Android aplikacija za prepoznavanje spomenika

Perić, Juraj

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Sveučilišni studij

Android aplikacija za prepoznavanje spomenika

Završni rad

Juraj Perić

Osijek, 2022.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

Obrazac Z1P - Obrazac za ocjenu završnog rada na preddiplomskom sveučilišnom studiju

Osijek, 15.09.2022.

Odboru za završne i diplomske ispite

Prijedlog ocjene završnog rada na preddiplomskom sveučilišnom studiju

Ime i prezime Pristupnika:	Juraj Perić
Studij, smjer:	Preddiplomski sveučilišni studij Računarstvo
Mat. br. Pristupnika, godina upisa:	R 4413, 22.07.2019.
OIB Pristupnika:	66729311429
Mentor:	Izv. prof. dr. sc. Mirko Köhler
Sumentor:	,
Sumentor iz tvrtke:	
Naslov završnog rada:	Android aplikacija za prepoznavanje spomenika
Znanstvena grana rada:	**Programsko inženjerstvo (zn. polje računarstvo)**
Zadatak završnog rad:	TEMA ZAUZETA ZA STUDENTA Juraj Perić. U završnom radu potrebno je osmisliti i izraditi mobilnu aplikaciju za prepoznavanje i informiranje o spomenicima. Potrebno je napraviti administratorski dio aplikacije za unos informacija o spomenicima i njihovo prepoznavanje. Potrebno je izraditi bazu znanja, koja sadrži fotografije spomenika i njihov opis. Također krajnjem korisniku treba omogućiti da za uslikanu
Prijedlog ocjene završnog rada:	Izvrstan (5)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 3 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 2 bod/boda Jasnoća pismenog izražavanja: 2 bod/boda Razina samostalnosti: 3 razina
Datum prijedloga ocjene od strane mentora:	15.09.2022.
Datum potvrde ocjene od strane Odbora:	21.09.2022.

Potvrda mentora o predaji konačne verzije rada:	Mentor elektronički potpisao predaju konačne verzije.
	Datum:

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA
I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

IZJAVA O ORIGINALNOSTI RADA

Osijek, 21.09.2022.

Ime i prezime studenta:	Juraj Perić
Studij:	Preddiplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	R 4413, 22.07.2019.
Turnitin podudaranje [%]:	10

Ovom izjavom izjavljujem da je rad pod nazivom**: Android aplikacija za prepoznavanje spomenika**

izrađen pod vodstvom mentora Izv. prof. dr. sc. Mirko Köhler

i sumentora ,

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

IZJAVA

o odobrenju za pohranu i objavu ocjenskog rada

kojom ja __Juraj Perić_____, OIB: __66729311429_____, student/ica Fakulteta elektrotehnike, računarstva i informacijskih tehnologija Osijek na studiju _Preddiplomski sveučilišni studij Računarstvo_ , kao autor/ica ocjenskog rada pod naslovom: _Android aplikacija za prepoznavanje spomenika**,**
dajem odobrenje da se, bez naknade, trajno pohrani moj ocjenski rad u javno dostupnom digitalnom repozitoriju ustanove Fakulteta elektrotehnike, računarstva i informacijskih tehnologija Osijek i Sveučilišta te u javnoj internetskoj bazi radova Nacionalne i sveučilišne knjižnice u Zagrebu, sukladno obvezi iz odredbe članka 83. stavka 11. *Zakona o znanstvenoj djelatnosti i visokom obrazovanju* (NN 123/03, 198/03, 105/04, 174/04, 02/07, 46/07, 45/09, 63/11, 94/13, 139/13, 101/14, 60/15).
Potvrđujem da je za pohranu dostavljena završna verzija obranjenog i dovršenog ocjenskog rada. Ovom izjavom, kao autor/ica ocjenskog rada dajem odobrenje i da se moj ocjenski rad, bez naknade, trajno javno objavi i besplatno učini dostupnim:

a) široj javnosti

b) studentima/icama i djelatnicima/ama ustanove

c) široj javnosti, ali nakon proteka 6 / 12 / 24 mjeseci (zaokružite odgovarajući broj mjeseci).

**U slučaju potrebe dodatnog ograničavanja pristupa Vašem ocjenskom radu, podnosi se obrazloženi zahtjev nadležnom tijelu Ustanove.*

__Osijek, 21.09.2022._____ _____

(mjesto i datum) (vlastoručni potpis studenta/ice)

SADRŽAJ

1. UVOD

Kulturna svijest te kreativno i umjetničko izražavanje jedne su od ključnih kompetencija koncepta cjeloživotnog učenja i funkcioniranja u društvu. Kultura i umjetnost od izrazitog su značaja za kvalitetu života, dok manjak ili nedostupnost kulturno-umjetničkih sadržaja utječe na smanjenje socijalnih interakcija osoba, slabi osjećaj pripadnosti te negativno utječe na psihofizičko zdravlje [1]. Osim vizualnog doživljaja umjetnosti i kulturnog dobra važno je i razumjeti značenje i vrijednost istog te smjestiti ga u povijesni okvir. Često su te informacije na licu mjesta teško dostupne i nedovoljno sažete za trenutak obilaska ili slučajnog prolaza uz neku od atrakcija koje nas zanimaju. Kako bismo ljudima približili informacije o kulturnim znamenitostima i omogućili im jednostavniji pristup istim tim informacijama potrebno je ponuditi interaktivni i brzi pristup tim sadržajima. Mobilni telefoni su danas općenito najbrži i najjednostavniji pristup informacija ljudima, a ujedno imaju i kameru koja omogućuje fotografiranje u bilo kojem trenutku. Stoga se aplikacija za mobilni telefon nameće kao najbolje rješenje za ovaj problem.

Mobilna aplikacija razvijena u ovom radu omogućava korisnicima brži, jednostavniji i interaktivniji pristup informacija o spomenicima na temelju slikane fotografije. Aplikacija bi trebala pružati informacije o spomenicima na području grada Osijeka, no za potrebe ovog rada obradit će se osam kipova na Kružnom pilu Svetog Trojstva u osječkoj Tvrđi zajedno s cijelim kužnim pilom. Aplikacija bi trebala primiti fotografiju(slikanu putem aplikacije ili iz galerije mobilnog telefona) te ju pružiti neuronskoj mreži koja obrađuje fotografiju i daje povratnu informaciju o kojem se spomeniku radi.

U drugom poglavlju rada opisana su slična programska rješenja koja već postoje na tržištu. Treće poglavlje predstavlja pregled tehnologija korištenih pri izradi aplikacije i njihovu svrhu u izradi aplikacije. U četvrtom poglavlju opisuje se izrada neuronske mreže potrebne za prepoznavanje spomenika na temelju fotografije i njezin izgled. Peto poglavlje opisuje izradu mobilnog dijela aplikacije i njezinih komponenti. U šestom poglavlju obrađuje se funkcionalnost i kompletan izgled aplikacije, nakon čega slijedi zaključak ovog rada.

1.1. Zadatak završnog rada

U završnom radu potrebno je osmisliti i izraditi mobilnu aplikaciju za prepoznavanje i informiranje o spomenicima. Potrebno je izraditi bazu znanja, koja sadrži fotografije spomenika i njihov opis. Također krajnjem korisniku treba omogućiti da za uslikanu fotografiju spomenika dobije informacije o njemu.

2. PREGLED POSTOJEĆIH RJEŠENJA

Postojanje neuronskih mreža omogućilo je razvitak raznih aplikacija za klasifikaciju slika, teksta ili govora. Na tržištu već postoje slična rješenja koja implementiraju prepoznavanje objekata na temelju fotografije. U nastavku će biti opisana neka od postojećih rješenja te njihova usporedba s rješenjem ovog rada.

2.1. Google Lens

Google Lens je tehnologija za prepoznavanje fotografija razvijena od strane Google-a, dizajnirana je da ponudi relevantne informacije povezane s objektima koje identificira pomoću neuronske mreže [2]. Lens pretražuje druge podatke na internetu te uspoređuje fotografiju s ostalim sadržajima na internetu i rangira ih na temelju sličnosti i relevantnosti. Osim što omogućuje pretraživanje cijelih fotografija također prepoznaje tekst, QR kodove, bar kodove i matematičke zadatke na fotografiji te omogućuje njihovu analizu, kopiranje ili prevođenje na drugi jezik. Izvorno je napravljena kao samostalna aplikacija pa kasnije postaje integrirana u kamere mobilnih telefona koji koriste Android. Google Lens je tehnologija koja ima mogućnost pretraživanja spomenika, no kako joj je spektar pretraživanja veći od samih spomenika nije potpuno precizna za manje poznate spomenike. Također, sučelje Google Lensa nam daje samo ime traženog objekta i mogućnost njegove pretrage na Google-u, stoga nemamo trenutačan i jednostavan pristup informacijama. Slika 2.1. prikazuje korištenje Google Lensa.

Sl. 2.1. Korištenje Google Lens-a[3]

2.2. Smartify

Smartify je mobilna aplikacija koja može prepoznati uslikana umjetnička djela u muzejima i dati informacije o njima [4]. Ideja za aplikaciju je nastala kao alat pristupačnosti slabovidnim korisnicima kojima je teško pročitati tekst koji opisuje umjetninu. Nakon što aplikacija prepozna umjetničko djelo korisnik ima mogućnost uvećavanja djela te čitanja ili preslušavanja njezinog opisa. Smartify je aplikacija koja je najsličnija aplikaciji koja će se obraditi u ovom radu, no ova aplikacija je većinom orijentirana na slike koje se nalaze u galerijama, a ne na kipove i spomenike.

2.3. Picture This

Picture This je mobilna aplikacija koja prepoznaje uslikanu biljku, pruža informacije o njoj kao i savjete o brizi i potencijalne bolesti biljke. Za razliku od aplikacije koja će se obraditi u ovome radu, ova aplikacija pruža klasifikaciju biljaka, no sam dizajn i princip rada aplikacije je vrlo sličan.

2.4. ID–Art

ID-Art je mobilna aplikacija koja pomaže identificirati ukradena kulturna dobra, smanjiti nezakonitu trgovinu i povećati šanse za povrat ukradenih predmeta [5]. Najčešće ju koriste policijski službenici, ali može ju koristiti bilo tko. Aplikacija pristupa INTERPOL-ovoj bazi podataka, provjerava uslikanu fotografiju te daje povratnu informaciju ako je objekt ukraden. Ova aplikacija također klasificira kulturna dobra no pretražuje samo bazu podataka ukradenih djela.

3. PREGLED KORIŠTENIH TEHNOLOGIJA

U ovom poglavlju biti će opisane tehnologije korištene pri izradi mobilne aplikacije. Kako se radi o mobilnoj aplikaciji za Android, korištene tehnologije vezane uz samu mobilnu aplikaciju biti će Android Studio, Kotlin i XML, za neuronsku mrežu koristit će se TensorFlow, Python i Google Colab, a za bazu znanja koja sadrži fotografije i opis spomenika koristit će se Firebase. Za smještanje modela na poslužitelj korišteni su Docker i Google Cloud.

3.1. Kotlin

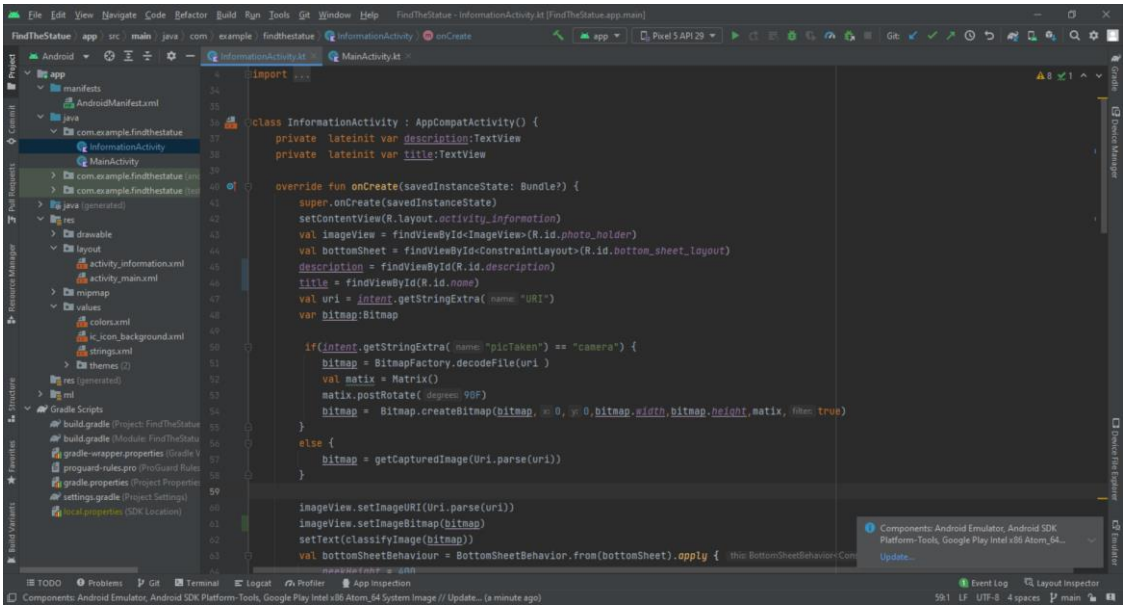
Kotlin je opće namjenski, besplatni, statički tipizirani programski jezik otvorenog koda koji je prvobitno dizajniran za JVM *(Java Virtual Machine*) i Android. Kombinira objektno orijentirane i funkcionalne programske značajke[6]. Nastao je po uzoru na jezike kao što su Java, C#, Python i Scala. Razvijen je od strane Jet Brainsa radi nedostataka potrebnih značajki u postojećim jezicima. Razvoj mobilnih aplikacija postaje primarno dizajniran u Kotlinu nakon Google-ove konferencije 2019 [7].

3.2. XML

XML (*Extensible Markup Language*) je jezik za označavanje sličan HTML-u, ali bez unaprijed definiranih oznaka za korištenje. Umjesto toga, definiraju se svoje vlastite oznake korištene za vlastite potrebe [8]. XML se koristi kao jezik za opis komponenti, odnosno rasporeda zaslona u Android aplikacijama.

3.3. Android Studio

Android Studio je službeno razvojno okruženje za razvoj Android aplikacija [9]. Nudi posljednje Java razvojne pakete(JDK) i programske razvojne pakete(SDK). Besplatan je za preuzimanje i nudi razne mogućnosti poput uređivača teksta i Android Emulatora koji služi za jednostavno i brzo pokretanje aplikacija. Osim mobilnih telefona, emulator nudi i postavke za tablete, Android Tv i Android Wear. Slika 3.1. prikazuje Kotlin kod u Android Studio razvojnom okruženju.



Sl. 3.1. Kotlin kod u Android Studio razvojnom okruženju

3.4. Python

Python je interpretirani, objektno orijentirani, programski jezik visoke razine s dinamičkom semantikom[10]. Jezik je koji ima jednostavnu sintaksu i samim time čitljivost koda je lakša što smanjuje održavanje programa. Podržava module i pakete što potiče modularnost programa i ponovnu upotrebu koda. U zadnje vrijeme najviše se koristi za razvoj umjetne inteligencije i strojnog učenja, ali ima i druge brojne primjene.

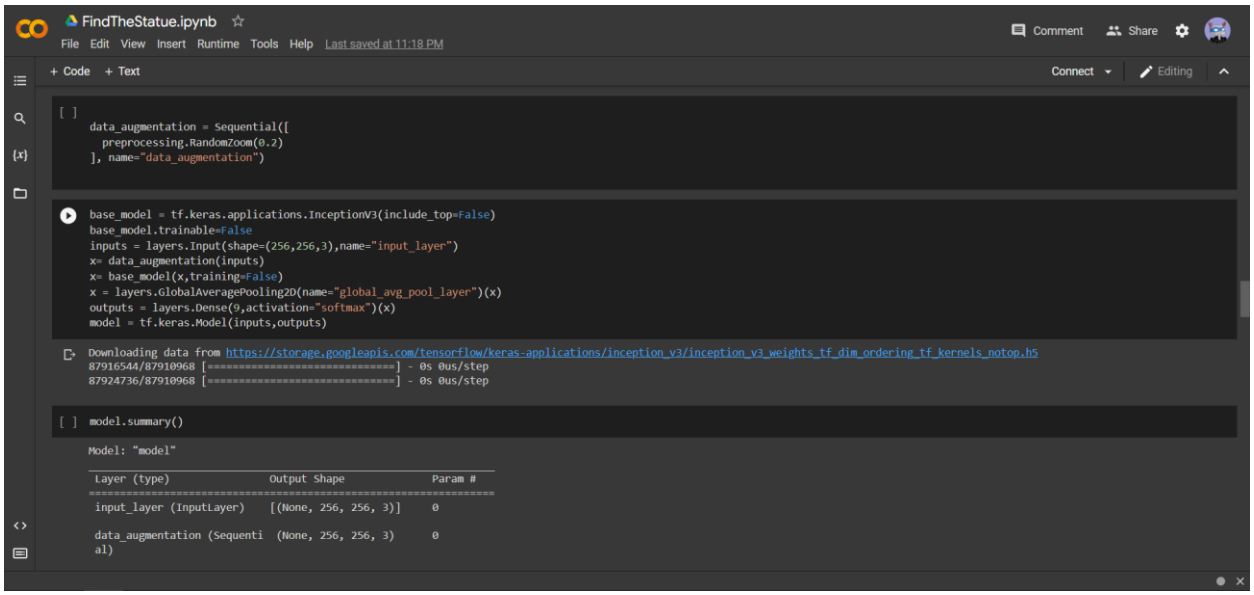
3.5. TensorFlow

TensorFlow je biblioteka otvorenog koda prilagođena Pythonu za numeričko računanje koja čini strojni učenje i razvoj neuronskih mreža bržim i lakšim[11]. Razvijen je od strane Google Brain tima za internu upotrebu u istraživanju i proizvodnji. Uz to što nudi često korištene slojeve neuronske mreže što olakšava modeliranje same mreže, nudi i već gotove istrenirane modele neuronskih mreža.

3.6. Google Colab

Colaboratory, ili skraćeno Colab, razvojno je okruženje koje je proizvod Google Research tima. Colab svakome omogućuje pisanje i izvršavanje proizvoljnog Python koda putem preglednika, a posebno je prikladan za strojno učenje, analizu podataka i obrazovanje [12]. Nudi besplatan pristup

računalnim resursima uključujući i grafičku karticu, što je posebno važno kako bi treniranje neuronske mreže bilo brže. Slika 3.2. prikazuje . TensorFlow kod u Google Colab razvojnom okruženju.



Sl. 3.2. TensorFlow kod u Google Colab razvojnom okruženju

3.7. Firebase

Google Firebase je platforma za razvoj aplikacija kojeg je razvio Google i koji programerima omogućuje razvoj iOS, Android i web aplikacija. Firebase pruža alate za praćenje analitike, izvještavanje i popravljanje padova aplikacija, kreiranje marketinga i eksperimenta s proizvodima [13]. Nudi brojne usluge koje olakšavaju razvoj poslužiteljske strane. U ovome radu koristit će se Firebase Storage za pohranu fotografija spomenika na poslužitelju i Firebase Realtime Database za pohranu opisa kipova i veza na fotografije koje se nalaze na poslužitelju.

3.8. Docker

Docker je skup proizvoda platforme kao usluge koji koriste virtualizaciju na razini operativnog sustava za isporuku programa u paketima koji se nazivaju kontejneri[14]. Kontejneri su međusobno izolirani i sadrže vlastiti programski paket, biblioteke i konfiguracijske datoteke, odnosno Docker slike. Mogu međusobno komunicirati kroz dobro definirane kanale[15]. U ovome radu Docker će se koristiti za isporuku neuronske mreže u kontejneru koji će obrađivati HTTP zahtjeve.

3.9. Google Cloud Platform

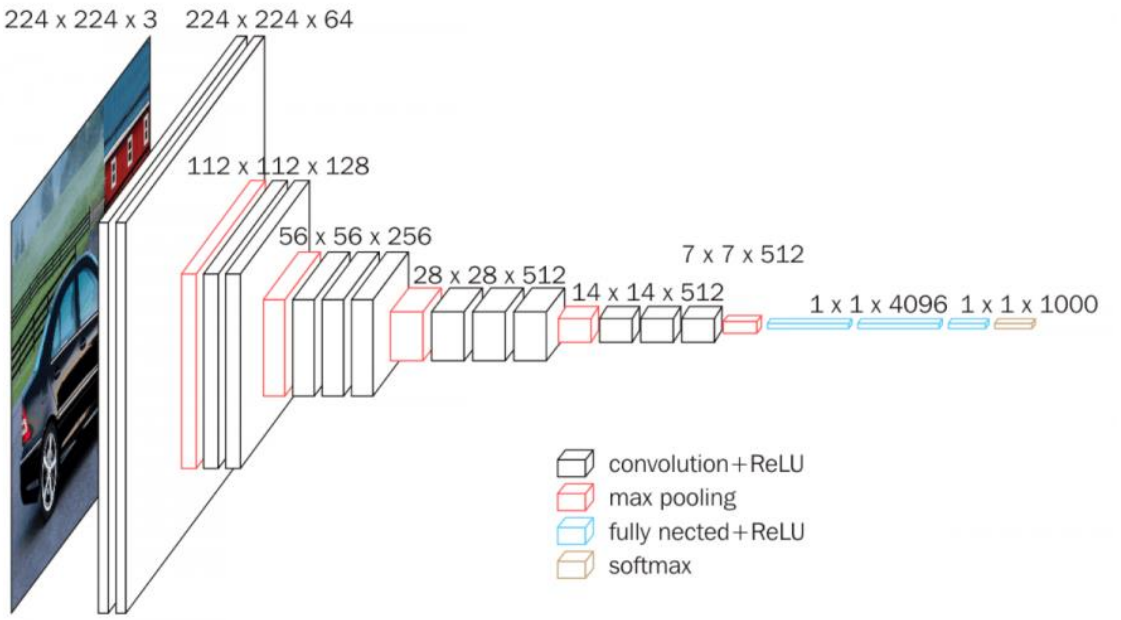
Google Cloud Platform, koju nudi Google, skup je usluga računalstva u oblaku koji radi na istoj infrastrukturi koju Google koristi interno za svoje proizvode za krajnje korisnike, kao što su Google pretraživanje, Gmail, Google disk i YouTube[16]. Pruža infrastrukturu kao uslugu, platformu kao uslugu i računalna okruženja bez poslužitelja. Okruženje bez poslužitelja je značajka koja će se koristiti u ovome radu za pokretanje Docker spremnika bez potrebe za vlastitim poslužiteljem koji će primati HTTP zahtjeve.

4. IZRADA NEURONSKE MREŽE

U ovom poglavlju biti će opisan proces izrade neuronske mreže koja će obavljati klasifikaciju spomenika u ovome radu. Kako se radi o klasifikaciji fotografija najprikladnija mreža za ovaj problem je konvolucijska neuronska mreža, model VGG-16 koji će biti dodatno opisan u nastavku.

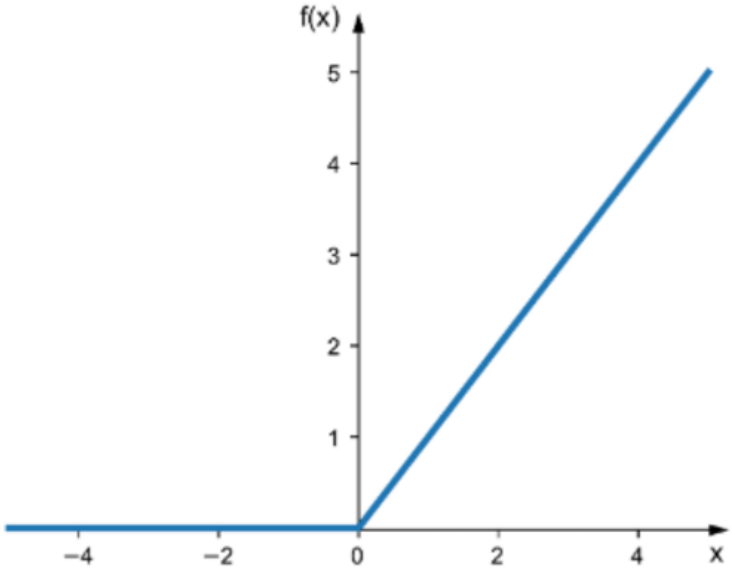
4.1. VGG – 16

VGG-16 odnosno *Visual Geometry Group from Oxford* je model konvolucijske neuronske mreže koji su predložili K. Simonyan i A. Zisserman sa Sveučilišta u Oxfordu u radu "*Very Deep Convolutional Networks for Large-Scale Image Recognition*". Slika 4.1. prikazuje arhitekturu VGG-16 neuronske mreže.



Sl. 4.1. VGG-16 arhitektura[17]

Ulaz u bilo koju konfiguraciju predstavlja slika fiksne veličine 224 x 224 s tri kanala boja - crvena(R), zelena(G), plava(B). Slika prolazi kroz prvi niz od 2 konvolucijska sloja vrlo male receptivne veličine 3 x 3, popraćeni s ReLU aktivacijama. *Rectified linear activation function* odnosno ReLu je aktivacijska funkcija koja će vratiti ulaz jednak samom sebi ako je pozitivan, a u suprotnom vraća nulu. Graf ReLU funckije prikazan je na slici 4.2.



Sl. 4.2. Graf ReLU aktivacijske funkcije[18]

Oba konvolucijska sloja sadrže 64 filtera. Konvolucijski pomak(engl. *stride*) fiksiran je na 1 piksel, a nadopuna(engl. *padding*) je također 1 piksel. Ova konfiguracija čuva prostornu razlučivost, a izlaz je jednak dimenzijama ulazne slike. Izlaz zatim prolazi kroz sloj sažimanja maksimalnom vrijednosti(engl. *Max Pooling*) s veličinom podmatrice od 2 x 2 piksela i korakom od 2 piksela. Ovo prepolavlja veličinu ulaza, stoga je izlaz iz ovog sloja dimenzija 112 x 112. Izlaz zatim prolazi kroz sličan drugi niz, ali sa 128 filtera umjesto 64. Tako su dimenzije izlaza 56 x 56 nakon čega slijedi treći niz od tri konvolucijska soja i sloja sažimanja maksimalne vrijednosti. Broj filtera primijenjenih ovdje je 256, nakon čega je izlaz iz ovog sloja dimenzija 28 x 28. nakon toga slijede dva niza od tri konvolucijska sloja, od kojih svaki sadrži 512 filtara. Izlaz iz oba niza je dimenzija 7 x 7. Nizove konvolucijskih slojeva slijede tri gusta sloja(engl. *Dense*). Posljednji od ova tri sloja je popraćen *Softmax* aktivacijskom funkcijom za kategorijsku klasifikaciju[17].

4.2. Prikupljanje podataka

Kako bi neuronska mreža ispravno generalizirala danu fotografiju potrebno je sastaviti raznolik skup fotografija. Za 9 spomenika koji se prepoznaju u ovome radu prikupljen je skup podataka od 419 fotografija, 40 do 60 fotografija po spomeniku. Fotografije su snimane više različitih dana, iz

različitih kutova u različito doba dana. Primjer snimljenih fotografija za kip Sv. Stjepana prikazan je na slici 4.3.

Sl. 4.3. Fotografije kipa Sv. Stjepana

4.3. Pretprocesiranje podataka

Prije treniranja neuronske mreže potrebno je podijeliti skup podatka na slike koje će se koristiti za trening i one koje će se koristiti za testiranje. Također, potrebno je slike dodatno prilagoditi kako bi treniranje bilo brže i preciznije. Tensorflow nudi biblioteku *keras* koja sadrži funkciju *image_dataset_from_directory* s kojom se može izvesti većina ovih koraka. Ova funkcija će vratiti skup slika iz odabranog direktorija podijeljenih u više podskupova zvanih *batches*. Podskupovi se

koriste kako bi model brže uočio pravilnosti između slika istih kipova pokušavajući imati više manjih prolazaka kroz podatke umjesto prolaska kroz cijeli skup podataka odjednom. Također funkcija će prilagoditi dimenzije slika na one zadane parametrom, te će izvršiti podjelu na slike koje će se koristiti za treniranje i testiranje. Prikaz poziva funkcije *image_dataset_from_directory* možemo vidjeti na slici 4.4.

Linija ***Kod***

```
1:      train_data = tf.keras.utils.image_dataset_from_directory(
2:      all_dir,
3:      shuffle = True,
4:      image_size=IMAGE_SHAPE,
5:      batch_size=BATCH_SIZE,
6:      label_mode="categorical",
7:      crop_to_aspect_ratio=True,
8:      seed=42,
9:      validation_split=0.15,
10:     subset='training',
11:     )
```

Sl. 4.4. Poziv funkcije *image_dataset_from_directory*

U varijable *test_data* i *train_data* bit će spremljene fotografije dimenzija 256 x 256, podijeljenih u skupine po 32 slike. Parametrom *validation_split* određuje se postotak slika koje će se koristititi za testiranje, stoga će *test_data* sadržavati 62 slike, a *train_data* 357. Kako ne bi došlo do preklapanja slika u varijablama *train_data* i *test_data* vrijednost parametra *seed* treba postaviti na jednaku vrijednosti. Posljednji korak pretprocesiranja je normalizacija podataka, odnosno podešavanje vrijednosti svih piksela fotografije da se nalaze u intervalu od 0 do 1. Normalizacija se postiže dodavanjem sloja u model neuronske mreže koji će vršiti dijeljenje svakog piksela sa 255, odnosno s maksimalnim brojem vrijednosti koje piksel može sadržavati.

4.4. Kreiranje modela

Osim VGG-16 modela koji predstavlja osnovni model ove neuronske mreže, kompletnu mrežu potrebno je dodatno proširiti kako bi odgovarala potrebama aplikacije. Tako će prvi sloj kompletnog modela predstavljati ulazni sloj koji ima četiri ulazne dimenzije 32 x 256 x 256 x 3, odnosno veličina podskupa pomnožena s dimenzijama slike i 3 kanala boja , idući u nizu je već spomenuti normalizacijski sloj te nakon njega dolazi VGG-16 model koji će u prvoj fazi treniranja imati zamrznute težine i parametre, odnosno taj dio mreže se neće trenirati. Iz VGG-16 mreže su

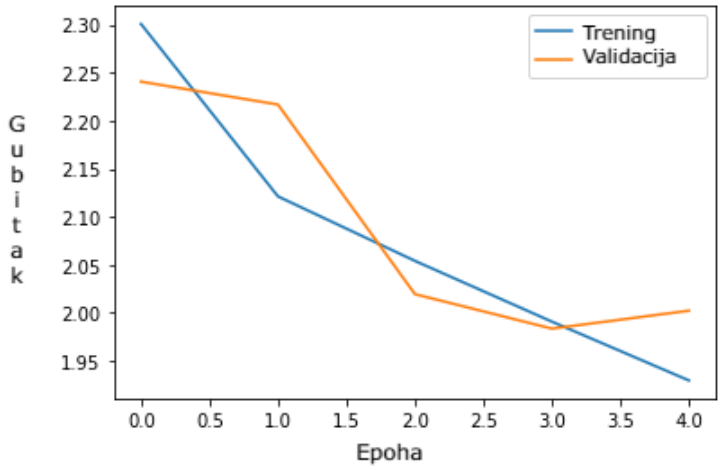
isključeni njezini završni slojevi, gusti slojevi, kako bi mreža mogla biti prilagođena na klasifikaciju kipova. Nakon VGG-16 sloja dolazi *GlobalAveragePooling2D* sloj koji je zadužen za pretvaranje četiri dimenzionalnog tenzora u dvodimenzionalni tenzor računanjem srednjih vrijednosti. *GlobalAveragePooling2D* je nužan kako bi dimenzija izlaza iz prijašnjeg sloja odgovarala ulazu u posljednji sloj modela, gusti sloj. Gusti sloj(engl. *Dense*) sloj s aktivacijskom *softmax* funkcijom je završni sloj neuronske mreže u kojem je njegovih 9 neurona(9 mogućih slika) potpuno povezano sa svim neuronima iz konvolucijskog dijela mreže, zbog čega je poznat i pod nazivom potpuno povezani sloj(engl. *fully connected layer*). Svaki neuron ovog sloja predstavljat će vjerojatnost pojedine klase modela.

4.5. Treniranje modela

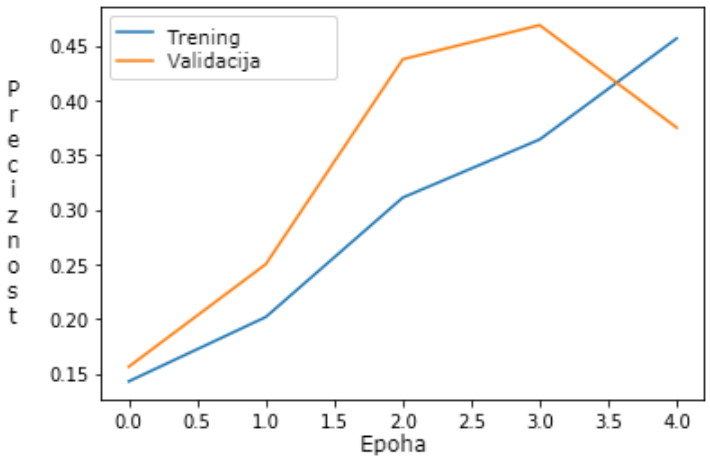
Model neuronske mreže trenira se prijenosnim učenjem(engl. *Transfer learning*). Prijenosno učenje je tip učenja u kojemu model nastoji iskoristiti prethodno naučena znanja i primijeniti ih na nove probleme. Tako će se u prvoj fazi trenirati svi slojevi modela neuronske mreže osim slojeva u osnovnom(VGG-16) modelu, ova faza se zove izdvajanje značajki(engl. *Feature extraction*). Druga faza prijenosnog učenja je fino podešavanje(engl. *Fine tuning)* i predstavlja treniranje svih slojeva modela neuronske mreže uključujući i zadnjih nekoliko slojeva osnovnog modela, ostali slojevi osnovnog modela ostaju zamrznuti.

4.5.1. Izdvajanje značajki

Za početak treniranja modela potrebno je podesiti funkciju gubitka za učenje modela na *categorical_crossentropy* jer se radi o kategoričnoj klasifikaciji te je kao metoda optimiziranja podešena funkcija *Adam*. Nakon toga za trening će se koristiti varijabla *test_data* ranije spomenuta te će se za validacijski skup podataka koristiti 50% *train_data* varijable. Validacijski skup podataka koristi se kao nepristrana provjera točnosti modela za vrijeme njegovog treniranja i podešavanja. Broj epoha za prvu fazu učenja postavljen je na 5. Rezultati izlazne veličine funkcije gubitka u ovisnosti o epohi za vrijeme treniranja prikazani su grafom na slici 4.5. Ovi rezultati bi trebali biti što manji svakom epohom. Preciznost modela u ovisnosti o epohi prikazana je grafom na slici 4.6. Rezultati preciznosti trebali bi rasti nakon svake epohe.



Sl. 4.5. Graf gubitka modela u ovisnosti o epohi nakon izdvajanja značajki

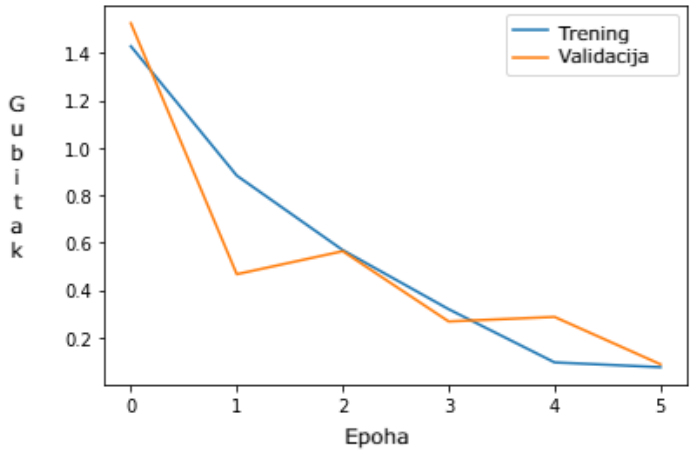


Sl. 4.6. Graf preciznosti modela u ovisnosti o epohi nakon izdvajanja značajki

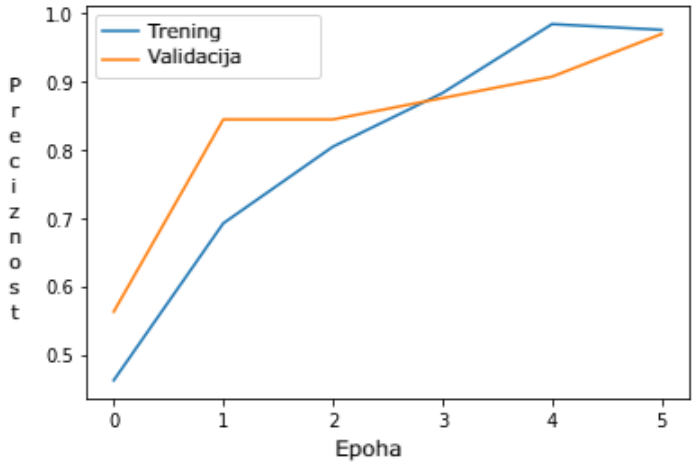
Plave linije predstavljaju rezultate modela nad skupom podataka zaduženom za treniranje, a narančaste prikazuju rezultate modela nad skupom podataka zaduženom za validaciju. Nakon 5 epoha preciznost modela nad cijelim skupom podataka za testiranje je 46.77%. Pristupa se sljedećoj fazi treninga te se vrijednosti trenutnih težina spremaju kako bi se trening mogao produljiti u slučaju da druga faza treniranja bude manje uspješna od prethodne.

4.5.2. Fino podešavanje

U početku druge faze treniranja potrebno je odmrznuti zadnjih nekoliko slojeva osnovnog modela. U ovome radu odmrznuto je zadnjih 15 slojeva VGG-16 modela neuronske mreže te je omogućeno njihovo treniranje. Ostali parametri treninga ostaju isti kao i u prvoj fazi treniranja. Model nastavlja trening za još 5 epoha. Rezultati izlazne veličine funkcije gubitka prikazani su grafom na slici 4.7. te su rezultati preciznosti prikazani grafom na slici 4.8.

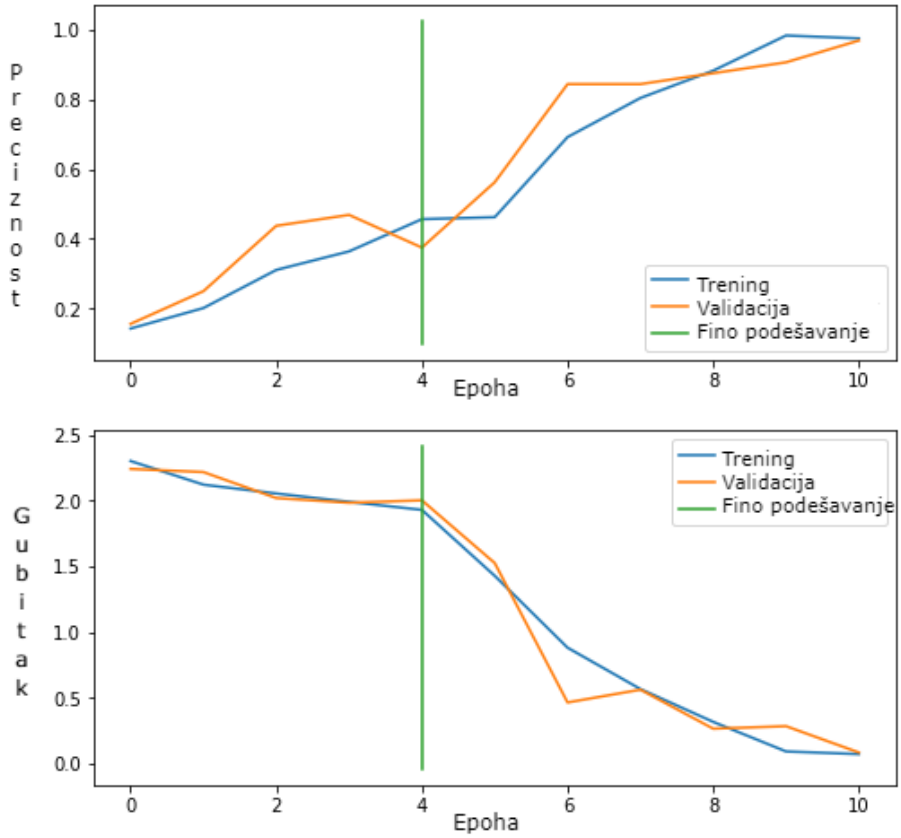


Sl. 4.7. Graf gubitka modela u ovisnosti o epohi nakon finog podešavanja



Sl. 4.8. Graf preciznosti modela u ovisnosti o epohi nakon finog podešavanja

Iz grafova se može zaključiti kako su vrijednosti za preciznost nastavile brže rasti, a gubitak opadati, što se povoljno odražava na preciznost neuronske mreže. Usporedbu izlaznih vrijednosti za dvije faze učenja možemo vidjeti u grafovima na slici 4.9.



Sl. 4.9. Grafovi preciznosti i gubitka prije i poslije finog podešavanja

Zelena linija predstavlja početak druge faze učenja, plava predstavlja rezultate nad skupom podataka korištenom za treniranje, a narančasta nad skupom korištenim za validaciju. Iz grafa se može primijetiti kako je trening ubrzao i postigao veću preciznost u drugoj fazi treniranja. Nakon završenih 10 epoha model je postigao preciznost od 98.39% koja će biti dovoljna za ispravno predviđanje spomenika.

5. POSTAVLJANJE MODELA NEURONSKE MREŽE NA GOOGLE CLOUD I IZRADA BAZE PODATAKA

Kako se aplikacija ne bi morala ažurirati svaki puta kada je dodan novi kip u neuronsku mrežu, model je potrebno postaviti na poslužitelj te iz aplikacije slati zahtjeve kada korisnik želi prepoznati uslikani kip. U ovom poglavlju opisati će se postupak pokretanja modela neuronske mreže u Docker spremniku te njegovo postavljanje na Google Cloud koji će se koristiti kao poslužitelj.

5.1. Postavljanje modela neuronske mreže u Docker sliku

TensorFlow serving je fleksibilan, sistem za serviranje modela strojnog učenja koji je dizajniran za tržišna okruženja. Omogućuje jednostavno mijenjanje serviranog modela i eksperimenata čuvajući jednaku arhitekturu servera i API (engl. *Application Programming Interface*)[19]. Također ga je jednostavno podesiti u Docker sliku i pokrenuti u Docker kontejneru koji će biti zadužen za obradu zahtjeva poslanog na poslužitelj. Nakon pokretanja postavljene Docker slike u spremniku biti će moguće slati zahtjeve s tijelom koje će sadržavati uslikanu fotografiju u obliku tenzora te će kao odgovor biti vraćen vektor s vjerojatnostima mogućih klasa. Zahtjeve će biti moguće slati prema računalu koje će pokretati Docker kontejner, stoga ga je potrebno postaviti na poslužitelj kako bi se zahtjevi mogli slati s različitih lokacija u bilo koje doba dana.

5.2. Pokretanje Docker kontejnera na Google Cloud-u

Za mogućnosti korištenja Google Clouda potrebno je napraviti račun i novi projekt. Prije pokretanja kontejnera na Google Cloudu, potrebno je spremiti Docker sliku s modelom na Google Cloudovu značajku Container Registry. Nakon što je slika spremljena na Google Cloud platformi moguće ju je pokrenuti u kontejneru pomoću Cloud Run mogućnosti. Također, Cloud Run će pokretati više instanci istog kontejnera kako bi se zahtjevi poslani prema poslužitelju brže izvršavali. Svaka instanca kontejnera imati će 1GiB memorije i 2 procesora na raspolaganju, vrijeme u kojemu zahtjev mora biti obrađen je 300 sekundi te je moguće slati 80 zahtjeva prema kontejneru. Najmanji broj pokrenutih instanci kontejnera je 1 kako korisnik ne bi morao dugo čekati pri obradi prvog zahtjeva te je najviše moguće imati 100 instanci kontejnera. Nakon pokretanja kontejnera dobivena je završna točka prema kojoj će biti moguće slati API pozive s mobilne aplikacije. Na ovaj način će poslužitelj vršiti obradu zahtjeva za prepoznavanje objekta umjesto mobilne aplikacije. Ovim pristupom mobilna aplikacija će izgubiti na veličini, u sebi neće sadržavati cijeli model neuronske mreže, te će dobiti na fleksibilnosti. U slučaju potrebe za novim

modelom potrebno je samo spremiti i pokrenuti Docker sliku s novim modelom na Google Cloud platformi umjesto ažuriranja cijele mobilne aplikacije.

5.3. Izrada baze podataka i pohrane slika

Mobilna aplikacija ovoga rada treba pružati korisniku podatke o imenu spomenika, njegovom opisu i slici koja će služiti kao provjera korisniku radi li se stvarno o pravom spomeniku. Slike svih spomenika korištenih za izradu mreže spremljene su u Firebase Storage podijeljenje po klasama. Firebase Storage služi kao pohrana slika iz kojega će se kasnije moći povući slike za izradu novih modela. Također, postoje poveznice na svaku fotografiju kako bi se mogle lakše spremiti u bazu podataka. Baza podataka koja se koristi za spremanje informacija o kipovima je vrlo jednostavna jer postoji samo jedan entitet. Podaci su spremani u Firebase realtime database. Svakom spomeniku je dodijeljen njegov indeks preko kojega će biti pronađen, a svaki indeks sadrži: ime spomenika, opis spomenika i poveznicu na fotografiju spomenika spremljenu u Firebase Storage. Primjer jedne grane baze podataka prikazan je na slici 5.1.

Sl. 5.1. Grana podataka u Firebase Real Time Database

6. RAZVOJ MOBILNE APLIKACIJE

U sljedećem poglavlju biti će opisane funkcionalnosti i izgled mobilne aplikacije izrađene u sklopu ovog rada. Za izradu mobilne aplikacije korišten je Android Studio s programskim jezikom Kotlin, a izgled zaslona je opisan pomoću XML jezika. Aplikacija je podijeljena u tri aktivnosti koje će redom biti opisivane.

6.1. Zaslon kamere

Prilikom pokretanja aplikacije korisnik se nalazi na zaslonu koji sadrži kameru. Prikaz zaslona s kamerom prikazan je na slici 6.1.

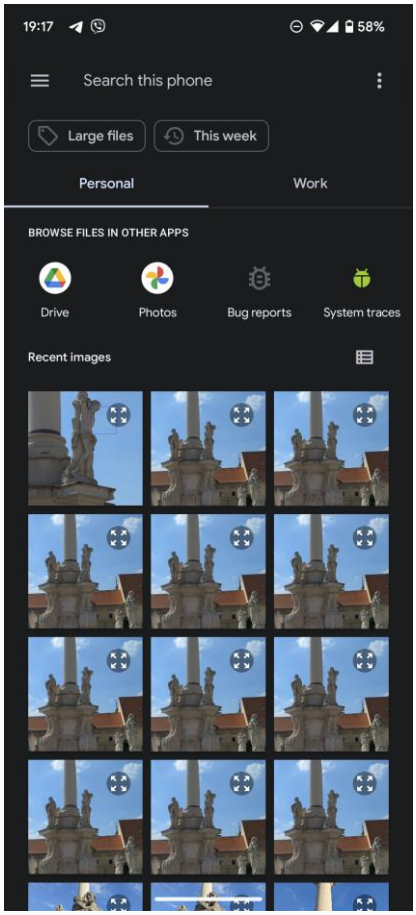
Sl. 6.1. Zaslon kamere

Upotreba kamere na zaslonu omogućena je korištenjem CameraX tehnologije. CameraX je Jetpack biblioteka koja omogućuje jednostavno dodavanje kamere u i njenih dodataka u android aplikaciju[20]. Osim same kamere na zaslonu nalaze se fokus, mogućnost uvećanja kamere i gumbi različitih funkcionalnosti. Fokus je dodan pomoću CamereX, omogućuje fokusiranje

kamere na mjestu gdje korisnik dodirne ekran, nakon korisnikovog dodira oko fokusiranog mjesta stvorit će se kvadrat u trajanju od 1 sekunde. Uvećanje kamere izvodi se pomoću dva prsta gestom smanjivanja i povećavanja. Korištenje fokusa s uvećanom kamerom prikazano je na slici 6.2. Ukoliko korisnik ne pritisne mjesto željenog fokusa, fokus će se primijeniti na sredinu ekrana automatski.

Sl. 6.2. Primjena fokusa i uvećanja kamere

Gornji lijevi gumb omogućuje mijenjanje postavki bljeskalice(upaljena, ugašena, automatski) korištene pri uzimanju fotografije. Postoje dvije mogućnosti učitavanja slika za prepoznavanje. Donjim lijevim gumbom omogućeno je učitavanje slike s telefona paljenjem vanjske aktivnosti, prikaz učitavanja slike s telefona može se vidjeti na slici 6.3, te je donjim srednjim gumbom omogućeno snimanje fotografije u stvarnom vremenu.



Sl. 6.3. Zaslon učitavanja slike s telefona

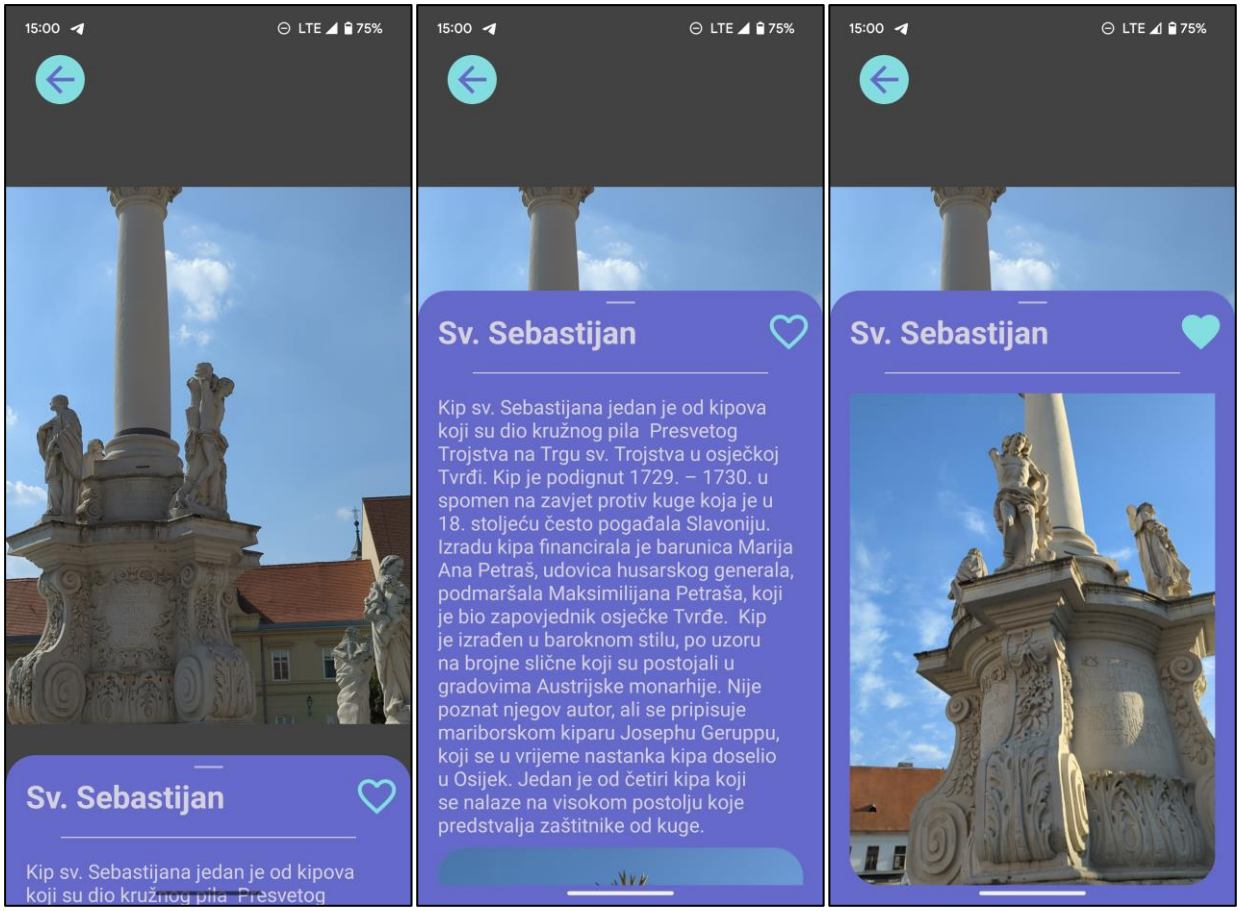
Nakon učitavanja fotografije korisnik je preusmjeren na zaslon informacija. Donji desni gumb odvest će korisnika na listu prepoznatih spomenika koje je dodao u listu favorita.

6.2. Zaslon informacija

Korisnik se preusmjerava na zaslon informacija nakon učitane fotografije. Fotografija učitana u prethodnoj aktivnosti šalje se prema Google Cloud poslužitelju na kojemu se nalazi model neuronske mreže. OkHttp je učinkovit HTTP & HTTP/2 klijent za Android i Java aplikacije korišten za slanje zahtjeva prema poslužitelju iz aplikacije[21]. Fotografija se obrađuje te poslužitelj vraća vektor s vjerojatnostima za svaku klasu. Za vrijeme čekanja odgovora od poslužitelja na zaslonu je prikazan *spinner*. Zaslon za vrijeme čekanja prikazan je na slici 6.4.

Sl. 6.4. Zaslon informacija za vrijeme čekanja odgovora s poslužitelja

Kada aplikacija dobije odgovor od poslužitelja vidljivost *spinnera* se sakriva te se prikazuje donji list s informacijama spomenika koji ima najveću vrijednost vjerojatnosti. Informacije o spomeniku se dohvaćaju iz Firebase realtime database preko indeksa spomenika s najvećom vjerojatnosti. Donji list s informacijama moguće je proširiti povlačenjem prsta ili dodirom na njega, list sadrži informacije o imenu, opisu i slici spomenika koja služi korisniku kao potvrda o predviđanju modela te ga je moguće listati. Osim informacija o spomeniku, u desnom kutu donjeg lista nalazi se gumb u obliku srca kojim je moguće spomenik dodati u listu favorita. Ukoliko se spomenik već nalazi u favoritima gumb srca će biti ispunjen. Primjeri zaslona nakon odgovora poslužitelja nalaze se na slici 6.5.



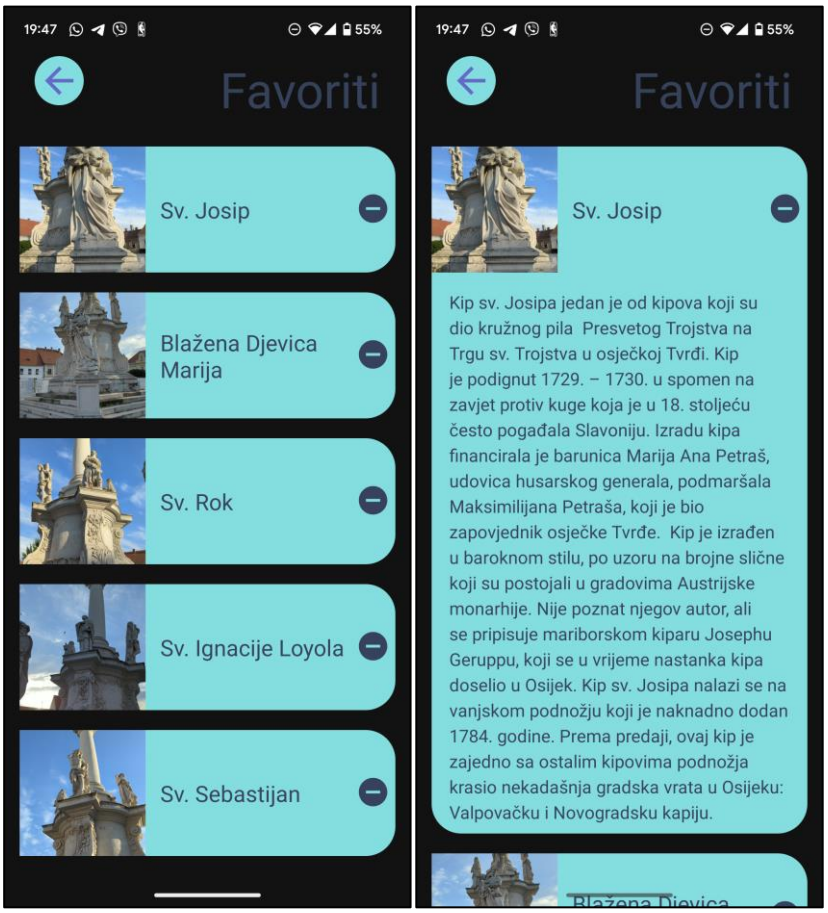
Sl. 6.5. Zaslon s informacijama nakon odgovora poslužitelja

U gornjem lijevom kutu nalazi se gumb kojim se korisnik vraća na zaslon kamere. Ukoliko je najveća vjerojatnost neke klase ispod 50% smatra se da je uslikani objekt nepoznat te će donji list prikazati informacije kako je potrebno ponovno uslikati objekt.

6.3. Zaslon favorita

Zaslon favorita prikazuje zaslon sa spomenicima koji su na zaslonu informacija dodani u listu favorita. Favorizirani spomenici se prikazuju unutar *RecyclerViewa*. *RecyclerView* olakšava učinkovito prikazivanje velikih skupova podataka dinamičkim stvaranjem definiranih komponenti samo kada su potrebne[22]. Svaka stavka u *RecyclerViewu* sadrži sliku, ime i gumb u obliku minusa kojim je moguće ukloniti spomenik iz liste favorita. Dodirom na željenu stavku kartica se proširuje te prikazuje i opis spomenika koji je dodirnut. Ponovnim dodirom kartica se opet smanjuje. Smanjivanje kartice prikazuje se animacijom povlačenja kartice prema gore. Kao i na

zaslonu informacija u gornjem lijevom kutu nalazi se gumb kojim se korisnik može vratiti na zaslon kamere. Slike zaslona favorita sa sažetim i proširenim karticama prikazan je na slici 6.6.



Sl. 6.6. Zaslon favorita s njihovim informacijama

7. ZAKLJUČAK

Tijekom obilazaka kulturnih i povijesnih znamenitosti u sklopu turističkog upoznavanja nekog mjesta, posjetiocima je bitno što jednostavnije saznati više podataka o znamenitosti. Kako bi to bilo moguće informacije im moraju biti lako i brzo dostupne.

U ovom radu razvijena je mobilna aplikacija koja omogućuje korisnicima prepoznavanje spomenika u gradu Osijeku na temelju uslikane fotografije. Osim samog prepoznavanja spomenika korisnik će dobiti i informacije o njemu. Za prepoznavanje fotografija korištena je konvolucijska neuronska mreža VGG-16 dodatno podešavana koristeći Tensorflow u Google Colab razvojnom okruženju. Dodatnim podešavanjima i treniranjem neuronske mreže nad raznovrsnim skupom fotografija različitih klasa postignuta je preciznost mreže od 98.39%. Model mreže je postavljen u Docker sliku koja se pokreće u kontejneru smještenom na Google Cloud platformi koja služi kao poslužitelj. Informacije o spomenicima zajedno sa fotografijama spremani su na Firebase platformu. Za izradu mobilne aplikacije korišteno je Android Studio razvojno okruženje te Kotlin i XML tehnologije. Mobilna aplikacija ima funkcionalnost slanja fotografije iz galerije ili uslikane u stvarnom vremenu na Google Cloud poslužitelj te nakon njegovog odgovora pruža informacije o prepoznatom spomeniku. Aplikacija je trenutno funkcionalna i prepoznaje osam kipova na Kružnom pilu Svetog Trojstva u osječkoj Tvrđi. Moguće ju je dodatno unaprijediti dodavanjem više klasa spomenika kako bi bila u mogućnosti prepoznavati sve spomenike u Osijeku. Osim proširivanja aplikacije većim brojem spomenika, aplikacija može biti unaprijeđena razvojem web administratorskog sučelja u koje bi administrator unosio informacije o spomenicima nakon kojeg bi se mreža automatski nanovo istrenirala. Također, aplikaciju je moguće unaprijediti i na korisničkoj strani dodavanjem čitača informacija za slabovidne.

LITERATURA

[1] Hermes, “Virtualna Umjetnost i Kultura – HERMES,” Hermes. https://hermes.hr/hr/vuk/ (pristupano 23.6.2022).

[2] [2]Timesofindia.Com, “Google Lens app gets two new features,” Times Of India, 29. svibnja 2019. [Na internetu]. Dostupno na: https://timesofindia.indiatimes.com/gadgets-news/google-lens-app-gets-two-new-features/articleshow/69568473.cms (pristupano 26.7. 2022).

[3] L. Little, “Multisearch could make Google Lens your search sensei,” TechRadar, 7. travnja 2022. [Na internetu]. Dostupno na: https://www.techradar.com/news/multisearch-could-make-google-lens-your-search-sensei (pristupano 26.6.2022.)

[4] M. Brown, “Smartify makes all museum audio tours free for rest of 2020,” The Guardian, 26. ožujka 2020. [Na internetu]. Dostupno na: https://www.theguardian.com/culture/2020/mar/26/smartify-makes-all-museum-audio-tours-free-for-rest-of-2020 (pristupano 26.6. 2022).

[5] INTERPOL, “ID-Art mobile app,” Interpol. https://www.interpol.int/Crimes/Cultural-heritage-crime/ID-Art-mobile-app (pristupano 26.6. 2022).

[6] M. Heller, “What is Kotlin? The Java alternative explained,” InfoWorld, 23. ožujka 2020. [Na internetu]. Dostupno na: https://www.infoworld.com/article/3224868/what-is-kotlin-the-java-alternative-explained.html (pristupano 26.6. 2022).

[7] JetBrains, “Kotlin for Android,” Kotlin Help, 6. rujna 2022. https://kotlinlang.org/docs/android-overview.html (pristupano 7.9. 2022).

[8] Mozilla, “XML introduction - XML: Extensible Markup Language,” MDN, 9. rujna 2022. https://developer.mozilla.org/en-US/docs/Web/XML/XML_introduction (pristupano 26.7. 2022).

[9] Google/Developers, “Android Studio features ,” Android Developers, 12. siječnja 2022. https://developer.android.com/studio/features (pristupano 26.6. 2022).

[10] Python Software Foundation, “What is Python? Executive Summary,” Python.org. https://www.python.org/doc/essays/blurb/ (pristupano 26.6. 2022).

[11] S. Yegulalp, “What is TensorFlow? The machine learning library explained,” InfoWorld, 3. lipnja 2022. [Na internetu]. Dostupno na: https://www.infoworld.com/article/3278008/what-is-tensorflow-the-machine-learning-library-explained.html (pristupano 26.6. 2022).

[12] Google, “Google Colab,” Google. https://research.google.com/colaboratory/faq.html (pristupano 26.6. 2022).

[13] L. Rosencrance, “Google Firebase,” TechTarget, 25. travnja 2019. [Na internetu]. Dostupno na: https://www.techtarget.com/searchmobilecomputing/definition/Google-Firebase (pristupano 26.6.2022)

[14] M. O’Gara, “Ben Golub, Who Sold Gluster to Red Hat, Now Running dotCloud,” WayBackMachine, 26. srpnja 2013. [Na internetu]. Dostupno na: https://web.archive.org/web/20190913100835/http://maureenogara.sys-con.com/node/274733 (pristupano 18.8.2022).

[15] Docker Inc., “Docker frequently asked questions (FAQ),” Docker Documentation, 9. rujna 2022. https://docs.docker.com/engine/faq/#what-does-docker-technology-add-to-just-plain-lxc (pristupano 18.8.2022).

[16] Google, “Why Google Cloud,” Google Cloud. https://cloud.google.com/why-google-cloud (pristupano 18.8.2022).

[17] G. L. Team, “What is VGG16 - Convolutional Network for Classification and Detection,” GreatLearning Blog: Free Resources what Matters to shape your Career!, 1. listopada 2021. [Na internetu]. Dostupno na: https://www.mygreatlearning.com/blog/introduction-to-vgg16/#VGG%2016%20Architecture (pristupano 26.8.2022).

[18] J. Brownlee, “A Gentle Introduction to the Rectified Linear Unit (ReLU),” Machine Learning Mastery, 8. siječnja 2019. [Na internetu]. Dostupno na: https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/ (pristupano 26.8.2022).

[19] Google, “For Production,” TensorFlow. https://www.tensorflow.org/tfx/guide/serving (pristupano 26.8. 2022).

[20] Google/Developers, “CameraX ,” Android Developers. https://developer.android.com/jetpack/androidx/releases/camera (pristupano 26.8.2022).

[21] baeldung, “A Guide to OkHttp,” Baeldung, 6. prosinca 2016. https://www.baeldung.com/guide-to-okhttp (pristupano 3.9.2022).

[22] Google/Developers, “Create dynamic lists with RecyclerView ,” Android Developers. https://developer.android.com/develop/ui/views/layout/recyclerview?gclid=Cj0KCQjwmdGYBhDRARIsABmSEeOed006pN7baZUeCeF6vrUnFYY3Y1sK6LEvas0y6PhNBJmMIZYAIoaAimeEALw_wcB&gclsrc=aw.ds (pristupano 3.9.2022).

SAŽETAK

U ovome radu razvijene su neuronska mreža i mobilna aplikacija koje omogućavaju korisnicima prepoznavanje i informiranje o kulturnom spomeniku na temelju uslikane fotografije spomenika. Razumijevanje umjetnosti i kulturnog dobra vrlo je važno jer se smatra jednim od ključnih kompetencija funkcioniranja u društvu. Za razvoj modela neuronske mreže korišteno je Google Colab razvojno okruženje i Tensorflow biblioteka. Postavljanje modela neuronske mreže na poslužitelj postignuto je Dockerom i Google Cloud platformom, a informacije o spomenicima čuvaju se na Firebase platformi. Mobilna aplikacija izrađena je u Android Studio razvojnom okruženju programskim jezikom Kotlin, a izgled aplikacije ostvaren je pomoću XML-a.

Ključne riječi: Android Studio, mobilna aplikacija, neuronska mreža, spomenici, Tensorflow

ABSTRACT

Android application for monument recognition

In this paper, a neural network and a mobile application were developed to enable users to recognize and learn about a cultural monument based on a photograph of the monument. Understanding art and cultural heritage is very important because it is considered as one of the key competencies for functioning in society. The Google Colab development environment and Tensorflow library were used to develop the neural network model. Deploying the neural network model to the server was achieved with Docker and the Google Cloud platform while the information about the monuments is stored on the Firebase platform. The mobile application was created in the Android Studio development environment using the Kotlin programming language, and the application layout was created using XML.

Key words: Android Studio, mobile application, monuments, neural network, Tensorflow

ŽIVOTOPIS

Juraj Perić rođen je u Osijeku 13. listopada 2000. godine. Pohađao je Osnovnu školu Frana Krste Frankopana te se nakon završetka 2015. godine upisuje u III. Gimnaziju Osijek. Nakon završetka gimnazije 2019. godine, upisuje se na preddiplomski sveučilišni studij Računarstva na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija u Osijeku.

_____

Potpis autora