ŠOLSKI CENTER VELENJE

ELEKTRO IN RAČUNALNIŠKA ŠOLA

Trg mladosti 3, 3320 Velenje

MLADI RAZISKOVALCI ZA RAZVOJ ŠALEŠKE DOLINE

RAZISKOVALNA NALOGA

RAČUNALNIŠKI VID IN PREVERJANJE COVID TESTOV

Tematsko področje: TEHNIŠKE VEDE

Avtor:

Tobija Žuntar, 3. letnik

Mentor:

Gregor Hrastnik

Velenje, 2022

Raziskovalna naloga je bila opravljena na Šolskem centru Velenje, Elektro in računalniški šoli, leta 2022.

Mentor: Gregor Hrastnik

Datum predstavitve: april 2022

KLJUČNA DOKUMENTACIJSKA INFORMACIJA

ŠD ŠC Velenje, šolsko leto 2021/2022

KG Računalniški vid in COVID-19 testi

AV ŽUNTAR, Tobija

SA HRASTNIK, Gregor

KZ 3320 Velenje, SLO, Trg mladosti 3

ZA ŠC Velenje, Elektro in računalniška šola, 2022

LI 2022

IN **RAČUNALNIŠKI VID IN PREVERJANJE COVID TESTOV**

TD Raziskovalna naloga

OP 24 str., 9 sl., 5 vir.

IJ SL

JI sl / en

AI Pri samotestiranju za COVID-19 lahko zelo enostavno pride do poneverbe rezultatov testa. Te sporoči testirana oseba sama, pri čemer ni nobenega mehanizma, ki bi varoval pred sporočanjem lažnih rezultatov. V namen preprečitve tega smo se odločili raziskati različne možnosti, kako bi z uporabo strojnega učenja lahko preverili, da se sporočeni rezultat testa in slika testne ploščice ujemata z dejanskim stanjem le-te. Pri tem smo preizkusili različne načine, s katerimi bi izboljšali točnost zaznavanja rezultata testa in preprečili poneverbe. Ustvarili smo tudi preprosto spletno rešitev, ki z uporabo modela strojnega učenja s slike testne ploščice odčita dejanski rezultat testa.

KEYWORD DOCUMENTATION

ND ŠC Velenje, Elektro in računalniška šola, 2022

CX Computer Vision and Rapid COVID-19 Antigen Tests

AU ŽUNTAR, Tobija

AA HRASTNIK, Gregor

PP 3320 Velenje, SLO, Trg mladosti 3

PB ŠC Velenje, Elektro in računalniška šola, 2022

PY 2022

TI **COMPUTER VISION AND RAPID COVID ANTIGEN TEST VERIFICATION**

DT Research work

NO 24 p., 9 fig., 5 ref.

LA SL

AL sl / en

AB Self-testing for COVID-19 is a straightforward procedure, yet it is very easy for the test results to be intentionally misinterpreted. There is currently no software solution put in place to guard against this. Hence, we have decided to research different methods of matching a picture of the test pad with the user-reported test results. Our research focused specifically on the use of Machine Learning technology in this process. Over the course of the project, we have evaluated a variety of techniques for improving the accuracy of our Machine Learning model. We have also created a proof-of-concept web service that uses the aforementioned model to interpret the test results based on an input picture.

KAZALO VSEBINE

KAZALO SLIK

1. UVOD

Naš cilj je ustvariti spletno aplikacijo, ki s pomočjo računalniškega vida prepozna hitre COVID-19 teste in ugotoviti, če takšna rešitev pripomore k večjemu zaupanju v uporabniško vnesene rezultate testov.

V nekaterih Evropskih državah imajo namreč uvedeno obvezno samotestiranje, vendar rezultate vnašajo uporabniki sami, preverjanja vnesenega rezultata pa zaenkrat ni. Naša spletna aplikacija bi torej od uporabnika zahtevala, da naloži sliko izvedenega testa, ter pogledala ali se rezultat na testu ujema s tem, kar je vnesel uporabnik.

Pri tem je potrebno imeti v mislih tudi to, da je sistem, ki je trenutno v uporabi, mogoče zelo preprosto ogoljufati s večkratnim slikanjem iste testne ploščice. V izogib temu bi bile na testnih ploščicah nameščene kode QR, ki jih naša rešitev skenira in se tako prepriča, da gre v resnici za novo testno ploščico.

1.1 HIPOTEZE

V okviru raziskovalne naloge smo si zastavili naslednji dve hipotezi:

1. Z računalniškim vidom je možno zanesljivo odčitati rezultat testa s testne ploščice za samotestiranje.

2. Pri testnih ploščicah s QR kodami bi te preprečile večkratni vnos rezultata iste ploščice.

2. PREGLED OBJAV

Raziskali smo, kako poteka preverjanje rezultatov testov za samotestiranje v različnih evropskih državah. Pogledali smo tudi, katere rešitve za izdelavo modelov strojnega učenja so nam na voljo in kako bi jih uporabili za izdelavo našega modela.

2.1 SAMOTESTIRANJE V VELIKI BRITANIJI

V Veliki Britaniji samotestiranje za COVID-19 poteka s pomočjo hitrih antigenskih testov, ki imajo na sprednji strani nameščene kode QR in so enolično oštevilčeni. Velika Britanija je trenutno edina evropska država, kjer uporabljajo tak sistem sledenja samotestiranju. Drugod po Evropi testi nimajo kod QR oz. te za izkazovanje veljavnosti rezultatov niso zahtevane.

V Veliki Britaniji testiranci teste izvedejo sami, na domu, in nato rezultate vnesejo na vladno spletno stran [1]. Ta je opremljena s tehnologijo za skeniranje kod QR. Stran tako prepreči večkratni vnos rezultata iste testne ploščice na podlagi njene enolične številke. Trenutno pa stran nima nobenega mehanizma za preverjanje verodostojnosti vnesenega rezultata testa [2].

2.2 GOOGLE TEACHABLE MACHINE

Teachable Machine je spletna platforma, razvita sprva kot eksperiment s strani Googla, ki omogoča preprosto izdelavo in učenje umetne inteligence na lastnem naboru podatkov. Namenjena je vsem, ki se želijo poigrati z umetno inteligenco in strojnim učenjem, vendar nimajo zadostnega znanja za ročno izdelavo in preizkus modelov. Podprte so možnosti izdelave modela za razločevanje slik, zvokov ter poz, pri čemer lahko model naučimo razlikovati med poljubnim številom slik/zvokov/poz.

Vse potrebne nastavitve na podlagi izbrane vrste modela nastavi Teachable Machine sam, pri čemer imamo možnost spremeniti število krogov učenja ter velikost paketov vhodnih podatkov. Model naučimo z enim klikom. Takoj zatem ga lahko preizkusimo bodisi z uporabo vgrajene spletne kamere oz. mikrofona v računalniku za neposreden vnos novih podatkov ali pa z nalaganjem datotek.

Kljub svoji začetnikom prijazni zasnovi pa je Teachable Machine uporaben tudi za razvoj resnih projektov, saj omogoča izvoz modela v obliki, združljivi z ogrodjem Tensorflow. To je odprtokodna platforma za strojno učenje, ki se uporablja za ročno učenje modelov in njihovo uporabo v aplikacijah. Njegovo vlogo v našem primeru samodejno opravi Teachable Machine. Ob izvoženem modelu so priložena tudi navodila za njegovo uporabo v programskih jezikih JavaScript, Java in Python, ter različnih izdajah knjižnic Tensorflow (npr. Tensorflow Lite). Podana koda nam pomaga pri vgradnji izdelanega modela v lastne spletne strani in mobilne aplikacije [3].

Prva javno dostopna različica storitve Teachable Machine je bila izdana že leta 2017, vendar takrat shranjevanje in izvoz naučenega modela nista bila mogoča, učenje pa je potekalo le na podlagi slik, pri čemer je bilo mogoče razlikovati med samo tremi različnimi slikami.

2.3 APPLE CREATE ML

Create ML je aplikacija, ki je del Applovega integriranega razvojnega okolja Xcode. Namenjena je izdelavi modelov strojnega učenja za uporabo v aplikacijah za Applove operacijske sisteme (macOS, iOS, iPadOS ...). Aplikacija uporablja klice API, ki jih lahko uporabimo za učenje modelov neposredno na napravah v lastnih aplikacijah.

Omogoča izdelavo modelov za zaznavanje in razlikovanje slik, predmetov na slikah, besedila na slikah, ključnih besed v besedilu, zvokov, dejavnosti v videoposnetkih ter iskanje vzorcev v naborih podatkov [4].

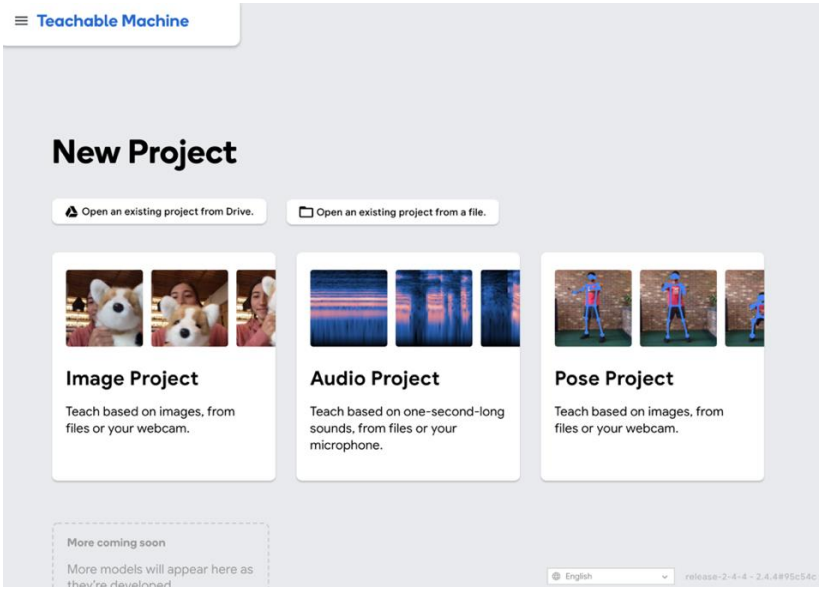
Podatke za učenje modelov vstavimo s pomočjo definicijskih datotek JSON. Te lahko ustvarimo v lastnih aplikacijah in tako ustvarimo modele med njihovim delovanjem, kar je tudi glavni namen tega pripomočka [5].

3. POTEK IZVAJANJA

Za pripravo modela bomo uporabljali storitev Google Teachable Machine, saj je ta za naš način uporabe izmed zgoraj opisanih dveh najbolj praktična. Applova rešitev je namreč namenjena sprotni izdelavi in uporabi modelov v mobilnih aplikacijah, v našem primeru pa bomo model izdelali vnaprej.

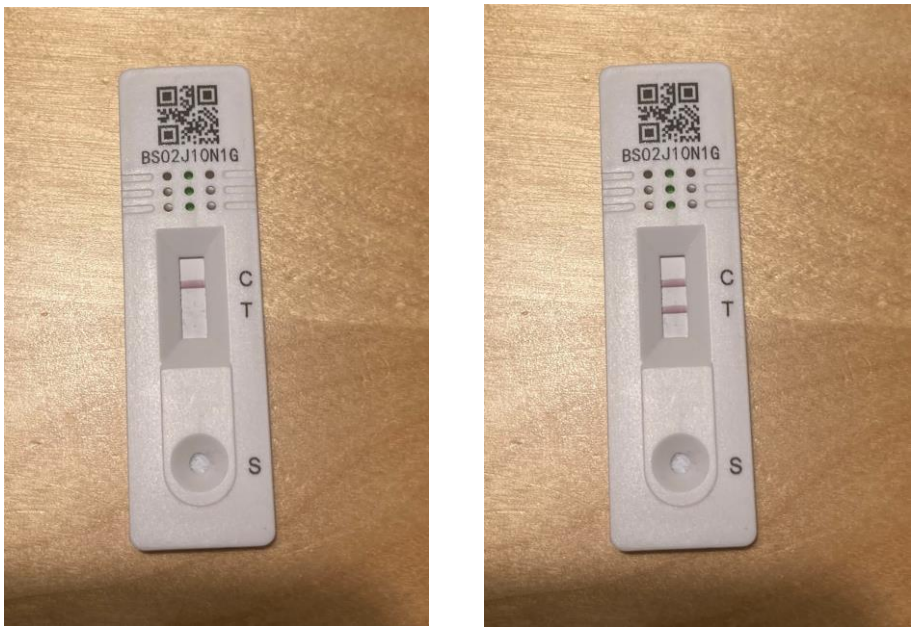
3.1 IZDELAVA NAUČENEGA MODELA

Za uporabo modela na spletni strani smo morali tega najprej izdelati, kar smo naredili s pomočjo Googlove spletne storitve Teachable Machine. Na njeni spletni strani smo ustvarili nov projekt z uporabo načina učenja za slike.

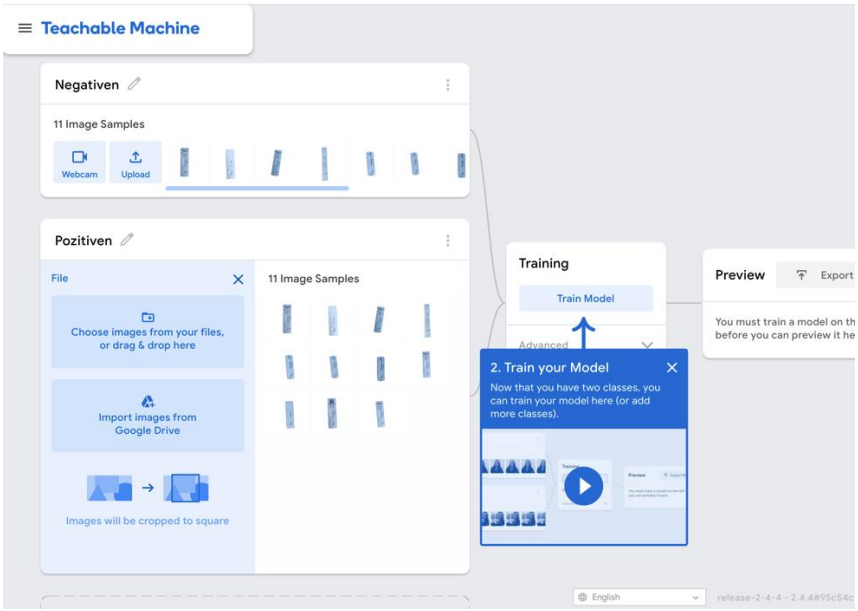


Slika 1: Pozdravna stran storitve Google Teachable Machine.

Nato smo naložili naše slike. Te smo razvrstili v dve skupini — tiste s pozitivnim izvidom testa ter tiste z negativnim izvidom testa:



Slika 2: Primer vhodnik slik negativnega in pozitivnega testa

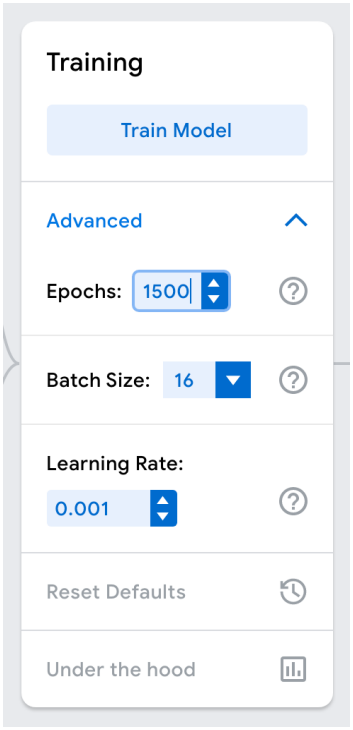


Slika 3: Vstavljanje nabora podatkov za učenje modela.

Nato smo nastavili možnosti učenja, pri čemer smo od privzetega morali spremeniti le število krogov (Epochs-ov). Ti določijo, kolikokrat se algoritem strojnega učenja prebije skozi vzorčna nabora podatkov. S povečanjem tega števila povečujemo intenzivnost učenja ter posledično dosežemo izboljšanje natančnosti našega naučenega algoritma.

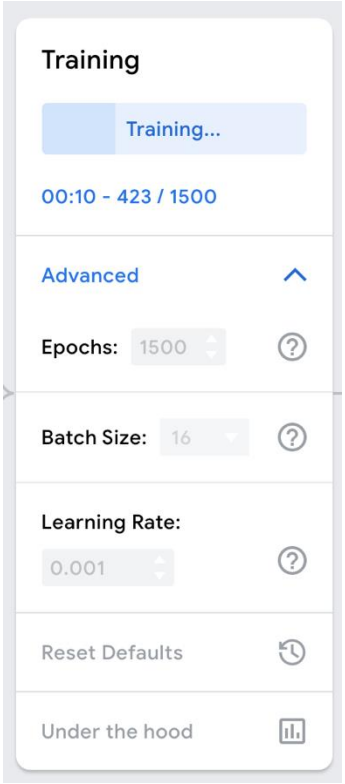
Naslednja nastavitev, ki nam je na voljo, je »Batch Size« oz. velikost sklopa slik, nad katerimi se vrši vsak krog učenja. Število slik v vhodnih paketih se razdeli na pakete te velikosti; ko se algoritem strojnega učenja prebije skozi vse te, se to šteje kot en dokončan krog.

Zadnja nastavitev, ki nam je na voljo, je »Learning Rate« oz. stopnja učenja. Zanjo sicer ni navedenih podrobnejših podatkov o njenem vplivu na učenje, razen opombe, da lahko njeno spreminjanje nepredvidljivo vpliva na rezultate učenja.



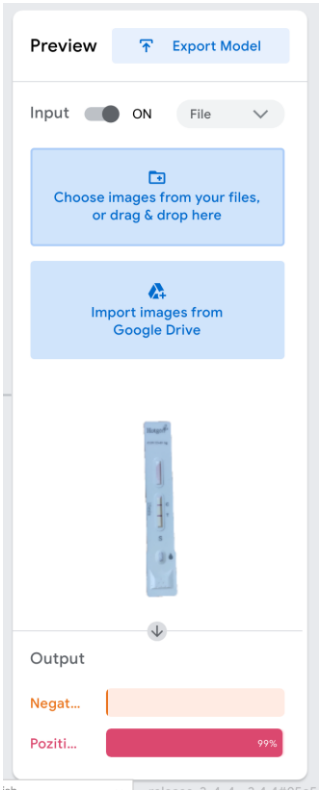
Slika 4: Spreminjanje število krogov strojnega učenja in nastavitev ostalih parametrov.

Za tem smo sprožili učenje s pritiskom na gumb »Train Model« ter počakali, da se to dokonča. Med potekom učenja mora biti zavihek s stranjo vedno aktiven, saj le to omogoča popoln dostop do strojnih virov, potrebnih za samo učenje. Hitrost učenja je odvisna od števila krogov in velikosti ter števila vhodnih naborov podatkov.



Slika 5: Potek učenja modela.

Po zaključenem učenju lahko model preizkusimo kar na strani, in sicer z nalaganjem slike oz. uporabo spletne kamere računalnika.



Slika 6: Testiranje naučenega modela z naloženo sliko pozitivnega testa; model le-tega pravilno zazna kot pozitivnega.

S klikom na gumb »Export« (Izvoz) pa so nam na voljo tudi navodila za prenos in vgradnjo modela v lastne spletne strani in aplikacije, kar smo kasneje s pridom izkoristili za izgradnjo naše spletne aplikacije. Model je združljiv s ogrodji Tensorflow (Tensorflow, Tensorflow.js in Tensorflow Lite), priložena pa so navodila za njegovo uporabo v programskih jezikih Python, JavaScript ter Java (za Android).

3.2 IZBOLJŠANJE NATANČNOSTI ZAZNAVANJA

3.2.1 Slike z ozadjem

Pri slikah z ozadjem model sicer razlikuje slike testov, vendar z natančnostjo med 40 in 60 odstotki. Pojavlja se tudi odstopanje zaradi različnih ozadij slik. Slike, na katerih je bil model naučen, imajo namreč različna ozadja, kar vpliva na natančnost zaznave. S sliko pozitivnega testa na rjavem ozadju lahko zavedemo model, ki je bil naučen na velikem številu slik negativnih testov na podobnem rjavem ozadju. Primer takšnih slik sta testa na sliki 2.

Sicer pa je ta način še najbližji osnovnemu namenu platforme Teachable Machine, torej izdelava modela, ki je sposoben razlikovati med različnimi nepredelanimi slikami.

3.2.2 Slike brez ozadja

Zaradi prejšnjih problemov z raznovrstnimi ozadji testov na sliki smo se odločili preizkusiti učenje na podlagi slik z nevtralnim (belim) ozadjem. Vsem slikam na našem vhodnem naboru podatkov smo odstranili ozadje ter iste slike uporabili kot negativne in pozitivne teste (dodali smo manjkajočo črto na slike, da so bili negativni testi videti kot pozitivni). Pričakovali smo, da bo model zaznal le razliko v črti in ga zato vse ostalo ne bi motilo.

Pri tem smo se krepko zmotili, saj so bili rezultati porazni. Model je s preko 90 odstotno prepričanostjo izbral napačen rezultat. Pri tem smo opazili tudi, da platforma Teachable Machine deluje veliko bolje na nepredelanih slikah, saj se ne osredotoča le na posamezen element slike, vendar nanjo kot celoto.

Slika 7: Slika testa z odstranjenim ozadjem.

3.2.3 Preizkus uporabe filtra Threshold

Ker se na vhodnih slikah črtici, ki označujeta rezultat testa ne vidita dobro, smo se odločili poskusiti povečati kontrast slik, da bi se ti dve bolje razločili od podlage. To smo storili s spreminjanjem kontrastnega praga slike (angl. Image Threshold), saj je ta način dovolj enostaven, da bi ga po potrebi lahko implementirali v naši rešitvi.

Z znatnim povečanjem kontrastnega praga slike dosežemo, da so na njej vidne le glavne črte in robovi predmetov na sliki (glej sliko 8). Pri tem smo predvidevali, da bo to modelu olajšalo zaznavanje, saj nanj ne bi vplivala barva in osvetlitev slike testne ploščice.

Na vseh slikah, ki smo jih ustvarili pri prejšnjem poskusu, smo s pomočjo urejevalnika slik spremenili kontrastni prag in ustvarili nov model ter ga naučili v 5000 krogih. Na njem smo preizkusili tako nespremenjene, kot tudi slike z povečanim kontrastnim pragom. Kljub temu se v nobenem izmed teh dveh primerov to ni obneslo. Model je večino časa s preko 90 odstotno prepričanostjo izbral napačen rezultat testa.

Slika 8: Primer slike z znatno povečanim kontrastnim pragom.

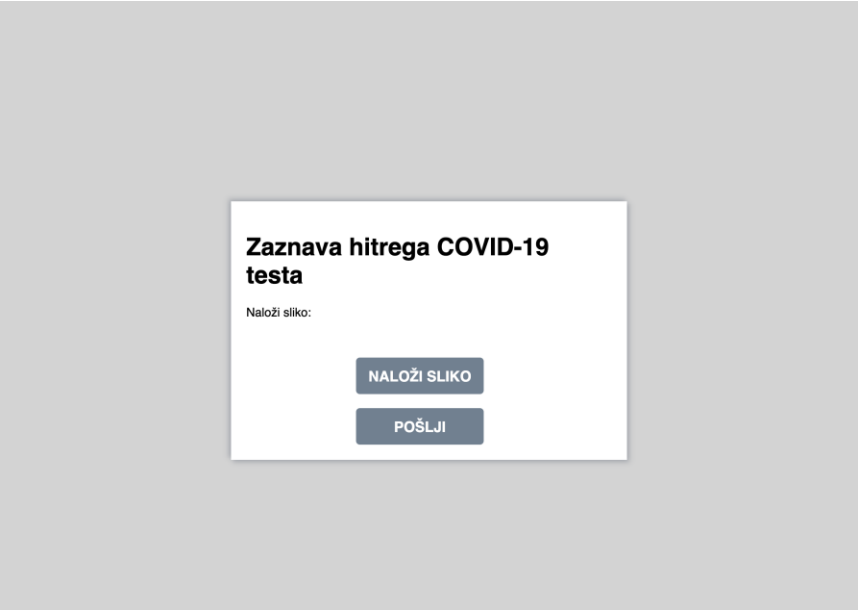
3.3 IZDELAVA SPLETNE APLIKACIJE ZA DEMONSTRIRANJE MODELA

Za namen demonstracije našega modela strojnega učenja smo se odločili izdelati preprosto spletno aplikacijo. V njej uporabniki s telefona naložijo skenirano sliko testne ploščice, s katere aplikacija s pomočjo našega modela poskusi odčitati rezultat testa. Če ima testna ploščica na sebi kodo QR, se tudi ta izpiše v aplikaciji.

Samo aplikacijo smo izdelali s pomočjo ogrodja Flask in programskega jezika Python; del, ki s slik odčitava rezultate testov na podlagi našega modela, pa uporablja programsko knjižnico Keras.

Aplikacija si ob zaznavi kode QR s testne ploščice to zabeleži in ob kasnejši zaznavi iste kode uporabnika o tem opozori. Kodo bi lahko spremenili tudi tako, da aplikacija zavrne vnesen rezultat, če je bila ista koda QR že zaznana.

Na enak način bi lahko države izdelale lastne aplikacije oz. spletne strani za preverjanje, v katere bi uporabniki vnesli rezultate testov za samotestiranje ter sliko, s pomočjo katere bi lete preverile verodostojnost vnesenih rezultatov. Prav tako je zaradi večstranskosti modelov možno te uporabiti tudi v mobilnih aplikacijah za sistema Android in iOS.



Slika 9: Spletna aplikacija za zaznavo rezultatov testov.

4. REZULTATI IN RAZPRAVA

Prvo hipotezo smo delno potrdili, saj smo ugotovili, da je rezultate testov možno odčitati z uporabo računalniškega vida, vendar nam ni uspelo zagotoviti popolne zanesljivosti. To smo poskušali izboljšati na več načinov.

Najprej smo slikam poskusili odstraniti ozadje, vendar je ta način v resnici še poslabšal točnost zaznavanja. Pri tem smo ugotovili, da Teachable Machine slike gleda kot celote in takšen način uporabe ne pride v poštev.

Nato smo poskusili povečati kontrastni prag slik z uporabo filtra. Ta način je točnost zaznavanja malenkost izboljšal, vendar to še vedno ni bilo to, kar smo si želeli.

Na koncu smo se odločili ostati pri prvi metodi, torej z uporabo nespremenjenih slik, vendar smo s povečanjem nabora podatkov za učenje izboljšali točnost zaznavanja dovolj, da je demonstracija modela možna.

Ugotavljamo, da bi bilo točnost zaznavanja najlažje doseči neposredno z računalniškim vidom, torej s programiranjem lastnega programa računalniškega vida, brez uporabe storitve Google Teachable Machine. Izdelava takšnega programa je veliko težja kot uporaba spletne storitve, vendar bi pri tem imeli popoln nadzor nad zaznavanjem. Pri Teachable Machine to ni možno, saj se na podlagi vzorcev za učenje sam odloči, kateri deli slike se mu zdijo bolj pomembni za zaznavanje in kateri manj.

Pri tem bi tak program na sliki poiskal območje dveh črtic testa ali pozval uporabnika, naj ga izbere, pred obdelavo pa bi sliko tudi primerno obdelal – povečal kontrastni prag, zmanjšal resolucijo za lažjo obdelavo ipd.

Drugo hipotezo smo potrdili, saj smo izdelali demonstracijsko aplikacijo, ki si sproti beleži prebrane kode QR testnih ploščic in ob poskusu ponovnega vnosa le-teh o tem opozori uporabnika. Na enak način je možno izdelati aplikacijo, ki bi zavrnila večkratni vnos rezultata in slike iste testne ploščice.

5. ZAKLJUČEK

V okviru te raziskovalne naloge smo poskušali ustvariti najbolj točen model za prepoznavo testnih ploščic testov za virus COVID-19. Pri tem nam je največji izziv predstavljalo izboljšanje natančnosti zaznavanja testov, kar je bila tudi osrednja tema raziskovalne naloge. Ravno v to je bilo vloženega ogromno časa za iskanje možnosti za izboljšave, nešteto preizkuševanj, iskanje optimalnega načina za prilagoditev nabora vzorcev za učenje modela ...

Skozi te izzive smo spoznali pomen ustreznega nabora podatkov v svetu strojnega učenja in umetne inteligence. Cilja nam sicer ni uspelo doseči tako dobro, kakor smo si prvotno zamislili, vendar smo se kljub temu naučili veliko uporabnih veščin ter spoznali tehnologije, povezane s strojnim učenjem.

6. ZAHVALE

Za pomoč pri izdelavi raziskovalne naloge se zahvaljujemo:

- Šolskemu centru Velenje, Elektro in računalniški šoli za uporabo računalnikov,
- Gregorju Hrastniku za mentorstvo in pomoč pri izvedbi raziskovalne naloge.

7. VIRI IN LITERATURA

[1] https://www.gov.uk/government/publications/instructions-for-covid-19-self-test

[2] https://www.gov.uk/report-covid19-result

[3] https://teachablemachine.withgoogle.com/faq#Tools-and-Resources

[4] https://developer.apple.com/machine-learning/create-ml/

[5] https://developer.apple.com/documentation/createml