55. srečanje mladih raziskovalcev Slovenije 2021

GOZDNI ČUVAJ

Raziskovalno področje:
APLIKATIVNI INOVACIJSKI PREDLOGI IN PROJEKTI

Avtor: Mark Berdnik
Mentor: Branko Potisk, Manja Sovič Potisk
Šola: Srednja elektro-računalniška šola Maribor

Maribor, 2021

55. srečanje mladih raziskovalcev Slovenije 2021

GOZDNI ČUVAJ

Raziskovalno področje:
APLIKATIVNI INOVACIJSKI PREDLOGI IN PROJEKTI

Avtor: Mark Berdnik
Mentor: Branko Potisk, Manja Sovič Potisk
Šola: Srednja elektro-računalniška šola Maribor

Maribor, 2021

Kazalo vsebine

Kazalo slik

Kazalo grafov

Kazalo tabel

1 Povzetek

Gozdni čuvaj je mobilna aplikacija, ki uporabnikom omogoča avtomatsko prepoznavanje različnih vrst gliv in prikaz njihovih lastnosti. Mobilna aplikacija je zgrajena za vse starostne skupine, saj prijazen izgled omogoča hitro spoznavanje z aplikacijo. Z enim klikom uporabnik zajame sliko izbrane glive, aplikacija pa jo avtomatsko posreduje na naš že trenirani model, ki s pomočjo računalniškega vida klasificira vrsto glive in izpiše dane lastnosti. Aplikacija deluje na principu globokega učenja, z uporabo različnih odprtokodnih ogrodij kot so: *Flutter*, *TensorFlow Lite*, *Node.js* itd. Celotna aplikacija je za končnega uporabnika brezplačna in vključuje vse funkcije, ki jih ponujamo brez dodatnih stroškov. S to aplikacijo želimo na zabaven način informirati javnost o pravilnem ravnanju in nabiranju gliv ter preprečiti zaužitje strupenih vrst.

Ključne besede: strojno učenje, računalniški vid, gozd, varovanje narave, glive

2 Zahvala

Sprva se želim zahvaliti mentorjema za vse nasvete in spodbude, ki sta mi jih dala v teku razvoja inovacijskega predloga. Posebej sem še hvaležen članom Gobarskega društva Lisička Maribor[1], ki so mi bili pripravljeni svetovati in podati začetne reference.

[1] http://www.gobe.si/

3 Uvod

Ni skrivnost, da večina Slovencev v jesenskem času preživlja svoj prosti čas v gozdovih, kjer nabira razne glive za užitje ali pa za pohvalo na socialnih omrežjih. Težave nastanejo, ker imajo nekateri ljudje nabiranje gliv za konjiček, nimajo pa potrebnega znanja, kar lahko vodi k resnim zapletom osebnega zdravja in tudi nepravilnemu posegu v naravo. Zaradi teh in še veliko več razlogov smo se odločil, da ustvarim mobilno aplikacijo, ki ni le poučna ampak tudi zabavna za uporabo in skladno s tem aplikativna za vse starostne skupine.

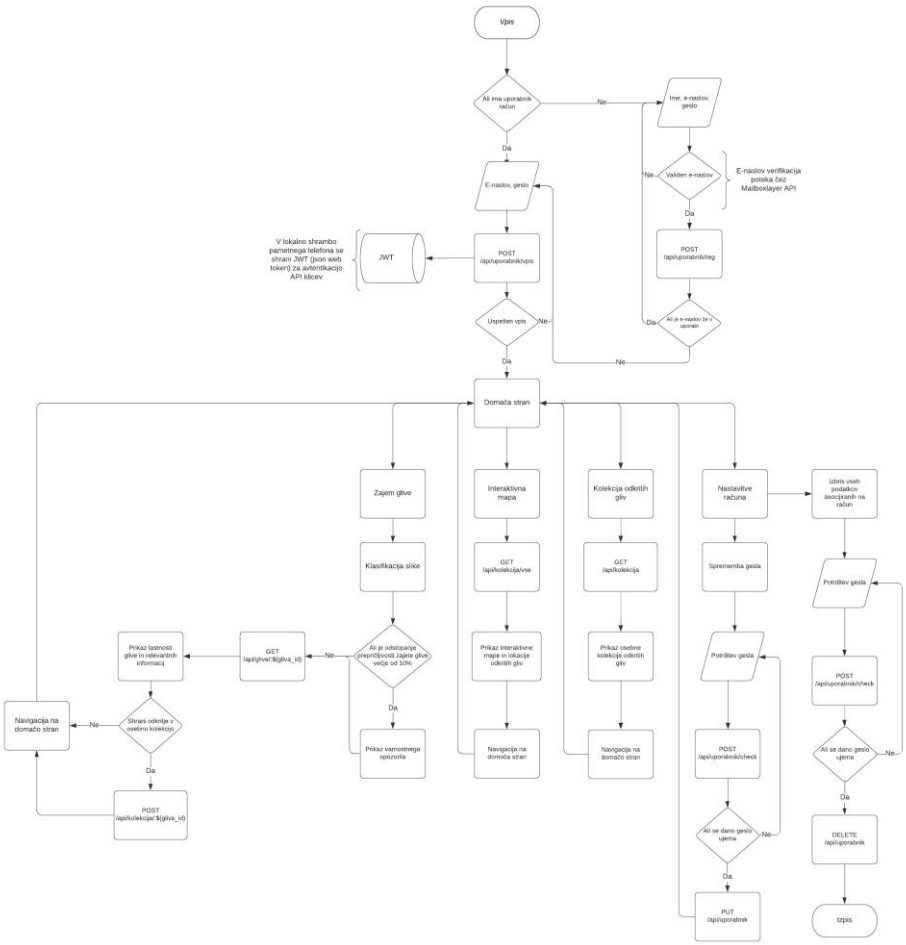
Mobilna aplikacije je zasnovana tako, da uvajanje traja le nekaj minut. Previdna izbira tipografije in splošna oblika gradnikov pripomoreta k lažji razumljivosti. Gozdni čuvaj z uporabo umetne inteligence uvrsti glivo v pravilno družino in izpiše vse relevantne informacije o njej, ki končnemu uporabniku koristijo. Končni uporabnik po klasifikaciji glive prejme možnost, da deli najdbo z ostalimi gobarji preko interaktivne mape, ki je vgrajena v aplikacijo ali pa shrani najdbo v osebno zbirko.

3.1 Cilji

Aplikacijo smo izdelali z namenom, da bo končni uporabnik seznanjen s tem, kako pravilno ravnati z določeno glivo in kakšne posledice ima lahko za posameznika. Na socialnih omrežjih opažamo zanemarjanje pravilnega ravnanja z glivami. Ta in še mnogo več razlogov nas je vzpodbudilo, da izdelamo aplikacijo, katere končni cilj je poučiti uporabnike, ne le o pravilnem ravnanju z glivami, temveč tudi navdušiti vse generacije o varovanju narave.

3.2 Metodologija dela

Najprej smo začeli z izdelavo diagrama poteka mobilne aplikacije (slika 1), kateri je znatno pomagal pri vizualizaciji projekta. Po dokončanem diagramu smo pričeli z izdelavo API-ja, ki služi kot medij prenosa in procesiranja podatkov med mobilno aplikacijo in podatkovno bazo. Pri izdelavi vmesnika smo bili pozorni na zaščito občutljivih podatkov, saj je ključno, da so dani podatki končnega uporabnika varno shranjeni v podatkovni bazi. Po dokončanem API-ju smo pričeli z zasnovo oblike mobilne aplikacije v urejevalniku vektorskih grafik *Adobe Illustrator*[2]. Uporaba omenjenega programa se nam je prvotno zdela malenkost kompleksa. Ob gradnji nevronskega omrežja se je pojavila težava razlikovanju gliv pri različnih družinah. Skladno s tem se je povečala nevarnost za uporabnike, saj so mnoge vrste gliv strupene za konzumiranje. Zaradi te ključne težave smo se obrnili na člane Gobarskega društva Lisička Maribor, ki so z veseljem pristopili na pomoč in predlagali, naj se osredotočimo na družino cevark (lat. *boletus*). Čeprav je učenje nevronskega omrežja potekalo precej časa, je izdelava osrednjega dela aplikacije v ogrodju *Flutter* zaradi dosedanjih izkušenj vzela manj časa. Ostala je še izdelava logotipa, saj je marsikatera priljubljena aplikacija zaznamovana s kakovostnim logotipom.



Slika 1: Diagram poteka mobilne aplikacije

[2] https://www.adobe.com/products/illustrator.html

3.3 Celostna grafična podoba

Ključna pri uspehu in prepoznavnosti aplikacije je celostna grafična podoba, ki zajema vse od vrhunskega logotipa do premišljeno izbranih barv, ki nastopajo v aplikaciji. Predhodno pa se moramo vprašati, kaj želimo z našim logotipom končnemu uporabniku sporočiti in kakšen vtis želimo pri uporabniku pustiti. Spodbudo za izdelavo logotipa smo dobili iz začetne ideje o ozaveščanju ljudi o varstvu narave, še posebej gozdov. To sporočilo smo želeli prenesti v celostno podobo mobilne aplikacije, ki zajema bistvo tega projekta.

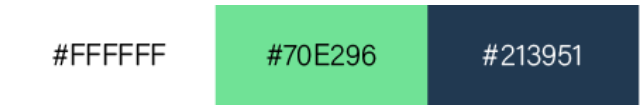
3.3.1 Logotip

Logotip (slika 2) ni kompleksno strukturiran, saj enostavna oblika omogoča hitro prepoznavnost med ljudmi. Temeljna struktura logotipa ponazarja gozd s prelivom v gorovje in sončni vzhod, ki je izhodišče celote. Izhodiščno sporočilo je varovanje narave, saj brez nje ne vidimo prihodnosti.

Slika 2: Prikaz vektorjev logotipa v namizni aplikaciji Adobe Illustrator

3.3.2 Barvna shema

V logotipu nastopajo tri različne barve: zelena, bela in modra (slika 3). Barve logotipa in celotne aplikacije predstavljajo pomembno vlogo v počutju končnega uporabnika. Z zeleno barvo želimo pri uporabniku vzbuditi občutek varnosti, saj je psihološki občutek te barve umirjenost, zadovoljstvo in uravnoteženost. Svojo vlogo ima tudi modra barva, ki vzbudi zaupanje, zvestobo in sprostitev.



Slika 3: Barvni spekter logotipa

3.4 Programski jeziki

3.4.1 Python

Zaradi pestre izbire odprtokodnih knjižnic in ogrodij smo se odločili, da bomo za izdelavo aplikacije uporabili objektno-orientiran programski jezik *Python*. *Python* v primerjavi z ostalimi objektno-orientiranimi programskimi jeziki z nekaj izjemami, ima sestavo sintakse, ki je zelo enostavna za vzdrževanje kode. Skupnost omenjenega programskega jezika je vedno pripravljena pristopiti k pomoči, v kolikor se pojavi kakšna težava z osebnim projektom ali dodeljeno nalogo.

3.4.2 JavaScript

Razvoj aplikacijskega vmesnika je narejen v programskem jeziku *JavaScript*, ki je kot *Python* objektno-orientiran. Vsestranskost jezika je precejšen bonus, ker omogoča razvoj *frontend*[3] in *backend*[4] aplikacij. Praviloma vsi sodobni brskalniki in tudi znana podjetja podpirajo *JavaScript*, to ga izpostavi med ene izmed najprepoznavnejših programskih jezikov tega časa.

3.4.3 Dart

Za izdelavo mobilne aplikacije je uporabljen objektno-orientiranim programskim jezikom *Dart*, ki se je prvič pojavil leta 2011. Prvotno se je ta jezik uporabljal za interno izdelavo spletnih strežnikov in mobilnih aplikacij pri podjetju Google. Leta 2017 je vzdignil zanimanje razvijalcev, ko je Google uradno napovedal odprtokodno ogrodje *Flutter* za razvoj mobilnih, kasneje tudi spletnih aplikacij. Od takrat naprej se je zanimanje za *Dart* občutno povečalo.

3.4.4 SQL

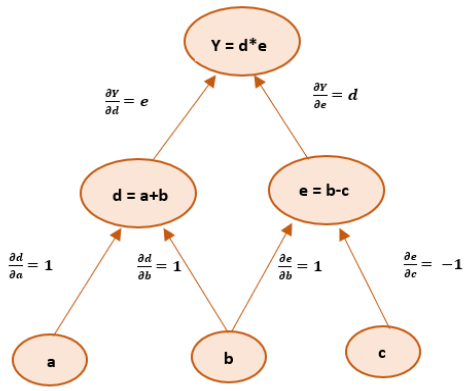
Obdelava in shramba podatkov v podatkovno bazo, se je izvedla s pomočjo programskega jezika *SQL* (ang. *simple query language*). Omenjen jezik je dobro dokumentiran in je primeren za delo z relacijskimi podatkovnimi bazami.

[3] Frontend – sprednji del spletne strani, kjer uporabnik komunicira s stranjo.
[4] Backend – zadnji del spletne strani, uporabnik kot obiskovalec strani ne vidi.

3.5 Ogrodja

3.5.1 Tensorflow Lite

Globoko učenje smo izvajali preko odprtokodnega ogrodja za strojno učenje *Tensorflow Lite*. To ogrodje ima prilagodljiv ekosistem orodij, knjižnic in virov, ki razvijalcem omogočajo enostavno treniranje modelov. Ogrodje zaradi fleksibilnosti omogoča istočasno uporabo več modelov ali več različic istega modela. Kalkulacije, ki jih izvaja ogrodje, so opisane z uporabo grafov pretoka podatkov, prikazane na sliki 4. Vsako vozlišče grafa (ang. *node*) predstavlja primer aritmetičnih operacij, vsak rob pa je večdimenzionalni nabor podatkov (ang. *tensor*), na katerem se izvajajo omenjene operacije.



Slika 4: Primer računskega grafa

3.5.2 Node.js

Aplikacijski vmesnik (ang. *application programming interface)* je razvit z odprtokodnim ogrodjem *Node.js*. Ta se uporablja kot privzet programski jezik *JavaScript* in gostuje ter procesira klice s strani virtualnega strežnika. Vloga ogrodja je obdelovanje CRUD-funkcij, ki jih končni uporabnik prikliče v mobilni aplikaciji kot asinhrono izvajalno okolje, *Node.js* je namenjen izdelavi razširjenih aplikacijskih vmesnikov, ki so pripravljeni za produkcijske namene.

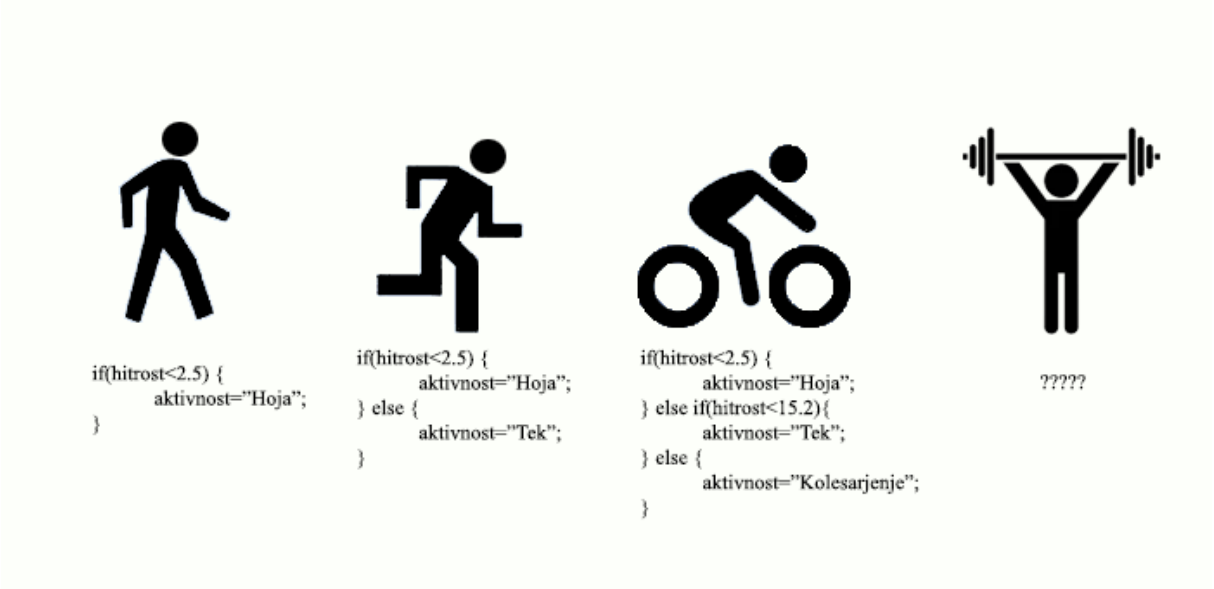
3.5.3 Flutter

Mobilna aplikacija je bila ustvarjena z odprtokodnim ogrodjem *Flutter*, ki temelji na osnovi gradnikov (ang. *widget*). Ena izmed mnogih prednosti ogrodja je možen razvoj medplatformskih aplikacij. *Flutter* vsebuje SDK (ang. *software development kit*), ki vključuje zbirko orodij, ki nam pomagajo pri razvoju aplikacije. Ogrodje uporablja programski jezik *Dart*, ki ga je skladno z ogrodjem razvilo podjetje Google. Od začetne izdaje leta 2017 do današnjega dne, si je *Flutter* razširil lastno prepoznavnost in interes med razvijalci mobilnih aplikacij, še posebej zaradi hitrega razvoja, ki ga lastna skupnost prispeva dnevno.

3.6 Klasifikacija gliv

3.6.1 Osnove strojnega učenja

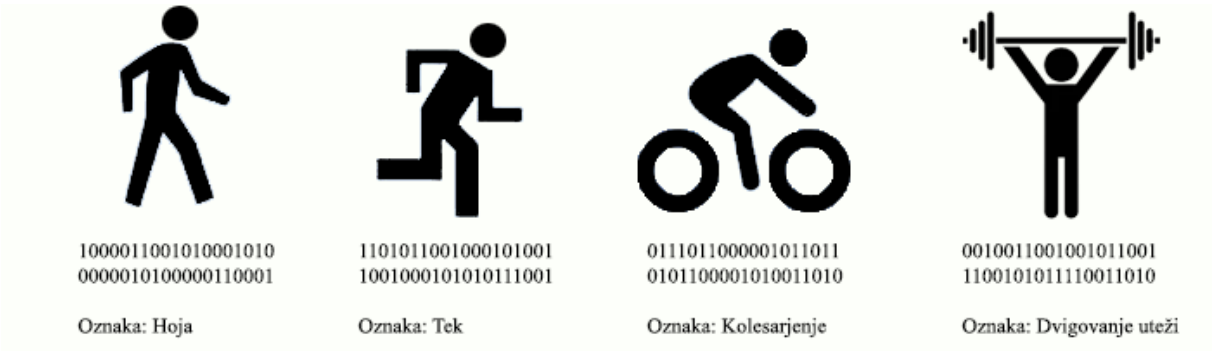
Zamislimo si besedno zvezo strojno učenje v metaforičnem kontekstu. Mi, kot razvijalci, spodbujamo računalnik, da programira sam. Začnimo s primerjavo konvencionalnega programiranja in strojnega učenja. Pri konvencionalnem programiranju zapisujemo program v tradicionalnem postopkovnem jeziku, kot je montažni jezik (ang. *assembly language*) ali jezik na visoki ravni (ang. *high-level compiler language*). Za lažje razumevanje smo ustvarili slikovni prikaz kako bi izgledal konvencionalen program, ki omogoča prepoznavanje aktivnosti posameznika na podlagi parametra za hitrost.



Slika 5: Primer konvencionalnega programiranja

Na zgornji sliki uporabljamo zgolj eno informacijo (hitrost), toda kako prepoznati dvigovanje uteži? Zaradi kompleksnosti te aktivnosti potrebujemo več kot le eno spremenljivko, da pridobimo dovolj informacij za prepoznavanje le-te. Praktično katerakoli aktivnost ustvari specifične in edinstvene podatke, ki predstavljajo trenutno dejavnost posameznika. Zato prepoznavanje aktivnosti samo s spremenljivko hitrost ni dovolj, ker imajo različni ljudje različen slog hoje, hkrati pa na njihovo premikanje vpliva še mnogo drugih dejavnikov.

Rešitev za takšen problem je strojno učenje, ker deluje z zbiranjem podatkov in prepoznavanjem vzorcev ter vključuje minimalno človekovo posredovanje. Skoraj vsak zaplet, ki nastane v konvencionalnem programiranju in je definiran z določenim sklopom pravil, je enostavno odpraviti in avtomatizirati s strojnim učenjem. Spodnja slika nakazuje rešitev omenjenega problema, ki je odpravljen, ki je odpravljen s pomočjo strojnega učenja.

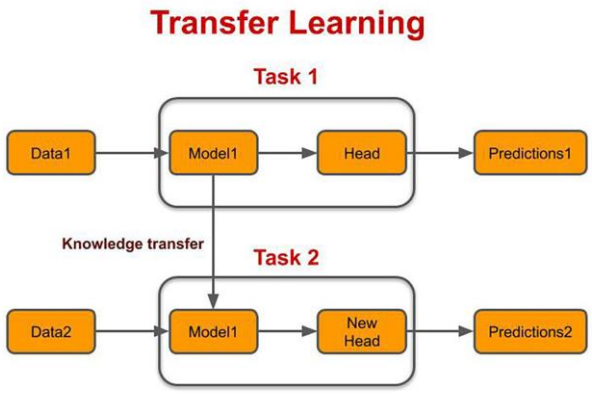


Slika 6: Primer klasifikacije slik s strojnim učenjem

3.6.2 Prenosno učenje

Metoda učenja s prenosom temelji na znanju že usposobljenega modela strojnega učenja, ki izkoristi pridobljena znanja za klasifikacijo povezanega problema. V prejšnji točki smo navidezno usposobili klasifikator za predvidevanje aktivnosti. To znanje, ki ga je model pridobil med treningom lahko uporabimo za prepoznavanje drugih stvari, v našem primeru je ta stvar kolekcija raznovrstnih gliv. S prenosnim učenjem je naš končni cilj izkoristiti naučen model za izboljševanje drugega. Splošna ideja je uporabiti znanje, ki se ga je model naučil iz naloge z mnogo več razpoložljivimi podatki o usposabljanju, ki ima skladno s tem svoje označbe v novi nalogi, ki nima toliko podatkov. Namesto, da bi učni proces začeli iz ničle, začnemo z vzorci, ki smo se jih naučili pri reševanju sorodne naloge.

Za primerjavo bomo vzeli računalniški vid, saj nevronske mreže načeloma poskušajo zaznati robove v prejšnjih slojih, oblike v srednjem sloju in nekatere značilnosti, ki so povezane z nalogo v poznejših plasteh. Pri učenju s prenosom, se uporablja zgolj zgodnji in srednji sloj, slednje pa samo preusmerimo. Ta način obdelovanja pomaga izkoristiti označene podatke o nalogi, za katero je bila prvotno usposobljena.



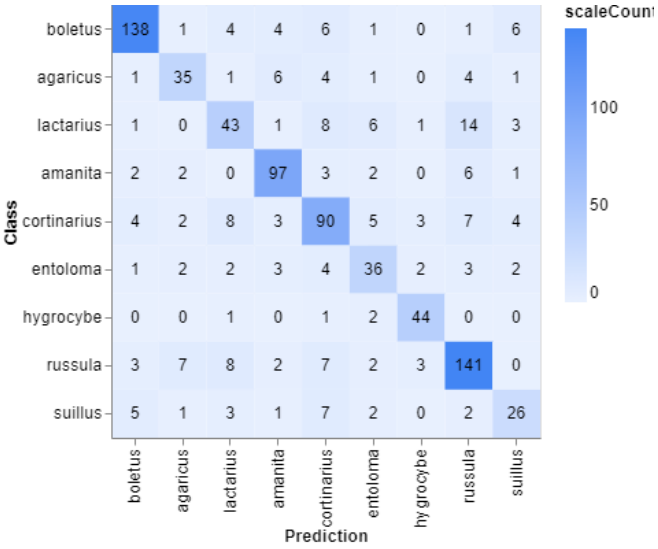
Slika 7: Prikaz prenosnega učenja

Prihranek časa za usposabljanje in boljša zmogljivost nevronskih mrež sta dve izmed mnogih prednosti prenosnega učenja. Načeloma je za usposabljanje nevronske mreže na začetku potrebnih veliko podatkov, vendar pride do zapletov pri zbiranju le-teh. Pri tem je prenosno učenje zelo koristno, saj lahko zgradimo trden model strojnega učenja s sorazmerno malo podatki o usposabljanju, ker je model že predhodno usposobljen. Zraven tega se čas za treniranje modela skrajša, ker se lahko zgodi, da nevronska mreža, ki nima nič predhodnih podatkov potrebuje dneve ali celo tedne za klasifikacijo kompleksnih nalog.

3.6.3 Natančnost oz. nenatančnost prepoznavanja gliv

Eden izmed izzivov, ki je nastopil pri izdelavi te aplikacije, je bilo vprašanje zanesljivosti treniranega modela, saj končni uporabnik načeloma zaupa presoji, ki jo izvede aplikacija. Problem se pojavi že pri splošni ideji prepoznavanja gliv, ki pripadajo različnim družinam, saj sta lahko glivi dveh različnih družin identični na prvi pogled in to pusti skrajno malo prostora za napake. Zaradi trenutnih razmer je bilo oteženo in dokaj omejeno pridobivanje kakovostnih primerkov gliv različnih družin. Ključno je, da imamo obširno kolekcijo istovrstnih gliv za treniranje modela, saj je izbran projekt skrajno kompleksen zaradi mnogovrstnosti gliv, kot je že omenjeno v prejšnjih točkah. Odločili smo se, da se bomo osredotočili le na eno družino gliv, to je družina cevark (lat. *boletus*).

V tej točki se bomo izključno posvetil treniranju modela za klasifikacijo vseh družin gliv, da lahko potem uvrstim ciljne subjekte v pravilno družino in se osredotočim na klasifikacijo le izbrane družine, torej cevark. Kolekcijo slik najpogostejših severnoevropskih vrst gliv, ki so uvrščene na podlagi pripadnosti družin, smo pridobili iz javne skupnosti za obdelavo in procesiranje podatkov *Kaggle*[5]. Začetni cilj je bil ugotoviti izstopanje prepričljivosti klasifikacije dodeljene glive v specifični družini. V praktičnem delu bo opisan postopek treniranja modela.
Učinkovitost klasifikacije strojnega učenja lahko preverimo s tehniko matric zmede (ang. *confusion matrix*). S to metod vnesemo testne subjekte, za katere so znane resnične vrednosti in kot končen izid izvemo natančnost našega treniranega modela.



Graf 1: Matrica zmede za treniran model

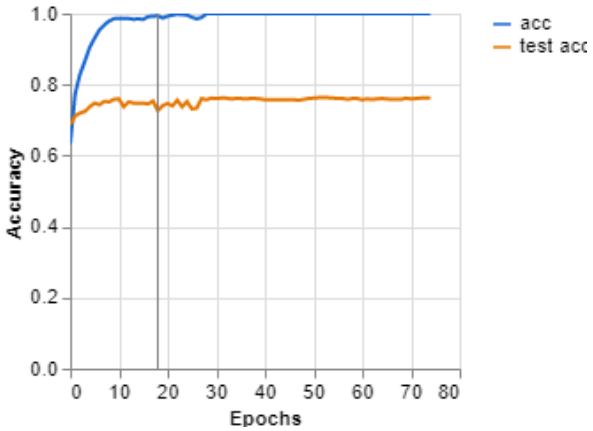
[5] https://www.kaggle.com/

Iz zgoraj razbranih podatkov lahko sklepamo, da ima naš model precej natančne napovedi zgolj za razred cevark. Kompletni nabor podatkov za treniranje modela je vseboval 5664 slik gliv različnih družin, od tega je naša ciljna družina vsebovala 1073 slik. Vsota testnih vzorcev, ki smo jih uporabili za preverjanje natančnosti družine cevark, je vsebovala 161 slik gliv, med temi je naš model pravilno prepoznal 138 pripadajočih tega razreda. Procentualno odstopanje je 14%, sicer ta podatek zaznamuje precejšno natančnost, vendar zaradi občutljivosti tega projekta je tega odstopanja izrazito preveč. Zaradi navedenega razloga potrebujemo več vzorcev slik gliv, ki pa jih nismo mogli pridobiti zaradi trenutne pandemije. Natančnost napovedi modela za vsak razred je predstavljena s tabelo, ki je prikazana spodaj.

Tabela 1: Natančnost napovedi za dani razred

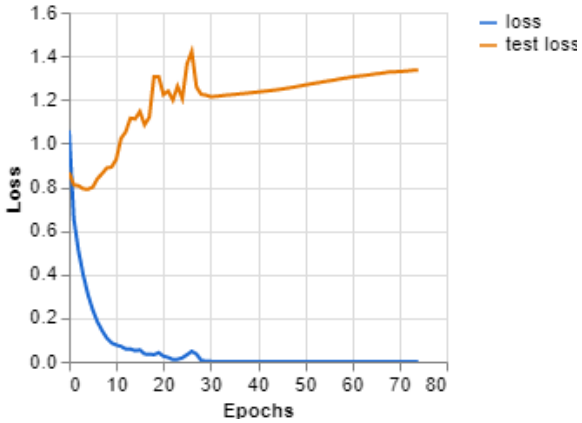
Razred	**Natančnost**	**Št. vzorcev**
Boletus	0.86	161
Agaricus	0.66	53
Lactarius	0.56	77
Amanita	0.86	113
Cortinarius	0.71	126
Entoloma	0.65	55
Hygrocybe	0.92	48
Russula	0.82	173
Suillus	0.55	47

Ključne natančnosti klasifikacije so vsekakor parametri, ki smo jih uporabili za treniranje tega modela. Število epoh (ang. *epoch*) je 75, kar pomeni, da je bil vsak vzorec glive 75-krat doveden skozi treniran model. Na splošno je večje število epoh boljše za natančnost klasifikacije. Velikost serije (ang. *batch*) za obdelavo vzorcev pred posodobitvijo modela znaša 16, to pomeni, da bodo podatki razdeljeni na 354 serij (5664/16). Ko bo skozi model dovedenih vseh 354 serij, bo končan en epoh. Ker želimo v prihodnje izboljšati treniran model, si oglejmo natančnost klasifikacije, ki je prikazana s pomočjo grafa.



Graf 2: Natančnost klasifikacij

Prikazana natančnost je odstotek klasifikacij, ki jih model uspešno prepozna med treningom. Če naš model klasificira 4712 vzorcev od 5664, je natančnost 0.8 (4712/5664). V povprečju je natančnost prepoznavnosti naših testnih vzorcev 0.8, kar ne zadošča za dokaj varno prepričanje o zanesljivosti treniranega modela zaradi mnogovrstnosti gliv. Izguba na epoh je prikazana na spodnjem grafu.



Graf 3: Izguba na epoh

Izguba je merilo za ovrednotenje modela, kako dobro se je naučil napovedati prave klasifikacije za podani skupek vzorcev. V zgornjem primeru je izguba testnih vzorcev večja od nič, kar pomeni, da so napovedi izrazito pomanjkljivi. Potrebno bi bilo pridobiti večji nabor vzorcev ter podvojiti število epoh za treniranje modela. Glede na navedeno zanesljivost našega modela, bi ga bilo potrebno še izboljšati, saj gre v končni fazi za aplikacijo za klasifikacijo gliv, ki jih mnogo končnih uporabnikov konzumira. Zato je pravilnost napovedi naše nevronske mreže nujna.

4 Izvedba projekta

4.1 Podatkovna baza

Ključno pri shrambi naših podatkov je podatkovna baza, ki ima štiri glavne entitete. Entiteta *uporabnik* skrbi za shrambo uporabniških podatkov in s tem omogoča prijavo in registracijo v mobilno aplikacijo. Atributa *e_naslov* in *geslo* vsebujeta osebne podatke uporabnika, zaradi tega je pomembno, da poskrbimo za varnost te podatkovne baze. Namen entitete *kolekcija* je shramba zajetih gliv končnih uporabnikov. Shranjuje vse od geolokacije najdene glive do klasificirane slike. Podrobni pogled v najdbo nam omogoča entiteta *gliva*, saj zajema informacije, ki so temeljne za identifikaciji glive. *Družina_gliv* vključuje bistvene podatke o družini gliv in spletno povezavo za dodatne reference.

Izbrali smo odprtokodno rešitev *MariaDB* (verzija 10.5.9) za relacijsko podatkovno bazo tega projekta, saj ponuja boljši pomnilniški mehanizem kot splošno znana MySQL podatkovna baza. Vizualizacijo relacijske sheme smo ustvarili s pomočjo spletnega oblikovalnika za izdelavo ER diagramov *dbdiagram*[6]. (slika 8)

Slika 8: Prikaz relacijske sheme

[6] https://dbdiagram.io/

4.2 Aplikacijski vmesnik

Komuniciranje in obdelavo podatkov med mobilno aplikacijo ter podatkovno bazo omogoča aplikacijski vmesnik (ang. *application programming interface* oz. *API)*. Ustvarili smo ga s pomočjo odprtokodnega ogrodja *Node.js*, saj je primeren za raznovrstno uporabo in ima obširno skupnost, ki je pripravljena pomagati, če nastane zaplet pri izdelavi. Aplikacijski vmesnik vsebuje 10 končnih točk (ang. *endpoint*), ki skrbijo za sprejem HTTP zahtev v obliki tako imenovanih *CRUD* (ang. *create*, *read*, *update*, *delete*) klicev. V primeru, da je klic uspešno izveden, vmesnik vrne mobilni aplikaciji skupek podatkov v podobi *JSON* formata za lažje obdelovanje podatkov.

Varstvo podatkov je primarna prioriteta, saj nam uporabniki zaupajo za varno obdelavo osebnih podatkov, kot so npr.: e-naslovi in gesla. Zaradi navedenega je za shranjevanje gesel uporabljena knjižnica *Bcrypt*, ki s pomočjo kodiranja zagotovi povečano varnost. Kljub vsemu je kodiranje občutljivih podatkov trivialno, če podatkovna baza ni zavarovana z močnim geslom in v primeru, da aplikacijski vmesnik uporablja metode, ki predstavljajo kompromis le-tem podatkom. Avtentikacija uporabnikov za določene klice končnih točk se izvaja z uporabo knjižnice *passport* in JWT-ja oz. spletnega žetona (ang. *json web token*). Spletni žeton preveri verodostojnost uporabnika, v primeru, da izvede klice na željeno končno točko (slika 10).

Slika 9: Prikaz ključnih podatkov izbrane glive v namizni aplikaciji Postman

```
app.get(
  "/glive/:id",
  passport.authenticate("jwt", { session: false }),
  (req, res) => {
    let sql = `SELECT g.ime AS 'gliva_ime', g.opis AS 'gliva_opis', g.nevarna AS 'gliva_nevarna',
     g.wiki_link AS 'gliva_wiki_link', d.ime AS 'druzina_ime', d.opis AS 'druzina_opis',
      d.wiki_link AS 'druzina_wiki_link' FROM gliva g JOIN druzina_gliv d ON g.id = d.id WHERE g.id = ? `;
    db.query(sql, req.params.id, function (err, data) {
      if (err) console.log("Napaka: ", err);
      try {
        if (data[0]) {
          res.json({
            gliva: data,
            uspeh: true,
          });
        }
      } catch (e) {
        console.log(e);
        res.json({
          uspeh: false
        })
      }
    });
  }
);
```

Slika 10: Prikaz končne točke

4.3 Treniranje nevronskega omrežja

Pridobljena zbirka vzorcev družin gliv in tudi posamičnih vrst iz ciljne skupine cevark je vsebovala 5664 slik gliv, ki pripadajo različnim družinam, in 121 slik treh vrst gliv, ki pripadajo omenjeni družini, kar je seveda mnogo premalo vzorcev za natančne napovedi treniranega modela. Zadošča pa za prikaz izvedbe projekta. Treniranje nevronske mreže je izveden v spletni aplikaciji *Google Colaboratory*[7], ki omogoča izvajanje programskega jezika *Python* preko spletnega brskalnika. Spletna aplikacija deluje kot izolirano okolje in je še posebej primerno za strojno učenje in predelavo podatkov.

Nevronsko mrežo smo trenirali z odprtokodnim ogrodjem za strojno učenje *Tensorflow Lite* in odprtokodno knjižnico za razvoj in ocenjevanje globokega učenja *Keras*. Skladno s tem smo uporabili metodo prenosnega učenja. To je zagotovilo konvolucijsko nevronsko omrežje *MobileNetV2*, ki se izredno dobro obnese v mobilnih napravah za strojno učenje. Pri treniranju modela smo uporabil 75 epoh (ang. *epoch*) in velikost serije 16 (ang. *batch*) s stopnjo učenja 0,01. V povprečju je treniranje nevronske mreže trajalo 3 ure (~ 2,5-min/epoh).

```
89  model = tf.keras.Sequential([osnovni_model,
90    tf.keras.layers.Conv2D(32,3, activation='relu'),
91    tf.keras.layers.Dropout(0.2),
92    tf.keras.layers.GlobalAveragePooling2D(),
93    tf.keras.layers.Dense(9, activation='softmax')
94  ])
```

Slika 11: Prikaz dodajanja slojev nevronski mreži

Vizualizacijo matrice zmede in grafa za izgubo ter natančnost učinkovitosti treniranega modela smo opravili z odprtokodno knjižnico *Matplotlib*, saj to je ključnega pomena za odkrivanje pomanjkljivosti pri klasifikaciji določenih vzorcev. Po dokončanem treniranju smo usposobljeno nevronsko mrežo pretvoril v *Tensorflow Lite* datoteko, ki ima podaljšavo *.tflite* in jo nato izvozil za implementacijo v mobilno aplikacijo.

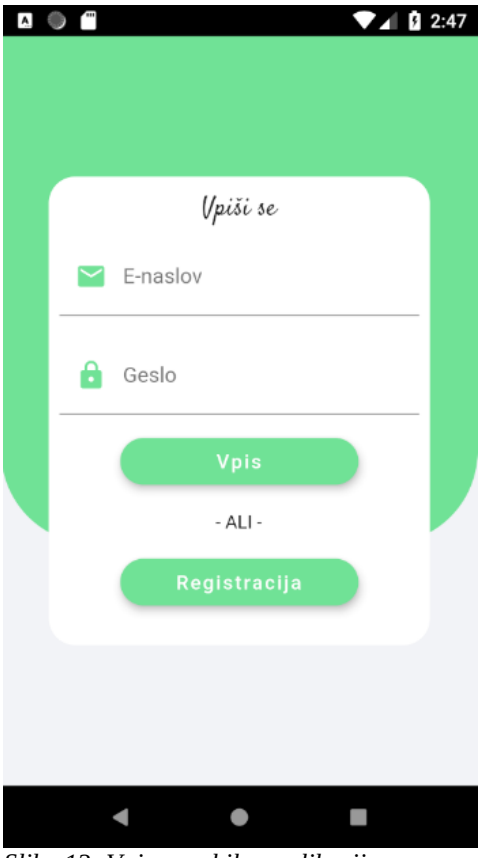
[7] https://colab.research.google.com/

4.4 Mobilna aplikacija

Izdelava mobilne aplikacije je potekala s pomočjo odprtokodnega ogrodja *Flutter*. Izdelava osnovnih skic individualnih strani pa v namizni aplikaciji za vektorsko obdelavo grafik *Adobe Illustrator*. Emulator[8] pametnega telefona nam je pomagal pri prihranitvi časa, saj je gradnja z njim veliko hitrejša kot, če bi uporabili konvencionalni način prenosa različic mobilne aplikacije na fizični telefon.

4.4.1 Vpis in registracija

Končni uporabnik po uspešni namestitvi mobilne aplikacije mora ustvariti nov račun za omogočanje funkcionalnosti shranjevanja gliv v osebno kolekcijo in deljenje najdb na interaktivni mapi. Ko vnese zahtevane podatke (*e-naslov, geslo in potrditev gesla*), se izzove *CRUD* klic za preverjanje validnosti e-naslova. V primeru da e-naslov ni zaseden in je skladno s tem tudi validen, se uporabniku ustvari račun, s katerim se lahko vpiše in dostopa do vseh funkcij, ki jih aplikacija ponuja. Vpis v aplikacijo ne predstavlja nobene težave, saj so gradniki za vhod podatkov in gumbov primerne velikosti za vse starostne skupine. (slika 12 in 13)



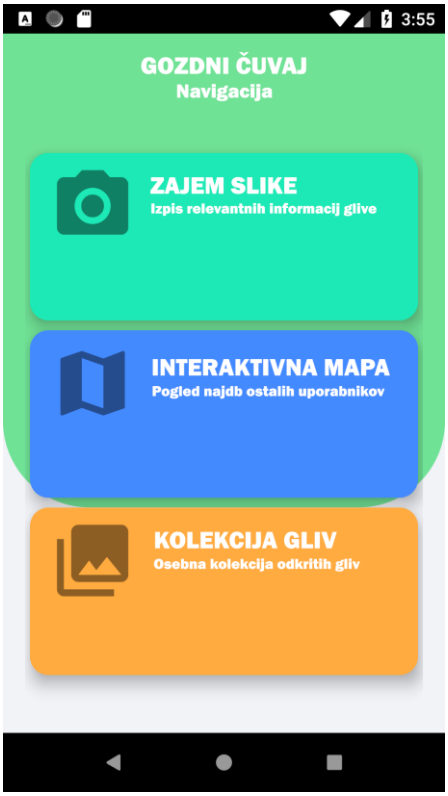
Slika 13: Vpis v mobilno aplikacijo

Slika 12: Registracija uporabnika

[8] Emulator – računalniški program, ki uporablja izolirano okolje za izvršitev programske opreme.

4.4.2 Naslovna stran

Osrednji del mobilne aplikacije je naslovna stran oz. navigacijska stran, kjer se končnemu uporabniku prikaže izbira treh sekcij. Vsaka sekcija predstavlja določeno funkcionalnost, ki jo ponuja aplikacija. Gradniki so tokrat v obliki kart, kjer s sodobno obliko in razberljivo tipografijo uporabnik enostavno poišče željeno sekcijo. S klikom na izbrano sekcijo se uporabnik premakne na novo stran, kjer ga pričaka minimalistična prehodna animacija med stranmi.



Slika 15: Naslovna stran

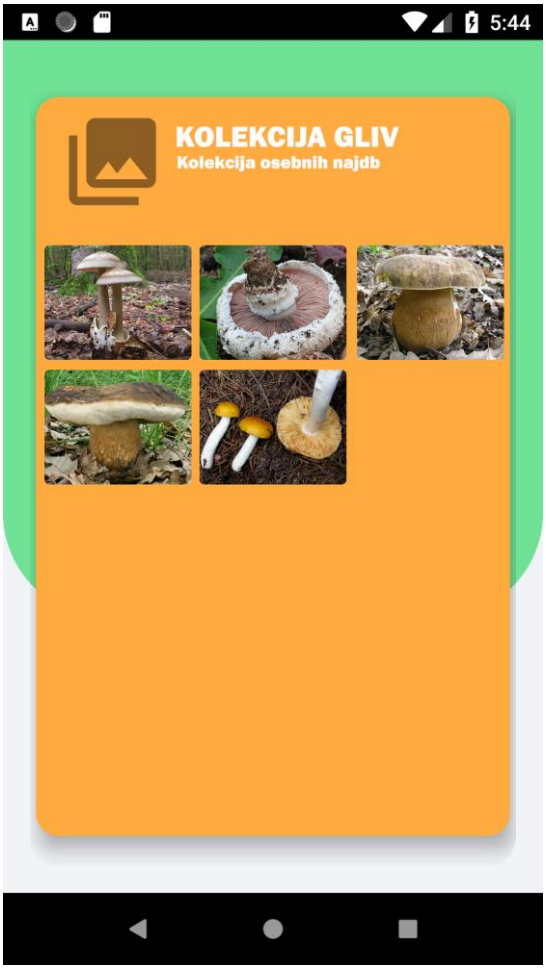
```
Widget build(BuildContext context) {
  return SafeArea(
    child: Scaffold(
      resizeToAvoidBottomPadding: false,
      backgroundColor: Color(0xfff2f3f7),
      body: Stack(
        children: <Widget>[
          Container(
            height: MediaQuery.of(context).size.height * 0.65,
            width: MediaQuery.of(context).size.width,
            child: Container(
              decoration: BoxDecoration(
                color: Palette.lightGreen,
                borderRadius: BorderRadius.only(
                  bottomLeft: const Radius.circular(70),
                  bottomRight: const Radius.circular(70),
                ), // BorderRadius.only
              ), // BoxDecoration
            ), // Container
          ), // Container
          Column(
            mainAxisAlignment: MainAxisAlignment.center,
            children: <Widget>[
              _buildCardFoto(),
              _buildCardKolekcija(),
              _buildCardMapa(),
            ], // <Widget>[]
          ) // Column
        ], // <Widget>[]
      ), // Stack
    ), // Scaffold
  ); // SafeArea
  }
}
```

Slika 14: Gradnik naslovne strani

4.4.3 Podrobne informacije in kolekcija gliv

Ob uspešnem zajetju in klasificiranju slike se končnemu uporabniku prikaže stran, ki vsebuje splošne informacije glive, kot npr.: *ime, družina, opis, prepričljivost* ipd. V primeru, da je prepričljivost manjša od 90%, se uporabniku izriše varnostno okence s potrditvenim poljem in splošnim opozorilom za samoodgovornost. V primeru, da nevronska mreža ne zazna nobene glive, ki pripada družini cevark, se podatki ne izpišejo, temveč se pojavi demonstracijsko obvestilo. Uporabniku je podana tudi opcija shrambe te slike v osebno kolekcijo gliv, kjer lahko dostopajo v pretekle najdbe brez kakršnekoli skrbi za izgubo teh slik ob formatiranju osebnega telefona, saj se vse shranijo v našo podatkovno bazo. Vsekakor, če se končni uporabnik odloči za shrambo te slike v osebno kolekcijo, se na interaktivni mapi prikaže lokacija najdišča za 5-dni, to je seveda vidno vsem registriranim uporabnikom v sekciji *interaktivna mapa*.

Slika 17: Podrobne informacije zajete glive



Slika 16: Osebna kolekcija gliv

4.4.4 Interaktivna mapa

Druženje gobarjev vseh starostnih skupin omogoča sekcija interaktivna mapa. Ob kliku na sekcijo se s pomočjo *Maps SDK* izriše mapa, centrirana na lokacijo končnega uporabnika v primeru, da je aplikaciji odobril dostop do trenutne lokacije. Na mapi so prikazana predhodna najdišča v obliki glive, ki so si jih gobarji shranili v osebno kolekcijo. Ob kliku na najdišče se s pomočjo geolokacije prikaže možnost določanja poti do odkrite glive v primeru, da cesta vodi do cilja. *Maps SDK* je storitev podjetja *Google*, ki zahteva vnos osebnega ključa za dostop do te storitve, saj beleži vsoto vseh klicev končnih točk. Čeprav je na začetku ta storitev brezplačna, se ob prekoračitvi določene kvote zahteva plačilo. Pridobljen dostopni ključ vnesemo v datoteko *AndroidManifest.xml*, kjer se shranjujejo ključne informacije o aplikaciji.

Slika 18: Interaktivna mapa

```
<meta-data android:name="com.google.android.geo.API_KEY"
        android:value=                                    />
```
Slika 19: Prikaz metapodatkov v manifestu aplikacije

5 Družbena odgovornost

Strojno učenje je prihodnost avtomatizacije, v katero se pomikamo. Nedolgo nazaj, so naši predniki morali prepisovati knjige, da bi širili znanje, dokler ni *J. Gutenberg* izumil tiskarskega stroja in spremenil način in hitrost širjenja informacij. Od leta 1950 naprej se je s prihodom računalnikov pričela doba digitalizacije. Tiskane knjige so se postopoma začele spreminjati v binarna števila, dostopna celotnemu svetu. Zaradi tehnoloških napredkov se kvaliteta naših življenj povečuje. Eden izmed pomembnih tehnoloških napredkov je tudi umetna inteligenca.

Področje botanike je precej obširno in zaradi tega je obdelava podatkov s strojnim učenjem precej zahtevna. Projekt, ki smo si ga zadali, temelji na izboljšanju teoretičnega znanja o glivah in preprečevanju nepravilnega ravnanja z njimi v gozdovih. Glede na to, da ima večina nabiralcev gliv v lasti pametni telefon, bi ga lahko s pomočjo naše aplikacije, koristno uporabila za izboljšanje poznavanja gliv in ravnanja z njimi. Nekoč so si gobarji in predvsem začetniki te dejavnosti pri prepoznavanju nabranih gliv pomagali z ducatom knjig. Danes nam ni več potrebno živeti v preteklosti, če si lahko pomagamo s tehnologijo, ki nam je na voljo. Napredek in uvajanje na področju tehnologije je ključnega pomena za razvoj družbe, kot jo poznamo.

6 Zaključek

Porast pametnih telefonov, cenejših kamer in vsakodnevne izboljšave na področju prepoznavanja slik, so odprla novo obdobje za umetno inteligenco. Podjetja v različnih sektorjih, kot so: avtomobilska industrija, igre na srečo in elektronsko poslovanje, uporabljajo to tehnologijo. Uspeh tehnologije za klasifikacijo slik in uporabnost strojnega učenja kot postopka avtomatizacije je odvisen od natančnosti le-tega. Umetna inteligenca lahko pomaga v primeru naravnih nesreč, ki se vsakodnevno dogajajo po svetu, ali pri iskanju znakov življenja med ostanki potresa bodisi za napovedi poplav, cunamija ali izbruha vulkana.

V našem primeru, umetna inteligenca pripomore k prepoznavanju mnogovrstnih družin gliv. Temeljni cilj je izobraziti ljudi o pravilnem ravnanju z glivami, saj lahko nepravilen pristop pri nabiranju gliv pomembno vpliva na naravo v tem območju. Pomembno se je zavedati, da so glive tudi hrana nekaterih živali in s posegi v naravo lahko porušimo živalski prehranjevalni splet.

Kljub trenutni situaciji, smo z razvito aplikacijo zadovoljni. V prihodnosti bi bilo potrebno izboljšati prepoznavnost in klasificiranje gliv. Slednje bi dosegli s treniranjem nevronske mreže na večjem številu vzorcev oz. slik gliv. Nedvomno bomo posvetili svoj čas tudi za oglaševanje aplikacije na različnih socialnih omrežjih z namenom večje prepoznavnosti in uporabnosti aplikacije.

7 Viri in literatura

Bhavsar, P. (2019). *Topbots*. An Ultimate Guide To Transfer Learning In NLP. Dostopno na: https://www.topbots.com/transfer-learning-in-nlp/ [Ogled 19.2.2021]

Lücking, R. (2020). *SpringerLink*. Unambiguous identification of fungi: where do we stand and how accurate and precise is fungal DNA barcoding? Dostopno na: https://link.springer.com/article/10.1186/s43008-020-00033-z [Ogled 19.2.2021]

Maksymenko, S. (2020). *Mobidev*. AI-based visual inspection for defect detection. Dostopno na: https://mobidev.biz/blog/ai-visual-inspection-deep-learning-computer-vision-defect-detection [Ogled 16.2.2021]

Ruder, S. (2017). *Ruder*. Transfer Learning – Machine Learning's Next Frontier. Dostopno na: https://ruder.io/transfer-learning/ [Ogled 27.2.2021]

Shigupta, D. (2020). *Geeksforgeeks*. Computational Graphs in Deep Learning. Dostopno na: https://www.geeksforgeeks.org/computational-graphs-in-deep-learning/ [Ogled 25.2.2021]