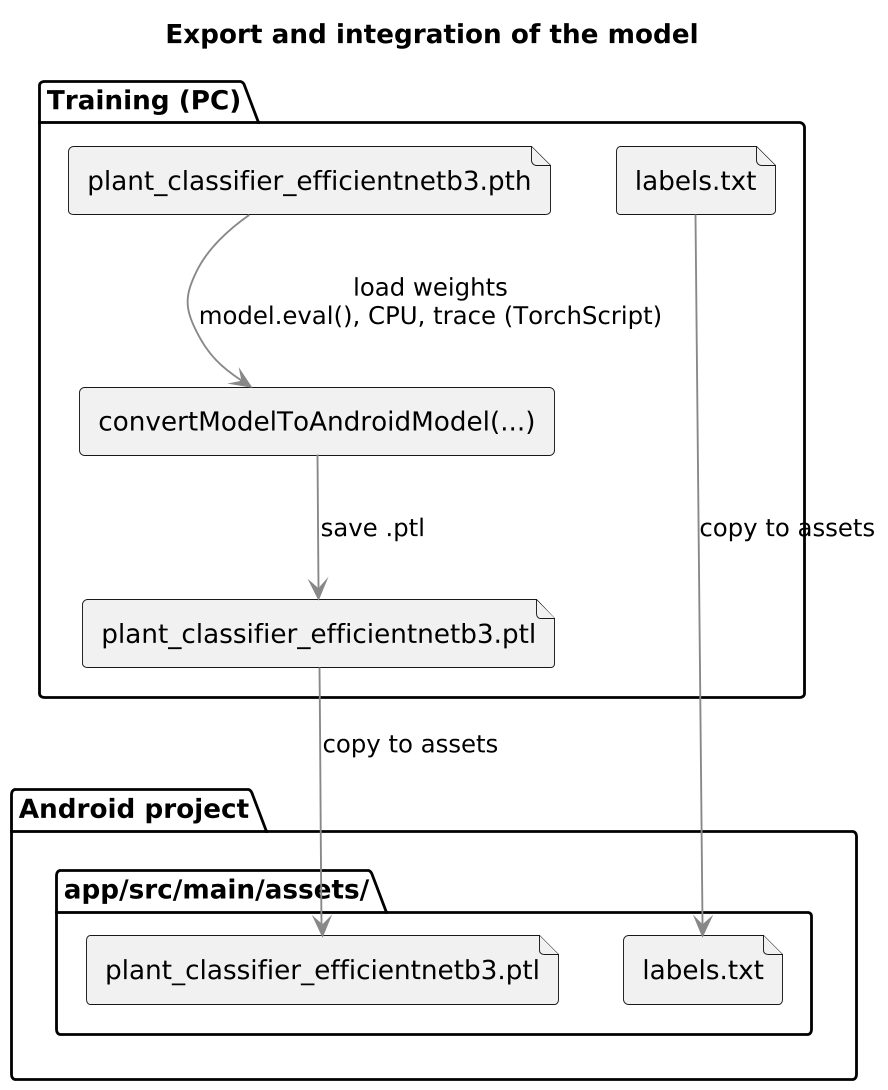
Krivulje učenja, trenirajući i validacijski gubitak te validacijska točnost po epohama, upućuju na očekivano ponašanje: kontinuirani pad trenirajućeg gubitka, početni pad i stabilizacija validacijskog gubitka te porast točnosti do plato-vrijednosti. To sugerira da je kombinacija arhitekture *EfficientNet-B3* i primijenjenih augmentacija prikladna za zadatak klasifikacije 74 vrsta na skupu od 54253 slika. Ipak, pogreške su učestalije kod vizualno sličnih vrsta i u otežanim uvjetima, okluzije, mali objekt u kadru, složena pozadina, što je uobičajen obrazac u praktičnim klasifikacijskim problemima. Neravnomjerna zastupljenost klasa može blago favorizirati češće klase, taj se učinak može dodatno adresirati ponderiranjem gubitka ili ciljanim obogaćivanjem podataka u budućem radu.

U ovoj fazi, zaključujemo da treniranje daje stabilan „najbolji“ model s najvećom val. Točnošću od 0,8465 u ep. 10. Rano zaustavljanje sprječava pretreniranje i nepotrebne epohe, a odabrani checkpoint služi kao polazište za konverziju u mobilno-prikladan format i integraciju u Android aplikaciju u sljedećim odjeljcima.

## Izvoz modela i pripreme za Android

Nakon dovršetka treniranja i odabira „najboljih” težina, model se pretvara u mobilno-prikladan artefakt i zajedno s mapom oznaka integrira u Android aplikaciju. Izvoz je realiziran funkcijom *convertModelToAndroidModel*, koja učitava spremljene težine, prebacuje mrežu u evaluacijski način i na CPU, te generira *TorchScript* trace nad fiksnim ulazom. Dobiveni artefakt sprema se kao datoteka *Models/AndroidModel/plant\_classifier\_efficientnetb3.ptl*, a zatim se kopira u *Application/app/src/main/assets/plant\_classifier\_efficientnetb3.ptl* kako bi bio dostupan unutar aplikacije. Time se osigurava da je izvršni model dostupan iz *assets* direktorija bez potrebe za mrežom i bez dodatne instalacije na uređaju.

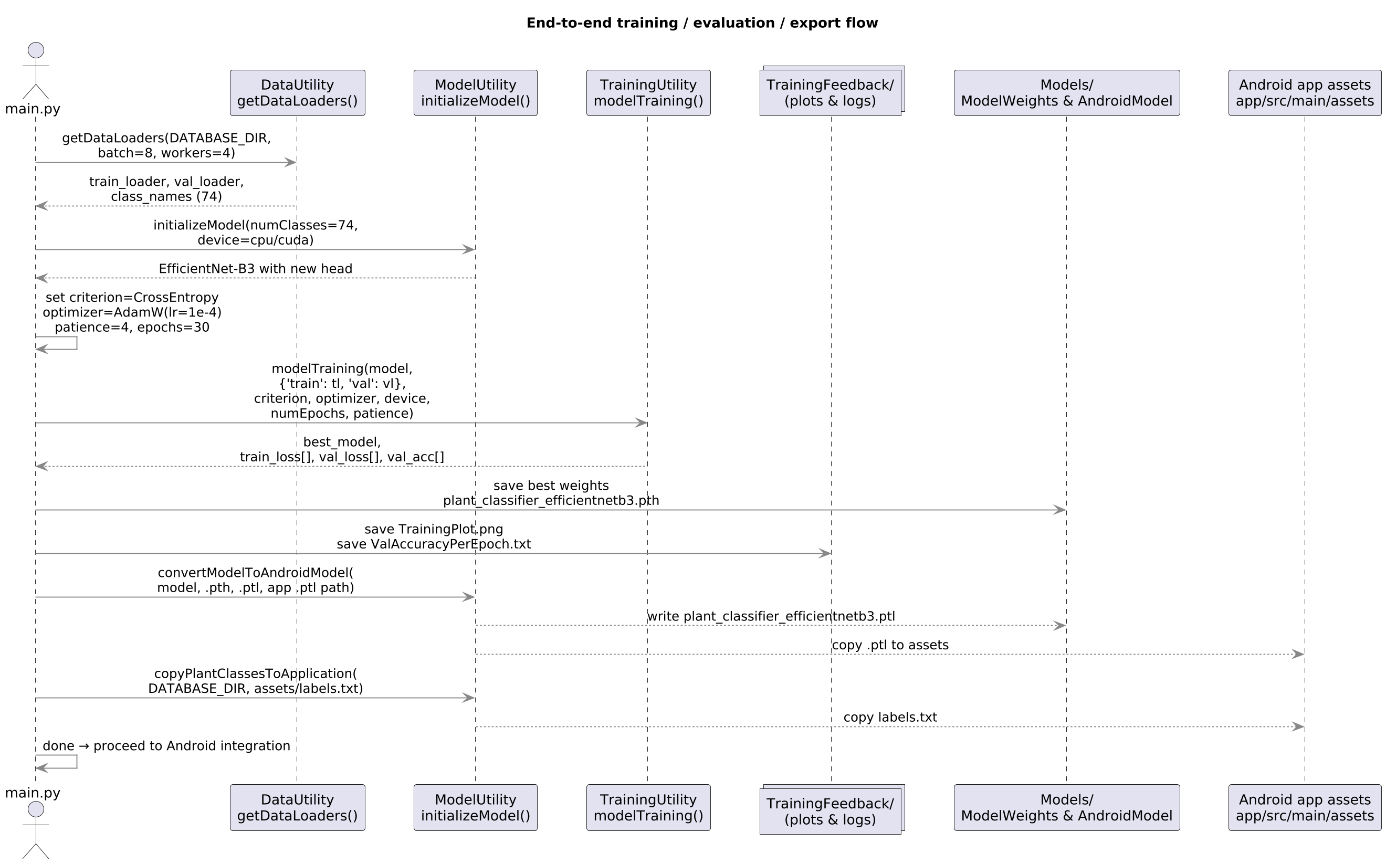
Uz model, aplikaciji je potrebna i dosljedna mapa klasa. Funkcija *copyPlantClassesToApplication* kopira popis oznaka, generiran tijekom izrade baze i treniranja, u *Application/app/src/main/assets/labels.txt*. Redoslijed ovih oznaka mora odgovarati redoslijedu na izlazu modela, čime se održava jednoznačno preslikavanje indeks, naziv vrste u svim fazama. Ovakva organizacija jamči da će prikazani rezultat u aplikaciji koristiti iste nazive i isti poredak kao i pri evaluaciji na računalu. Cijeli proces izvoza i prebacivanja u Android model mogu se vidjeti na slici Slika 4‑3.



Slika ‑ Izvoz modela i prebacivanjeu Android model (izrađeno u web editoru <https://plantuml.com/>)

U kontekstu on-device inferencije, očekivanja modela nad ulazom prate trening pipeline: slika se pretvara u RGB, skalira/izrezuje na kvadratnu ciljnu rezoluciju, normalizira na *ImageNet* statistiku, *mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]*, a zatim se, kao tenzor oblika *1×3×H×W* tipa *float32*, prosljeđuje *TorchScript* modulu. Izlazi su sve 74 klase, a tijekom inferencije u aplikaciji tipično se primjenjuje softmax kako bi se dobile vjerojatnosti, izabire se top-1 ili top-k i taj se indeks preslikava u naziv klase čitanjem iz *labels.txt*. Ovakav pristup omogućuje potpuno offline izvođenje, čuva privatnost korisnika na način da slika ne napušta uređaj i eliminira latenciju povezanu s mrežnim pozivima.

Proces izrade mobilnog artefakta usklađen je s praktičnim zahtjevima integracije na Androidu, model je serijaliziran u *TorchScript Lite* formatu , .ptl formatu, prikladnom za *PyTorch Mobile runtime*, smješten je u *assets,* s ciljem pojednostavljenja distribucije i učitavanja, a pripadajuća datoteka s oznakama osigurava semantičku dosljednost rezultata. Dodatne optimizacije poput kvantizacije radi smanjenja memorijskog otiska i latencije predstavljaju prirodnu nadogradnju i moguće su u kasnijim iteracijama bez promjene sučelja aplikacije. U sljedećem poglavlju fokus se premješta na integraciju modela u Android aplikaciju, uključujući pretprocesiranje slika, pozivanje *TorchScript* modula i prezentaciju rezultata korisniku. Cjelokupni tijek skripte objasnjene kroz prethodne odjeljke nalazi se na slici Slika 4‑4.



Slika ‑ Tijek skripte za treniranje, evaluaciju i izvoz modela (izrađeno u web editoru <https://plantuml.com/>)

## Ograničenja i potencijalna poboljšanja

Trenutni tijek daje stabilan rezultat, ali ima nekoliko praktičnih ograničenja vezanih uz implementaciju. Skup augmentacija u *DataUtility.py* je namjerno „jak“, rotacije, affine/perspektiva, *vertical flip*, *ColorJitter*, *RandomErasing*, što povećava robusnost, ali može uvesti distribucijski pomak u odnosu na realne uvjete korištenja aplikacije. U idućim iteracijama smisleno je prilagoditi intenzitet transformacija onome što će korisnici najčešće snimati, npr. smanjiti vertikalna zrcaljenja ako nisu prirodna za većinu biljnih uzoraka.

U *TrainingUtility.py* treniranje se oslanja na fiksni korak učenja, *Adam*, i rano zaustavljanje s *patience=4*, uz praćenje točnosti kao glavne metrike. Ovakav protokol je jednostavan i učinkovit, ali može stati prije potencijalnog poboljšanja. U praksi se kao nadogradnja često koristi raspored stope učenja, npr. *ReduceLROnPlateau*, ili kratka druga faza finog podešavanja s manjom stopom učenja.

# ANDROID APLIKACIJA

## Pregled

Ovo poglavlje prikazuje kako je istrenirani klasifikator iz poglavlja 4 integriran u Android aplikaciju te kako izgleda cjelokupni tok na uređaju. Model u *TorchScript* formatu i pripadna mapa klasa smješteni su u *assets* i učitavaju se pri pokretanju aplikacije. Ulazna slika dolazi iz kamere, *CameraX* ili iz galerije, po potrebi se korigira orijentacija (*EXIF*), zatim se pretprocesira u tenzor, RGB, promjena veličine na 300×300, normalizacija na ImageNet statistiku i prosljeđuje modelu. Izlazi se mapiraju na nazive vrsta čitanjem iz *labels.txt* i rezultat se prikazuje korisniku u obliku *pop-up* prozora, zajedno s osnovnim informacijama o vrsti iz lokalne baze podataka aplikacije.

Sva obrada odvija se lokalno na uređaju, bez mrežne ovisnosti: time se čuvaju privatnost i smanjuje latencija. U nastavku poglavlja sažeto su objašnjeni ključni elementi implementacije, ovisnosti projekta i smještaj artefakata, dohvat slike kamerom ili iz galerije, pretprocesiranje i poziv modela, mapiranje izlaza na nazive te prikaz rezultata kako bi se jasno povezali skup podataka iz poglavlja 3 i treniranje modela iz poglavlja 4 s praktičnom, upotrebljivom Android aplikacijom.

## Postavke projekta

Aplikacija je izrađena u *Kotlinu* s uključenim *Jetpack Compose UI* slojem, *CameraX* komponentama za rad s kamerom i *PyTorch Mobile* okruženjem za izvođenje *TorchScript* modela. *Gradle* konfiguracija koristi *minSdk 24*, *targetSdk 36* i *compileSdk 36*, a *Java/Kotlin* su podešeni na verziju 11, pri čemu je *Compose* eksplicitno omogućen.

Artefakti modela, odnosno datoteke *plant\_classifier\_efficientnetb3.ptl* i *labels.txt*, smješteni su u *app/src/main/assets/* te se učitavaju prilikom pokretanja aplikacije kako bi bili dostupni offline. Oslonac na *PyTorch Mobile 1.13.1* i prateći *pytorch\_android\_torchvision* modul omogućuje pretvorbu bitmapa u tenzore i normalizaciju u skladu s treniranjem, dok *CameraX* paketi zaduženi za pregled, životni ciklus i snimanje osiguravaju stabilan rad kamere. Korisničko sučelje temelji se na *AndroidX Compose-*u i *Materialu 3* uz podršku *AppCompat* biblioteka.

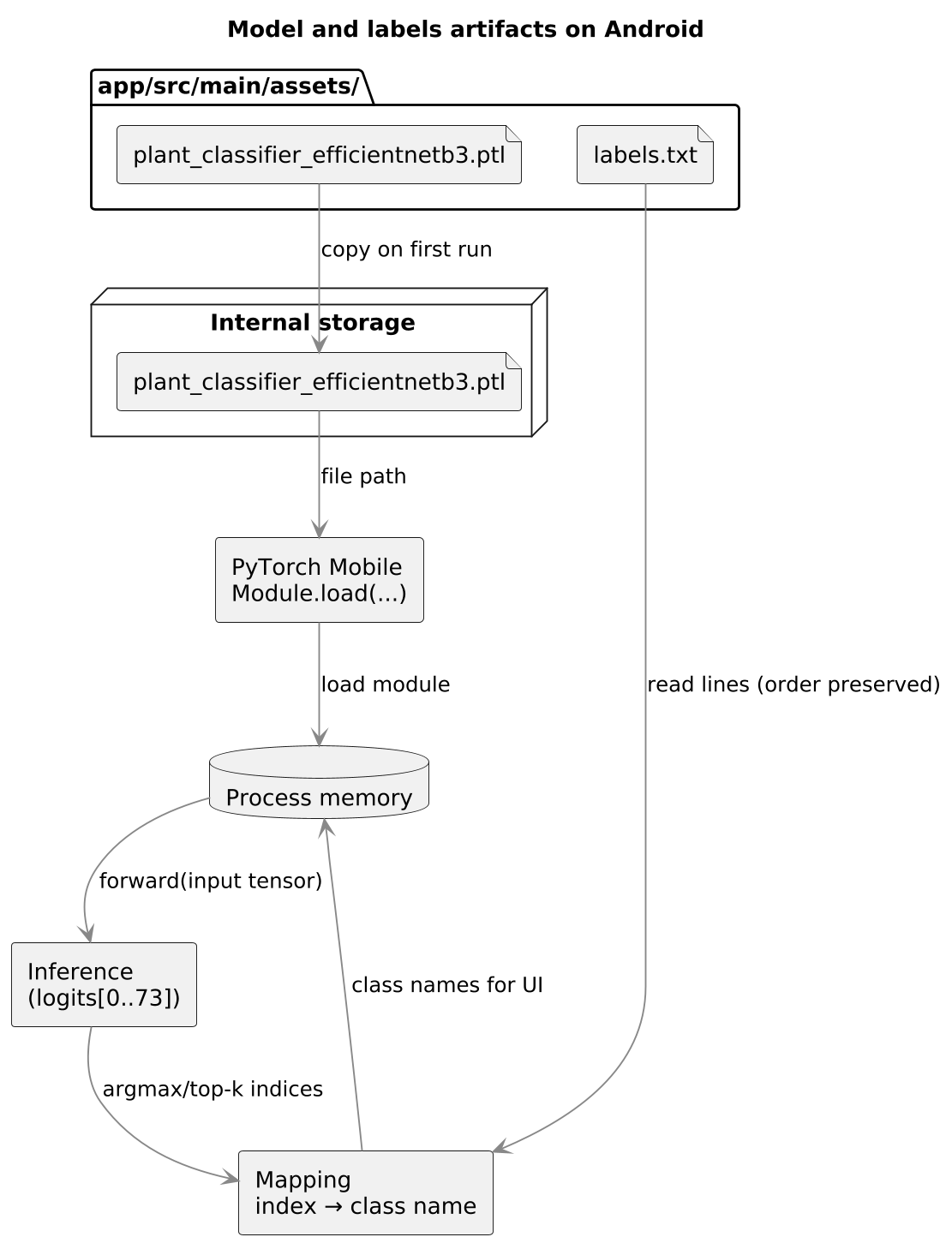
U izdanju za produkciju minifikacija je trenutačno isključena. Ako se naknadno uključi, potrebno je zadržati *PyTorch* klase pravilima za zadržavanje paketa *org.pytorch* i *org.pytorch.torchvision* kako bi učitavanje modela i utilita ostalo pouzdano. Ovakva postava osigurava pouzdano izvođenje modela na uređaju, jednostavan pristup artefaktima iz *assets* direktorija i stabilno rukovanje kamerom.

## Artefakti modela

Model istreniran u poglavlju 4 distribuira se u *TorchScript* formatu kao datoteka *plant\_classifier\_efficientnetb3.ptl*, dok se preslikavanje izlaznih indeksa na nazive vrsta čuva u tekstualnoj datoteci *labels.txt*. Oba artefakta smještena su u *app/src/main/assets/*, čime su dostupna offline i ne ovise o mreži. Pri pokretanju aplikacije model se prvo kopira iz *assets* u internu pohranu i učitava pozivom *Module.load(...)*. Ovakav korak je praktičan jer *PyTorch* Mobile očekuje putanju do obične datoteke, a ne do *asset stream*-a, te omogućuje pouzdano inicijalno učitavanje bez privremenih ovisnosti o vanjskim dozvolama pohrane.

Datoteka *labels.txt* učitava se redak-po-redak u memoriju i normalizira na mala slova radi robusnog uspoređivanja. Presudan je redoslijed: indeks i u izlazu modela jednoznačno odgovara i-tom retku u *labels.txt*, pa se ovim poretkom zadržava semantička dosljednost između treniranja, izvoza i prikaza u aplikaciji. Takva organizacija eliminira potrebe za dodatnim metapodacima i smanjuje mogućnost pogrešnog mapiranja naziva ako se skup klasa mijenja. U slučaju nedostupnosti ili oštećenja bilo kojeg od artefakata, aplikacija jasno prijavljuje grešku i onemogućuje klasifikaciju, čime se izbjegavaju tihe, pogrešne predikcije.

Odabirom *.ptl* formata i *assets* distribucije postiže se nekoliko ciljeva: privatnost (slike ne napuštaju uređaj), predvidljiva latencija (nema mrežnih poziva), jednostavno verzioniranje (zamjenom para *model.ptl* i *labels.txt* jednim *build*-om), te minimalna integracija na Androidu jer su oba resursa dostupna odmah nakon instalacije. Ovaj raspored artefakata čini temelj kasnijeg toka pretprocesiranja i inferencije opisanog u sljedećim odjeljcima. Čitav proces u ovom odjeljku može se vidjeti na slici Slika 5‑1.



Slika ‑ Distribucija artefakata: model.ptl i labels.txt u assets, kopiranje modela u internu pohranu, učitavanje putem Module.load(...) i mapiranje izlaznih indeksa na nazive klasa (izrađeno u web editoru <https://plantuml.com/>)

## Dohvaćanje slika preko kamere i galerije

Ulazna slika u aplikaciju dolazi iz dvaju izvora, kamere i galerije, pri čemu je tok rada usklađen s Androidovim mehanizmima dozvola i životnog ciklusa. Za kameru se koristi *CameraX* s *PreviewView* komponentom za prikaz uživo te *ImageCapture* konfiguriranim u načinu minimizirane latencije kako bi se vrijeme okidanja svelo na minimum. Nakon što korisnik odobri potrebne dozvole, *ProcessCameraProvider* veže *Preview* i *ImageCapture* uz životni ciklus aktivnosti i pokreće prikaz. Pritiskom na gumb okidača fotografija se sprema u predmemoriju aplikacije, a rezultat snimanja vraća se kroz povratni poziv. Snimljena datoteka dekodira se u bitmapu, a zatim se, prema EXIF metapodatcima snimke, korigira orijentacija kako bi prikaz odgovarao stvarnom smjeru snimanja. Korigirana slika prikazuje se u prekrivnom *ImageView* elementu dok se prikaz kamere privremeno skriva, čime korisnik odmah vidi ulaz koji će se klasificirati.

Odabir fotografije iz galerije provodi se implicitnom namjerom *ACTION*\_PICK s filtrom za slike. Nakon što korisnik odabere zapis, sustav vraća Uri koji se dekodira u bitmapu i prikazuje na istom mjestu kao i snimljena fotografija. Pristup galeriji ovisi o verziji sustava pa se na novijim uređajima koristi dozvola *READ\_MEDIA\_IMAGES*, dok se na starijim primjenjuje *READ\_EXTERNAL\_STORAGE.* Zahtjev i obrada odgovora rješavaju se u aktivnosti prije otvaranja odabira. U oba slučaja, čim je slika vidljiva pokreće se kratka animacija „skeniranja” kao vizualni pokazatelj da je obrada u tijeku, a dobivena bitmapa prosljeđuje se sljedećem koraku pretprocesiranja. Ovakva organizacija osigurava pouzdan i brz prijenos slike iz kamere ili medijske pohrane do modela, uz jasno korisničko iskustvo i pravilnu orijentaciju sadržaja.

## Pretprocesiranje na uređaju

Pretprocesiranje na uređaju priprema ulaznu fotografiju u oblik koji *TorchScript* model očekuje i pritom strogo slijedi isti redoslijed koraka pri svakoj klasifikaciji. Nakon što je slika dohvaćena i eventualno korigirana po *EXIF* orijentaciji, provodi se provjera formata bitmapa. Ako je ulaz u *Config.HARDWARE*, pretvara se u softverski *ARGB\_8888* kako bi se omogućilo sigurno čitanje piksela bez iznimki u *runtime*-u. Time se osigurava da su podaci dostupni procesoru slike i da daljnje operacije ne ovise o GPU-podršci sustava.

Sljedeći korak je promjena veličine slike na kvadratnu rezoluciju od 300×300 piksela. Ova ciljna dimenzija usklađena je s izvezenim *TorchScript* modelom i omogućuje determinističan, brz put do tenzora bez dodatnih međukoraka. Nakon skaliranja, slika se pretvara u tenzor s rasporedom kanala *CHW* i dubinom *float32*, pri čemu se vrijednosti najprije skaliraju u raspon [0, 1], a zatim normaliziraju na *ImageNet* statistiku koristeći srednje vrijednosti [0.485, 0.456, 0.406] i standardne devijacije [0.229, 0.224, 0.225]. Ovaj korak replicira očekivanja modela u pogledu distribucije ulaza te smanjuje razliku između uvjeta treniranja i uvjeta izvođenja. Rezultat je tenzor oblika 1×3×300×300 bez alfa kanala, spreman za prosljeđivanje u modul.

Cijeli postupak je determinističan i bez stohastičkih transformacija: na uređaju se namjerno ne primjenjuju augmentacije kako bi se zadržala ponovljivost, stabilna latencija i predvidljivost rezultata. Usklađenost dimenzija, normalizacije i redoslijeda kanala s onime što je korišteno pri izvozu modela presudna je za konzistentnu točnost; stoga su koraci promjene veličine, pretvorbe u tenzor i normalizacije implementirani identičnim parametrima pri svakoj inferenciji.

## Inferencija i mapiranje rezultata

Nakon pretprocesiranja, ulazni tenzor prosljeđuje se *TorchScript* modulu u pozadinskoj niti, pri čemu se poziva metoda za naprijedni prolaz i dobiva jednodimenzionalni niz čija je duljina jednaka broju klasa. Budući da je cilj izdvojiti najvjerojatniju klasu, dovoljno je izračunati indeks najveće vrijednosti u nizu. primjena *softmaks*-a nije potrebna za izdvajanje poretka jer ne mijenja *argmax*. Ovakav pristup izbjegava dodatnu numeričku i vremensku cijenu, što je korisno u mobilnom okruženju.

Dobiveni indeks izravno se preslikava na naziv vrste koristeći listu oznaka učitanu iz labels.txt. Redoslijed u toj listi mora ostati identičan poretku izlaznih klasa kako bi mapiranje bilo jednoznačno, a u implementaciji se nazivi normaliziraju na mala slova radi robusnog uspoređivanja u ostatku aplikacije. Ako je polje rezultata neočekivano prazno ili je izračunati indeks izvan granica liste oznaka, postupak se prekida uz zapis poruke u log i kratku obavijest korisniku, čime se sprječava prikaz nekonzistentnog ili pogrešnog ishoda.

Rezultat inferencije tretira se kao *top-1* predikcija koja se predaje sljedećem sloju aplikacije. Time se odvajaju odgovornosti: ovaj odjeljak rješava izračun i semantičko mapiranje, a prikaz i obogaćivanje informacija preuzima UI sloj opisan u idućem odjeljku. Po potrebi, isti mehanizam lako se proširuje na *top-k* pristup zadržavanjem nekoliko najvećih klasa i njihovih indeksa te mapiranjem na više kandidata, bez promjena u načinu učitavanja modela ni u formatu *labels.txt*.

## Prikaz rezultata

Nakon što je određena *top-1* klasa, rezultat se prikazuje u donjoj ploči koja se inflatira iz *bottom\_sheet\_layout.xml* i preklapa glavni zaslon. Ploča sadrži naslov s engleskim i hrvatskim nazivom vrste, latinski naziv u kurzivu te opis koji je smješten u proširujuće (engl. *scrollable*) područje kako bi se uredno prikazali dulji tekstovi. Informacija o jestivosti naglašena je zasebnim redom na dnu. Podaci se dohvaćaju iz lokalne baze *PlantDatabase.kt*, pri čemu se ključ predikcije normalizira na mala slova kako bi se izbjegla nepodudaranja u zapisima. Ako za neku prepoznatu oznaku nema pripadajućeg zapisa, ploča se i dalje prikazuje s nazivom iz *labels.txt*, a ostala polja popunjavaju se neutralnim porukama, čime se postupak čini robusnim na parcijalne podatke.

Otvaranje ploče vremenski je usklađeno s dovršetkom animacije skeniranja kako bi korisnik dobio jasan osjećaj završetka obrade. U trenutku prikaza ploče i ulazna fotografija ostaje vidljiva u pozadini, što pomaže povezati rezultat s konkretnim uzorkom. Zatvaranjem ploče aplikacija automatski vraća prikaz kamere i skriva statičnu sliku, čime se korisniku omogućuje neprekinut ciklus snimanja i prepoznavanja. Rješenje je prilagođeno lokalizaciji jer se naslovi i opisi generiraju iz domaće baze podataka, dok se stvarni ključne oznake i redoslijed klasa zadržavaju identični onima u *labels.txt*, pa je veza između modela i sučelja dosljedna bez obzira na jezik prikaza. Izgled sučelja nakon prepoznavanja prikazan je na slici Slika 5‑2.



Slika ‑ Sučelje nakon prepoznavanja vrste sa slike

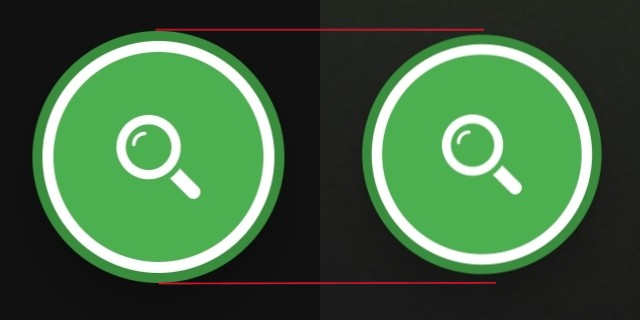
## UX povratne informacije i animacije

Vizualni povratni signal u aplikaciji namjerno je jednostavan i stalno povezan s radnjama korisnika. Tijekom klasifikacije iznad prikaza kamere pojavljuje se uska „scan” traka definirana u *scanning\_bar.xml*, koja se animira vertikalnim pomakom dok je obrada u tijeku i nestaje kada je rezultat spreman. Animacija je sinkronizirana tako da završi barem jedan puni ciklus prije gašenja, čime se izbjegava nagli prekid vizualne informacije i korisniku jasno daje do znanja da je postupak zaključen. Traka je prikazana kao lagani *overlay* koji ne blokira interakciju te se po završetku obrade odmah skriva kako bi se korisnik mogao vratiti snimanju bez dodatnih koraka. Na slici Slika 5‑3 prikazan je izgled animacije skeniranja.



Slika ‑ Animacija skeniranja

Sam čin okidanja popraćen je kratkom mikro-animacijom gumba okidača pomoću shutter\_button\_scale.xml, koja na pritisak neznatno smanjuje skalu i zatim je vraća na početno stanje. Ovaj „taktilni” dojam potvrđuje da je akcija zaprimljena i poboljšava doživljaj odzivnosti bez utjecaja na trajanje obrade. Pozadina gumba sastavljena je slojevito u *shutter\_button\_background.xml*, što osigurava dobru čitljivost kontura i stabilan kontrast na raznim pozadinama kamere. Kombinacija diskretne scan animacije i kratkog pritisnog feedbacka postiže ravnotežu između informiranja korisnika i nenametljivosti, pri čemu je naglasak ostavljen na prikazu fotografije i rezultatu u donjoj ploči. Slika 5‑4 prikazuje dizajn i razliku u veličini gumba prije i tijekom pritiska.



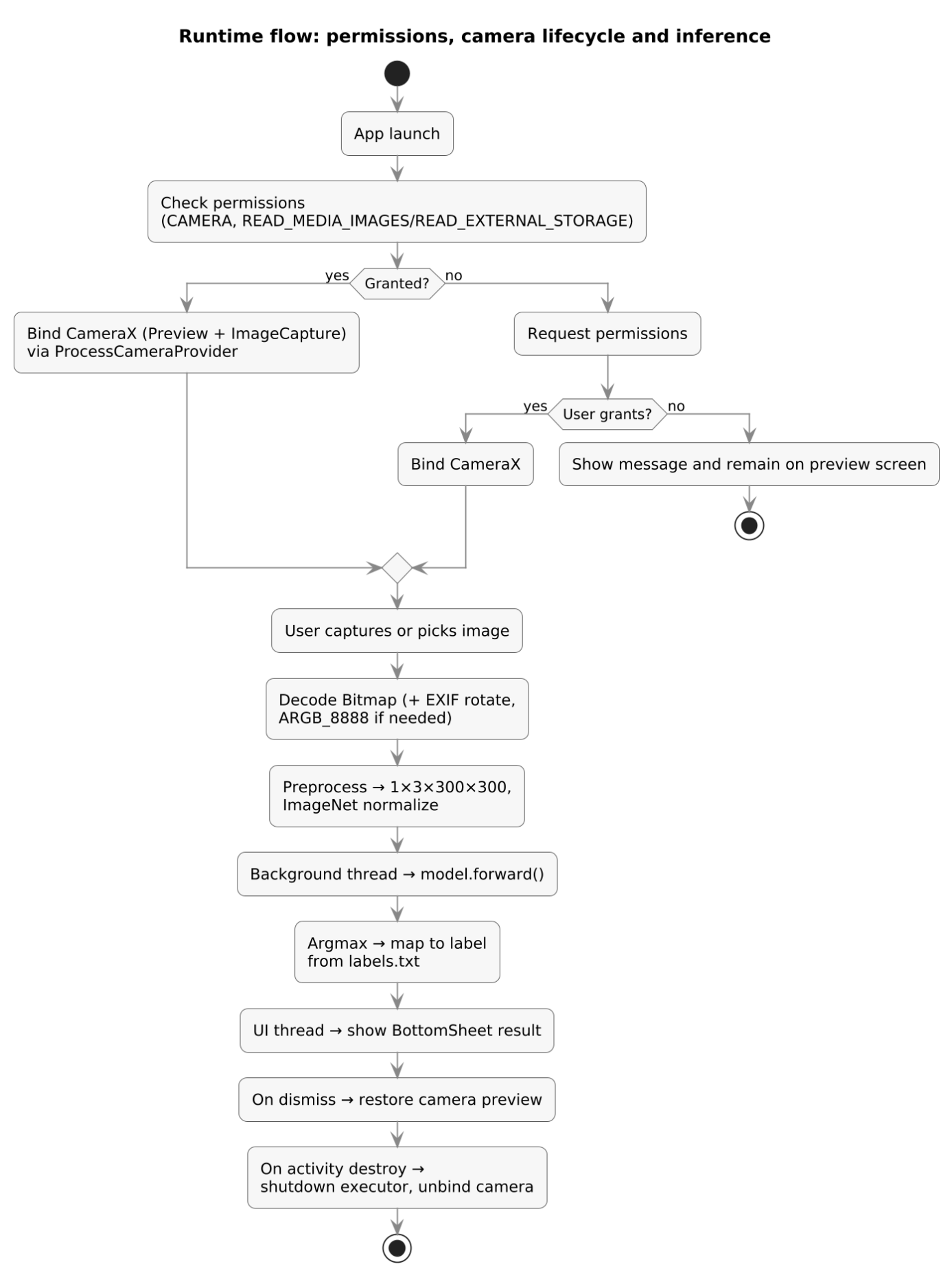
Slika ‑ Usporedba veličina prije pritiska i tijekom pritiska

## Operativni aspekti rada na uređaju

Aplikacija upravlja dozvolama i resursima tako da cijeli tok ostaje stabilan i neovisan o mreži. Dozvole se traže u vrijeme izvođenja, pri čemu se za novije verzije sustava koristi pristup medijima putem *READ\_MEDIA\_IMAGES*, a na starijim uređajima čitanje datoteka rješava se preko *READ\_EXTERNAL\_STORAGE*. Pokretanje kamere oslanja se na *ProcessCameraProvider*, koji povezuje *Preview* i *ImageCapture* s životnim ciklusom aktivnosti. Na taj se način kamera automatski pauzira i obnavlja pri promjenama stanja bez ručnog upravljanja. U trenutku gašenja aktivnosti resursi se uredno zatvaraju, a izvršna nit za klasifikaciju gasi se pozivom *shutdown*, čime se izbjegavaju curenja resursa i zaključane datoteke u predmemoriji.

Izvedba inferencije odvija se na pozadinskoj niti kako bi sučelje ostalo odzivno, dok se sve promjene prikaza, pokazivanje fotografije, otvaranje i zatvaranje donje ploče, poruke korisniku, objavljuju na glavnoj niti. Ulazne slike se po potrebi rotiraju prema *EXIF* metapodacima i, ako su u „hardware” formatu, privremeno pretvaraju u *ARGB\_8888* kako bi bile pogodne za pretprocesiranje na CPU-u. Model i mapa klasa distribuiraju se uz aplikaciju u *assets* direktoriju. Model se pri prvom učitavanju kopira u internu pohranu i otvara s lokalne putanje, a *labels.txt* čita se izravno u memoriju. Takva organizacija omogućuje rad u potpunosti offline i jamči predvidljivu latenciju jer nema mrežnih poziva niti promjenjivih vanjskih ovisnosti.

Rukovanje pogreškama osigurava robustan povrat u stabilno stanje. Ako dođe do iznimke pri dekodiranju slike, učitavanju modela ili mapiranju rezultata, aplikacija zapisuje dijagnostičke poruke u log i korisniku prikazuje kratku obavijest, zatim vraća prikaz kamere kako bi se moglo nastaviti sa snimanjem bez ponovnog pokretanja. Istodobno, vizualne animacije tijekom obrade sinkronizirane su tako da uvijek dovrše barem jedan ciklus, čime se izbjegava nagli prekid i održava dosljedan doživljaj. Ovakav skup operativnih praksi spaja pouzdanost, privatnost i jednostavnost uporabe, pa je cijeli tok klasifikacije spreman za praktičnu, ponovljivu uporabu na uređaju. Pregled čitavog operativnog toka nalazi se na slici Slika 5‑5.



Slika ‑ Operativni tok aplikacije

U ovom poglavlju prikazan je cjelovit tok integracije modela u Android aplikaciju: artefakti *plant\_classifier\_efficientnetb3.ptl* i *labels.txt* smješteni su u *assets*, učitavaju se pri pokretanju, a ulazne fotografije preuzimaju se kamerom (*CameraX*) ili iz galerije, po potrebi rotiraju prema *EXIF* metapodacima i pretprocesiraju na 300×300 uz *ImageNet* normalizaciju. Tenzor se prosljeđuje *TorchScript* modelu, iz izlaza izdvaja se najvjerojatnija klasa i mapira na naziv iz *labels.txt*, a rezultat se prikazuje u donjoj ploči zajedno s osnovnim informacijama o vrsti iz lokalne baze.

Posebna je pažnja posvećena korisničkom doživljaju i stabilnosti izvođenja, animirana „scan” traka i mikro-animacija gumba pružaju jasan, nenametljiv *feedback*, dok se dozvole, niti i životni ciklus kamere upravljaju tako da aplikacija pouzdano radi potpuno offline. Ovim je zaokružena praktična primjena sustava razvijenog u prethodnim poglavljima i pripremljen je teren za završne zaključke rada.

# ZAKLJUČAK

U radu je izrađen cjelovit sustav za automatsku klasifikaciju biljaka i drveća koji obuhvaća tri ključne cjeline: izradu i dokumentiranje vlastite baze podataka, treniranje dubokog modela te integraciju gotovog klasifikatora u Android aplikaciju. Baza s 74 vrste i ukupno 54 253 slika prikupljena je automatizirano preko *iNaturalist* API-ja uz poštivanje CC0/CC-BY licenci i bilježenje izvora i atribucija. Skup je podijeljen na učenje i validaciju, a priložena je i skripta za potpunu rekonstrukciju baze iz zapisa URL-ova, čime su osigurani ponovljivost i transparentnost.

Model zasnovan na *EfficientNet-B3* treniran je na pripremljenom skupu uz kombinaciju prostornih i fotometrijskih augmentacija, optimizaciju *AdamW* i rano zaustavljanje. Tijekom epoha bilježen je očekivan pad gubitka i rast točnosti na validaciji, s dosegnutom razinom točnosti oko 0,85, što potvrđuje da odabrana arhitektura i protokol učenja daju dobar odnos točnosti i računalne učinkovitosti za mobilnu primjenu. Model je potom izvezen u *TorchScript* formatu i uparen s mapom klasa, čime je osigurano stabilno i jednoznačno preslikavanje izlaznih indeksa na nazive vrsta u svim fazama.

Android aplikacija demonstrira potpun *on-device* tok: dohvat slike kamerom (*CameraX*) ili iz galerije, korekciju orijentacije i pretprocesiranje na 300×300 s *ImageNet* normalizacijom, izvođenje inferencije *TorchScript* modelom te prikaz rezultata u donjoj ploči uz osnovne informacije o vrsti iz lokalne baze. Budući da se sve odvija lokalno na uređaju, bez mrežnih poziva, postižu se predvidljiva latencija i očuvanje privatnosti korisnika. Sučelje je upotpunjeno nenametljivim vizualnim povratnim informacijama (scan traka i animacija gumba), a operativni dio pažljivo upravlja dozvolama, dretvama i životnim ciklusom kamere.

Glavna ograničenja proizlaze iz neuravnoteženosti broja uzoraka po klasama i fine sličnosti među pojedinim taksonima. Ona se mogu ublažiti daljnjim čišćenjem i balansiranjem skupa, uvođenjem dodatnih ili ciljano prilagođenih augmentacija te dopunom protokola učenja (npr. rasporedi stope učenja, *focal loss*, kalibracija izlaza). Na strani mobilne primjene, prirodni nastavak rada uključuje lagane optimizacije modela (npr. kvantizacija) i sustavnija mjerenja na većem broju uređaja.

Sveukupno, rad spaja prikupljanje podataka, treniranje i mobilnu integraciju u funkcionalnu cjelinu koja pokazuje da je prepoznavanje biljaka moguće ostvariti pouzdano i bez oslanjanja na mrežu. Postavljena je čvrsta tehnička osnova koja se može nadograditi širím skupom podataka, dodatnim mjerama robusnosti i optimizacijama za uređaje, čime sustav postaje još korisniji u stvarnim terenskim uvjetima.

LITERATURA

1. Russell, S., & Norvig, P. Artificial Intelligence: A Modern Approach. Pearson, 2020.
2. Goodfellow, I., Bengio, Y., & Courville, A. Deep Learning. MIT Press, 2016.
3. IBM. “Supervised vs. Unsupervised Learning.” Dostupno na: <https://www.ibm.com/think/topics/supervised-vs-unsupervised-learning>, zadnji pristup: 25.08.2025.
4. Sutton, R. S., & Barto, A. G. Reinforcement Learning: An Introduction. MIT Press, 2018.
5. Rosenblatt, F. The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain. Psychological Review, 1958.
6. Goodfellow, I., Bengio, Y., & Courville, A. Deep Learning. MIT Press, 2016.
7. Krizhevsky, A., Sutskever, I., & Hinton, G. ImageNet Classification with Deep Convolutional Neural Networks. NeurIPS, 2012.
8. Szeliski, R. Computer Vision: Algorithms and Applications. Springer, 2022.
9. Gonzalez, R. C., & Woods, R. E. Digital Image Processing. Pearson, 2018.
10. Kamilaris, A., & Prenafeta-Boldú, F. X. Deep learning in agriculture: A survey. Computers and Electronics in Agriculture, 2018.
11. Grinblat, G. L., Uzal, L. C., Larese, M. G., & Granitto, P. M. (2016). Deep learning for plant identification using vein morphological patterns. Computers and Electronics in Agriculture, 127, 418–424.
12. Hughes, D. P., & Salathé, M. (2015). An open access repository of images on plant health to enable the development of mobile disease diagnostics. arXiv:1511.08060.
13. Lee, S. H., Chan, C. S., Mayo, S. J., & Remagnino, P. (2017). How deep learning extracts and learns leaf features for plant classification. Pattern Recognition, 71, 1–13.
14. Goëau, H., Bonnet, P., & Joly, A. (2017). Plant identification in an open-world (LifeCLEF 2017). CLEF Working Notes.
15. Affouard, A., Goëau, H., Bonnet, P., & Joly, A. (2017). PlantNet App in the Era of Deep Learning. ICCV Workshops.
16. Google. (2023). Google Lens. Dostupno na: <https://lens.google/>, zadnji pristup: 25.08.2025.

PRILOZI

Kazalo slika i tablica

Kazalo slika

[Slika 2‑1 Prikaz kako radi nadzirano strojno učenje. (preuzeto s https://maddevs.io/blog/semi-supervised-learning-explained/ ) 3](#_Toc207539424)

[Slika 2‑2 Prikaz kako radi nenadzirano strojno učenje (preuzeto s https://maddevs.io/blog/semi-supervised-learning-explained/) 4](#_Toc207539425)

[Slika 2‑3 Prikaz podržanog strojnog učenja (preuzeto s https://www.mathworks.com/help/reinforcement-learning/ug/what-is-reinforcement-learning.html) 5](#_Toc207539426)

[Slika 2‑4 Primjer izgleda višeslojnog perceptrona, tzv. MLP-a (preuzeto s https://learnopencv.com/understanding-convolutional-neural-networks-cnn/) 7](#_Toc207539427)

[Slika 2‑5 Izgled konvolucijskih slojeva u VGG-16 konvolucijskoj neuralnoj mreži (preuzeto s https://learnopencv.com/understanding-convolutional-neural-networks-cnn/) 8](#_Toc207539428)

[Slika 2‑6 Neuronska mreža trenirana od počeka (preuzeto s https://www.danrose.ai/blog/transfer-learning-from-a-business-perspective) 9](#_Toc207539429)

[Slika 2‑7 Neuronska mreža nakon upotrebe tranfer learning tehnike (preuzeto s https://www.danrose.ai/blog/transfer-learning-from-a-business-perspective) 9](#_Toc207539430)

[Slika 2‑8 Usporedba ljudskog vida i računalnog vida (preuzeto s https://imagevision.ai/blog/understanding-computer-vision-a-technical-overview/) 10](#_Toc207539431)

[Slika 2‑9 Prikaz kako neuronska mreža vidi različite značajke sa slike (preuzeto s https://www.nvidia.com/en-gb/glossary/computer-vision/) 11](#_Toc207539432)

[Slika 3‑1 Sažetak ciljeva, pravnih okvira, tehničkih parametara i planirane strukture baze podataka (izrađeno u web editoru https://plantuml.com/) 14](#_Toc207539433)

[Slika 3‑2 Tijek dohvaćanja i filtriranja po licenci (izrađeno u web editoru https://plantuml.com/) 16](#_Toc207539434)

[Slika 3‑3 Formiranja i provjere odgovora 18](#_Toc207539435)

[Slika 3‑4 Sažetak programske logike za jedno opažanje 19](#_Toc207539436)

[Slika 3‑5 Funkcija za preuzimanje slika 20](#_Toc207539437)

[Slika 3‑6 Dijagram toka za izradu baze podataka (izrađeno u web editoru https://plantuml.com/) 22](#_Toc207539438)

[Slika 3‑7 Struktura skupa podataka (izrađeno u web editoru https://plantuml.com/) 23](#_Toc207539439)

[Slika 3‑8 Funkcija za raspodjelu slika u na set za treniranje i set za validaciju 24](#_Toc207539440)

[Slika 3‑9 Pregled baze podataka (izrađeno u web editoru https://plantuml.com/) 27](#_Toc207539441)

[Slika 3‑10 Funkcija za parsiranje datoteke sa izvorima slika 28](#_Toc207539442)

[Slika 3‑11 Dijagram toka za rekonstrukciju baze podataka (izrađeno u web editoru https://plantuml.com/) 30](#_Toc207539443)

[Slika 4‑1 Implementacija ranog zaustavljanja 35](#_Toc207539444)

[Slika 4‑2 Krivulje učenja: trenirajući i validacijski gubitak te validacijska točnost po epohama. 36](#_Toc207539445)

[Slika 4‑3 Izvoz modela i prebacivanjeu Android model (izrađeno u web editoru https://plantuml.com/) 38](#_Toc207539446)

[Slika 4‑4 Tijek skripte za treniranje, evaluaciju i izvoz modela (izrađeno u web editoru https://plantuml.com/) 39](#_Toc207539447)

[Slika 5‑1 Distribucija artefakata: model.ptl i labels.txt u assets, kopiranje modela u internu pohranu, učitavanje putem Module.load(...) i mapiranje izlaznih indeksa na nazive klasa (izrađeno u web editoru https://plantuml.com/) 43](#_Toc207539448)

[Slika 5‑2 Sučelje nakon prepoznavanja vrste sa slike 47](#_Toc207539449)

[Slika 5‑3 Animacija skeniranja 49](#_Toc207539450)

[Slika 5‑4 Usporedba veličina prije pritiska i tijekom pritiska 50](#_Toc207539451)

[Slika 5‑5 Operativni tok aplikacije 52](#_Toc207539452)

Kazalo tablica

[Tablica 1 Usporedba između nadziranog, nenadziranog i podržanog strojnog učenja 5](#_Toc207539453)

Popis oznaka i kratica

CNN Convolutional Neural Networks

MLP Multi Layered Perceptron

API Application Programming Interface

engl Engleski

SAŽETAK/ABSTRACT I KLJUČNE RIJEČI/KEYWORDS

**Sažetak**

*Rad prikazuje cjelovit postupak izgradnje sustava za automatsku klasifikaciju biljaka i drveća te njegovu primjenu na Android uređaju. Skup podataka izrađen je automatiziranim preuzimanjem fotografija s iNaturalist API-ja uz poštivanje CC0/CC-BY licenci, pri čemu je prikupljeno 74 klasa i ukupno 54 253 slika. Implementirani su zapis izvora i atribucija, slučajna podjela na skupove učenja i provjere te skripta za potpunu rekonstrukciju baze iz dnevnika URL-ova, čime su osigurani ponovljivost i transparentnost. Model zasnovan na EfficientNet-B3 treniran je uz kombinaciju prostornih i fotometrijskih augmentacija, optimizaciju AdamW i rano zaustavljanje; validacijska točnost doseže oko 0,84–0,85, što potvrđuje prikladnost odabrane arhitekture za zadatak fine klasifikacije.*

*Za mobilnu primjenu model je izvezen u TorchScript formatu i uparen s mapom klasa (labels.txt). Android aplikacija, temeljena na CameraX-u, omogućuje dohvat slike kamerom ili iz galerije, korekciju orijentacije, pretprocesiranje na 300×300 uz ImageNet normalizaciju te izvođenje inferencije offline, bez mrežne ovisnosti. Rezultat se prikazuje u donjoj ploči (BottomSheet) s osnovnim informacijama o vrsti iz lokalne baze, dok vizualni povratni signal (scan traka i mikro-animacija gumba) poboljšava doživljaj korištenja. Rad doprinosi praktičnim smjernicama za izradu vlastite baze podataka, treniranje i izvoz modela te njegovu integraciju na uređaju, a ističe i ograničenja vezana uz neuravnoteženost klasa i sličnost pojedinih taksona, što otvara prostor za daljnje proširenje skupa i optimizacije.*

**Ključne riječi**

*računalni vid, klasifikacija biljaka, EfficientNet-B3, TorchScript, Android, iNaturalist, on-device inferencija*

***Development of an Android Application for Image-Based Classification of Plants and Trees***

**Summery**

*This thesis presents an end-to-end system for automatic classification of plants and trees and its deployment on Android devices. The dataset was built by programmatically downloading photographs via the iNaturalist API with CC0/CC-BY compliance, yielding 74 classes and a total of 54,253 images. Source URLs and attributions were recorded, the data were randomly split into training and validation sets, and a reconstruction script enables full dataset re-creation from the logged sources, ensuring reproducibility and transparency. An EfficientNet-B3 model was trained with a mix of spatial and photometric augmentations, the AdamW optimizer, and early stopping; the resulting validation accuracy reached approximately 0.84–0.85, confirming a good accuracy–efficiency balance for mobile use. For deployment, the model was exported to TorchScript and paired with a class map (labels.txt). The Android application, built on CameraX, acquires images from the camera or gallery, corrects EXIF orientation, preprocesses to 300×300 with ImageNet normalization, and performs fully offline inference. Predictions are presented in a BottomSheet together with basic species information from a local in-app database, while subtle UI feedback (a scanning bar and a shutter-press animation) improves the user experience. The work contributes practical guidelines for dataset construction, model training and export, and on-device integration, and it discusses limitations stemming from class imbalance and the fine-grained similarity of certain taxa, pointing to dataset expansion and mobile-oriented optimizations as natural next steps.*

**Keywords**

*computer vision, plant classification, EfficientNet-B3, TorchScript, Android, iNaturalist, on-device inference*