Integracija openCV biblioteke u React Native android aplikaciju

Malnar, Luka

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Stručni studij računarstva

Integracija OpenCV biblioteke u React Native android aplikaciju

Završni rad

Luka Malnar

Osijek, 2024.

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

Obrazac Z1S: Obrazac za ocjenu završnog rada na stručnom prijediplomskom studiju

Ocjena završnog rada na stručnom prijediplomskom studiju

Ime i prezime pristupnika:	Luka Malnar
Studij, smjer:	Stručni prijediplomski studij Računarstvo
Mat. br. pristupnika, god.	R 4392, 21.10.2020.
JMBAG:	0165082136
Mentor:	doc. dr. sc. Hrvoje Leventić
Sumentor:	
Sumentor iz tvrtke:	
Predsjednik Povjerenstva:	doc. dr. sc. Krešimir Romić
Član Povjerenstva 1:	doc. dr. sc. Hrvoje Leventić
Član Povjerenstva 2:	Marin Benčević, univ. mag. ing. comp.
Naslov završnog rada:	Integracija openCV biblioteke u React Native android aplikaciju
Znanstvena grana završnog rada:	**Programsko inženjerstvo (zn. polje računarstvo)**
Zadatak završnog rada:	Istražiti i opisati metode i biblioteke dostupne za obradu slike unutar React Native programskog okvira. Opisati openCV biblioteku i metodu integracije u React Native. Za praktični dio razviti Android aplikaciju pomoću React Native okvira koja će integrirati openCV biblioteku. Implementirati u aplikaciju odabrane metode obrade slike pomoću openCV-a. Rezervirano za: Luka Malnar

Datum ocjene pismenog dijela završnog rada od strane mentora:	23.09.2024.
Ocjena pismenog dijela završnog rada od strane mentora:	Izvrstan (5)
Datum obrane završnog rada:	27.9.2024.
Ocjena usmenog dijela završnog rada (obrane):	Izvrstan (5)
Ukupna ocjena završnog rada:	Izvrstan (5)
Datum potvrde mentora o predaji konačne verzije završnog rada čime je pristupnik završio stručni prijediplomski studij:	27.09.2024.

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

IZJAVA O IZVORNOSTI RADA

Osijek, 27.09.2024.

Ime i prezime Pristupnika:	Luka Malnar
Studij:	Stručni prijediplomski studij Računarstvo
Mat. br. Pristupnika, godina upisa:	R 4392, 21.10.2020.
Turnitin podudaranje [%]:	5

Ovom izjavom izjavljujem da je rad pod nazivom**: Integracija openCV biblioteke u React Native android aplikaciju**

izrađen pod vodstvom mentora doc. dr. sc. Hrvoje Leventić

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 pristupnika:

Sadržaj

1. UVOD

Mobilne aplikacije postaju sveprisutne u svakodnevnom životu, nudeći korisnicima razne funkcionalnosti i rješenja za svakodnevne izazove. Jedan od takvih izazova je rješavanje sudoku zagonetki, popularne igre logike koja zahtijeva preciznost i brzinu. U ovom radu predstavljena je integraciju OpenCV biblioteke u React Native Android aplikaciju za rješavanje sudoku zagonetki. OpenCV (Open Source Computer Vision Library) je biblioteka otvorenog koda koja pruža alate za obradu slika i računalni vid. Integracija OpenCV biblioteke u mobilnu aplikaciju omogućuje korištenje naprednih tehnika obrade slika kako bi se prepoznale i riješile sudoku zagonetke. Prvi korak u procesu je prepoznavanje sudoku polja na slici. Koristeći OpenCV, analizirana je slika snimljena kamerom mobilnog uređaja ili odabrana iz galerije te je određeno gdje se nalazi sudoku polje. Nakon toga, primjenjujemo različite metode obrade slike kako bismo poboljšali kvalitetu slike i istaknuli pojedine elemente. Nadalje, korišten je Convolutional Neural Network (CNN) model za klasifikaciju kako bi se prepoznalo koje znamenke ili znakovi se nalaze unutar pojedinih sudoku polja. Model je treniran u Pythonu, a zatim implementiran unutar naše aplikacije koristeći TensorFlow.js, omogućujući nam brzu i preciznu klasifikaciju. Nakon prepoznavanja znamenki unutar sudoku polja, korišteni su algoritmi za rješavanja sudoku zagonetki kako bismo pronašli rješenje. Rješenje se zatim prikazuje korisniku na slici sudoku ploče, uz pomoć OpenCV-a, čineći aplikaciju intuitivnom i korisnički prijateljskom. Ovaj rad pruža detaljan pregled procesa integracije OpenCV biblioteke u React Native Android aplikaciju za rješavanje sudoku zagonetki, koristeći napredne tehnike obrade slika i klasifikacije. Kroz opisani proces, korisnicima se nudi intuitivno i efikasno rješenje za brzo i precizno rješavanje sudoku zagonetki putem mobilnog uređaja.

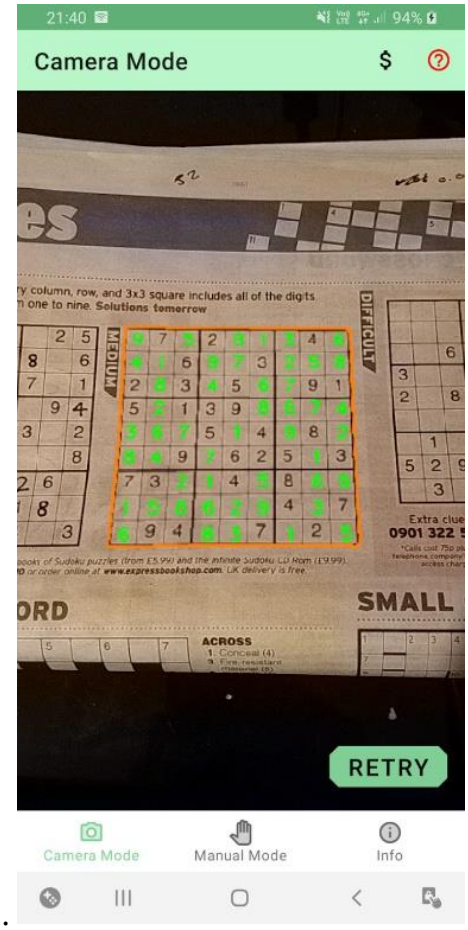
1.1. Zadatak završnog rada

Cilj ovog završnog rada je istražiti i implementirati integraciju OpenCV biblioteke u React Native Android aplikaciju za rješavanje sudoku zagonetki. Konkretno, istražujemo kako koristiti napredne tehnike obrade slika i klasifikacije, omogućene OpenCV-om i CNN modelom, kako bismo automatski prepoznali i riješili sudoku zagonetke putem mobilnog uređaja.

2. POSTOJUĆA RJEŠENJA

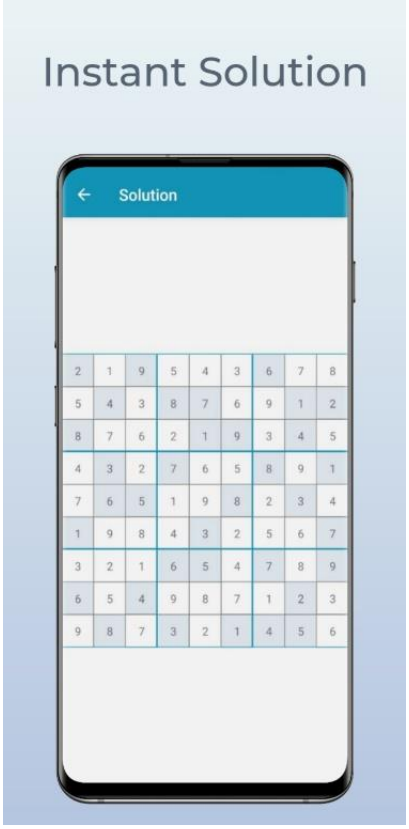
2.1.Aplikacije za rješavanje sudoku zagonetki

U području mobilnih aplikacija, sve veći broj korisnika traži rješenja za svakodnevne izazove poput rješavanja sudoku zagonetki. Aktualni dosezi u području rješavanja sudoku zagonetki putem mobilnih aplikacija obuhvaćaju aplikacije kao što su Sudoku Solver , SudoSolver i Sudoku Solver [Camera].



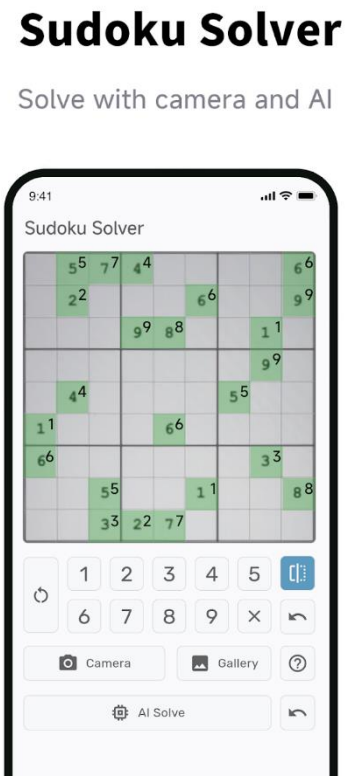
Slika 2.1.1 Sudoku Solver aplikacija

Aplikacija Sudoku Solver [1], omogućuje korisnicima jednostavno rješavanje klasičnih Sudoku zagonetki uz pomoć kamere mobilnog uređaja. Korisnici mogu usmjeriti kameru na papirnatu Sudoku zagonetku, a aplikacija će prepoznati brojeve i automatski izračunati rješenje. Ova funkcionalnost čini aplikaciju izuzetno korisnom za one koji žele brza rješenja bez potrebe za ručnim unosom brojeva. Jedna od ključnih prednosti aplikacije je njena jednostavnost i praktičnost za rješavanje standardnih 9x9 Sudoku zagonetki. Međutim, aplikacija ne podržava varijante s većim dimenzijama, što može biti ograničenje za naprednije korisnike.



Slika 2.1.2 SudoSolver aplikacija

SudoSolver [2] aplikacija je dizajnirana za rješavanje klasičnih 9x9 sudoku zagonetki, ali u usporedbi s aplikacijom Sudoku Solver, korisnici u SudoSolveru moraju ručno unijeti znamenke u prazna polja. Ova aplikacija pruža jednostavno i pregledno sučelje, koje korisnicima omogućuje intuitivno unošenje vrijednosti u prazna polja sudoku ploče. SudoSolver nudi osnovne funkcije rješavanja zagonetki, poput provjere ispravnosti unesenih brojeva, ali ne posjeduje napredne značajke poput automatskog prepoznavanja znamenki putem kamere.



Slika 2.1.2 Sudoku Solver [Camera] aplikacija

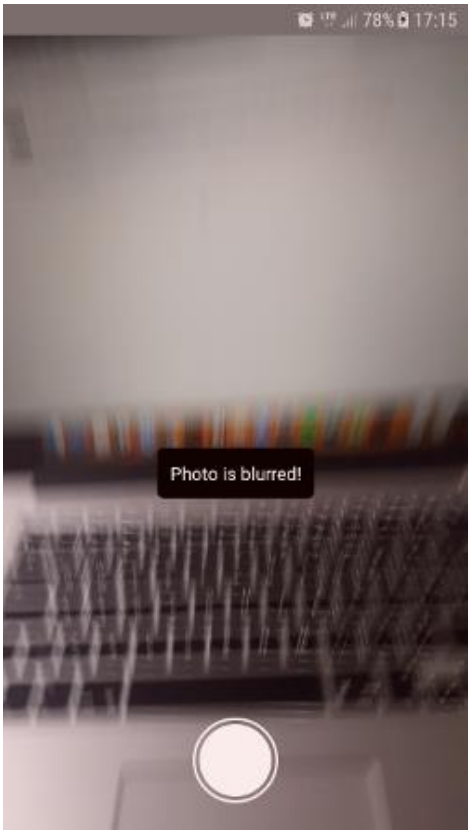
Sudoku Solver [Camera] [23] aplikacija korisnicima omogućuje skeniranje sudoku polja pomoću kamere te automatsko prepoznavanje i rješavanje zagonetke. Uz prepoznavanje brojeva putem kamere, aplikacija također nudi mogućnost ručnog unosa brojeva. Na taj način, ukoliko aplikacija krivo prepozna broj ili ga ne prepozna uopće, korisnici imaju opciju ispraviti grešku ili dodati broj. Osim automatskog rješavanja zagonetki, aplikacija omogućuje i ručno rješavanje sudoku zagonetki, što pruža korisnicima fleksibilnost da sami dovrše zagonetku. Također, aplikacija nudi opciju generiranja nasumičnih sudoku polja različitih težina, čime se korisnicima omogućuje da biraju između lakših i težih zagonetki, ovisno o njihovim preferencijama. Međutim, aplikacija podržava isključivo klasične 9x9 sudoku zagonetke, što je ograničenje za korisnike koji preferiraju složenije varijante.

2.2.Aplikacije koje demonstriraju integraciju OpenCV-a

Integracija OpenCV biblioteke u mobilne aplikacije omogućuje napredne funkcionalnosti obrade slike i računalnog vida. Sljedeće aplikacije i alati demonstriraju kako se OpenCV može integrirati u React Native aplikacije.

Slika 2.2.1 React Native Document Scanner

React Native Document Scanner [3] razvijen od strane Michael Villeneuvea je aplikacija za skeniranje dokumenata unutar React Native aplikacija, podržavajući Android i iOS platforme. Ova aplikacija koristi OpenCV za obradu slike te omogućuje korisnicima da koriste kameru uređaja za skeniranje dokumenata, s mogućnostima detekcije granica dokumenta u stvarnom vremenu, korekcije perspektive, primjene filtera slike (svjetlina, zasićenost, kontrast), te spremanja rezultirajućih slika.



Slika 2.2.2 React Native Document Scanner

React Native OpenCV Tutorial [22], iako primarno predstavljen kao tutorial, omogućuje kreiranje funkcionalne aplikacije koja koristi OpenCV biblioteku unutar React Native okruženja. Ova aplikacija koristi osnovne značajke OpenCV-a za obradu slike, poput detekcije rubova i obrade kontura. Koristeći kameru mobilnog uređaja, aplikacija prepoznaje i analizira objekte unutar slike te primjenjuje OpenCV funkcije za filtriranje i manipulaciju slike u stvarnom vremenu.

3. PRIMIJENJENE TEHNOLOGIJE I ALATI

3.1.OpenCV

OpenCV (Open Source Computer Vision Library) je popularna biblioteka otvorenog koda [4] koja se koristi za računalni vid i strojno učenje. Razvijena je kako bi pružila zajedničku infrastrukturu za aplikacije računalnog vida i ubrzala upotrebu strojnog prepoznavanja u komercijalnim proizvodima. OpenCV je započeo kao istraživački projekt [5] u Intelu 1999. godine s ciljem stvaranja zajedničke infrastrukture koja bi mogla ubrzati istraživanja i razvoj u računalnom vidu. Prva verzija OpenCV-a puštena je u javnost 2000. godine. Od tada, OpenCV je prošao kroz mnoge iteracije i proširenja, te sada podržava više programskih jezika kao što su C++, Python, Java i MATLAB.

Podržava različite operativne sustave [4], uključujući Windows, Linux, MacOS, Android i iOS, što ga čini fleksibilnim alatom za razvoj multiplatformskih aplikacija. Biblioteka sadrži više od 2500 optimiziranih algoritama za računalni vid koji se mogu koristiti za različite zadatke kao što su detekcija i prepoznavanje lica, prepoznavanje objekata, klasifikacija scena, rekonstrukcija 3D modela, praćenje pokreta, segmentacija slike, i ekstrakcija objekata.

Također podržava paralelno procesiranje [6] putem više jezgrenih CPU-a i GPU-a, što omogućuje bržu obradu velikih skupova podataka i složenih algoritama. Kompatibilan je s mnogim drugim bibliotekama i alatima kao što su TensorFlow, PyTorch, Keras, NumPy, SciPy i Matplotlib, što omogućuje integraciju s alatima za strojno učenje i znanost o podacima.

OpenCV se koristi u raznim industrijama i aplikacijama [5]. U automobilskoj industriji koristi se za razvoj autonomnih vozila i naprednih sustava pomoći vozaču (ADAS). U zdravstvu pomaže u analizi medicinskih slika i razvoju dijagnostičkih alata. U sigurnosnim sustavima koristi se za prepoznavanje lica i praćenje. U robotici omogućuje robotima percepciju i interakciju s okolinom.

3.2.React Native

React Native je popularan okvir otvorenog koda za razvoj mobilnih aplikacija [7] koji omogućuje izradu nativnih aplikacija za Android i iOS koristeći JavaScript i React. Razvijen od strane Facebooka i prvi put objavljen 2015. godine, React Native je dizajniran kako bi programerima omogućio korištenje već postojećih React vještina za izradu mobilnih aplikacija s nativnim izgledom i osjećajem. Jedna od ključnih prednosti React Native-a je njegova sposobnost da koristi isti kod za više platformi, što značajno smanjuje vrijeme razvoja i troškove. Umjesto pisanja odvojenih kodova [8] za Android i iOS, programeri mogu koristiti jedan zajednički kod za obje platforme. React Native koristi nativne komponente, što omogućuje bolju performansu i korisničko iskustvo u usporedbi s web-based

aplikacijama.

Podržava širok raspon dodataka i modula [10] koji omogućuju proširenje funkcionalnosti aplikacija. Ovi moduli obuhvaćaju različite aspekte mobilnog razvoja kao što su pristup kameri, GPS-u, senzorskim podacima, push obavijestima i mnogim drugim značajkama. Zahvaljujući širokoj zajednici programera, dostupne su brojne biblioteke i alati koji olakšavaju razvoj složenih aplikacija.React Native omogućuje "hot reloading" značajku [9], koja omogućuje trenutne promjene u kodu bez potrebe za ponovnim pokretanjem aplikacije, što značajno ubrzava razvojni proces.

Koristi se u raznim industrijama za izradu različitih vrsta aplikacija [7]. U e-trgovini se koristi za izradu aplikacija za kupovinu, u zdravstvu za razvoj aplikacija za praćenje zdravlja i fitnessa, u financijama za izradu mobilnih bankarskih aplikacija, te u mnogim drugim sektorima. Popularne aplikacije poput Facebooka, Instagrama, Airbnba i Skypea koriste React Native za svoje mobilne platforme, što svjedoči o njegovoj pouzdanosti i fleksibilnosti.

3.3.Tensorflow.js

TensorFlow.js je moćna biblioteka otvorenog koda razvijena od strane Google Brain tima [13], koja omogućuje razvoj i treniranje modela strojnog učenja unutar web preglednika, na Node.js platformi, kao i u React Native aplikacijama. Pružajući fleksibilnost i performanse potrebne za moderne aplikacije, TensorFlow.js koristi hardversku akceleraciju putem WebGL-a za treniranje modela direktno u pregledniku [14], čime se smanjuje potreba za prijenosom podataka na server i povećava privatnost korisničkih podataka.

Podržava širok raspon algoritama i arhitektura strojnog učenja [15], uključujući duboke neuronske mreže (DNN), konvolucijske neuronske mreže (CNN) i rekurentne neuronske mreže (RNN). Kompatibilan je s drugim JavaScript bibliotekama kao što su React i Angular, što ga čini idealnim za razvoj aplikacija koje zahtijevaju napredne funkcionalnosti strojnog učenja.

Biblioteka također nudi podršku za paralelno procesiranje [14] putem više jezgrenih CPU-a i GPU-a, omogućujući bržu obradu i treniranje modela. Korištenjem TensorFlow.js-a, programeri mogu stvoriti interaktivne aplikacije koje koriste modele strojnog učenja za inteligentne funkcionalnosti, kao što su prepoznavanje slika, analiziranje sentimenta i personalizaciju sadržaja.

3.4.Python

Python je programski jezik visoke razine i opće namjene [16], koji je poznat po svojoj jednostavnoj sintaksi, čitljivosti i svestranosti. Razvijen od strane Guido van Rossuma i prvi put predstavljen 1991. godine, Python je postao jedan od najpopularnijih programskih jezika u svijetu. Dizajniran je s naglaskom na čitljivost koda, omogućujući programerima da izraze koncepte u manje redaka koda u usporedbi s drugim jezicima kao što su C++ ili Java.

Jedna od ključnih prednosti je njegova široka primjena u različitim područjima, uključujući web razvoj, znanost o podacima, strojno učenje, umjetnu inteligenciju, analizu podataka, automatizaciju sustava i razvoj softvera. Python podržava različite paradigme programiranja, uključujući proceduralno, objektno-orijentirano i funkcionalno programiranje.

Poznat je po svojoj opsežnoj standardnoj biblioteci [17] koja pruža module i funkcije za rad s datotekama, web servisima, protokolima, GUI razvojem i mnogim drugim područjima. Osim standardne biblioteke, Python ima veliki ekosustav dodatnih okvira kao što su Django i Flask za web razvoj i biblioteka [18], kao što su NumPy i pandas za znanstvene podatke, TensorFlow i PyTorch za strojno učenje te Matplotlib i Seaborn za vizualizaciju podataka.

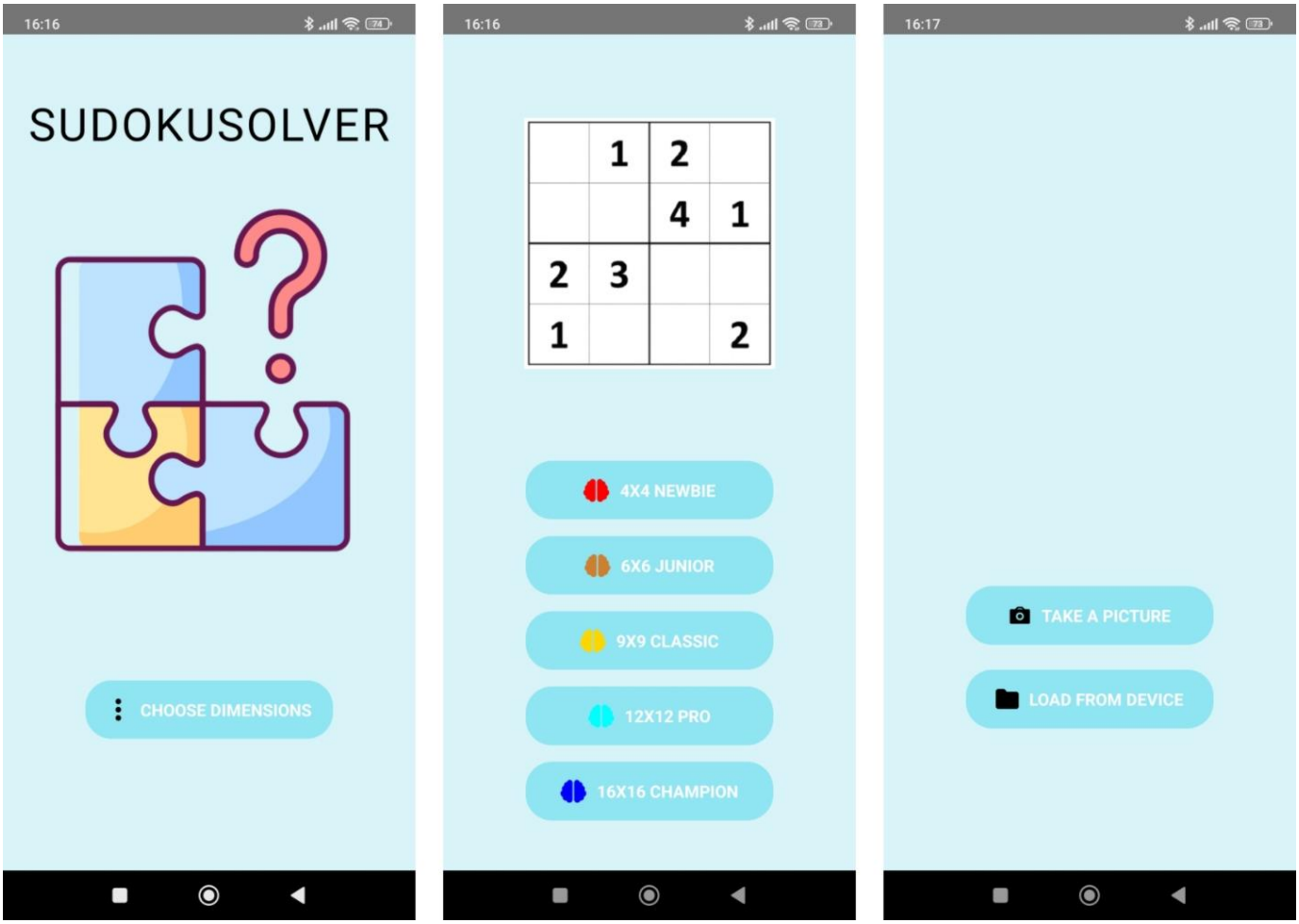
Pythonova popularnost [16] također se može pripisati njegovoj interoperabilnosti s drugim jezicima i platformama, kao i snažnoj podršci za integraciju s raznim tehnologijama. Python skripte mogu se koristiti za automatizaciju zadataka, dok se Python moduli mogu integrirati s aplikacijama napisanim u drugim jezicima kao što su C, C++ i Java.

4. PRIKAZ RADA APLIKACIJE

U ovom je poglavlju prikazan je izgled svakog dijela mobilne aplikacije te primjer kako to izgleda kada korisnik koristi aplikaciju.

4.1.Navigiranje aplikacijom

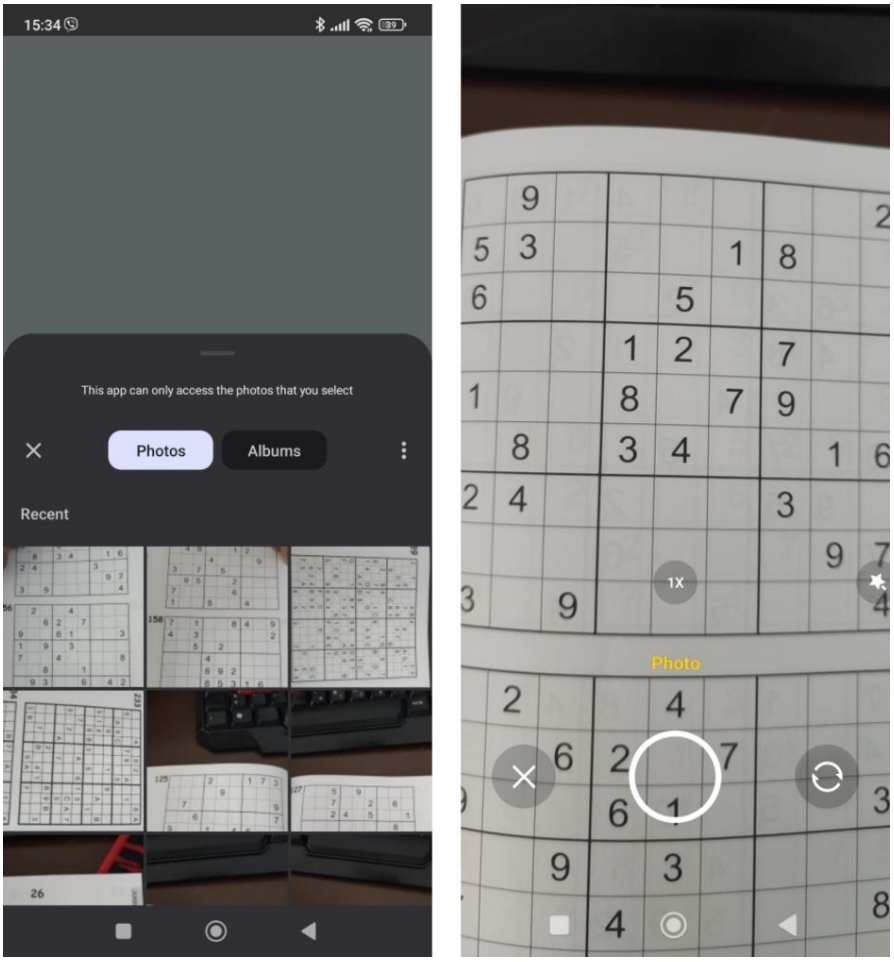
Na početnoj stranici se nalazi slika zagonetke koja predstavlja ne riješeni sudoku. Klikom na gumb CHOOSE DIMENSIONS prelazimo na drugi dio aplikacije gdje su nam ponuđeni izbornici kojima je omogućeno odabiranje dimenzija sudoku polja koje ćemo rješavati. Zatim se nam ponuđene 2 opcije odabira slike, prva opcija nam omogućava slikanje koristeći kameru na mobitelu, a druga opcija nam omogućava odabir već postojeće slike iz memorije uređaja.



Slika 4.1.1 Navigiranje aplikacijom

4.2.Odabir slike

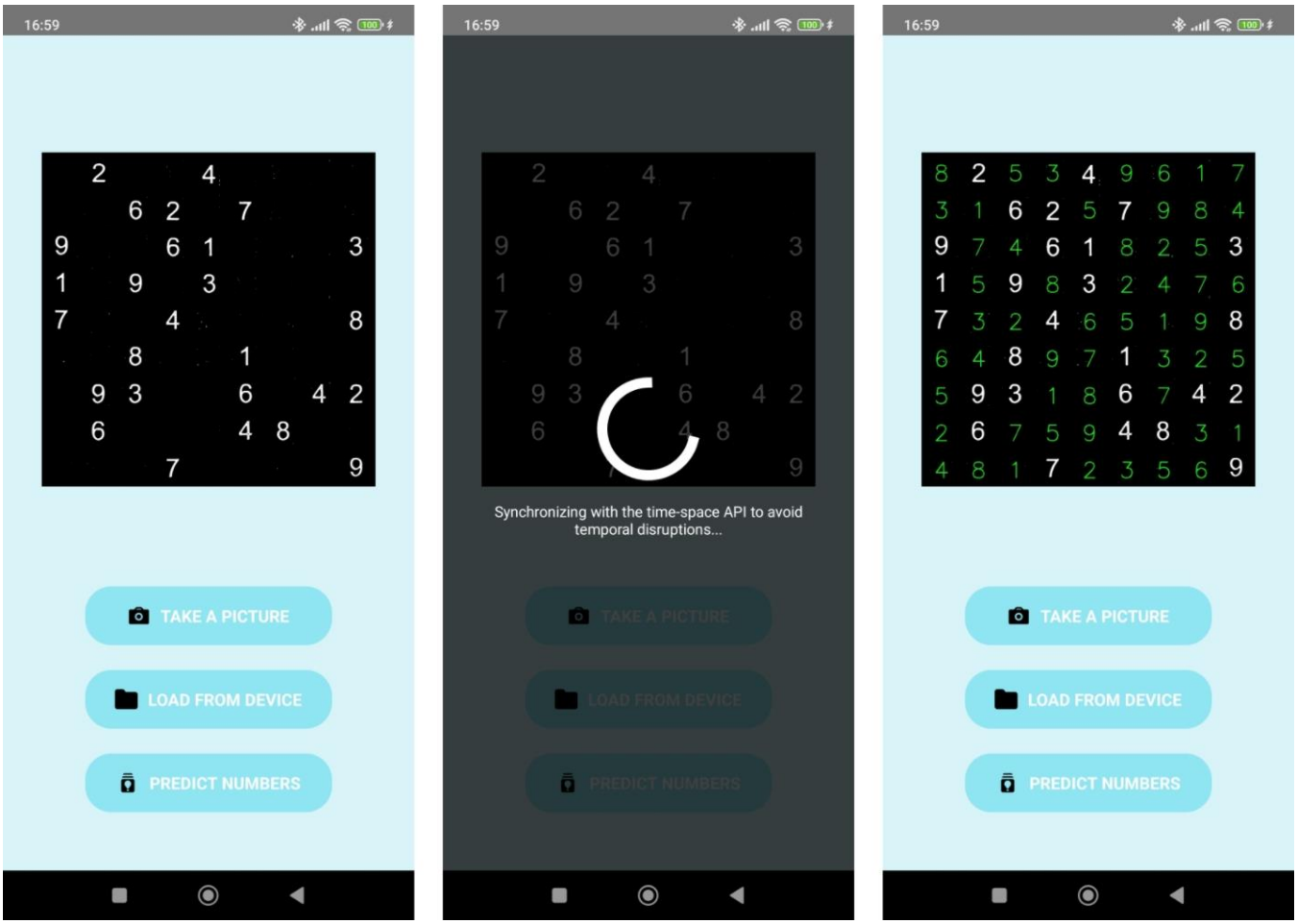
Na lijevoj strani slike prikazan je proces odabira opcije "LOAD FROM DEVICE". U odjeljku "Photos" možete pregledati nedavne slike, dok pritiskom na gumb "Albums" možete pristupiti različitim direktorijima poput "Kamera", "Screenshots", "Viber" ili "WhatsApp". Na desnoj strani slike prikazan je proces odabira opcije "TAKE A PICTURE". Klikom na ovu opciju, otvara se kamera uređaja omogućujući vam da fotografirate sudoku koji želite riješiti.



Slika 4.2.1 Prikaz različitog načina odabira slika

4.3.Rješavanje

Na lijevoj strani slike vidimo kako izgleda odabrana i očitana slika. Prije očitavanja brojeva i rješavanja sudokua, slika prolazi kroz potrebnu obradu. Ova prikazana slika omogućuje korisniku da provjeri kvalitetu pred obrade slike. Ako slika nije dobro obrađena, korisnik može pokušati ponovno slikati ili odabrati drugu sliku. Mutne slike s mnogo šuma mogu otežati ili onemogućiti rješavanje sudokua. U središtu slike prikazan je proces nakon odabira opcije "PREDICT NUMBERS". Pojavljuje se indikator aktivnosti koji signalizira rješavanje sudoku polja. Na desnoj strani slike vidimo riješeno sudoku polje.



Slika 4.3.1 Prikaz rješavanja sudoku polja

5. POSTUPAK RIJEŠAVANJA SUDOKU POLJA

U ovom poglavlju detaljno ćemo opisati postupak rješavanja sudoku polja u našoj aplikaciji. Koristili smo OpenCV biblioteku za obradu slika, implementirali smo je u Android dio aplikacije, te smo razvili vlastiti CNN model u Pythonu, koji smo kasnije integrirali u aplikaciju kao TensorFlow.js (tfjs) model.

5.1. Integracija OpenCV biblioteke

Prije početka integracije OpenCV biblioteke u React Native Android aplikaciji, bilo je potrebno preuzeti i pokrenuti skriptu downloadAndInsertOpenCV.sh [21]. Koristili smo metode opisane u [19] i [20]

```
1   version=4.8.0
2   base_url=https://sourceforge.net/projects/opencvlibrary
3   /files/${version}/
4   #android
5   curl -LO ${base_url}/opencv-${version}-android-sdk.zip
6   unzip opencv-${version}-android-sdk.zip
7   cd android/app/src/main
8   mkdir jniLibs
9   cp -r ./../../../../OpenCV-android-sdk/sdk/native/libs/ ./jniLibs
10  cd ../../../../
11  rm -rf opencv-${version}-android-sdk.zip
12  rm -rf OpenCV-android-sdk/
13
14
```

Kod 5.1.1 DownoloadAndInsertOpencv skripta

Ova skripta preuzima OpenCV SDK verziju 4.8.0, raspakirava ju, kopira potrebne biblioteke u direktorij jniLibs unutar vašeg projekta, te zatim briše privremene datoteke.

Nakon pokretanja ove skripte, otvorio sam projekt u Android Studiju. Preuzeo sam 4.8.0 verziju OpenCV-a za Android sa službene stranice kako bih imao pristup najnovijim značajkama i ispravcima grešaka. Uveo sam OpenCV u Android Studio putem opcije "File" -> "New" -> "Import Module" i odabrao "sdk/java" mapu unutar raspakiranog OpenCV arhiva, čime sam dodao OpenCV kao modul u moj projekt. Sljedeći korak bio je ažuriranje build.gradle datoteke unutar uvezenog OpenCV modula. Ažurirao sam četiri polja kako bi odgovarala postavkama u mojoj glavnoj build.gradle datoteci projekta: compileSdkVersion, buildToolsVersion, minSdkVersion i targetSdkVersion. Time sam

osigurao da su verzije sinkronizirane između glavnog projekta i OpenCV modula. Nakon ažuriranja build.gradle datoteke, dodao sam ovisnost modula. Otvorio sam postavke modula unutar aplikacije, odabrao karticu "Dependencies" i kliknuo na ikonu "+" na dnu. Odabrao sam "Module Dependency" i zatim odabrao uvezeni OpenCV modul. Unutar direktorija android/app/src/java kreirao sam paket pod nazivom "com.reactlibrary" kako bih organizirao svoje kodne datoteke. Također sam ažurirao manifest datoteku s odgovarajućim dozvolama za kameru i pristup vanjskoj pohrani. Zatim sam unutar novo kreiranog paketa napravio datoteku RNOpenCvLibraryModule.java i ispunio je metodama koje su mi bile potrebne, umjesto onih definiranih u instrukcijskom videu. Napravio sam i datoteku RNOpenCvLibraryPackage.java koja nam omogućuje React Native okruženju da prepozna i koristi RNOpenCvLibraryModule, čime se osigurava da su sve potrebne nativne funkcionalnosti ispravno registrirane i dostupne JavaScript kodu. Konačno, inicijalizirao sam OpenCV u metodi onCreate unutar MainApplication klase i dodao BaseLoaderCallback, također, u metodi onConfigurationChanged osigurao sam da se promjene konfiguracije ispravno obrađuju.

```
1   public void onCreate() {
2         super.onCreate();
3         SoLoader.init(this, /* native exopackage */ false);
4
5         if (!OpenCVLoader.initDebug()) {
6           Log.d("OpenCv", "Error while init");
7         }
8
9         // Initialize OpenCV
10        if (!OpenCVLoader.initDebug()) {
11          Log.d("OpenCV", "Internal OpenCV library not found.
12           Using OpenCV Manager for initialization");
13          OpenCVLoader.initAsync(OpenCVLoader.OPENCV_VERSION_3_0_0,
14           this, mLoaderCallback);
15        } else {
16          Log.d("OpenCV", "OpenCV library found inside the package.
17           Using it!");
18          mLoaderCallback.onManagerConnected
19          (LoaderCallbackInterface.SUCCESS);
20        }
21    }
```

Kod 5.1.2 Metoda onCreate

5.2.Korištene tehnike obrade digitalne slike

Učitavanje slike u format kompatibilan s OpenCV-om je ključni korak u obradbi slike. Slike se najčešće pohranjuju u različitim formatima kao što su JPEG, PNG, BMP, TIFF, itd. Kada radimo s OpenCV-om, ove slike treba pretvoriti u interni format koji OpenCV može efikasno koristiti za daljnju obradu i analizu.

OpenCV koristi format dvodimenzionalne matrice, poznat kao Mat, za pohranu slika [5]. Mat objekt u OpenCV-u predstavlja dvodimenzionalnu matricu piksela gdje su podaci spremljeni u kontinuiranom bloku memorije. Svaki element matrice odgovara pikselu u slici, a vrijednosti unutar matrice ovisi o tipu slike (npr. grayscale ili RGB).

Resize, ili promjena veličine slike, je tehnika obrade slika koja se koristi za prilagodbu dimenzija slike. Ova operacija može uključivati povećanje (zoom in) ili smanjenje (zoom out) slike, ovisno o potrebama. Funkcija koja se koristi za ovu svrhu u OpenCV-u je cvResize().

Funkcija cvResize() prilagođava izvornu sliku točno veličini odredišne slike. Ako je postavljen ROI (Region of Interest) u izvornoj slici, tada će se taj ROI promijeniti kako bi se uklopio u odredišnu sliku. Isto tako, ako je ROI postavljen u odredišnoj slici, izvor će se promijeniti kako bi se uklopio u ROI."

Pretvorba slike iz RGB (Red, Green, Blue) prostora boja u grayscale (nijanse sive) je čest korak u obradi slika. Ovaj proces se koristi kako bi se smanjila količina podataka koje je potrebno obraditi, te kako bi se olakšala daljnja analiza slike. Pretvorba iz RGB u grayscale može se postići primjenom različitih težina na RGB komponente slike i zbrajanjem rezultata. Formula koja se najčešće koristi za ovu pretvorbu je:

$$Grayscale = 0.299 * R + 0.587 * G + 0.114 * B \tag{5-1}$$

Ova formula koristi različite težine za crvenu, zelenu i plavu komponentu, jer ljudsko oko različito percipira intenzitet ovih boja.

Gaussian Blur, ili Gaussovo zamućenje, je jedna od ključnih tehnika obrade slika koja se često koristi za smanjenje šuma i artefakata iz slike. Svaka točka u ulaznoj slici se zamjenjuje težinskim prosjekom okolnih točaka, pri čemu se težine određuju Gaussovom funkcijom. Ovo zamućenje je korisno jer smanjuje šum, a pritom manje zamućuje rubove u usporedbi s jednostavnim srednjim zamućenjem.

Thresholding, ili određivanje praga, je tehnika obrade slika koja se koristi za segmentaciju slike, pretvarajući sivu sliku u binarnu sliku. Ova tehnika pojednostavljuje analizu slike naglašavanjem važnih objekata i uklanjanjem nevažnih informacija.

Klasično određivanje praga uključuje odabir jedne vrijednosti praga, pri čemu svi pikseli s intenzitetom iznad te vrijednosti postaju bijeli (255), a svi pikseli ispod te vrijednosti postaju crni (0).

Adaptive thresholding je naprednija verzija ovog postupka koja koristi različite vrijednosti praga za različite dijelove slike, čime se postiže bolja segmentacija kod slika s neujednačenom osvijetljenošću. Jedna od popularnih metoda adaptivnog određivanja praga je ADAPTIVE_THRESH_GAUSSIAN_C, koja koristi lokalne srednje vrijednosti za određivanje praga.

Perspective transformation, ili perspektivna transformacija, je tehnika obrade slika koja omogućuje promjenu perspektive slike, dajući joj izgled kao da je snimljena iz drugačijeg kuta. Ova tehnika je korisna za ispravljanje perspektivnih iskrivljenja ili za izravnavanje slike kako bi se olakšala analiza.

U OpenCV-u, perspektivna transformacija se postiže korištenjem funkcija getPerspectiveTransform i warpPerspective. Funkcija getPerspectiveTransform koristi četiri uparena skupa točaka izvorne i ciljne slike kako bi izračunala transformacijsku matricu. Zatim se funkcija warpPerspective koristi za primjenu te matrice na izvornu sliku, čime se dobiva transformirana slika.

Contours, ili konture, su bitan koncept u obradi slika, koji omogućuje prepoznavanje i analizu oblika i granica objekata unutar slike. Konture se koriste za prepoznavanje objekata, njihovo segmentiranje, te za različite analize oblika. U OpenCV-u, konture se mogu pronaći pomoću funkcije findContours, dok se njihove karakteristike, poput površine, mogu analizirati pomoću funkcije contourArea.

Konture su linije koje spajaju sve kontinuirane točke (piksela) koje imaju iste vrijednosti intenziteta. One pomažu u identificiranju oblika objekata u binarnoj slici, gdje su objekti predstavljeni bijelim pikselima, a pozadina crnim.

Hough transform, ili houghova transformacija, je tehnika u računalnom vidu koja se koristi za detekciju oblika kao što su linije i krugovi unutar slike. Ova tehnika omogućuje prepoznavanje oblika čak i u prisutnosti šuma. Houghova transformacija temelji se na ideji da se oblici mogu prepoznati transformiranjem točaka u slike u parametarski prostor, gdje se traže lokalni maksimumi koji predstavljaju prisutnost oblika.

Standardna Houghova transformacija (SHT) koristi sve točke u slici za akumulaciju podataka u parametarskom prostoru, dok progresivna probabilistička Houghova transformacija (PPHT) koristi samo dio točaka, što značajno smanjuje računalnu složenost bez gubitka preciznosti.

Erode, ili erozija, je tehnika obrade slika koja se koristi za smanjenje veličine objekata u binarnoj slici. Ova operacija uklanja piksele na rubovima objekata, čineći ih manjima i eliminirajući sitne, izolirane šumove.

Dilate, ili dilatacija, je tehnika obrade slika koja povećava objekte u binarnoj slici dodavanjem piksela na njihove rubove, što može popuniti sitne praznine. Ova operacija koristi strukturni element, poput kvadratnog, kružnog ili križnog oblika, koji određuje područje koje se ispituje, zamjenjujući svaki piksel maksimalnom vrijednošću piksela iz njegove okolice.

5.3.Algoritam za pred obradu slike

Algoritam obrade slike sudoku polja koristi se za izdvajanje i pripremu područja interesa (ROI - Region of Interest) sa slike, kako bi se olakšalo prepoznavanje brojeva. Ovaj proces se provodi pomoću biblioteke OpenCV, a sastoji se od nekoliko važnih faza, počevši s učitavanjem slike pa sve do uklanjanja suvišnih linija s konačne slike.

Prvi korak uključuje učitavanje slike pomoću URI-ja koji se pretvara u putanju slike. Slika se zatim učitava pomoću metode Imgcodecs.imread. Ako učitavanje slike ne uspije, vraća se poruka o grešci putem povratnog poziva, čime se signalizira neuspjeh.

```
1   Mat img = Imgcodecs.imread(imagePath);
2
3   if (img.empty()) {
4       Log.e("RNOpenCvLibrary", "Failed to load image.");
5       callback.invoke("Failed to load image.");
6       return;
7   }
```

Kod 5.3.1 Učitavanje slike

Nakon uspješnog učitavanja slike, potrebno ju je pretvoriti u crno-bijeli format (grayscale), kako bi se olakšala daljnja obrada. To se postiže korištenjem metode Imgproc.cvtColor. Kako bi se smanjio šum na slici, primjenjuje se Gaussovo zamućenje (Gaussian Blur). Ova tehnika zamućuje sliku, smanjujući utjecaj sitnih detalja i pomaže u detekciji važnijih elemenata. Nakon toga, koristi se adaptivna binarna metoda određivanja praga (adaptive thresholding) kako bi se izdvojile jasne granice na slici, čime se priprema za prepoznavanje kontura.

```
1   Mat imgGrayed = new Mat();
2   Imgproc.cvtColor(img, imgGrayed, Imgproc.COLOR_BGR2GRAY);
3   Mat imgBlured = new Mat();
4   Size size = new Size(5, 5);
5   Imgproc.GaussianBlur(imgGrayed, imgBlured, size, 1);
6
7   Mat imgTreshed = new Mat();
8   Imgproc.adaptiveThreshold(imgBlured, imgTreshed, 255,
9    Imgproc.ADAPTIVE_THRESH_GAUSSIAN_C,
10          Imgproc.THRESH_BINARY_INV, 13, 3);
11
```

Kod 5.3.2 Obrada slike

S pripremljenom slikom, sljedeći korak je detekcija kontura pomoću metode Imgproc.findContours, koja identificira sve rubove i oblike na slici. Nakon toga, koristeći pomoćnu funkciju biggestContour, pretražuje se lista pronađenih kontura kako bi se odabrala najveća, koja predstavlja rub Sudoku polja.

```
List<MatOfPoint> contours = new ArrayList<>();
Mat hierarchy = new Mat();
Imgproc.findContours(imgTreshed, contours, hierarchy,
 Imgproc.RETR_EXTERNAL, Imgproc.CHAIN_APPROX_SIMPLE);

MatOfPoint biggestContour = biggestContour(contours);
```

Kod 5.3.3 Poziv funkcije biggestContour

Pomoćna metoda biggestContour provjerava svaku konturu, izračunava njezinu površinu, te procjenjuje odgovara li uvjetima za sudoku polje. Kontura mora imati četiri ruba i dovoljno veliku površinu (više od 50 piksela). Ako je kontura dovoljno velika i odgovara ovim uvjetima, ona se vraća, inače se vraća null.

```
private MatOfPoint biggestContour(List<MatOfPoint> contours) {
    MatOfPoint biggest = null;
    double maxArea = 0;

    for (MatOfPoint contour : contours) {
        double area = Imgproc.contourArea(contour);
        double perimeter = Imgproc.arcLength(new MatOfPoint2f
        (contour.toArray()), true);

        if (area > 50 && perimeter > 0) {
            MatOfPoint2f approx = new MatOfPoint2f();
            Imgproc.approxPolyDP(new MatOfPoint2f(contour.toArray()),
             approx, 0.02 * perimeter, true);
            int vertices = approx.toList().size();

            if (area > maxArea && vertices == 4) {
                biggest = contour;
                maxArea = area;
            }
        }
    }
    return biggest;
}
```

Kod 5.3.4 Metoda biggestContour

Ako je pronađena najveća kontura, primjenjuje se perspektivna transformacija kako bi se izdvojilo područje interesa (ROI). Korištenjem reorganiziranih točaka konture, slika se izravnava i prilagođava željenom obliku, što omogućuje da se sudoku polje izdvaja i pravilno poravna. Slika se zatim prilagođava dimenzijama sudoku polja pomoću metode Imgproc.resize.

```
Mat imgWarped = findRoi(imgTreshed, Arrays.asList(biggestContour),
 imgTreshed.cols(), imgTreshed.rows());
Imgproc.resize(imgWarped, imgWarped, new Size(100 * sudokuDimensions,
 100 * sudokuDimensions));
```

Kod 5.3.5 poziv metode findRoi

Pomoćna metoda findRoi koristi perspektivnu transformaciju kako bi preoblikovala konturu u pravokutni format. Ova metoda uzima ulazne i odredišne točke te pomoću metode Imgproc.getPerspectiveTransform vrši transformaciju slike.

```
private Mat findRoi(Mat img, List<MatOfPoint> contours,
 int widthImg, int heightImg) {
    MatOfPoint biggest = biggestContour(contours);
    if (biggest != null) {
        MatOfPoint2f biggestPoints = reorder(new MatOfPoint2f(
            biggest.toArray()));
        MatOfPoint2f roiCorners1 = biggestPoints;
        MatOfPoint2f roiCorners2 = new MatOfPoint2f(
            new Point(0, 0), new Point(widthImg, 0),
                new Point(0, heightImg), new Point(
                    widthImg, heightImg));
        Mat imgCroppedSudoku = Imgproc.getPerspectiveTransform(
            roiCorners1, roiCorners2);
        Mat imgWarped = new Mat();
        Imgproc.warpPerspective(img, imgWarped, imgCroppedSudoku,
         new Size(widthImg, heightImg));
        return imgWarped;
    } else {
        return null;
    }
}
```

Kod 5.3.6 Metoda findRoi

Nakon što je slika prilagođena poziva se pomoćna metoda removeLines koja detektira linije koristeći Imgproc.HoughLinesP, a zatim crta crne linije preko detektiranih elemenata, čime se efektivno uklanjaju iz slike.

```
private Mat removeLines(Mat imgWarped, int sudokuDimensions) {
    Mat lines = new Mat();
    int minLineLength = (int) (Math.min(imgWarped.cols(),
     imgWarped.rows()) / sudokuDimensions * 0.8);

    Imgproc.HoughLinesP(imgWarped, lines, 1, Math.PI / 180, 50,
     minLineLength, 10);

    for (int x = 0; x < lines.rows(); x++) {
        double[] vec = lines.get(x, 0);
        Point start = new Point(vec[0], vec[1]);
        Point end = new Point(vec[2], vec[3]);
        Imgproc.line(imgWarped, start, end, new Scalar(0, 0, 0),
         3);
    }

    return imgWarped;
}
```

Kod 5.3.*7* Metoda removeLines

Konačno, nakon obrade slike, ona se pretvara u bitmap format i sprema u predmemoriju aplikacije. Putanja spremljene slike vraća se putem povratnog poziva, čime se omogućuje daljnja upotreba slike.

```
Bitmap resultBitmap = Bitmap.createBitmap(imgWarped.cols(),
 imgWarped.rows(), Bitmap.Config.ARGB_8888);
Utils.matToBitmap(imgWarped, resultBitmap);

File outputDir = getReactApplicationContext().getCacheDir();
String processedImageFileName = "processed_image" + ".jpg";
File processedImageFile = new File(outputDir,
 processedImageFileName);
processedImageFile.createNewFile();

FileOutputStream outputStream =
 new FileOutputStream(processedImageFile);
resultBitmap.compress(Bitmap.CompressFormat.JPEG, 100,
 outputStream);
outputStream.close();

callback.invoke(Uri.fromFile(processedImageFile).toString());
```

Kod 5.3.8 Spremanje slike

5.4.Podjela slike na pojedinačna polja

U procesu podjele slike sudoku polja na individualna polja, koristi se metoda splitImageIntoBoxes, koja je zadužena za razdvajanje slike sudoku polja na pojedinačne ćelije (box-ove). Ova metoda prima putanju slike, dimenzije sudoku polja te povratni poziv (callback) kojim se vraća rezultat obrade. Algoritam koristi OpenCV biblioteku za manipulaciju slikom.

Prvi korak je učitavanje slike sudoku ploče s navedene putanje. Slika se učitava pomoću metode Imgcodecs.imread, a u slučaju neuspjeha učitavanja, ispisuje se poruka o grešci te se povratni poziv koristi za vraćanje informacije o neuspjehu. Nakon uspješnog učitavanja, slika se prosljeđuje funkciji splitInBoxes, koja je zadužena za stvarnu podjelu slike na individualna polja.

```
1   Mat img = Imgcodecs.imread(imagePath);
2
3   if (img.empty()) {
4       Log.e("RNOpenCvLibrary", "Failed to load image.");
5       callback.invoke("Failed to load image.");
6       return;
7   }
```

Kod 5.4.1 Učitavanje slike

Pomoćna metoda splitInBoxes prihvaća sliku i dimenzije sudoku polja kao parametre te vraća listu objekata tipa Mat, gdje svaki element predstavlja jednu ćeliju sudoku polja. Proces podjele se temelji na broju redaka i stupaca, koji su definirani dimenzijama sudoku polja. Za svaki redak i stupac, metoda izračunava pravokutnik (ROI - Region of Interest) koji predstavlja granice pojedinačnog polja na slici. Zatim se pomoću tih pravokutnika izdvajaju polja iz originalne slike koristeći metodu clone(). Ova metoda osigurava da se svako polje kopira kao zasebni objekt, čime se sprječava bilo kakva modifikacija originalne slike.

```
1   private List<Mat> splitInBoxes(Mat img, int sudokuDimensions) {
2       int rows = sudokuDimensions;
3       int cols = sudokuDimensions;
4       int boxWidthHeight = img.cols() / cols;
5       List<Mat> boxes = new ArrayList<>();
6
7       for (int i = 0; i < rows; i++) {
8           for (int j = 0; j < cols; j++) {
9               Rect roi = new Rect(j * boxWidthHeight, i *
10               boxWidthHeight, boxWidthHeight, boxWidthHeight);
11              Mat box = new Mat(img, roi).clone();
12              boxes.add(box);
13          }
14      }
15
16      return boxes;
17  }
18
```

Kod 5.4.2 Metoda splitInBoxes

Nakon što su sva polja izdvojena, svako polje se dodatno obrađuje kako bi bilo spremno za daljnju analizu. Prvo se na svako polje primjenjuju morfološke transformacije, erozije i dilatacije, čime se poboljšava kvaliteta slike. Ove transformacije se koriste za uklanjanje šuma i jačanje rubova na slici, što omogućava jasniju obradu brojeva u kasnijim fazama. Nakon primjene morfoloških operacija, svako pojedinačno polje se skalira na veličinu od 50x50 piksela. Ova fiksna dimenzija je odabrana kako bi slika svakog polja bila konzistentna.

```
1   Mat kernel2 = Imgproc.getStructuringElement(Imgproc.MORPH_RECT,
2    new Size(3, 3));
3   Imgproc.erode(box, box, kernel2);
4   Imgproc.dilate(box, box, kernel2);
5   Imgproc.resize(box, box, new Size(50, 50));
6
```

Kod 5.4.3 Transformacije slike

Kada su polja prilagođena, svako polje se pretvara u Bitmap objekt te se sprema kao JPEG datoteka u privremenu mapu unutar aplikacije. Na ovaj način se osigurava da su sva polja spremljena kao zasebne slike, koje se mogu koristiti za daljnju analizu. Putanja do svake slike se sprema u listu boxUris, koja se kasnije vraća putem povratnog poziva.

```
Bitmap boxBitmap = Bitmap.createBitmap(box.cols(), box.rows(),
 Bitmap.Config.ARGB_8888);
Utils.matToBitmap(box, boxBitmap);

String boxFileName = "box_" + i + ".jpg";
File boxFile = new File(cacheDir, boxFileName);
boxFile.createNewFile();

FileOutputStream outputStream = new FileOutputStream(boxFile);
boxBitmap.compress(Bitmap.CompressFormat.JPEG, 100, outputStream);
outputStream.close();

boxUris.add(Uri.fromFile(boxFile).toString());

WritableArray uriArray = Arguments.fromList(boxUris);
callback.invoke(uriArray);


```

Kod 5.4.4 Spremanje slika u listu

5.5.Algoritam za rješavanje sudoku polja

Metoda solveSudoku koristi se za rekurzivno rješavanje sudoku zagonetke. Ova metoda prima matricu sudoku polja i dimenzije polja, te pokušava pronaći rješenje koristeći metode pretraživanja praznih polja i ispitivanja valjanosti poteza. Cijeli proces je strukturiran kroz tri glavne funkcije koje zajednički omogućuju učinkovito rješavanje problema.

Na početku, metoda solveSudoku traži prazno polje u sudoku matrici. Ovo se postiže pomoću pomoćne funkcije findEmpty, koja prolazi kroz matricu i vraća koordinate prvog praznog polja. Prazno polje je definirano kao ono koje ima vrijednost 0. Ako nema praznih polja, to znači da je sudoku već riješen, te metoda vraća kopiju trenutne matrice.

```
1   function solveSudoku(matrixToSolve, size) {
2         const empty = findEmpty(matrixToSolve, size);
3         if (!empty) {
4           return matrixToSolve.map((row) => [...row]);
5         } else {
6           const [row, col] = empty;
```

Kod 5.5.1 Funkcija solveSudoku

Ako su pronađena prazna polja, funkcija prolazi kroz sve moguće vrijednosti od 1 do veličine sudoku polja. Za svaki broj, provjerava može li se taj broj valjano postaviti na trenutnu poziciju koristeći funkciju isValidMove. Ova funkcija osigurava da broj nije već prisutan u istom retku, stupcu ili u malom kvadratu kojem pripada trenutno polje.

```
1   for (let num = 1; num <= size; num++) {
2         if (isValidMove(matrixToSolve, num, row, col, size)) {
3           const matrixCopy = matrixToSolve.map((row) => [...row]);
4           matrixCopy[row][col] = num;
5
```

Kod 5.5.2 Petlja za ispitivanje mogućih brojeva

Kada se pronađe valjan potez, funkcija stvara kopiju trenutne matrice kako bi osigurala da originalna matrica ostane nepromijenjena. Nakon postavljanja broja na trenutno polje, solveSudoku rekurzivno poziva samu sebe, koristeći novu kopiju matrice. Ako rekurzivni poziv pronađe rješenje, to rješenje se vraća. Ako se ne može naći rješenje, funkcija se ponovno poziva s drugim brojem na istom polju. Ovaj postupak se ponavlja sve dok se ne pronađe rješenje za cijelo sudoku polje, ili dok se ne utvrdi da rješenja nema.

```
            const solution = solveSudoku(matrixCopy, size);
            if (solution) {
              return solution;
            }
          }
        }
      }
      return null;
    }
```

Kod 5.5.3 Rekurzivni poziv funkcije solveSudoku

Funkcija isValidMove provjerava je li određeni broj legalan potez na zadanom polju. Funkcija prolazi kroz redak i stupac kako bi provjerila je li broj već prisutan. Također provjerava maleni kvadrat na koji polje pripada, dijeleći sudoku polje na podmatrice koje predstavljaju kvadratiće unutar sudoku ploče. Ako se broj već nalazi u istom kvadratu, redku ili stupcu, potez se smatra nevažećim.

```
function isValidMove(matrixToSolve, num, row, col, size) {
      for (let i = 0; i < size; i++) {
        if (matrixToSolve[row][i] == num ||
         matrixToSolve[i][col] == num) {
          return false;
        }
      }

      const boxSizeX = Math.ceil(Math.sqrt(size));
      const boxSizeY = Math.floor(Math.sqrt(size));
      const boxX = Math.floor(col / boxSizeX) * boxSizeX;
      const boxY = Math.floor(row / boxSizeY) * boxSizeY;

      for (let i = 0; i < boxSizeY; i++) {
        for (let j = 0; j < boxSizeX; j++) {
          if (matrixToSolve[boxY + i][boxX + j] == num) {
            return false;
          }
        }
      }
      return true;
    }
```

Kod 5.5.4 Funkcija isValidMove

Funkcija findEmpty prolazi kroz cijelu matricu kako bi pronašla prvo prazno polje. Kada pronađe polje s vrijednošću 0, vraća njegove koordinate u obliku niza [i, j]. Ako nema praznih polja, funkcija vraća null, što signalizira da je sudoku polje riješeno.

```
1   function findEmpty(matrixToSolve, size) {
2         for (let i = 0; i < size; i++) {
3           for (let j = 0; j < size; j++) {
4             if (matrixToSolve[i][j] == 0) {
5               return [i, j];
6             }
7           }
8         }
9         return null;
10      }
11
```

Kod 5.5.5 Funkcija findEmpty

Ove tri funkcije zajedno čine osnovu rekurzivnog algoritma za rješavanje sudoku zagonetke. solveSudoku iterira kroz sve mogućnosti, koristeći provjere iz funkcije isValidMove, dok findEmpty osigurava da funkcija traži prazne pozicije na sudoku ploči sve dok se ne pronađe rješenje.

5.6.Prikaz riješenog polja

Metoda drawSolutions omogućuje prikaz riješenog Sudoku polja na slici. Proces prikaza sastoji se od nekoliko ključnih koraka, koji uključuju učitavanje slike, crtanje brojeva ili slova na odgovarajućim pozicijama na slici, te spremanje modificirane slike u privremeni direktorij aplikacije.

Prvo, URI slike se parsira u objekt tipa Uri kako bi se dohvatila stvarna putanja slike. Slika se zatim učitava pomoću OpenCV metode Imgcodecs.imread. U slučaju da slika ne može biti učitana, metoda bilježi poruku o grešci i vraća je putem povratnog poziva.

```
1   Mat img = Imgcodecs.imread(imagePath);
2
3   if (img.empty()) {
4       Log.e("RNOpenCvLibrary", "Failed to load image.");
5       callback.invoke("Failed to load image.");
6       return;
7   }
8
```

Kod 5.6.1 Učitavanje slike

Nakon uspješnog učitavanja slike, izračunava se širina i visina svake ćelije sudoku polja dijeljenjem širine slike s dimenzijama polja. Ovi podaci omogućuju pravilno određivanje koordinata za ispis riješenih vrijednosti unutar svake ćelije. Zatim se koristi dvostruka petlja kako bi se iteriralo kroz svaku ćeliju Sudoku polja. Za svaku ćeliju dohvaća se odgovarajuća vrijednost iz niza solvedSudoku. Ako je vrijednost različita od nule, metoda izračunava koordinate na kojima će broj ili slovo biti ispisano unutar te ćelije.

```
1   int boxWidthHeight = img.cols() / sudokuDimensions;
2
3   for (int i = 0; i < sudokuDimensions; i++) {
4           for (int j = 0; j < sudokuDimensions; j++) {
5               int value = solvedSudoku.getInt(i * sudokuDimensions + j);
6
7               if (value != 0) {
8                   int x = j * boxWidthHeight + boxWidthHeight / 3;
9                   int y = (int) (i * boxWidthHeight + boxWidthHeight / 1.3);
10                  value = displayValue(value);
11
12
```

Kod 5.6.2 Izračunavanje koordinata

Pomoćna metoda displayValue provjerava je li vrijednost između 10 i 16. Ako jest, vrijednost se konvertira u odgovarajuće slovo ('A' do 'G'), dok se ostale vrijednosti (brojevi) zadržavaju nepromijenjene. Ovaj mehanizam osigurava pravilno prikazivanje brojeva i slova u Sudoku polju.

```
1   private String displayValue(int value) {
2         if (value >= 10 && value < 16) {
3             return String.valueOf((char) ('A' + (value - 10)));
4         }
5         return String.valueOf(value);
6     }
7
```

Kod 5.6.3 Metoda displayValue

Nakon što je vrijednost konvertirana, koristi se metoda Imgproc.putText za ispisivanje odgovarajućeg teksta na slici. Tekst se ispisuje pomoću fonta FONT_HERSHEY_SIMPLEX, zelene boje (new Scalar(0, 255, 0)) i veličine 2, što osigurava da su brojevi ili slova vidljivi i čitljivi.

```
1   Imgproc.putText(img, String.valueOf(value), new Point(x, y),
2       Imgproc.FONT_HERSHEY_SIMPLEX, 2, new Scalar(0, 255, 0), 2);
```

Kod 5.6.3 Ispisivanje teksta na slici

Nakon što su svi brojevi ili slova iscrtani na slici, slika se konvertira u Bitmap objekt pomoću metode Utils.matToBitmap. Zadnji korak uključuje spremanje modificirane slike kao JPEG datoteke u privremeni direktorij aplikacije. Datoteka se sprema s odgovarajućim imenom, a URI slike se zatim vraća putem povratnog poziva, čime je proces prikazivanja riješenog Sudoku polja završen.

```
1   Bitmap resultBitmap = Bitmap.createBitmap(img.cols(), img.rows(),
2    Bitmap.Config.ARGB_8888);
3   Utils.matToBitmap(img, resultBitmap);
4   File outputDir = getReactApplicationContext().getCacheDir();
5   String modifiedImageFileName = "modified_image_" + ".jpg";
6   File modifiedImageFile = new File(outputDir, modifiedImageFileName);
7   modifiedImageFile.createNewFile();
8
9   FileOutputStream outputStream = new FileOutputStream(modifiedImageFile);
10  resultBitmap.compress(Bitmap.CompressFormat.JPEG, 100, outputStream);
11  outputStream.close();
12
13  callback.invoke(Uri.fromFile(modifiedImageFile).toString());
14
15
```

Kod 5.6.3 Spremanje riješenog polja

6. PREPOZNAVANJE ZNAKOVA IZ SLIKE

Kako bi aplikacija funkcionirala bilo je potrebno razviti CNN model za prepoznavanje znakova iz slike. Ukupno imamo 17 različitih znakova koje treba prepoznati, to su brojevi od 1-9 te znakovi od A-G, te prazna polja i koja će biti zapisana kao znak 0. Korišten je Python programski jezik, zajedno s nekoliko važnih biblioteka kao što su OpenCV (cv2), NumPy, Matplotlib, scikit-learn i Keras. Nakon što je model istreniran i spremljen kao ".h5" datoteka, korišten je alat tensorflowjs_converter da bismo napravili pretvorbu modela čineći ga kompatibilnim s React Native aplikacijom koristeći TensorFlow.js

6.1.Prikupljanje slika za treniranje

Kako bi model radio što preciznije odvojene su konkretne slike koje su bile obrađene na našem uređaju. Slike su prvobitno bile dimenzija 100x100. Sakupljeno je 200-tinjak slika za svaki pojedini znak su pohranjene u direktoriju data. Zatim su preimenovane slike te ih premještene u drugi folder kako bi bile sortirane.

Slika 6.1.1 Prikaz četiri sakupljene nasumične slike

6.2.Augmentiranje i obrada podataka

Kako bi povećali raznolikost skupa podataka i osigurali bolje performanse modela za prepoznavanje znakova, primijenjena je tehnika augmentacije podataka. Augmentacija omogućava modelu da postane otporniji na varijacije u podacima tako što se iz svake izvorne slike generira nekoliko varijacija. U ovom procesu primijenjene su transformacije kao što su rotacija, promjena svjetline, kontrasta te pomicanje svake slike. Tehnika augmentacije je ključna jer model uči prepoznavati znakove i u uvjetima gdje nisu savršeno poravnati ili imaju različite osvjetljenje i kontrast.

Prvi korak u procesu augmentacije je definiranje parametara za rotaciju, svjetlinu, kontrast i pomicanje slika, te priprema direktorija za pohranu rezultata augmentacije. Slike se učitavaju iz izvornog direktorija, a za svaku klasu znakova kreira se odgovarajući direktorij u kojem će se pohraniti augmentirane slike.

```
1   rotationRange = 2
2   brightnessRange = 0.09
3   contrastRange = 0.09
4   positionShiftRange = 6
5
6   sourceDir = "orderedImages"
7   destinationDir = "augmentedImages"
8
9   augmentationFactor = 3
10  os.makedirs(destinationDir, exist_ok=True)
11
12  for classFolder in os.listdir(sourceDir):
13      classFolderPath = os.path.join(sourceDir, classFolder)
14      destinationClassDir = os.path.join(destinationDir, classFolder)
15      os.makedirs(destinationClassDir, exist_ok=True)
16
17      imageFiles = [f for f in os.listdir(classFolderPath)
18       if f.endswith(".jpg")]
19
```

Kod 6.2.1 Definiranje parametara augmentacije

Nakon učitavanja slika, svaka slika prolazi kroz niz transformacija. Rotacija omogućava modelu da prepozna znakove čak i kada su blago nagnuti, dok promjena svjetline i kontrasta simulira različite uvjete osvjetljenja. Pomicanje slike stvara dodatnu varijabilnost u položaju znaka unutar slike, što pomaže modelu da nauči prepoznati znakove koji nisu strogo centrirani. Svaka slika se transformira nekoliko puta kako bi se dobile različite varijacije, a svaka nova verzija se pohranjuje u odgovarajući direktorij.

```
for imageFile in imageFiles:
    srcPath = os.path.join(classFolderPath, imageFile)
    image = cv2.imread(srcPath, cv2.IMREAD_GRAYSCALE)

    for i in range(1, augmentationFactor + 1):
        angle = random.uniform(-rotationRange, rotationRange)
        M = cv2.getRotationMatrix2D((image.shape[1] / 2,
         image.shape[0] / 2), angle, 1)
        image = cv2.warpAffine(image, M, (image.shape[1],
         image.shape[0]))

        brightness = 1 + random.uniform(-brightnessRange,
         brightnessRange)
        contrast = 1 + random.uniform(-contrastRange,
         contrastRange)
        image = cv2.convertScaleAbs(image, alpha=brightness,
         beta=0)
        image = cv2.convertScaleAbs(image, alpha=contrast,
         beta=0)

        shiftX = random.randint(-positionShiftRange,
         positionShiftRange)
        shiftY = random.randint(-positionShiftRange,
         positionShiftRange)
        M = np.float32([[1, 0, shiftX], [0, 1, shiftY]])
        image = cv2.warpAffine(image, M, (image.shape[1],
         image.shape[0]))

        augmentedFilename =
         f"{imageFile.replace('.jpg', '')}Aug{i}.jpg"
        dstPath = os.path.join(destinationClassDir,
         augmentedFilename)
        cv2.imwrite(dstPath, image)

```

Kod 6.2.2 Primjena augmentacije

Nakon augmentacije, sve slike trebaju biti uniformne u smislu dimenzija i formata. Kako bi se osiguralo da su sve slike pravilno skalirane i u crno-bijelom formatu, implementiran je postupak obrade slika. Svaka slika se smanjuje na dimenzije 50x50 piksela, a sve boje se konvertiraju u grayscale kako bi se model fokusirao isključivo na prepoznavanje oblika znakova, bez distrakcija uzrokovanih bojama. Obrada se primjenjuje na slike iz oba skupa – izvornih i augmentiranih slika, te se obrađene slike pohranjuju u zasebni direktorij.

```
def process_image(src_path, dst_path, size=(50, 50)):
    image = cv2.imread(src_path, cv2.IMREAD_GRAYSCALE)
    image = cv2.resize(image, size)
    cv2.imwrite(dst_path, image)

for class_folder in os.listdir(ordered_images_dir):
    for image_file in os.listdir(os.path.join(ordered_images_dir,
     class_folder)):
        src_path = os.path.join(ordered_images_dir, class_folder,
         image_file)
        dst_path = os.path.join(processed_data_dir, class_folder,
         image_file)
        process_image(src_path, dst_path)

for class_folder in os.listdir(augmented_images_dir):
    for image_file in os.listdir(os.path.join(augmented_images_dir,
     class_folder)):
        src_path = os.path.join(augmented_images_dir, class_folder,
         image_file)
        dst_path = os.path.join(processed_data_dir, class_folder,
         image_file)
        process_image(src_path, dst_path)

```

Kod 6.2.3 Obrada slike i sortiranje

6.3.Podešavanje i treniranje modela

U ovom dijelu je prikazano kako se pripremaju podaci za treniranje modela, kako se postavlja arhitektura konvolucijske neuronske mreže (CNN) te kako se provodi treniranje.

U nastavku je prikazan kod koji učitava slike iz direktorija ProcessedData, normalizira ih dijeljenjem s 255 (kako bi sve vrijednosti bile između 0 i 1), te postavlja oznake (label) za svaku sliku prema klasi kojoj pripada.

```
images = []
labels = []
label_mapping = {'0': 0,'1': 1, '2': 2, '3': 3, '4': 4,
 '5': 5, '6': 6, '7': 7, '8': 8, '9': 9, 'A': 10,
  'B': 11, 'C': 12, 'D': 13, 'E': 14, 'F': 15, 'G': 16}

path = 'ProcessedData/'

dir_list = os.listdir(path)
for i in dir_list:
    dir = os.path.join(path, i)
    file_list = os.listdir(dir)
    for j in file_list:
        files = os.path.join(dir, j)
        img = cv2.imread(files, cv2.IMREAD_GRAYSCALE)
        img = np.array(img, dtype=np.float32)
        img = img / 255  # Normalizacija
        images.append(img)
        labels.append(i)

```

Kod 6.3.1 Učitavanje i priprema podataka

U ovom dijelu koda nasumično se odabire slika iz skupa podataka te se prikazuje zajedno s njezinom oznakom. Nakon toga, slike i oznake se pripremaju za treniranje modela. Funkcije LabelEncoder, shuffle, i train_test_split koriste se za pripremu podataka.

```
random_index = random.randint(0, len(images) - 1)

plt.imshow(images[random_index], cmap='gray')
plt.title(f'Label: {labels[random_index]}')
plt.show()

X = np.array(images)
y = [label_mapping[label] for label in labels]  # Mapiranje oznaka

le = LabelEncoder()
y = le.fit_transform(y)
X, y = shuffle(X, y, random_state=42)
X_train, X_test, y_train, y_test = train_test_split(X, y,
 test_size=0.2, random_state=55)

```

Kod 6.3.2 Prikaz slike podjela na trening i test skup

Izabralan je sekvencijalni model iz Keras biblioteke. Model se sastoji od pet konvolucijskih slojeva koji izvlače prostorne karakteristike iz slika. Svaki konvolucijski sloj ima jezgru veličine 3x3 i koristi ReLU aktivacijsku funkciju koja pojačava važne karakteristike i suzbija redundantne informacije. Max pooling slojevi se ubacuju između konvolucijskih slojeva kako bi se smanjila prostorna dimenzija podataka i poboljšala učinkovitost treniranja. Oni uzimaju maksimalnu vrijednost iz određenog susjedstva unutar slike, čime se postiže translacijska invarijantnost, sposobnost modela da prepozna znak bez obzira na njegov položaj unutar slike. Batch normalization slojevi su umetnuti nakon svake konvolucije. Oni poboljšavaju stabilnost i brzinu konvergencije treninga model. Nakon konvolucijskih slojeva, slijede dva dense sloja koji obavljaju klasifikaciju. Prvi sloj ima 512 neurona, a drugi 256 neurona. Oba koriste ReLU aktivaciju. Uključen je i Dropout sloj sa vjerojatnošću odustajanja od 50% neurona tijekom treninga. Ova tehnika regulacije pomaže spriječiti prekomjerno uklapanje i omogućava modelu da nauči robusnije karakteristike. Izlazni sloj ima broj neurona jednak broju klasa znakova (17 uključujući i prazno polje) i koristi Softmax aktivacijsku funkciju. Softmax funkcija daje vjerojatnosti za svaku klasu, a klasa sa najvećom vjerojatnoćom se smatra prepoznatom. Model se kompajlira uz pomoć Adam optimizatora. Izbor CNN arhitekture i parametara je odabran kako bi se postigla izuzetno visoka preciznost uzimajući u obzir i prednost akceleracije GPU-om.

```
model = Sequential()
model.add(Conv2D(32, (3, 3), activation='relu',
 input_shape=(100, 100, 1)))
model.add(BatchNormalization())
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(256, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))
model.add(Flatten())
model.add(Dense(512, activation='relu'))
model.add(BatchNormalization())
model.add(Dense(256, activation='relu'))
model.add(BatchNormalization())
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))

model.compile(optimizer='adam', loss='sparse_categorical_crossentropy',
 metrics=['accuracy'])

```

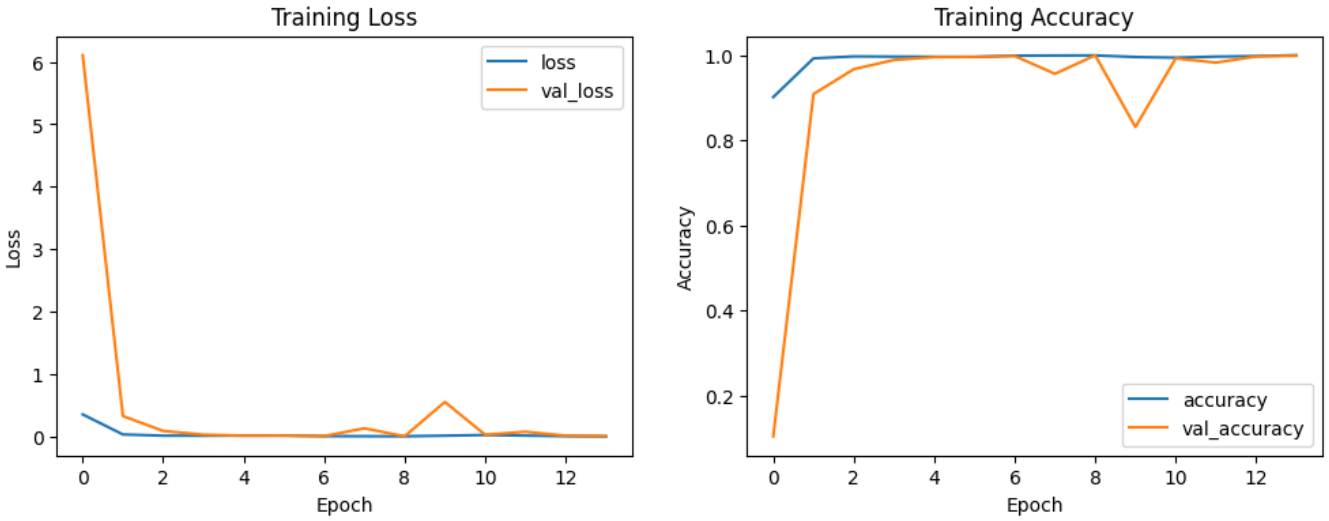
Kod 6.3.3 Definiranje CNN modela

Model se trenira na skupu podataka s podjelom na trening i validacijski skup, gdje je 20% podataka korišteno za validaciju tijekom treniranja. Također je implementirana tehnika EarlyStopping koja zaustavlja treniranje kada se model prestane poboljšavati. Nakon treniranja, model se evaluira na test skupu, te se sprema za kasniju upotrebu.

```
history = model.fit(
    X_train,
    y_train,
    validation_split=0.2,
    batch_size=25,
    epochs=100,
    callbacks=[early_stopping]
)

test_loss, test_accuracy = model.evaluate(X_test, y_test)
print(f'Test Loss: {test_loss}, Test Accuracy: {test_accuracy}')

model.save('ModelOCR.h5')

```

Kod 6.3.4 Treniranje modela i spremanje

Analizom grafa, možemo zaključiti da CNN model uspješno trenira i postiže izuzeto visoku preciznost u prepoznavanju znakova. Brzo smanjenje gubitka i porast preciznosti na trening i validacijskom skupu potvrđuju sposobnost modela da efikasno uči iz podataka. Primjećujemo blago pogoršanje validacijskog gubitka i preciznosti nakon određenog broja epoha, što sugerira moguće prekomjerno prilagođavanje modela. Implementacija "EarlyStopping" funkcije, koja automatski zaustavlja učenje kada se prestane poboljšavati validacijski gubitak nam pomaže u sprječavanju ovog problema.



Slika 6.3.1 Graf treniranja modela

7. ZAKLJUČAK

U ovom radu prikaza postupak integracije OpenCV biblioteke u React Native Android aplikaciju za rješavanje sudoku zagonetki. Implementacija uključuje prepoznavanje sudoku polja, obradu slike pomoću OpenCV-a, te prepoznavanje znamenki korištenjem konvolucijske neuronske mreže (CNN) trenirane u Pythonu i implementirane u aplikaciji putem TensorFlow.js. Ova kombinacija tehnologija omogućuje automatsko prepoznavanje i rješavanje sudoku zagonetki na mobilnom uređaju. Kroz ovaj proces, od prepoznavanja sudoku polja na slici, preko obrade slike do prepoznavanja znamenki i konačnog rješavanja zagonetke, korisnicima se pruža intuitivno i efikasno rješenje. Opisani koraci osiguravaju visoku preciznost i brzinu rješavanja zagonetki, čineći aplikaciju korisnički prijateljskom. Integracija OpenCV-a u mobilnu aplikaciju pokazala se kao moćan alat za obradu slika i računalni vid, dok upotreba CNN modela za prepoznavanje znakova dodatno poboljšava točnost i funkcionalnost aplikacije. Ovaj rad demonstrira kako napredne tehnike obrade slika i strojnog učenja mogu biti uspješno integrirane u mobilne aplikacije za rješavanje stvarnih problema, poput rješavanja sudoku zagonetki.

LITERATURA

[1] Robinson Industries, Sudoku Solver, Google Play, 2024., dostupno na: https://play.google.com/store/apps/details?id=com.RobinsonIndustries.SudokuSolver&hl=hr&gl=US [Posljednje pristupljeno: 10.08.2024.]

[2] Schoolmate, Sudoku, Google Play, 2024., dostupno na: https://play.google.com/store/apps/details?id=com.schoolmate.sudoku&hl=hr&gl=US [Posljednje pristupljeno: 04.06.2024.]

[3] Michael Villeneuve, React Native Document Scanner, GitHub, 2024., dostupno na: https://github.com/Michaelvilleneuve/react-native-document-scanner [Posljednje pristupljeno: 10.08.2024.]

[4] OpenCV.org, OpenCV, n.d., dostupno na: https://opencv.org/ [Posljednje pristupljeno: 10.08.2024.]

[5] G. Bradski i A. Kaehler, Learning OpenCV: Computer Vision with the OpenCV Library, O'Reilly Media, 2008., dostupno na: https://www.bogotobogo.com/cplusplus/files/OReilly%20Learning%20OpenCV.pdf [Posljednje pristupljeno: 10.08.2024.]

[6] OpenCV.org, CUDA Platforms, n.d., dostupno na: https://opencv.org/platforms/cuda/ [Posljednje pristupljeno: 10.08.2024.]

[7] React Native, Introduction to React Native, n.d., dostupno na: https://reactnative.dev/ [Posljednje pristupljeno: 10.08.2024.]

[8] React Native, Introduction to React Native Components, n.d., dostupno na: https://reactnative.dev/docs/intro-react-native-components [Posljednje pristupljeno: 10.08.2024.]

[9] React Native, Introducing Hot Reloading, 2016., dostupno na: https://reactnative.dev/blog/2016/03/24/introducing-hot-reloading [Posljednje pristupljeno: 10.08.2024.]

[10] React Native, Introduction to Native Modules, n.d., dostupno na: https://reactnative.dev/docs/native-modules-intro [Posljednje pristupljeno: 10.08.2024.] [11] https://en.wikipedia.org/wiki/JavaScript

[11] Wikipedia, JavaScript, 2024., dostupno na: https://en.wikipedia.org/wiki/JavaScript [Posljednje pristupljeno: 10.8.2024.]

[12] Mozilla Developer Network (MDN), JavaScript, n.d., dostupno na:

https://developer.mozilla.org/en-US/docs/Web/JavaScript [Posljednje pristupljeno: 10.08.2024.]

[13] W3Schools, TensorFlow Introduction, n.d., dostupno na: https://www.w3schools.com/ai/ai_tensorflow_intro.asp [Posljednje pristupljeno: 10.08.2024.]

[14] D. Smilkov, C. Nicholson, T. Snelgrove, H. Ree, N. Kreeger, i C. Mewald, TensorFlow.js: Machine Learning for the Web and Beyond, 2019., dostupno na: https://arxiv.org/abs/1901.05350 [Posljednje pristupljeno: 10.8.2024.]

[15] Wikipedia, Python (programming language), 2024., dostupno na: https://en.wikipedia.org/wiki/Python_(programming_language) [Posljednje pristupljeno: 10.8.2024.]

[16] Wikipedia, Python (programming language), 2024., dostupno na: https://en.wikipedia.org/wiki/Python_(programming_language) [Posljednje pristupljeno: 10.8.2024.]

[17] GeeksforGeeks, Libraries in Python, n.d., dostupno na: https://www.geeksforgeeks.org/libraries-in-python/ [Posljednje pristupljeno: 10.8.2024.]

[18] GeeksforGeeks, Differences Between Django vs Flask, n.d., dostupno na: https://www.geeksforgeeks.org/differences-between-django-vs-flask/ [Posljednje pristupljeno: 10.8.2024.]

[19] Brainhub.eu, "Integrating OpenCv in React Native," 2024., dostupno na: https://brainhub.eu/library/opencv-react-native-image-processing [Posljednje pristupljeno: 10.08.2024.]

[20] YouTube, "Integrating OpenCv in React Native," YouTube Video, objavio/la ELYL2fFi1SE, 2024., dostupno na: https://www.youtube.com/watch?v=ELYL2fFi1SE [Posljednje pristupljeno: 10.08.2024.]

[21] Brainhub, "Download and Insert OpenCV Script," GitHub Repository, dostupno na: https://github.com/brainhubeu/react-native-opencv-tutorial/blob/master/downloadAndInsertOpenCV.sh [Posljednje pristupljeno: 12.09.2024.]

[22] Brainhub, "React Native OpenCV Tutorial," GitHub Repository, dostupno na: https://github.com/brainhubeu/react-native-opencv-tutorial [Posljednje pristupljeno: 12.09.2024.]

[23] Google Play, "Sudoku Solve," dostupno na: https://play.google.com/store/apps/details?id=com.yoyodev.sudokusolve [Posljednje pristupljeno: 14.09.2024.]

SAŽETAK

Naslov: Integracija OpenCV biblioteke u React Native android aplikaciju

Sažetak: U ovom radu istražena je i implementirana integracija OpenCV biblioteke u React Native Android aplikaciju s ciljem automatizacije rješavanja sudoku zagonetki. Prikazan je cijeli postupak, počevši od preuzimanja i postavljanja OpenCV-a, preko obrade slike za prepoznavanje sudoku polja, do korištenja konvolucijske neuronske mreže (CNN) za prepoznavanje znamenki unutar polja. Razvijeni CNN model treniran je u Pythonu i integriran u aplikaciju putem TensorFlow.js, omogućujući brzu i točnu klasifikaciju znamenki. Aplikacija omogućava korisnicima da jednostavno slikaju sudoku zagonetku, prepoznaju znamenke i dobiju rješenje u stvarnom vremenu. Rezultati pokazuju visoku preciznost i efikasnost rješenja, što aplikaciju čini korisnički prijateljskom.

Ključne riječi: Konvolucijska neuronska mreža, Obrada slike, OpenCV, Računalni vid, React Native, Strojno učenje, Sudoku

ABSTRACT

Title: Integration of OpenCV library in React Native Android application

Summary: This work explores and implements the integration of the OpenCV library into a React Native Android application with the aim of automating the solving of Sudoku puzzles. The entire process is detailed, beginning with downloading and setting up OpenCV, followed by image processing to recognize Sudoku grids, and utilizing a Convolutional Neural Network (CNN) for recognizing digits within the grid. The developed CNN model was trained in Python and integrated into the application via TensorFlow.js, enabling fast and accurate digit classification. The application allows users to easily capture a Sudoku puzzle with their camera, recognize the digits, and obtain a solution in real-time. The results demonstrate high accuracy and efficiency, making the application user-friendly.

Keywords: Computer Vision, Convolutional Neural Network, Image Processing, Machine Learning, OpenCV, React Native, Sudoku

PRILOZI

Poveznica na GitLab repozitorij: https://gitlab.com/LuQchor/zavrsni-rad

Poveznica na GitLab branch gdje se nalazi apk:

https://gitlab.com/LuQchor/zavrsni-rad/-/tree/BuiltApk?ref_type=heads

Poveznica na google disk koji sadrži slike preko kojih je CNN model treniran:

https://drive.google.com/file/d/1AsLS8J2YIi4AuRpc_cdD4QF2VqLI2Tvs/view