Autentifikacija korisnika raspoznavanjem uzoraka na Android platformi

Kordić, Luka

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

Diplomski studij

AUTENTIFIKACIJA KORISNIKA RASPOZNAVANJEM UZORAKA NA ANDROID PLATFORMI

Diplomski rad

Luka Kordić

Osijek, 2019.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Obrazac D1: Obrazac za imenovanje Povjerenstva za obranu diplomskog rada

Osijek, 18.09.2019.

Odboru za završne i diplomske ispite

Imenovanje Povjerenstva za obranu diplomskog rada

Ime i prezime studenta:	Luka Kordić
Studij, smjer:	Diplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	D 869 R, 26.09.2018.
OIB studenta:	18273456504
Mentor:	Prof.dr.sc. Goran Martinović
Sumentor:	Dr.sc. Bruno Zorić
Sumentor iz tvrtke:	
Predsjednik Povjerenstva:	Izv. prof. dr. sc. Alfonzo Baumgartner
Član Povjerenstva:	Doc.dr.sc. Zdravko Krpić
Naslov diplomskog rada:	Autentifikacija korisnika raspoznavanjem uzoraka na Android platformi
Znanstvena grana rada:	**Programsko inženjerstvo (zn. polje računarstvo)**
Zadatak diplomskog rada:	U teorijskom dijelu rada potrebno je opisati moderne metode za raspoznavanje i autentifikaciju korisnika u računalnim sustavima. Poseban naglasak potrebno je staviti na metode zasnovane na postupcima strojnog učenja i raspoznavanja uzoraka. U praktičnom dijelu ostvariti aplikaciju za Android operacijski sustav koja na temelju potpisa fotografiranog pomoću uređaja dopušta korisniku pristup sadržaju. (sumentor: dr.sc. Bruno Zorić)
Prijedlog ocjene pismenog dijela ispita (diplomskog 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: 3 bod/boda Jasnoća pismenog izražavanja: 3 bod/boda Razina samostalnosti: 3 razina
Datum prijedloga ocjene mentora:	18.09.2019.

Potpis mentora za predaju konačne verzije rada u Studentsku službu pri završetku studija:	Potpis:
	Datum:

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

IZJAVA O ORIGINALNOSTI RADA

Osijek, 10.10.2019.

Ime i prezime studenta:	Luka Kordić
Studij:	Diplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	D 869 R, 26.09.2018.
Ephorus podudaranje [%]:	10

Ovom izjavom izjavljujem da je rad pod nazivom**: Autentifikacija korisnika raspoznavanjem uzoraka na Android platformi**

izrađen pod vodstvom mentora Prof.dr.sc. Goran Martinović

i sumentora Dr.sc. Bruno Zorić

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:

SADRŽAJ:

1. UVOD

Komunikacijske, informacijske i digitalne tehnologije vrlo se brzo razvijaju, a Internet je postao neizostavan dio svakodnevnog života. Često se ne pridodaje dovoljno pozornosti utjecajima koje ove tehnologije imaju na svoje korisnike, barem ne onim lošijim utjecajima. Svakodnevno se o korisnicima prikupljaju velike količine podataka, a u velikoj većini slučajeva se ti podaci skupljaju u svrhu nekog oblika identifikacije, poput onih u bankama, trgovinama, obrazovnim ustanovama, zdravstvenim ustanovama ili na društvenim mrežama. Ono o čemu se često ne razmišlja je način na koji se prikupljeni podaci koriste, koliko su oni sigurni, tko se zapravo brine o zaštiti identiteta ili tko je sve ovlašten za pristup tim podacima. Na razvoj tehnologije se uglavnom gleda kao na stvaranje naprednijeg i sigurnijeg okruženja, ali sa druge strane to ostavlja različite mogućnosti zlouporabe i neovlaštenog pristupa podacima te zadiranju u privatnost ljudi. Razvojem informacijskih i komunikacijskih tehnologija pojavljuje se i potreba za razvojem kompleksnijih i robusnijih načina sigurnosti u modernim računalnim sustavima. U ostvarivanju takvih sigurnosnih sustava biometrijske metode često imaju ključnu ulogu. Biometrija se može definirati kao skup metoda za jedinstveno prepoznavanje ljudi s obzirom na jednu ili više fizičkih ili ponašajnih osobina.

U teorijskom dijelu ovog diplomskog rada proučene su i opisane moderne metode za raspoznavanje i autentifikaciju korisnika u računalnim sustavima. Poseban naglasak stavljen je na metode zasnovane na postupcima strojnog učenja i raspoznavanju uzoraka. Kao praktični dio rada izrađena je aplikacija za Android operacijski sustav koja na temelju potpisa fotografiranog pomoću uređaja, omogućava korisniku pristup nekom sadržaju. U drugom poglavlju objašnjena je važnost i načini raspoznavanja korisnika. Opisana je biometrijska identifikacija, njeni tipovi i primjena. Opisani su neki od suvremenih načina primjene strojnog učenja i klasifikacije u raspoznavanju korisnika. Također su istražena i postojeća rješenja na Android platformi. U trećem poglavlju objašnjeni su pojmovi strojno učenje i klasifikacija. Navedeni su neki od problema koje je moguće riješiti strojnim učenjem i opisani su neki od načina njegove primjene. Proučena je uloga podataka i uzoraka u strojnom učenju i opisane su tri metode za izdvajanje značajki te tri algoritma za klasifikaciju. U četvrtom poglavlju objašnjeno je programsko rješenje koje je korišteno u praktičnom dijelu ovog diplomskog rada. Rješenje se zasniva na prepoznavanju uzoraka u fotografiranom potpisu. Na temelju rezultata analize prepoznatih uzoraka potpis se klasificira kao autentičan ili krivotvoren. Za prepoznavanje uzoraka korištena je metoda lokalnih binarnih

uzoraka (engl. *Local Binary Patterns – LBP)*, a za klasifikaciju, *kNN* klasifikator. U ovom poglavlju također su dani zahtjevi na sustav, opisana je platforma te alati i tehnologije korištene pri izradi programskog rješenja. Nakon izvedenog rješenja napravljena je statistička analiza rješenja. U zaključku su dani osvrti na postignute rezultate, opisane su prednosti i nedostatci realiziranog projekta, te dane smjernice za buduća moguća poboljšanja.

2. RASPOZNAVANJE KORISNIKA

U ovom poglavlju definirat će se pojam identifikacije i biometrijske identifikacije. Pokazat će se neki od trenutno najkorištenijih načina raspoznavanja korisnika u računalnim sustavima. Bit će objašnjeni principi rada pojedinih metoda. Prikazat će se i načini implementacija biometrijskih metoda identifikacije na uređajima s Android operacijskim sustavom. Analizirat će se učinkovitost i sigurnost korištenja pojedine metode u svakodnevnom korištenju mobilnih uređaja. Također će se ukazati na potencijalne probleme koji se mogu pojaviti pri korištenju pojedinih biometrijskih metoda.

2.1. Važnost i načini raspoznavanja korisnika

U ovom radu raspoznavanje korisnika promatrat će se u kontekstu sigurnosti računalnih sustava. Prema [1]: „informacijska sigurnost je stanje povjerljivosti, cjelovitosti i raspoloživosti podatka, koje se postiže primjenom propisanih mjera i standarda informacijske sigurnosti te organizacijskom potporom za poslove planiranja, provedbe, provjere i dorade mjera i standarda."

Svakako najpoznatiji i najraširenij način raspoznavanja korisnika u računalnim sustavima je kombinacija korisničkog imena i lozinke. U jednom sustavu ne može postojati više korisnika s istim korisničkim imenom, a svaki korisnik mora imati tajnu lozinku s kojom samo on može pristupiti sustavu. Ovakav način autentifikacije u upotrebi je od pojave prvih računalnih sustava i s vremenom su se pojavili brojni problemi. Jedan od najčešćih problema pri korištenju ovog načina autentifikacije je niska razina sigurnosti odabranih lozinki. Ljudi se nerijetko odlučuju za vrlo jednostavne lozinke jer žele što prije pristupiti svom računu ili nekom sadržaju. Odabir jednostavnih lozinki čini njihove račune vrlo ranjivima i podložnim napadima. Bilo da se radi o korištenju nekih od algoritama ili programa za otkrivanje lozinki ili jednostavno pukim pokušajima pogađanja. Drugi veliki problem ovog pristupa nadovezuje se na prethodni i naziva se recikliranje lozinki. Korisnici često koriste iste zaporke za različite vrste računa. Prema [2] čak 52% korisnika upotrebljava istu ili vrlo sličnu zaporku za različite usluge na Internetu. Još više je zabrinjavajuć podatak da se čak 85% recikliranih lozinki koristi u *online* trgovinama, a 62% za email.

U posljednjih nekoliko godina razvojem tehnologije pojavile su se neke nove, naprednije, brže i što je najvažnije, sigurnije metode raspoznavanja korisnika. Biometrijska identifikacija je jedna od tih metoda i njena je primjena danas sve češća u računalnim sustavima.

2.2. Biometrijska identifikacija

Prije definiranja biometrijske identifikacije, potrebno je prvo definirati pojam identifikacije. Prema [3]: "Identifikacija osobe je utvrđivanje istovjetnosti nepoznatog s otprije poznatim, na temelju određenih identifikacijskih obilježja. To je postupak usporedbe određenog broja identifikacijskih obilježja, pri čemu se ustanovljava podudarnost ili različitost između objekata koji se uspoređuju.“ Potrebno je razlikovati pojmove utvrđivanje identiteta i provjere identiteta. Provjera identiteta podrazumijeva uvid u javnu ispravu i usporedbu s već postojećim podacima u bazi podataka. Utvrđivanje identiteta puno je složeniji postupak koji se provodi kada identitet osobe nije poznat ili se sumnja u njegovu ispravnost.

Riječ biometrija potječe iz grčkog jezika od riječi: *bios*, što znači život i *metron*, što znači mjera. Prema tome, biometrija predstavlja mjerenje ponašajnih i tjelesnih osobina čovjeka. Biometrija se može definirati i na sljedeći način: „Biometrija je znanost o automatiziranim postupcima za jedinstveno prepoznavanje ljudi na temelju jednog ili više urođenih tjelesnih obilježja, ili obilježja čovjekovog ponašanja“ [3]. Kako bi se identifikacijska obilježja mogla koristiti pri raspoznavanju korisnika moraju odgovarati sljedećim zahtjevima:

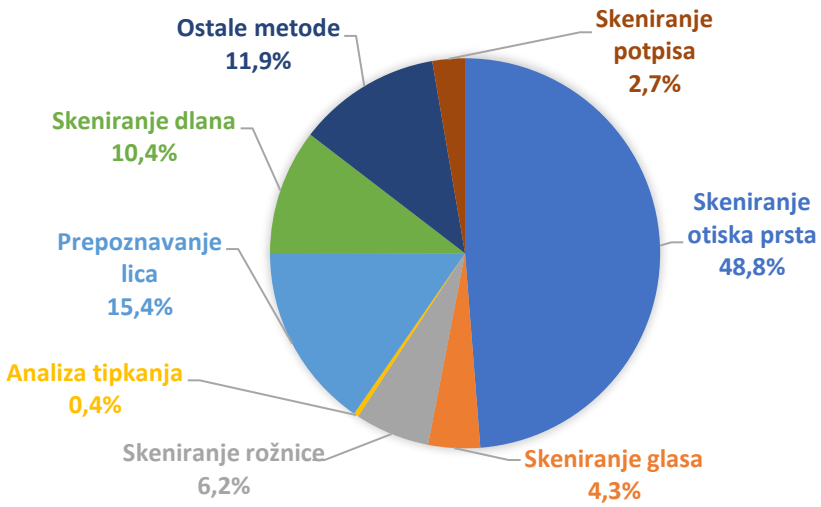
- univerzalnost – svaka osoba ga posjeduje,
- jedinstvenost – obilježje je različito kod svake osobe,
- trajno je i nepromjenjivo,
- moguće ga je izdvojiti iz ukupnosti obilježja – ovo je važno zbog mogućnosti pohrane u baze podataka i usporedbe,
- jednostavno prikupljanje i korištenje.

Svaki sustav koji se koristi za biometrijsku identifikaciju najčešće se sastoji od 4 glavna dijela:

- ulazne jedinice – služi za mjerenje obilježja,
- ekstraktora – služi za izdvajanje obilježja iz cjeline,
- baze podataka – sadrži pohranjena obilježja,
- jedinice za verifikaciju i usporedbu – provjerava i uspoređuje ulazna obilježja s uzorcima pohranjenim u bazi.

2.2.1. Tipovi biometrijske identifikacije

Biometrijska identifikacija može se podijeliti na fizičku biometriju i biometriju ponašanja. U fizičku biometriju pripadaju obilježja poput otiska prsta, DNK zapisa, šarenice oka, crta lica, geometrije šake, provjere vena itd. Biometriju čovjekova ponašanja može se promatrati kroz prepoznavanje glasa, prepoznavanje rukopisa ili potpisa, dinamiku tipkanja, dinamiku hodanja ili mirisa. Iako se oba tipa identifikacije mogu koristiti za raspoznavanje korisnika u računalnim sustavima, zbog jednostavnosti korištenja, cijene realizacije i jednostavnosti implementacije u usporedbi sa drugim oblicima, najrašireniju primjenu pronašli su čitači otiska prsta. Ponajviše u modernim mobilnim uređajima i računalima. Slika 2.1 prikazuje trenutni postotak upotrebe biometrijskih metoda identifikacije u računalnim sustavima.



Sl. 2.1. Korištenje biometrijskih metoda identifikacije, izrađena prema [4]

2.2.2. Otisak prsta

Otisak prsta je najstarija metoda biometrijske identifikacije i daleko je najčešće identifikacijsko obilježje koje se koristi u biometrijskoj identifikaciji. Kao što je to slučaj i sa svim ostalim obilježjima, svaka osoba ima jedinstven otisak prsta. Otisak prsta drugačiji je na svakom prstu osobe, čak i kod jednojajčanih blizanaca.

Metoda raspoznavanja korisnika pomoću otiska prsta zasniva se na daktiloskopiji. Prema [5]: daktiloskopija je grana kriminalistike koja se bavi proučavanjem papilarnih linija s ciljem identifikacije živih ili mrtvih osoba te nepoznatih osoba. Nekada su se otisci prstiju uspoređivali ručno, a danas postoje sustavi automatizirane identifikacije otisaka prstiju (engl. *Automated fingerprint identification system,* AFIS). U ovim sustavima danas se koristi nekoliko različitih tehnologija za prepoznavanje. Neke od tih tehnologija su optički senzori, termo-električni senzori, kapacitivni senzori, senzori električnog polja te senzori osjetljivi na pritisak. Od svih nabrojanih tehnologija najzastupljenija i najpoznatija je tehnologija optičkih senzora.

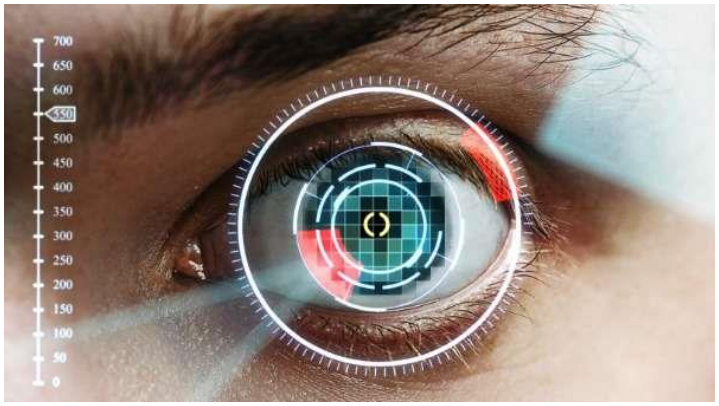
Razlog velike zastupljenosti ovih sustava (slika 2.1.) je u tome što je jedna od komercijalno najdostupnijih biometrijskih tehnologija. Mnogi proizvođači mobilnih uređaja te stolnih i prijenosnih računala danas implementiraju čitače otiska prsta u svoje uređaje kao moguću zamjenu za lozinke i pinove. Na slici 2.2. vidljiv je najčešći oblik implementacije ovog senzora.

Sve češće se kod modernijih mobilnih uređaja s OLED ili AMOLED zaslonima, mogu pronaći skeneri otiska prsta ugrađeni u zaslon. Nazivaju se još i *in-display* senzori. Vrlo pojednostavljen način rada ove tehnologije može se prikazati na sljedeći način: nakon što korisnik prisloni prst na zaslon, zaslon se uključuje i osvjetljava prst. Optički senzor prikuplja svjetlost koja se reflektira od prsta kroz razmake između piksela. Potrebno je blago pritisnuti zaslon kako bi senzor uspio prepoznati razlike između udubljenja i ispupčenja na prstu [6].

Sl. 2.2. Skener otiska prsta na mobilnom uređaju [7]

2.2.3. Skeniranje oka

Ljudsko oko idealan je kandidat za biometrijsku identifikaciju. U procesu identifikacije može se koristiti i šarenica i mrežnica oka. Šarenica je dio oka koji mu daje karakterističnu boju, a funkcija joj je da regulira količinu svjetla koja ulazi u oko (sl. 2.3.). Svaka osoba ima karakterističan i jedinstven izgled šarenice. Mrežnica je unutarnja ovojnica oka. Kao i šarenica, jedinstvena je i karakteristična za svakog čovjeka, međutim ona je jedno od najsigurnijih obilježja za identifikaciju jer nije moguće stvoriti repliku unutarnje strukture oka.



Sl. 2.3. Skeniranje šarenice oka [8]

Sustavi za raspoznavanje korisnika koji koriste ovu metodu smatraju se veoma pouzdanim jer se ne mogu prevariti lećama, staklenim okom, a čak niti pravim okom odstranjenim sa čovjeka. Sustavi imaju implementirane algoritme koji prepoznaju nosi li osoba leće. Za preostala dva

slučaja koristi se laser koji osvijetli oko na vrlo kratko vrijeme i u tom vremenu prati reakciju zjenice, tj. prati skupljanje i širenje zjenice pri obasjavanju.

Proces autentifikacije skeniranjem šarenice traje u prosjeku nekoliko sekundi i smatra se nenametljivim jer nije potreban kontakt osobe sa skenerom. Za razliku od njega proces skeniranja mrežnice nešto je duži pa može potrajati i do 15-ak sekundi. Kako bi proces bio uspješan potrebno je skinuti naočale ukoliko ih osoba nosi, približiti oko vrlo blizu skeneru i fokusirati se na jednu točku. Ovaj način skeniranja smatra se nametljivim jer je oko za vrijeme procesa osvijetljeno blagom svijetlosti, ali značajno je sigurniji i koristi se u slučajevima kada je potrebna visoka razina sigurnosti i gdje cijena nije odlučujući čimbenik.

2.2.4. Prepoznavanje lica

Izgled ljudskog lica određen je građom lubanje, rasporedom mišića lica, kvalitetom i vrstom kože, izgledom pojedinih dijelova poput nosa, usta, očiju i dr. Neke od karakteristika koje se uzimaju u obzir pri prepoznavanju lica su: razmak između očiju, veličina očiju, razmak i položaj jagodičnih kostiju, dimenzije i oblik nosa, oblik čela, oblik usta, dimenzije i oblik brade itd.

Biometrijska tehnologija prepoznavanja lica ne zahtijeva posebnu opremu za svoju implementaciju. Za prepoznavanje lica dovoljno je imati računalo i kameru. Ovakvi sustavi zbog svoje brzine se često koriste na mjestima gdje je potrebna provjera velike količine ljudi. Snimke lica ne zauzimaju puno memorije pa je sustav u mogućnosti usporediti veliki broj fotografija u kratkom vremenu. Budući da su karakteristike lica relativno nepromjenjive, osoba se često može prepoznati i nakon kirurških korekcija lica (sl. 2.4).

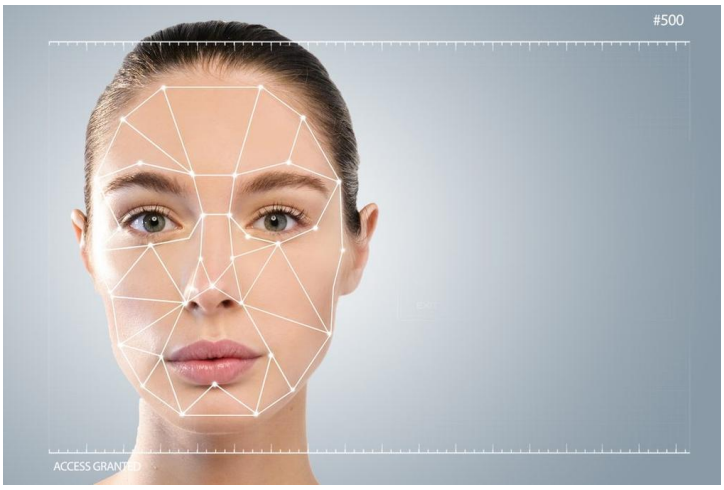
Sustavi za prepoznavanje lica koriste različite algoritme i metode strojnog učenja u svojim implementacijama. Prema [9], tri najčešće korištene metode su:

- holističke metode podudaranja
- metode temeljene na značajakama – strukturalne metode
- hibridne metode

Kod holističkih metoda cjelokupno područje lica se uzima u obzir i koristi kao ulazni podatak u sustav za prepoznavanje. Neki od najboljih primjera holističkih metoda podudaranja su *Eigenfaces, Principal Component Analysis, Linear Discriminant Analysis* i *Independent Component Analysis.*

U metodama temeljenim na značajkama lokalne značajke poput očiju, nosa ili usta se prvo izdvajaju i njihove lokacije i lokalne statistike prosljeđuju se strukturalnom klasifikatoru. Veliki izazov za metode izdvajanja značajki je tzv. obnova značajki (engl. *Feature restoration)* – kada sustav pokušava pronaći značajke koje su nevidljive zbog velikih varijacija (npr. pozicija glave kada se uspoređuje slika sprijeda i slika iz profila). Sustavi s hibridnim metodama koriste kombinaciju holističkih metoda podudaranja i metoda temeljenih na značajkama. Najčešće se u hibridnim metodama koriste 3D slike. Slika ljudskog lica pretvara se u 3D, što omogućuje sustavu da detektira stvari poput zakrivljenosti očiju, oblika brade ili čela. Može se koristiti čak i slika iz profila jer sustav koristi dubinu i os mjerenja, što mu daje dovoljno informacija za konstrukciju cijelog lica. Proces prepoznavanja kod ovih metoda je sljedeći: detekcija lica, određivanje veličine, lokacije i nagiba glave, mjerenje i stvaranje predložaka, pretvaranje predložaka u numeričku reprezentaciju i samo uspoređivanje sa slikama u bazi podataka [9].

Prepoznavanje lica danas ima mnoge upotrebe. Neke od najpoznatijih su svakako detekcija lica na Facebooku, detekcija lica na kamerama mobilnih uređaja ili *FaceID* tvrtke *Apple*. Osim ovih prepoznavanje lica može se pronaći i u mnogim drugim sustavima poput multimedijskih sustava, web-trgovinama i u raznim sigurnosnim i nadzornim sustavima. *FaceID* sustav dizajniran je kao zamjena za dosadašnji *TouchID,* a omogućuje otključavanje mobitelnog uređaja, plaćanje putem *Apple Pay-a* i pristupanje osjetljivim podacima. Sklopovlje za ovaj sustav sastoji se od senzora s tri modula. Jedan modul projicira mrežu sitnih, infracrvenih točaka na lice korisnika. Drugi modul čita rezultirajući uzorak i stvara 3D mapu lica, a treći modul je infracrvena kamera koja snima sliku korisnika. Ova mapa lica uspoređuje se s pohranjenom mapom i korisnik se autentificira ukoliko se mape poklapaju [10].



Sl. 2.4. Prepoznavanje lica osobe [11]

2.2.5. Prepoznavanje glasa

Ovakav oblik identifikacije najčešće se koristi u kombinaciji s nekim drugim tehnologijama. Razlog tomu je način na koji se ovakav sustav može zaobići. Korisnik mora izgovoriti fraze ili rečenice unaprijed spremljene u bazi kako bi se uspješno autentificirao. Pri izgovoru sustav za prepoznavanje prati karakteristike poput modulacije, frekvencije, boje glasa i govornih mana. Međutim, snimkom korisnikova izgovora ovakvi sustavi se u većini slučajeva mogu zaobići. Iako, najnoviji algoritmi za raspoznavanje imaju sposobnost prepoznavanja prirodnog govora. U modernim računalnim i mobilnim sustavima prepoznavanje glasa često se koristi za pretvaranje govora u tekstualne zapise. Tako je pomoću glasovnih naredbi moguće pisati poruke, slati elektroničku poštu, birati brojeve iz imenika ili vršiti interakciju sa digitalnim asistentima na mobitelima ili računalima (slika 2.5.).

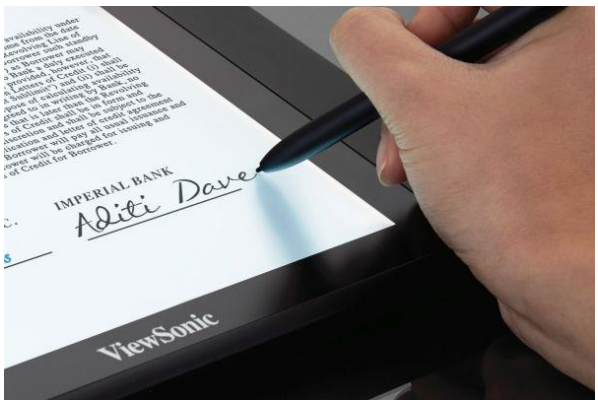
Sl. 2.5. Google asistent kontroliran pomoću glasovnih naredbi [12]

2.2.6. Prepoznavanje potpisa i rukopisa

Ako se uzme u obzir da svaka osoba ima jedinstven rukopis, onda se potpis može iskoristiti u svrhu autentifikacije. Potpis se koristi svakodnevno za davanje suglasnosti, bilo da se radi o ugovoru, peticiji, radnom dokumentu, potpisu transakcije ili nekoj drugoj radnji koja zahtijeva autentifikaciju. Na izgled potpisa mogu utjecati razni faktori poput psihološkog i fizičkog stanja osobe u trenutku potpisivanja, položaja tijela, podloge na kojoj se piše ili olovke kojom se piše. Potpisi iste osobe mogu se značajno razlikovati u službenom i neslužbenom okruženju. Okolina osobe poput buke, osvjetljenja ili temperature također može utjecati na izgled potpisa.

Budući da uvijek postoji mogućnost da će netko pokušati krivotvoriti potpis, potrebno je imati način točnog određivanja autora potpisa. Ovo se može postići analizom općih i posebnih ponašajnih biometrijskih karakteristika pri potpisivanju. U opće karakteristike mogu se svrstati stupanj ispisanosti, raspored teksta, odnos prema liniji pisanja, veličina potpisa, povezanost slova, nagib pisanja i ukrašavanje. Ono što čini potpis skoro nemogućim za kopiranje su osobine poput nagiba olovke, brzine pisanja, jačine pritiska na podlogu ili duljine poteza ruke [3].

Sustavi za prepoznavanje potpisa mogu se podijeliti na *online* i *offline* sustave. Ova podjela ovisi o načinu prikupljanja podataka. U *online* sustavima koriste se tableti osjetljivi na pritisak (sl. 2.6.) ili *web* kamere, dok se u *offline* sustavima koriste skenirane ili fotografirane slike potpisa. U ovom diplomskom radu koristit će se *offline* sustav za prepoznavanje.



Sl. 2.6. Uređaj za digitalno potpisivanje [13]

3. STROJNO UČENJE I KLASIFIKACIJA

Strojno učenje i umjetna inteligencija vrlo su popularna i zanimljiva područja u računalnoj industriji, posebno u posljednjih nekoliko godina. Ova dva pojma često se poistovjećuju, iako se njihova značenja razlikuju. Strojno učenje smatra se podskupom umjetne inteligencije. Jedan od načina kako se može protumačiti značenje pojma umjetna inteligencija je sposobnost strojeva da obavljaju zadaće koje su karakteristične za ljudsku inteligenciju. Prema tome, strojno učenje je jedan od načina kako se može postići umjetna inteligencija, a može se definirati kao sposobnost učenja bez eksplicitnog programiranja. Točnije, strojnim učenjem smatra se proučavanje i primjena algoritama koje računalni sustavi koriste kako bi izvršili neki zadatak bez korištenja konkretnih naredbi, oslanjajući se pritom na podatke i uzorke u podacima koje algoritmi pronalaze. Algoritmi u strojnom učenju koriste se kako bi se izradio matematički model na osnovu ulaznih podataka ili podatka za treniranje kako se najčešće nazivaju. Koristeći podatke za treniranje, model uči kako nešto predvidjeti ili donijeti odluku [14].

Razlog zašto je strojno učenje postalo iznimno popularno u proteklih nekoliko godina su velike količine podatka koje se generiraju svaki dan. Te velike količine podataka počele su se iskorištavati za treniranje sve učinkovitijih modela. Još jedan od razloga zašto se strojno učenje počelo sve intenzivnije koristiti je i razvoj sklopovlja. Nerijetko se za potrebe strojnog učenja prikupljaju izrazito velike količine podataka za treniranje. Za obradu takvih podataka i za provedbu matematičkih izračuna potrebno je imati adekvatno sklopovlje. Danas se u tu svrhu najčešće koriste grafičke kartice jer pružaju značajno bolje performanse u usporedbi s procesorima. Razlog tomu je način na koji su dizajnirane. U odnosu na procesore, koji su dizajnirani za općenitije izračune, grafičke kartice su manje fleksibilne, tj. napravljene su tako da mogu izvršavati veći broj istih operacija istovremeno. Njihova primjena posebno je značajna u dubokom učenju, gdje svaki sloj neuronske mreže može obavljati i na tisuće jednakih izračuna.

Postoje mnoge primjene strojnog učenja u različitim područjima. Jedna od najznačajnijih i najčešćih primjena u posljednje vrijeme je svakako implementacija na mobilnim uređajima. Strojno učenje pokreće digitalne pomoćnike poput *Google Assistanta, Siri i Alexe*. Značajne primjene strojnog učenja mogu se vidjeti i u medicini gdje se koriste za detekciju različitih bolesti koristeći klasifikaciju slika. Isto tako, strojno učenje pronalazi primjenu u predviđanju cijena

dionica. Vidljivo je, iz navedenih primjera, da je primjena strojnog učenja široka i svaki od problema zahtjeva nešto drugačiji pristup. Prema tome, strojno učenje može se podijeliti na dvije glavne vrste, nadzirano i nenadzirano učenje.

3.1. Nenadzirano učenje

Nenadzirano učenje je vrsta strojnog učenja gdje algoritmi pronalaze uzorke u podacima koji nisu klasificirani ili označeni. Neke on najčešćih primjena ovakvog načina strojnog učenja su:

- grupiranje – algoritmi pronalaze sličnosti u neoznačenom skupu podataka i prema tim sličnostima automatski svrstavaju podatke u grupe
- pronalaženje anomalija u podacima – automatski pronalaze neobične unose u skupu podataka. Ovo može biti korisno u raznim područjima gdje je potrebno pronaći neispravne unose u velikim i neoznačenim skupovima podataka
- pronalaženje asocijacija i relacija – algoritmi pronalaze skupove podataka koji se najčešće pojavljuju zajedno. Ovakav način učenja može se primijeniti za analizu košarice u web trgovinama. Algoritmi prepoznaju koje proizvode korisnici najčešće kupuju zajedno i prema tome analitičari i marketinški stručnjaci mogu razviti učinkovitije marketinške i prodajne strategije

Nenadzirano učenje može se koristiti i u kombinaciji s nadziranim učenjem. Uzorci koji se otkriju pomoću nenadziranog učenja mogu biti korisni pri treniranju modela u nadziranom učenju. Obično ovakvi sustavi, koji koriste nenadzirano učenje mogu obavljati složenije zadatke od sustava baziranih na nadziranom učenju, ali su nepredvidljiviji.

3.2. Nadzirano učenje

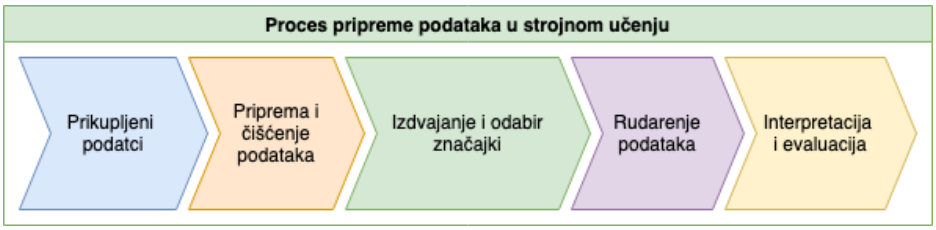
Nadzirano učenje, za razliku od nenadziranog, radi sa podacima koji su kategorizirani. To znači da su podaci koji se koriste za treniranje nekog modela označeni pripadnim imenom ili klasom. Budući da je praktični dio ovog rada aplikacija koja ima mogućnost prepoznavanja fotografiranog potpisa, korišteno je nadzirano učenje, a podaci su podijeljeni u dvije kategorije: “original” i “kopija”. Način implementacije detaljnije je razrađen u sljedećim poglavljima.

Dva najčešća problema koja se rješavaju nadziranim strojnim učenjem su klasifikacijski i regresijski problemi. Kod rješavanja klasifikacijskih problema cilj je predvidjeti diskretne vrijednosti na izlazu. Primjerice, diskretne vrijednosti izlaza mogu biti važna ili neželjena elektronička pošta, pasmina psa – labrador, retriver i slično. Više o klasifikatorima bit će pojašnjeno u poglavlju 3.8. Regresijski algoritmi nastoje predvidjeti neprekidne vrijednosti koje nije moguće klasificirati. Primjeri regresijskih problema mogu biti predviđanje cijena goriva ili predviđanje cijene stana ili kuće.

3.3. Podaci (Uzorci)

Podaci su jedan od najvažnijih dijelova strojnog učenja. Kako bi sustav mogao odraditi postavljeni zadatak, potrebno ga je to naučiti i dati mu podatke s kojima će raditi. Za učenje se koristi skup podataka koji ima svoje značajke i oznake klase (samo kod nadziranog učenja). Prvi korak u radu sa strojnim učenjem je prikupiti podatke. Početni skup uzoraka najčešće nije u potpunosti prikladan za treniranje modela jer može sadržavati nepotpune ili neispravne podatke, npr. prazne numeričke vrijednosti, odrezane slike i slično. Zbog toga je početni skup podataka potrebno pripremiti prije nego se koristi za treniranje. Proces pripreme podataka i odabira značajki za korištenje prikazan je na slici 3.1.

Ulazni skup podataka najčešće se dijeli na dva dijela: skup podataka za treniranje i skup podataka za testiranje i to tako da 60-80% podataka pripada skupu za treniranje a 20-40% podataka skupu za testiranje [15]. Svrha skupa podataka za testiranje je da se provjeri točnost i učinkovitost istreniranog modela s podacima koji nisu prethodno viđeni u procesu treniranja. Potrebno je voditi računa o dvije stvari pri odabiru skupa podataka za testiranje. Skup podataka treba biti dovoljno velik za statističku obradu i ne bi trebao imati drugačije karakteristike od skupa podataka za treniranje. Važno je napomenuti da se skup podataka za treniranje nikada ne bi trebao koristiti za testiranje.



Sl. 3.1. Proces pripreme podataka i odabira značajki za treniranje modela, izrađeno prema [16]

3.4. Značajke

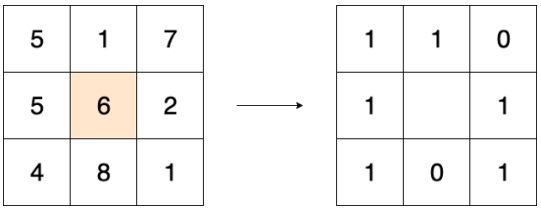
Kao što je navedeno u prethodnom poglavlju, svaki ulazni skup podataka ima svoje značajke. Značajke su individualne mjere uzorka koji je potrebno predvidjeti ili klasificirati. Veliku ulogu u kvaliteti treniranja modela imaju upravo odabrane značajke. U većini slučajeva veći broj značajki omogućuje bolje rezultate pri treniranju modela jer je dostupno više informacija o predmetima koje je potrebno predvidjeti ili klasificirati. Iako je poželjno imati veći broj značajki, nekada ne znači nužno veću točnost treniranog modela. Naprotiv, u nekim slučajevima značajke koje nisu dobro definirane mogu smanjiti efikasnost algoritma. Stoga je vrlo važno odabrati one značajke koje što bolje opisuju subjekt koji se predviđa. Primjerice, ako je potrebno trenirati klasifikator koji razvrstava jabuke i naranče, ulazni skup podataka može sadržavati značajke poput težine, veličine, boje i teksture. U ovom skupu, težina, boja i tekstura su značajke pomoću kojih je moguće vrlo lagano odrediti razliku između dvije klase. Veličina u ovom slučaju nije pretjerano korisna značajka jer su naranče i jabuke često približne veličine i pomoću nje se ne može lako napraviti razlika između njih. Isto tako, trebale bi se izbjegavati redundantne značajke. Ako postoji redundancija, algoritam može pogrešno zaključiti da značajke koje se ponavljaju imaju veću važnost, a zapravo opisuju isto svojstvo subjekta koji se predviđa. Dakle, pažljivim odabirom značajki može se uvelike poboljšati efikasnost algoritma.

U velikom broju slučajeva početni skup podataka je prevelik za upravljanje ili ima veliku redundanciju između značajki. Prema tome, preliminarni korak u raznim primjenama strojnog učenja i raspoznavanja uzoraka sastoji se od izdvajanja podskupova značajki ili stvaranja novog, smanjenog skupa kako bi postupak treniranja bio učinkovitiji. Pronalaženje i izdvajanje najboljih značajki nerijetko je vrlo kompliciran zadatak i zahtijeva poprilično razinu poznavanja metoda izdvajanja i iskustva u radu sa strojnim učenjem. Prema [17], stručnjaci za strojno učenje provode čak 80% vremena na prikupljanje i pripremu podataka. Proces izdvajanja značajki moguće je

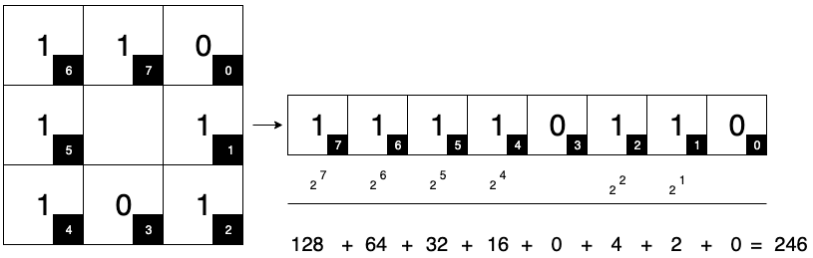
automatizirati i na taj način pomoći znanstvenicima u pripremi. Automatizacija pomaže tako da se stvori nekoliko potencijalno korisnih značajki iz skupa podataka. Neke od tih značajki mogu se odabrati i koristiti za treniranje [18]. Izdvajanje značajki je skup metoda koje omogućavaju sustavu da automatski otkrije potrebne značajke za učenje iz cjelokupnog ulaznog skupa podataka.

3.5. Lokalni binarni uzorci (engl. *Local Binary Patterns – LBP*)

Lokalni binarni uzorci su algoritam pomoću kojega je moguće pronaći rubove i oblike na slikama. Prije same implementacije *LBP-a* potrebno je originalnu sliku pretvoriti u sliku sive boje (engl. *grayscale*). Ako se radi sa slikama u boji, potrebno je cjelokupni postupak ponoviti tri puta. Po jednom za svaki od RGB kanala na slici. *LBP* u svom izvornom obliku analizira 9 susjednih piksela slike u jednom trenutku, uzimajući u obzir blok od 3x3 piksela. U tom bloku algoritam uspoređuje vrijednost srednjeg piksela s vrijednostima svih susjednih piksela i pretvara tih 9 vrijednosti u izlaznu vrijednost koja predstavlja jedan uzorak. Način na koji algoritam radi usporedbu je sljedeći: ukoliko je vrijednost srednjeg piksela manja u odnosu na vrijednost susjednog piksela, na njegovo se mjesto upisuje 0, a ukoliko je veća ili jednaka upisuje se 1 (slika 3.2.). Nakon što se napravi usporedba za svaki od piksela u bloku, nove vrijednosti zapišu se kao 8 bitni binarni broj i to na način da se odabere jedan od piksela u bloku, npr. gornji desni i zapisuju se vrijednosti u smjeru kazaljke na satu ili obrnuto od smjera kazaljke na satu. Odabir početnog piksela i smjera je proizvoljan, ali vrlo je bitno da se prati uvijek isti smjer i da se za sve blokove u slici koristi uvijek isto početno mjesto u bloku. Nakon što se dobije 8 bitni binarni broj potrebno ga je pretvoriti u decimalni zapis (slika 3.3.). Ova nova vrijednost u decimalnom zapisu sada predstavlja cjelokupni blok kao jednu značajku. Budući da se radi o 8 bitnom broju, decimalne vrijednosti koje predstavljaju značajku mogu biti od 0 do 255. Ovaj algoritam potrebno je ponoviti za sve piksele u zadanoj slici.

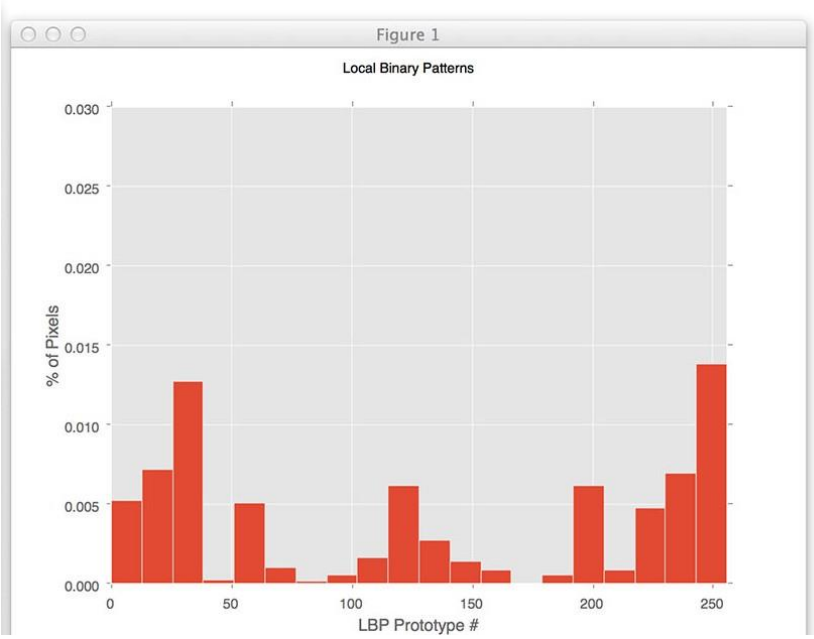


Sl. 3.2. Usporedba vrijednosti srednjeg piksela s vrijednostima susjednih piksela u LBP algoritmu, izrađeno prema [19]



Sl. 3.3. Pretvaranje 8 bitnog binarnog zapisa u decimalni broj , izrađeno prema [19]

Nakon što se ovaj postupak ponovi za svaki piksel u slici, može se napraviti histogram koji pokazuje koliko često se pojedini *LBP* uzorak pojavljuje. Mougće je napraviti jedan histogram, koristeći vrijednosti cijele slike (slika 3.4.), ili više histograma koji predstavljaju pojedine dijelove cjelokupne slike. Veličina histograma iznosi 2^P , gdje P predstavlja broj susjednih piksela koji se uspoređuju. U ovom radu P iznosi 8, tj. svaki piksel uspoređuje se s 8 susjednih piksela. Dobiveni histogram može se iskoristiti kao vektor značajki koji će se koristiti za treniranje klasifikatora.



Sl. 3.4. Histogram LBP vrijednosti [19]

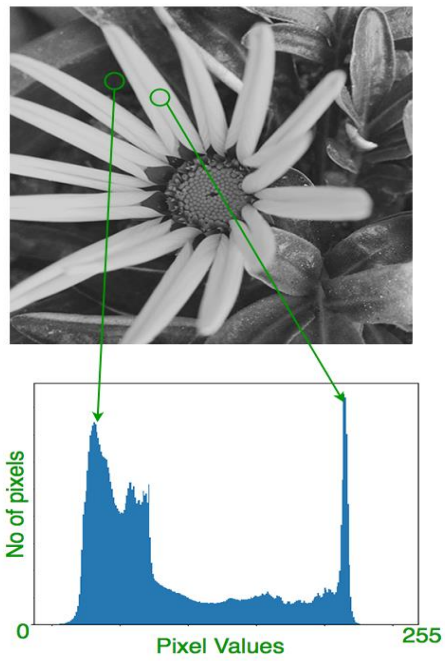
Važno je spomenuti i koncept *LBP* uniformnosti. *LBP* se smatra uniformnim ako ima najviše 2 prijelaza između 1 i 0 ili 0 i 1. Primjerice, uzorci 00001000 i 10000000 smatraju se uniformnima. Prvi uzorak ima dva prijelaza (0-1-0), dok drugi ima samo jedan (1-0). Primjer ne-uniformnog

uzorka može izgledati ovako 01010010. Ovaj uzorak ima 6 prijelaza između 1 i 0. Broj uniformnih uzoraka u *LBP-u* ovisan je o broju susjednih piksela **p** [20].

Uniformni *LBP* uzorci često se koriste pri izdvajanju značajki iz slike jer pružaju određenu razinu neovisnosti o rotaciji i osvjetljenju. Primjerice, ako je za određeni dio slike izračunat *LBP* uzorak i nakon toga se taj isti dio slike osvijetli te ponovno izračuna *LBP* uzorak, velika je vjerojatnost da će uzorci biti isti jer se ne uzimaju u obzir konkretne vrijednosti piksela nego omjeri njihovih vrijednosti. Tako će oni pikseli, koji su prije osvjetljenja imali manju vrijednost od svojih susjeda i dalje imati manju vrijednost nakon osvjetljenja, pod pretpostavkom da su svi pikseli slike ravnomjerno osvijetljeni [21].

3.6. Histogram

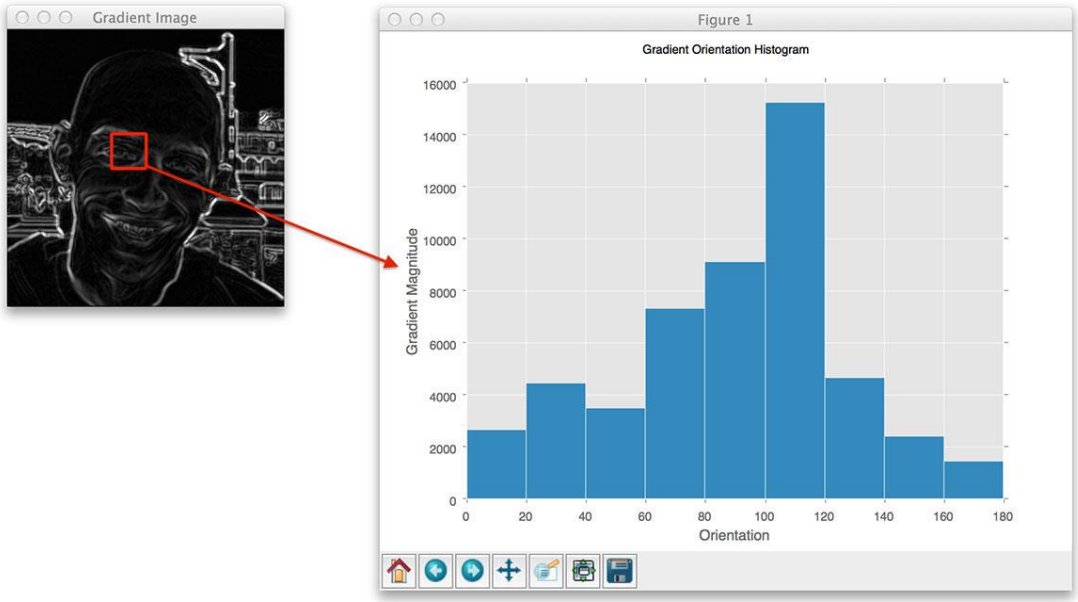
U kontekstu obrade slike, histogram najčešće predstavlja raspodjelu vrijednosti piksela. Histogram je graf koji prikazuje broj piksela za svaku drugačiju vrijednost intenziteta koju pronađe na slici. Primjer histograma prikazan je na slici 3.5. Za 8-bitnu, crno-bijelu sliku postoji 256 mogućih različitih intenziteta. Prema tome, histogram će grafički prikazati distribuciju piksela za svaki od 256 brojeva, odnosno prebrojat će koliko puta se svaka od 256 vrijednosti pojavila na slici. Histogrami se mogu dobiti i od slika u boji i to na dva načina. Jedna opcija je da se naprave tri histograma - svaki od tri grafa predstavlja jedan od kanala crvene, zelene i plave boje. Druga opcija je jedan, trodimenzionalni histogram čije osi predstavljaju tri kanala boje, a svjetlina svake točke predstavlja broj piksela [22].



Sl. 3.5. Primjer slike i pripadajućeg histograma [23]

3.7. Histogram usmjerenih gradijenata

Histogram usmjerenih gradijenata (engl. *histogram of oriented gradients*, HOG*)* je opisnik značajki (engl. *feature descriptor*) koji se najčešće koristi u računalnom vidu i obradi slike za prepoznavanje objekata. Može se još koristiti i za mjerenje i prikazivanje oblika i tekstura. Prvi je puta predstavljen u radu *Dalala* i *Triggsa, „Histograms of Oriented Gradients for Human Detection“* [24]. Osnovna ideja *HOG* algoritma je da se izgled objekta može modelirati distribucijom gradijenata intenziteta unutar pravokutnih dijelova slike. Najvažniji parametri *HOG* algoritma su orijentacija, broj piksela po ćeliji i broj ćelija po bloku. Ova tri parametra kontroliraju dimenzionalnost konačnog vektora značajki. Da bi se implementirao ovaj algoritam, potrebno je podijeliti sliku u nekoliko manjih dijelova – ćelija. Ćelija je pravokutni dio slike čija je veličina definirana brojem piksela koji sadrži (npr. 4x4). Zatim je potrebno izračunati *HOG* za sve piksele unutar ćelije i postupak ponoviti za sve ćelije. Prije računanja histograma, potrebno je definirati broj orijentacija. Ovaj broj određuje koliko stupaca će imati rezultirajući histogram, a predstavlja određeni raspon vrijednosti. Slika 3.6. prikazuje primjer histograma za jednu ćeliju s 9 raspona vrijednosti. Što znači da je u ovom primjeru broj orijentacija također 9. Nakon izračuna histograma za svaku ćeliju, može se kreirati konačni vektor značajki za cijelu sliku [25].



Slika 3.6. Primjer histograma jedne ćelije [25]

Kako bi promjene osvjetljenja i kontrasta što manje utjecale na rezultate, moguće je normalizirati vrijednosti gradijenata lokalno. Za ovaj postupak potrebno je ćelije grupirati u veće, povezane blokove. Ovi blokovi najčešće se preklapaju, što znači da svaka ćelija doprinosi konačnom vektoru značajki više od jedanput. Prema [25], najčešće se koriste blokovi ćelija veličine 2x2 ili 3x3, jer daju dovoljno dobre rezultate u većini slučajeva. Za svaki blok izračunava se vektor na način da se nadovezuju histogrami pojedinih ćelija unutar bloka, a zatim se primjenjuje L1 ili L2 normalizacija na dobiveni vektor. Nakon što su svi blokovi normalizirani, vektori svakog pojedinog bloka nadovezuju se, kreirajući tako jedan konačni vektor značajki. Ovaj vektor može se dalje koristiti na razne načine u kombinaciji sa strojnim učenjem. Najčešće se, u praksi, *HOG* algoritam koristi u kombinaciji s linearnim *SVM* klasifikatorom.

3.8. Klasifikatori (algoritmi)

Klasifikator je matematička funkcija ili algoritam strojnog učenja koji prima podatke na ulazu i prema tim podacima previđa izlaz. Može se reći da preslikava ulazne podatke u kategoriju na izlazu. Klasifikator se može vrlo pojednostavljeno prikazati matematičkim zapisom funkcije $f(x)=y$, gdje x predstavlja ulazne podatke, tj. značajke, a y predstavlja izlaz, odnosno kategoriju.

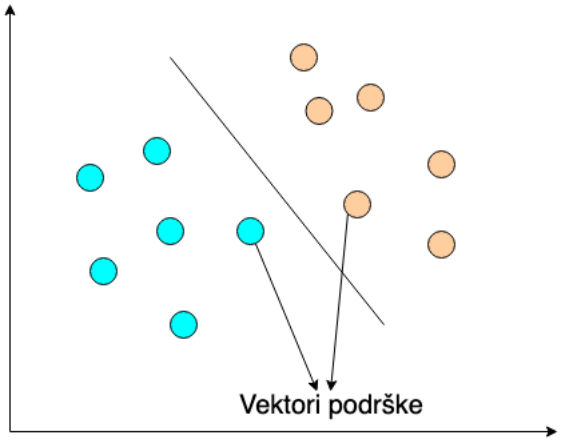
Klasifikator se može usporediti i sa funkcijom u programiranju. Funkcija prima neke parametre i vraća određeni rezultat operacije koja se izvršava u tijelu funkcije. Parametri funkcije predstavljaju ulazne podatke, odnosno značajke koje klasifikator koristi, dok vrijednost koju funkcija vraća predstavlja kategoriju koju klasifikator predviđa. Razlika između funkcije u programiranju i klasifikatora je u tijelu funkcije. U klasičnom programiranju tijelo funkcije piše programer. Suprotno tome, klasifikator se ne programira nego kroz podatke i algoritme uči kako odraditi neki posao. Za rješavanje klasifikacijskih problema moguće je koristiti različite algoritme koji mogu biti binarni kada postoje samo dvije kategorije ili višeklasni kada postoji više kategorija za predviđanje. Najčešći algoritmi koji se koriste pri klasifikaciji su:

- *Decision forest* algoritam
- *Decision tree* algoritam
- Logistička regresija
- SVM – *Support Vector Machine*
- Neuronska mreža
- *Naive Bayes* algoritam
- *K-Nearest Neighbours*

Kada se postavlja pitanje koji algoritam za strojno učenje koristiti, odgovor je uvijek isti – ovisi. Odabir algoritma ovisi o veličini i kvaliteti podataka, o problemu koji se nastoji riješiti algoritmom ili o potrebnom vremenu. Prije odabira algoritma najčešće je potrebno provesti nekoliko eksperimenata s različitim algoritmima i napraviti njihovu evaluaciju. Kada se donosi odluka nakon evaluacije, uzimaju se u obzir karakteristike kao što su preciznost, vrijeme potrebno za treniranje, linearnost, broj parametara i broj značajki.

3.8.1. Stroj s vektorima podrške

Stroj s vektorima podrške *(*engl. *Support vector machine - SVM)* je algoritam koji se koristi za nadzirano strojno učenje. Iako se može koristiti za klasifikacijske i regresijske probleme, najčešću primjenu pronalazi pri rješavanju problema klasifikacije. *SVM* algoritam temelji se na ideji pronalaženja plohe koja najbolje dijeli skup podataka na dvije klase. Klasični primjer primjene vidljiv je na slici 3.6.

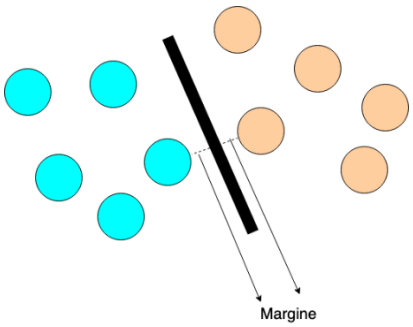


Sl. 3.6. Primjer jednostavnog SVM klasifikatora, izrađeno prema [26]

Važnu ulogu u radu ovog algoritma imaju pomoćni vektori. Pomoćni vektori su podaci najbliži hiperravnini koja razdvaja klase. Odnosno, predstavljaju podatke koji će, ako se uklone, promijeniti položaj hiperravnine koja ih je razdvajala. Zbog ovog svojstva mogu se smatrati ključnim elementima skupa podataka [26].

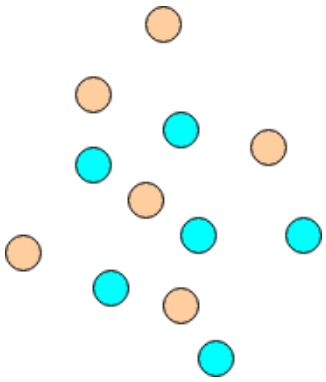
U jednostavnom primjeru kao na slici 3.6. hiperravnina je predstavljena kao pravac koji linearno razdvaja i klasificira skup podataka. Što su pojedini podaci više udaljeni od plohe to je vjerojatnost veća da su ispravno klasificirani. Prema tome uvijek se nastoji da podaci budu što dalje od plohe, a da su istovremeno na ispravnoj strani.

Pronalazak hiperravnine (pravca u ovom jednostavnom dvodimenzionalnom primjeru) koja će dovoljno dobro razdvojiti podatke često je vrlo kompleksan zadatak. Kako bi se pronašla odgovarajuća hiperravnina, potrebno je objasniti pojam margine. Margina predstavlja udaljenost od pravca do najbliže točke iz skupa podataka s obje strane pravca (slika 3.7.).



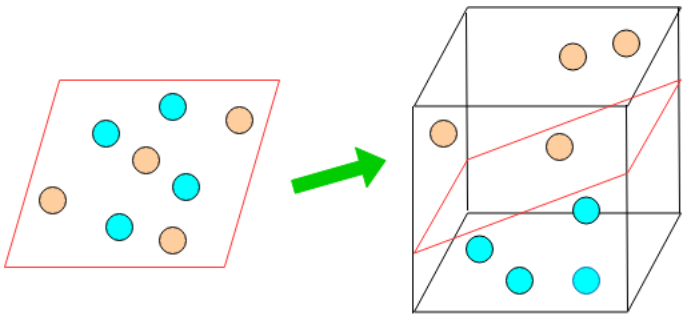
Sl. 3.7. Primjer margine, izrađeno prema [26]

Cilj algoritma je odabrati pravac s najvećom mogućom marginom između pravca i neke od točaka u skupu podataka kako bi se povećala vjerojatnost ispravne klasifikacije podataka. U prethodnom primjeru moguće je relativno lagalo odrediti liniju jer su podaci jednostavno raspoređeni. Problem je što skup podataka gotovo nikada nije ovako čisto i jednostavno raspoređen. U praksi se češće pojavljuju skupovi podataka kao na slici 3.8.



Sl. 3.8. Primjer kompleksnijeg skupa podataka, izrađeno prema [26]

Podatke na slici 3.8. nije moguće linearno razdvojiti, tj. nije moguće odrediti liniju koja će dovoljno dobro razdvojiti podatke i klasificirati ih. Kako bi bilo moguće klasificirati ovakav skup podataka, potrebno ih je preslikati u prostor više dimenzije. U ovom slučaju to znači prijelaz iz dvodimenzionalnog prikaza u trodimenzionalni prikaz, a postiže se korištenjem funkcije jezgre (engl. *kernel).* Ilustracija prijelaza iz 2D u 3D prikaz može se vidjeti na slici 3.9.



Sl. 3.9. Prijelaz iz dvodimenzionalnog u trodimenzionalni prikaz skupa podataka, izrađeno prema [26]

Budući da je skup podataka sada prikazan u tri dimenzije, ploha za razdvajanje više neće biti pravac nego ravnina. Da bi se elementi skupa podataka pravilno klasificirali potrebno je odrediti

ravninu u prostoru koja ih razdvaja. Ideja ovakvog pristupa je da se podaci preslikavaju u višu dimenziju dok ne postane moguće odrediti plohu koja će ih dovoljno dobro razdvojiti.

3.8.2. Algoritam K najbližih susjeda (engl. *k Nearest Neighbor - kNN*)

Ovaj algoritam koristi se u nadziranom strojnom učenju za rješavanje regresijskih i klasifikacijskih problema. Ideja ovog algoritma je pronaći unaprijed definirani broj uzoraka iz skupa podataka za treniranje koji su najbliži novom podatku. Broj uzoraka označava se s k i predstavlja broj najbližih susjeda. Ovaj broj može biti određen od strane korisnika algoritma ili se može mijenjati ovisno o gustoći točaka (učenje bazirano na broju točaka unutar nekog radijusa) [27]. Iako, u teoriji, udaljenost može biti bilo koja mjera, u praksi se najčešće koristi euklidska udaljenost. Prema [27], euklidska udaljenost između točaka p i q može se definirati kao duljina linije koja ih povezuje. Ako su p i q dvije točke u n-dimenzionalnom kartezijevom koordinatnom sustavu, udaljenost između njih može se izračunati prema formuli (3-1).

$$d(p,q) = d(q,p) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2} = \sqrt{\sum_{i=1}^{n} (q_i + p_i)^2} \tag{3-1}$$

Klasifikacija pomoću ovog algoritma je vrsta strojnog učenja koja ne pokušava stvoriti poopćeni interni model za predviđanje, nego jednostavno sprema uzorke podataka iz skupa za treniranje. Klasifikacija se postiže tako da se točki u razmatranju dodijeli klasa čijih elemenata ima najviše u k najbližih susjeda oko točke. Dva najčešća načina primjene ovog algoritma u klasifikaciji su učenje bazirano na k najbližih susjeda od zadane točke i učenje bazirano na broju najbližih susjeda unutar fiksnog radijusa r svake točke iz skupa podataka za treniranje [27].

Jedna od najvećih prednosti ovog algoritma je svakako jednostavnost njegova razumijevanja i implementacije. Unatoč tome, može se koristiti u rješavanju klasifikacijskih i regresijskih problema. Budući da ovaj algoritam drži podatke u memoriji, klasifikator se može lako prilagoditi novim podacima. Još jedna prednost ovog algoritma nad mnogim drugim

algoritmima korištenim u klasifikaciji, jest da se u odnosu na ostale algoritme vrlo lako može prilagoditi za višeklasnu klasifikaciju [28]. Ovaj algoritam ima i nekoliko važnih nedostataka koje je potrebno razmotriti. *k-NN* algoritam nije primjeren za velike skupove podataka jer mu se efikasnost i brzina vrlo brzo smanjuju s povećanjem broja podataka. Također, ovaj algoritam zahtijeva homogene značajke, tj. zahtijeva da značajke imaju istu skalu za mjerenje udaljenosti. Još jedan od problema koji se može pojaviti pri korištenju algoritma su nejednaki podaci. Ako je većina podataka iz skupa za treniranje označena jednom klasom, onda će model, naposljetku, dati veću važnost toj klasi. Ovo može dovesti do toga da se manjinska klasa pogrešno klasificira.

3.8.3. Slučajne šume

Algoritam slučajnih šuma (engl. *Random Forest)* koristi se u nadziranom strojnom učenju i može se koristiti za klasifikacijske i regresijske probleme. Kao što mu ime sugerira, ovaj algoritam radi na principu kreiranja „šume“ višestrukih stabala odluke (engl. *decision tree)* tijekom procesa treniranja. „Slučajno“ u imenu odnosi se na dvije vrste nasumičnosti do kojih dolazi pri treniranju modela stabla odluke. Na primjer, ako se za treniranje koristi skup podataka koji ima 1000 instanci i 30 značajki, nasumičnost se može postići na razini instanci i na razini značajki. Svako stablo odluke prima nasumični uzorak iz skupa podataka za treniranje (npr. 10%). To znači da će svaki model biti treniran neovisno, na 100 nasumično odabranih instanci iz skupa od 1000. Druga vrsta nasumičnosti je na razini značajke. Ne koristi se svih 30 značajki pri treniranju svakog od modela. Odabire se određeni dio (npr. 10%) koji će biti proslijeđen svakom modelu za treniranje. To znači da će, po ovom primjeru, 3 nasumično odabrane značajke biti poslane svakom modelu. Za konačni rezultat klasifikacije uzimaju se u obzir rezultati iz svakog stabla odluke te se podatku koji se klasificira dodjeljuje klasa koju je predvidjelo više modela.

Neke od prednosti ovog algoritma prema [29] su:

- korištenjem ovog algoritma može se izbjeći pretjerana prilagodba trening podacima (engl. *overfitting),* pod uvjetom da se koristi dovoljno velik broj modela stabala odluke,
- isti algoritam može se koristiti za klasifikacijske i regresijske zadatke

- ovaj algoritam može se koristiti i za pronalazak najvažnijih značajki u skupu podataka za treniranje

Prema [30], neki od nedostataka ovog algoritma su:

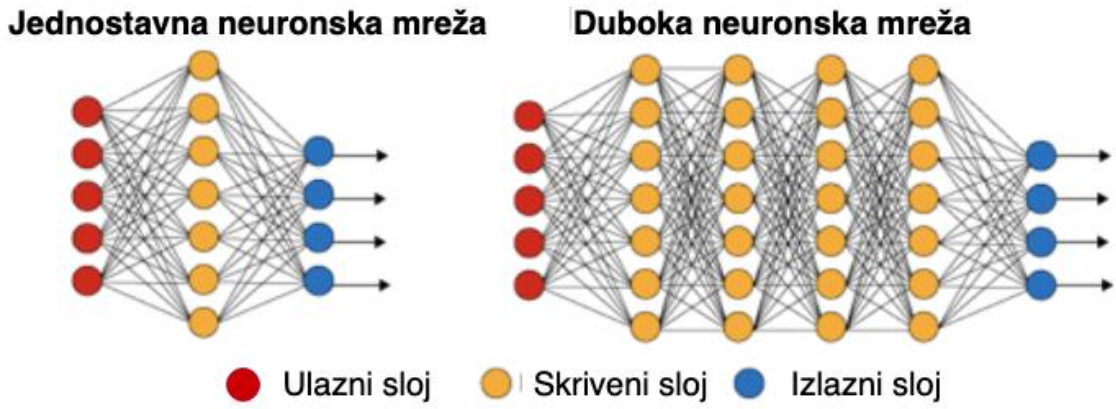
- ne daje dobre rezultate pri radu s manjim skupovima podataka jer ne može uočiti uzorke
- postoji problem interpretacije – teško je uočiti poveznicu između odziva algoritma i varijabli
- vrijeme treniranja nekada može biti poprilično veliko s obzirom da se treniraju višestruki modeli stabala odluke

3.8.4. Dubinsko učenje

Dubinsko učenje pojavilo se prvi puta prije otprilike 40 godina. Međutim, najveći procvat ove tehnologije dogodio se tijekom prošlog i ovog desetljeća ponajviše zahvaljujući velikim količinama podataka koje se generiraju proteklih godina, a mogu se iskoristiti za treniranje neuronskih mreža. Drugi veliki razlog razvoja ove tehnologije je razvoj sklopovlja, posebice grafičkih kartica. Vrijeme treniranja neuronskih mreža značajno se smanjilo korištenjem novijih komponenti. Poboljšanju treniranja neuronskih mreža najviše su doprinijele novije komponente koje omogućavaju brzo, paralelno izvođenje. Nove grafičke kartice se u tom području posebno ističu.

Umjetne neuronske mreže inspirirane su radom biološke neuronske mreže ljudskog mozga i najvažniji su dio dubinskog učenja [31]. Imaju sposobnost progresivnog učenja, tj. njihova sposobnost da riješe zadani problem se povećava analiziranjem podataka i primjera. Neuronske mreže sastoje se od paralelnih čvorova (jedinice za obradu – neuroni) međusobno povezanih i raspoređenih po slojevima. Svaki od neurona individualno je prejednostavan kako bi samostalno mogao nešto naučiti. Snaga neuronskih mreža je u povezanosti i načinu komunikacije između čvorova.

Ovisno o problemu koji mreža treba riješiti odlučuje se koliko će mreža imati slojeva i koliko će svaki sloj imati neurona. Slojevi između ulaznog i izlaznog sloja nazivaju se skriveni slojevi. Na slici 3.10. može se vidjeti kako izgleda jednostavna mreža sa samo jednim skrivenim slojem i mreža za dubinsko učenje koja se sastoji od nekoliko skrivenih slojeva.



Sl.3.10. Jednostavna neuronska mreža i dubinska neuronska mreža, izrađeno prema [32]

Skup podataka postavlja se na ulazni sloj neuronske mreže. Podaci s ulaza prosljeđuju se prema sljedećim slojevima mreže preko njihovih veza. Svaki sloj u mreži može imati drugačiju zadaću. Na primjer, ako mreža treba odrediti nalazi li se na slici pas ili mačka, jedan sloj mreže može analizirati dio slike koji prikazuje glavu, drugi sloj može analizirati šape, itd. Svakoj vezi pri ulazu u čvor dodjeljuje se težina. Težina se može pojednostavljeno opisati kao utjecaj pojedine značajke na izlaz. Težine se prosljeđuju po slojevima sa podacima skroz do izlaznog sloja. Svaki čvor ili neuron u mreži prije treniranja inicijaliziran je na neku neodređenu vrijednost, tj. izlaz se određuje pogađanjem. Nakon početnog pokušaja izračunava se razlika između ovog nasumičnog izlaza i stvarne vrijednosti koja bi trebala biti na izlazu. Prema izračunatoj pogrešci mogu se prilagoditi težine veza između neurona i tako poboljšati mogućnost točnog izlaza [31].

Dubinsko učenje i neuronske mreže najčešću upotrebu danas imaju u raspoznavanju slika, računalnom vidu, prepoznavanju ljudskog govora, raznim filterima na društvenim mrežama, kao pomoć u prijevoznim sredstvima, pomoć u medicini, robotima ili programima za igranje igara na ploči, poput šaha ili GO-a.

3.8.5. Pokazatelji kakvoće klasifikatora

Prije implementacije programskog rješenja na Android platformi, provedeno je nekoliko eksperimenata, kreiranih u *Python* programskom jeziku. Svrha ovih eksperimenata je utvrditi koji klasifikator daje najbolje rezultate. Za analizu rezultata korištene su četiri najčešće korištene mjere kvalitete: točnost, preciznost, odziv (engl - *recall*) i *F1*.

Prema [33], točnost klasifikatora je vrijednost koja se može interpretirati kao omjer broja točnih klasifikacija i ukupnog broja podataka koje je potrebno klasificirati. Ova vrijednost se često pogrešno interpretira, tj. visoka točnost ne znači uvijek visoku kvalitetu klasifikatora. Točnost je vrijednost koja dobro opisuje kvalitetu samo u slučajevima kada je broj podataka jednak ili vrlo sličan za svaku klasu. Na primjer, ako klasa A ima 90 uzoraka, a klasa B 10 uzoraka, klasifikator bi postigao točnost od 90%, ako za svaki novi uzorak jednostavno pretpostavi da pripada klasi A. Problem u ovakvim slučajevima nastaje kada je cijena pogrešne klasifikacije velika.

Preciznost je mjera koja se nerijetko poistovjećuje s točnosti, ali te dvije vrijednosti se bitno razlikuju. Prema [34], preciznost govori u koliko slučajeva je klasifikator ispravno predvidio pozitivnu vrijednost. Tj. može se definirati kao omjer točnih pozitivnih predviđanja i ukupnog broja pozitivnih predviđanja. U ovom radu, krivotvoreni potpis predstavlja pozitivan uzorak, dok originalni potpis označava negativan uzorak. Prema tome, preciznost klasifikatora u ovom radu mjeri se kao omjer predviđenih krivotvorenih potpisa i svih krivotvorenih potpisa.

Odziv (engl. *Recall)* mjera kvalitete odgovara na pitanje koliki je udio stvarnih pozitivnih uzoraka ispravno klasificiran [34]. Drugim riječima, *recall* je sposobnost klasifikatora da pronađe sve pozitivne uzorke. Ova mjera naziva se još i osjetljivost, a može se prikazati kao omjer broja točnih pozitivnih predviđanja i ukupnog broja pozitivnih uzoraka [35].

F1 mjera kvalitete je težinski prosjek preciznosti i odziva. *F1* uzima u obzir netočne pozitivne klasifikacije i netočne negativne klasifikacije. Ova mjera je obično bolji pokazatelj kvalitete klasifikatora od točnosti, posebno ako je distribucija podataka nejednaka između klasa. *F1* mjera može se izračunati prema formuli 3-2 [36].

(3-2)

$$F1 = 2 \cdot \frac{Recall \cdot Precision}{Recall + Precision}$$

3.9. Postojeća rješenja na Android platformi

U proteklih nekoliko godina Android platforma doživjela je veliki napredak u raspoznavanju korisnika i primjeni strojnog učenja za rješavanje različitih problema. U području autentifikacije Android telefoni danas imaju nekoliko rješenja. Standardni načini raspoznavanja korisnika korištenjem pinova, crtanjem uzoraka ili upisivanjem lozinki gotovo su u potpunosti zamijenjeni modernijim, sofisticiranijim i jednostavnijim metodama navedenim u sljedećim odlomcima.

3.9.1. Skeniranje otiska prsta

Zbog svoje sigurnosti i jednostavnosti korištenja, prepoznavanje otiska prsta trenutno je najkorišteniji oblik autentifikacije na mobilnim uređajima s Android operacijskim sustavom. Za prepoznavanje se koristi senzor koji je najčešće implementiran u gumb za početni zaslon ili na poleđini uređaja. Gotovo svi uređaji koriste kapacitivni senzor jer se on pokazao najsigurnijim. Ovakav oblik autentifikacije na Androidu podržan je od *SDK* verzije 23 (*Marshmallow).* Iako se najčešće koristi za otključavanje početnog zaslona, moguće ga je implementirati i u aplikacije koje zahtijevaju autentifikaciju prilikom plaćanja ili pristupa osjetljivim podacima, kao zamjenu za lozinke ili pinove. Polaganjem prsta na senzor stvara se virtualna slika otiska analizirajući razine naboja između udubljenja i ispupčenja na vrhovima prstiju. Stvorena virtualna slika pri postavljanju ovog sustava sprema se na uređaj i služi za usporedbu svaki idući puta kada korisnik pokuša otključati uređaj. Najveći sigurnosni problem za ovu metodu čine tzv. „gumeni prsti“ i fotografije visoke razlučivosti isprintane provodljivom tintom. Gumeni prsti su replike vrhova prstiju napravljene pomoću materijala sličnog gumi koje imaju sposobnost uhvatiti dovoljno detalja kako bi uspješno prevarile kapacitivni senzor.

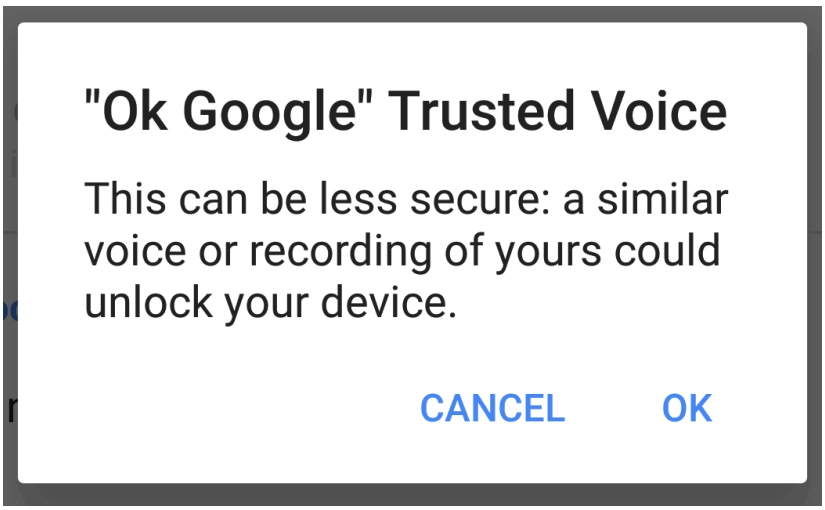
3.9.2. Prepoznavanje lica

Jedna od metoda autentifikacije na Android platformi je i prepoznavanje lica pomoću kamere na uređaju. Prepoznavanje lica smatra se biometrijskom metodom, ali na mobilnim uređajima se

ne koristi često zbog svoje relativno niske razine sigurnosti. Ovaj sustav može se u nekim slučajevima zaobići pokazivanjem ispisane fotografije visoke razlučivosti ili pokazivanjem lica na zaslonu drugog telefona ili računala. Uzevši u obzir ovaj problem i činjenicu da u današnje vrijeme društvene mreže sadrže veliki broj fotografija sa licima korisnika, ova metoda jednostavno nije dovoljno sigurna kako bi se koristila za autentifikaciju, barem ne sa trenutnom razinom kvalitete prepoznavanja stvarnog lica korisnika. Neki uređaji zahtijevaju od korisnika da trepne prilikom postupka prepoznavanja kako bi bili sigurni da se radi o živoj osobi. Unatoč tome ova metoda još uvijek nije pronašla svoju upotrebu u značajnijoj mjeri poput otiska prsta.

3.9.3. Prepoznavanje glasa

Prepoznavanje glasa također je jedan od oblika biometrijske identifikacije koji se može pronaći u upotrebi na Android platformi, najčešće putem Google značajke *Trusted Voice*. Ova značajka omogućava otključavanje telefona izgovaranjem fraze „*Ok Google*“. Ista fraza pokreće i Google asistenta ukoliko je dostupan na telefonu. Problem kod prepoznavanja glasa na mobilnim uređajima sličan je problemu sa prepoznavanjem lica. Korisnikov glas može se snimiti i u slučaju dovoljno kvalitetne snimke, iskoristiti kako bi se neovlašteno pristupilo uređaju. Pri uključivanju ove značajke na uređaju sustav pokaže upozorenje kako ovo nije najsigurnija opcija za korištenje jer se uređaj može otključati snimkom ili sličnim glasom (slika 3.11.).



Sl.3.11. Upozorenje pri aktivaciji značajke Trusted Voice [37]

4. PROGRAMSKO RJEŠENJE

Kao praktični dio diplomskog rada izrađena je aplikacija za Android operacijski sustav koja demonstrira primjenu strojnog učenja u raspoznavanju uzoraka. Program na temelju potpisa fotografiranog pomoću uređaja odlučuje može li korisnik pristupiti sadržaju aplikacije. Pri razvoju programskog rješenja najviše značaja pridodano je obradi slike, tj. pronalaženju značajki koje će se koristiti za treniranje modela te treniranju samog modela.

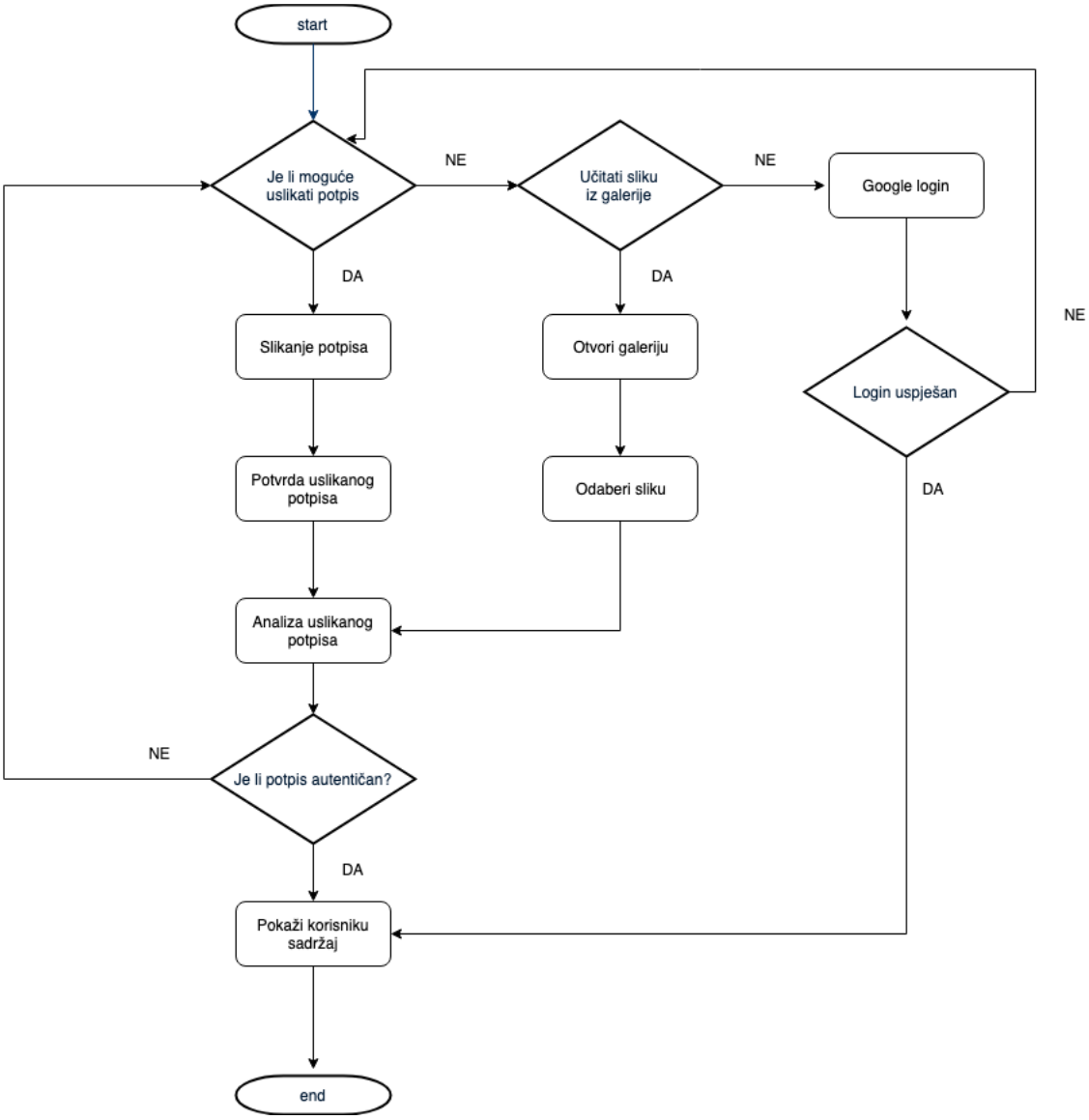
4.1. Zahtjevi na sustav

Tablica 4.1. prikazuje najvažnije zahtjeve na sustav. Osnovne mogućnosti sustava uključuju fotografiranje potpisa, njegovu klasifikaciju, prijavu u sustav u slučaju kada je potpis klasificiran kao originalan, prikaz sadržaja korisniku i odjavu iz sustava. Dodatne mogućnosti sustava su: mogućnost učitavanja fotografije potpisa iz galerije, ukoliko nije moguće napraviti novu fotografiju potpisa te mogućnost prijave putem Google računa, ako korisnik nije u mogućnosti napraviti novu fotografiju ili nema pohranjenu već postojeću fotografiju potpisa na uređaju.

Tab. 4.1. Popis zahtjeva na sustav

ID	Prioritet	Opis
		Generalni zahtjevi korisnika
1	1	Mogućnost fotografiranja potpisa
2	2	Mogućnost učitavanja fotografije s uređaja
3	1	Klasifikacija fotografiranog potpisa
4	1	Prijava u sustav u slučaju da je potpis klasificiran kao originalan
5	1	Prikaz poruke o pogrešci ukoliko potpis nije klasificiran kao original
6	2	Postoji alternativni način prijave u sustav
7	1	Mogućnost odjave iz sustava
8	1	Prikaz sadržaja korisniku

Dijagram toka sustava prikazan je na slici 4.1.



Sl. 4.1. Dijagram toka sustava

4.2. Opis platformi, alata i tehnologija

Programsko rješenje razvijeno je na Android platformi. Android je operacijski sustav tvrtke Google Inc. koji se danas koristi u pametnim telefonima, tabletima, televizorima, satovima, automobilima pa čak i u hladnjacima. Android je u njegovom začetku razvijala tvrtka Android Inc. čiji je cilj bio razviti operacijski sustav za fotoaparate. 2005. godine Google je kupio tvrtku i nastavio sa razvojem operacijskog sustava, ali za mobilne telefone. Prvi uređaj sa Android

operacijskim sustavom predstavljen je 2008. godine. 2011. godine Android je postao najprodavaniji mobilni operacijski sustav, a tu poziciju drži i danas. Posljednja verzija sustava je Android Pie koji je izdan u kolovozu 2018. godine.

Android se temelji na *Linux* jezgri, a njegov izvorni kod je *open-source,* tj. dostupan je pod *Apache* licencom. Većina aplikacija za Android pisana je u Java programskom jeziku koristeći Android SDK (*Software Development Kit*). 2017. godine Google je podržao *Kotlin* kao službeni programski jezik za razvoj Android aplikacija. Android studio je službeno integrirano razvojno okruženje (engl. IDE – *Integrated Development Environment*) tvrtke *Google* za razvoj Android aplikacija, a temeljeno je na *IntelliJ IDEA* okruženju tvrtke *JetBrains*. Android Studio je glavni alat korišten za razvoj ovog programskog rješenja, uz *Kotlin* programski jezik i *XML* jezik za označavanje.

Za treniranje modela i rad sa slikama korišten je Python programski jezik. Python je najčešće korišteni jezik za rad sa strojnim učenjem zbog svoje jednostavnosti i čitljivosti, ali i zbog velikog broja postojećih biblioteka koje uvelike olakšavaju razvoj. Neke od biblioteka korištenih u ovom radu su:

- *scikit-learn*
- *scikit-image*
- *openCV*
- *NumPy*

Scikit-learn je biblioteka iz skupa znanstvenih biblioteka za Python programski jezik koja sadrži unaprijed razvijene alate za lakši rad sa strojnim učenjem. U ovoj biblioteci mogu se pronaći alati koji olakšavaju rad s nadziranim učenjem, nenadziranim učenjem ili odabirom i evaluacijom modela. Također pruža alate za razne načine učitavanja podataka i vršenje transformacija nad njima [38]. Iz *scikit-learn* biblioteke korišten je *Linear Support Vector Machine,* točnije korištena je klasa *LinearSVC* za klasifikaciju potpisa. Ovaj klasifikator detaljnije je objašnjen u poglavlju 3.8.1.

Scikit-image je kolekcija algoritama koji služe za obradu slike. Neki od algoritama koji se mogu pronaći u biblioteci su algoritmi za segmentaciju, geometrijske transformacije, manipulaciju bojama, filtriranje, detekciju značajki i još mnogi drugi [39]. *Feature* modul iz ove biblioteke korišten je za izdvajanje značajki iz slike potpisa kao što je prethodno objašnjeno u poglavlju 3.5.

OpenCV je vrlo popularna *open source* biblioteka koja je razvijena kako bi korisnicima omogućila lakši rad u aplikacijama koje se baziraju na računalnom vidu. Biblioteka ima više od 2500 optimiziranih algoritama, što uključuje klasične i neke od najnovijih algoritama za strojno učenje. Neki od tih algoritama mogu se koristiti za detekciju i prepoznavanje lica, prepoznavanje objekata, praćenje pokreta kamere i slično [40]. *OpencCV* korišten je u ovom diplomskom radu za učitavanje slika i njihovo pretvaranje u *grayscale* format.

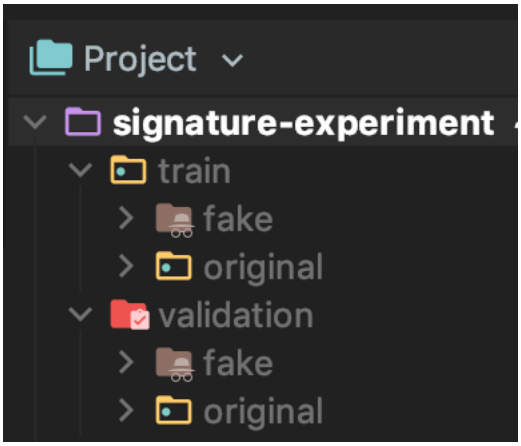
NumPy je osnovna biblioteka za znanstvene izračune u *Python* programskom jeziku. Pruža mogućnost korištenja višedimenzionalnih nizova, različitih izvedenih objekata iz spomenutih nizova i izbor rutina za brzo izvođenje operacija nad nizovima. To uključuje matematičke i logičke operacije, promjene oblika, sortiranje, osnovnu linearnu algebru, statističke operacije te razne druge operacije. Slike se u radu sa strojnim učenjem najčešće pohranjuju u raznim oblicima *NumPy* nizova [41]. U tu svrhu se biblioteka koristi i u ovom radu.

TensorFlow je *open source* biblioteka koja se koristi za zahtjevne brojčane izračune i dolazi sa podrškom za strojno učenje i dubinsko učenje. Koristi *Python* programski jezik kako bi pružio jednostavno sučelje za izgradnju aplikacija, a aplikacije se izvode u C++ programskom jeziku. *TensorFlow* omogućuje programerima da naprave grafove toka podataka (engl. *Dataflow graphs*) - strukture koje opisuju kako se podaci prosljeđuju kroz skup čvorova za obradu u sustavu. Svaki od čvorova u grafu predstavlja matematičku operaciju, a svaka poveznica između čvorova je višedimenzionalni niz podataka – tenzor. *TensorFlow* aplikacije mogu se pokrenuti na lokalnom računalu, u oblaku ili na mobilnim uređajima s iOS i Android operacijskim sustavom. Velika prednost korištenja *TensorFlowa* za strojno učenje je apstrakcija. Na taj način programeri se mogu fokusirati na samu logiku programa, a ne moraju razmišljati o svim detaljima implementacije pojedinih algoritama i matematičkim izračunima [42].

4.3. Usporedba rezultata različitih pristupa

Prije treniranja samog modela za raspoznavanje uzoraka potrebno je prikupiti podatke za treniranje. Za rješavanje ovog problema klasifikacije potpisa prikupljeno je ukupno 523 uzorka. Od ukupnih 523 uzoraka, 192 uzorka označena su kao krivotvoreni, a 331 kao originalni potpisi.

Uzorci su podijeljeni na skup podataka za treniranje i skup podataka za testiranje u omjeru 90:10, što znači da skup podataka za treniranje sadrži 471 uzorak, a skup za testiranje 52 uzorka. Na slici 4.1. prikazana je struktura podataka unutar projekta.



Sl. 4.1. Struktura podataka unutar projekta

U nastavku su u tablicama 4.2., 4.3. i 4.4. prikazane mjere kvalitete pojedinog klasifikatora s *LBP* metodom za izdvajanje značajki. Tablice 4.5., 4.6. i 4.7. prikazuju mjere kvalitete klasifikatora kada je za izdvajanje značajki korištena *HOG* metoda. Za dobivanje rezultata provedeno je 6 eksperimenata kreiranih na računalu, u programskom jeziku *Python*.

Tab. 4.2. Mjere kvalitete SVM klasifikatora s LBP metodom izdvajanja značajki

Lokalni binarni uzorci - SVM				
	Točnost	**Preciznost**	**Odziv**	***F1***
1.	0,98	1,00	0,96	0,98
2.	0,98	1,00	1,00	1,00
3.	0,98	1,00	1,00	1,00
4.	1,00	1,00	0,96	0,98
5.	1,00	1,00	1,00	1,00
6.	0,98	1,00	1,00	1,00
7.	1,00	1,00	1,00	1,00
8.	1,00	1,00	1,00	1,00
9.	1,00	1,00	0,96	0,98
10.	1,00	1,00	1,00	0,98
Min.	0,98	1,00	0,96	0,98
Max.	1,00	1,00	1,00	1,00
Std. dev.	0,01	0,00	0,02	0,01
Prosjek	0,99	1,00	0,99	0,99

U tablici 4.2. prikazani su rezultati analize kvalitete *SVM* klasifikatora u kombinaciji s *LBP* metodom za izdvajanje značajki. Provedeno je 10 mjerenja s nasumično odabranim podacima za testiranje. Iz tablice je vidljivo kako ova kombinacija postiže vrlo visoke rezultate prema svim prikazanim mjerama kvalitete.

Tab. 4.3. Mjere kvalitete kNN klasifikatora s LBP metodom izdvajanja značajki

Lokalni binarni uzorci - kNN				
	Točnost	**Preciznost**	**Odziv**	***F1***
1.	0,98	1,00	0,96	0,98
2.	0,98	1,00	1,00	1,00
3.	0,98	1,00	1,00	1,00
4.	1,00	1,00	0,96	0,98
5.	1,00	1,00	1,00	1,00
6.	0,98	1,00	1,00	1,00
7.	1,00	1,00	1,00	1,00
8.	1,00	1,00	1,00	1,00
9.	1,00	1,00	1,00	1,00
10.	1,00	0,96	1,00	0,98
Min.	0,98	0,96	0,96	0,98
Max.	1,00	1,00	1,00	1,00
Std. dev.	0,01	0,01	0,02	0,01
Prosjek	0,992	0,99	0,99	0,99

U tablici 4.3. prikazani su rezultati klasifikacije korištenjem *kNN* klasifikatora u kombinaciji s *LBP* metodom za izdvajanje značajki. U svih 10 mjerenja, postignute vrijednosti su vrlo slične prethodnom eksperimentu sa *SVM* klasifikatorom.

Tab. 4.4. Mjere kvalitete Random forest klasifikatora s LBP metodom izdvajanja značajki

Lokalni binarni uzorci - Random forest				
	Točnost	**Preciznost**	**Odziv**	***F1***
1.	0,98	1,00	0,96	0,98
2.	1,00	1,00	1,00	1,00
3.	0,98	1,00	1,00	1,00
4.	1,00	1,00	0,96	0,98
5.	1,00	1,00	1,00	1,00
6.	0,98	1,00	1,00	1,00
7.	1,00	1,00	1,00	1,00
8.	1,00	1,00	1,00	1,00
9.	1,00	0,96	1,00	0,98
10.	1,00	1,00	0,96	1,00
Min.	0,98	0,96	0,96	0,98
Max.	1,00	1,00	1,00	1,00
Std. dev.	0,01	0,01	0,02	0,01
Prosjek	0,99	0,99	0,99	0,99

Korištenje algoritma slučajnih šuma za klasifikaciju također daje vrlo slične rezultate kao i klasifikatori u prethodna dva eksperimenta. Sve četiri korištene mjere kvalitete imaju vrlo visok prosjek i nisko odstupanje od prosjeka.

Tab. 4.5. Mjere kvalitete *SVM* klasifikatora s *HOG* metodom izdvajanja značajki

HOG - SVM				
	Točnost	**Preciznost**	***Recall***	***F1***
1.	0,90	0,91	0,85	0,88
2.	0,88	0,86	0,92	0,89
3.	0,89	0,90	0,96	0,93
4.	0,88	1,00	1,00	1,00
5.	0,92	0,92	1,00	0,96
6.	1,00	1,00	1,00	1,00
7.	0,96	0,9	0,96	0,93
8.	1,00	0,92	0,96	0,94
9.	0,92	0,84	0,92	0,88
10.	0,94	1,00	0,85	0,96
Min.	0,88	0,84	0,85	0,88
Max.	1,00	1,00	1,00	1,00
Std. dev.	0,06	0,06	0,06	0,04
Prosjek	0,93	0,93	0,94	0,94

Ako se usporede rezultati analize kvalitete *SVM* klasifikatora pri korištenju *LBP* metode s rezultatima kada se korisit u kombinaciji s *HOG* metodom, može se vidjeti da u drugom slučaju klasifikator postiže neznatno lošije vrijednosti. Prosjek svih mjera i dalje iznosi preko 90%.

Tab. 4.6. Mjere kvalitete kNN klasifikatora s HOG metodom izdvajanja značajki

HOG - kNN				
	Točnost	**Preciznost**	**Odziv**	***F1***
1.	0,86	0,96	0,92	0,94
2.	0,94	0,84	0,92	0,88
3.	0,94	0,9	0,92	0,91
4.	0,88	1,00	0,96	0,98
5.	0,9	0,87	1,00	0,93
6.	0,98	1,00	0,92	0,96
7.	0,92	0,92	1,00	0,96
8.	0,96	0,97	0,96	0,91
9.	0,96	0,97	0,96	0,91
10.	0,9	1,00	0,92	0,96
Min.	0,86	0,84	0,92	0,88
Max.	0,98	1,00	1,00	0,98
Std. dev.	0,04	0,06	0,03	0,03
Prosjek	0,92	0,94	0,95	0,93

U tablici 4.6. može se vidjeti kako korištenje *kNN* klasifikatora u kombinaciji s *HOG* metodom za izdvajanje značajki daje nešto slabije rezultate u usporedbi s prvim eksperimentom gdje se *kNN* koristio s *LBP* metodom. Prosječna vrijednost rezultata mjerenja na 10 različitih primjera za sve mjere kvalitete iznosi preko 90%. Ovakvi rezultati su također vrlo zadovoljavajući.

Tab. 4.7. Mjere kvalitete Random forest klasifikatora s HOG metodom izdvajanja značajki

HOG - Random forest				
	Točnost	**Preciznost**	**Odziv**	***F1***
1.	0,79	0,96	0,73	0,83
2.	0,83	1,00	0,80	0,89
3.	0,75	0,91	0,73	0,81
4.	0,90	0,92	0,92	0,92
5.	0,83	0,87	0,85	0,86
6.	0,92	1,00	0,92	0,96
7.	0,86	0,97	0,80	0,875
8.	0,96	0,83	0,77	0,80
9.	0,88	0,77	0,85	0,81
10.	0,81	1,00	0,77	0,92
Min.	0,75	0,77	0,73	0,80
Max.	0,96	1,00	0,92	0,96
St. devijacija	0,06	0,0791693	0,07	0,06
Prosjek	0,85	0,923	0,81	0,87

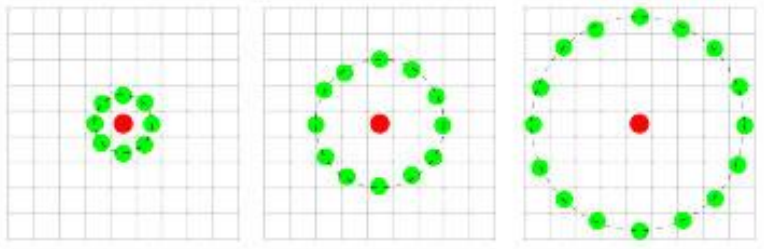
Analizom rezultata pojedinih klasifikatora i metoda za izdvajanje značajki ustavljeno je da korištenje metode lokalnih binarnih uzoraka pruža bolje rezultate u odnosu na korištenje *HOG* metode. Sva tri klasifikatora daju vrlo slične rezultate u kombinaciji s *LBP* metodom. Stoga je odlučeno da se za implementaciju sustava na Android platformi koristi *LBP* metoda za izdvajanje značajki i *kNN* klasifikator. Budući da je aplikacija izrađena u svrhu demonstracije rješenja, odabran je *kNN* klasifikator zbog jednostavnosti njegove implementacije.

4.3.1. LBP i kNN

Pri kreiranju eksperimenata u *Pythonu,* za pronalaženje značajki u slici potpisa korištena je implementacija lokalnih binarnih uzoraka iz *scikit-image* biblioteke, točnije iz *feature* modula. Kreirana je klasa *LocalBinaryPatterns* koja ima 2 atributa:

- *numPoints* – predstavlja broj točaka oko središnjeg piksela koje se uzimaju u obzir pri usporedbi
- *radius* – radijus kružnice oko središnjeg piksela

Prvi kvadrat na slici 4.2. pikazuje *numPoints=8* i *radius=1* atribute s njihovim pripadnim vrijednostima koji se koriste u ovom radu.



Sl. 4.2. Primjer različitih vrijednosti parametara numPoints i radius [19]

Unutar *LocalBinaryPatterns* klase definirana je metoda *computeLbp(image)* (slika 4.3.) koja prima samo jedan parametar – sliku koju je potrebno analizirati. Unutar *computeLbp* metode koristi se *local_binary_pattern* metoda iz prethodno spomenutog *feature* modula za konkretno izračunavanje lokalnih binarnih uzoraka. Ova metoda kao argumente prima sliku, *numPoints i radius* atribute te „*uniform*“ argument, koji je prethodno objašnjen u poglavlju 3.5. Rezultat koji ova metoda vraća ne može se izravno koristiti kao vektor značajki jer je rezultat dvodimenzionalni niz podataka, istih dimenzija kao i početna slika. Da bi se konstruirao vektor značajki, potrebno je napraviti histogram koji prikazuje broj pojavljivanja svakog od uzoraka pronađenih na slici. Za konstruiranje histograma koristi se *histogram* metoda iz *numpy* biblioteke. Dobiveni histogram predstavlja vektor značajki i kao takav se može koristiti pri treniranju klasifikatora.

```
class LocalBinaryPatterns:
    def __init__(self, numPoints, radius):
        self.numPoints = numPoints
        self.radius = radius

    def computeLbp(self, image, eps=1e-7):
        lbp = feature.local_binary_pattern(image, self.numPoints,
                                           self.radius, method="uniform")
        (hist, _) = np.histogram(lbp.ravel(),
                                 bins=np.arange(0, self.numPoints + 3),
                                 range=(0, self.numPoints + 2))

        # normalize the histogram
        hist = hist.astype("float")
        hist /= (hist.sum() + eps)

        # return the histogram of Local Binary Patterns
        return hist
```

Sl. 4.3. LocalBinaryPattern klasa i computeLbp metoda zadužene za kreiranje LBP histograma

Za treniranje klasifikatora kreirana je nova *Python* datoteka. U njoj je kreirana instanca prethodno napisane *LocalBinaryPatterns* klase te dva niza u koje će biti pohranjeni vektori

značajki i oznake klasa. Za svaku sliku iz skupa podataka za treniranje potrebno je napraviti sljedeće:

- učitati sliku
- pretvoriti ju u crno-bijelu sliku
- pronaći *LBP* histogram i spremiti ga u kreirani niz

Kod na slici 4.4. prikazuje prethodno opisane korake.

```
# initialize the local binary patterns descriptor along with
# the data and label arrays
desc = LocalBinaryPatterns(8, 1)
data = []
labels = []

# looping through training images
for imagePath in paths.list_images(args['training']):
    # load image, convert to grayscale and describe it
    image = cv2.imread(imagePath)
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    hist = desc.computeLbp(gray)

    # extract the label from image path then update the label and data lists
    labels.append(imagePath.split(os.path.sep)[-2])
    data.append(hist)
```

Sl. 4.4. Učitavanje slika i pronalazak LBP histograma za svaku sliku

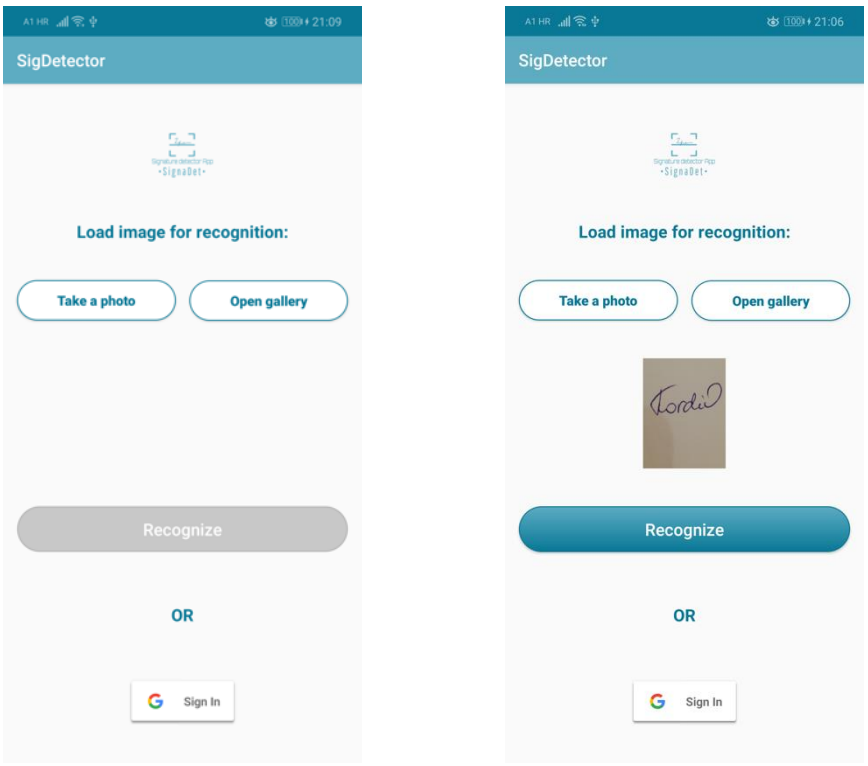
Nakon što su sve slike pretvorene u crno-bijeli format i nakon što je za svaku sliku određen *LBP* histogram, moguće je započeti treniranje modela. U ovom rješenju, za treniranje je korišten algoritam k najbližih susjeda. Potrebno je kreirati instancu klase *KNeighborsClassifier* koja se nalazi u *scikit-learn* biblioteci i na njoj pozvati metodu *fit(data, labels).* Metoda *fit* prima dva parametra – *data* i *labels.* Prvi parametar predstavlja niz koji sadrži *LBP* histograme slika iz skupa za treniranje dok drugi parametar predstavlja njihove oznake.

```
# train a classifier on the data
knn_classifier = KNeighborsClassifier()
knn_classifier.fit(data, labels)
```

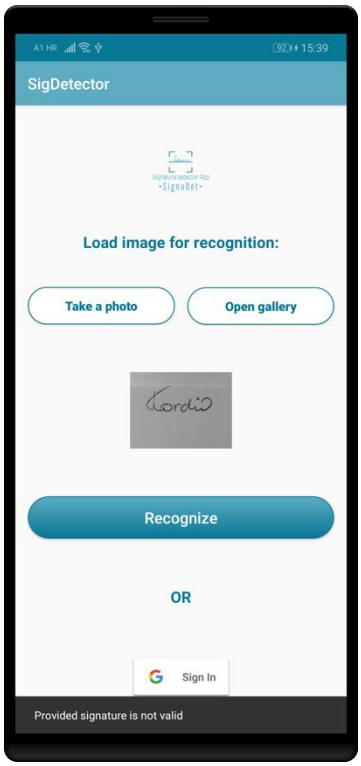
Sl. 4.5. Treniranje kNN klasifikatora

4.4. Način rada sustava

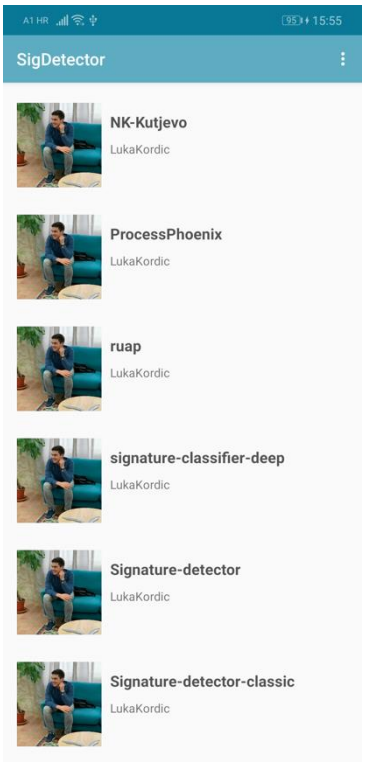
Pri pokretanju aplikacije korisnik ima mogućnost odabrati želi li se u aplikaciju prijaviti korištenjem Google računa ili analizom fotografije potpisa (slika 4.6.). Ukoliko se odluči za drugi način, ima mogućnost napraviti novu fotografiju, ili učitati neku od postojećih iz galerije uređaja. Mogućnost prijave u aplikaciju putem Google računa omogućena je kao alternativa ukoliko korisnik trenutno nije u mogućnosti fotografirati potpis ili nema prethodno spremljenu sliku potpisa na uređaju. Nakon što je odabrana željena fotografija, započinje proces njene klasifikacije. U slučaju kada sustav prepozna potpis kao autentičan, korisnik se automatski prijavljuje u sustav i prikazuje mu se sadržaj aplikacije. U suprotnom, prikazuje se odgovarajuća poruka pogreške (slika 4.7.). Zaslon koji se otvara nakon uspješne prijave u sustav prikazuje popis repozitorija s Github-a (slika 4.8.). Klikom na gumb u dodatnim opcijama ovog zaslona, korisnik se može odjaviti iz sustava, nakon čega se preusmjerava na početni zaslon.



Sl. 4.6. Početni zaslon aplikacije prije odabira fotografije (lijevo) i nakon odabira fotografije (desno)

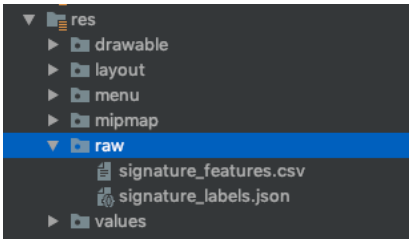


Sl. 4.7. Prikaz poruke o pogrešci



Sl.4.8. Prikaz sadržaja aplikacije

Za analizu fotografija na Android platformi korišteni su podaci iz prethodno kreiranog eksperimenta na računalu. Nakon izdvajanja značajki i kreiranja histograma u *Pythonu*, podaci se zapisuju u datoteku *.csv* formata. Svaki red u ovoj datoteci predstavlja broj ponavljanja lokalnih binarnih uzoraka za pojedinu sliku. Oznake iz skupa podataka za treniranje pohranjuju se u datoteku *.json* formata. Obje datoteke uključene su u Android aplikaciju, a pohranjene su u *raw* direktoriju unutar projekta (slika 4.9.).



Sl. 4.9. Lokacija datoteka unutar Android projekta

Nakon što korisnik odabere fotografiju potpisa kojom se želi prijaviti u sustav, ponavlja se postupak koji je prethodno proveden za sve slike u *Python* eksperimentu. Na slici se pronalaze

lokalni binarni uzorci i kreira se njihov histogram. Dobiveni histogram se zatim prosljeđuje *kNN* klasifikatoru kao vektor značajki. Budući da se radi o *kNN* klasifikatoru, ne postoji klasični proces treniranja, nego se svi podaci iz skupa za treniranje drže u memoriji. Kako bi odredio je li potpis originalan, algoritam računa i uspoređuje euklidsku udaljenost novonastalog vektora značajki sa svim podacima iz skupa za treniranje. Tijekom ovog postupka prati se najkraća trenutna udaljenost i indeks na kojemu se nalazi element s najkraćom udaljenosti. Nakon njenog pronalaska, algoritam kao rezultat klasifikacije vraća oznaku elementa na dobivenom indeksu.

5. ZAKLJUČAK

Zadatak ovog diplomskog rada bio je proučiti i implementirati autentifikaciju korisnika raspoznavanjem uzoraka na Android operacijskom sustavu. U teorijskom dijelu rada analizirane su i objašnjene neke od metoda za raspoznavanje korisnika u računalnim sustavima. Naglasak je stavljen na biometrijske metode raspoznavanja korisnika i strojno učenje. Proučeni su neki od najčešće korištenih načina biometrijske identifikacije i za svaku biometrijsku metodu objašnjene su njene specifičnosti te navedene prednosti i nedostatci. Prikazani su potencijalni problemi i sigurnosni propusti metoda pri njihovom korištenju na mobilnim uređajima. Objašnjen je pojam strojnog učenja te neki od načina na koje se ono može primijeniti. Najviše pažnje pridodano je nadziranom strojnom učenju i klasifikaciji. Pomoću eksperimenata provedenih u *Python* programskom jeziku, napravljena je usporedba triju klasifikatora i triju metoda izdvajanja značajki. Analizom rezultata eksperimenata utvrđeno je da najbolje rezultate daje kombinacija metode lokalnih binarnih uzoraka za izdvajanje značajki i *SVM* ili *kNN* klasifikatora. Prema tome, *kNN* klasifikator i *LBP* metoda izabrani su za implementaciju rješenja u praktičnom dijelu rada.

U praktičnom dijelu rada razvijena je aplikacija za Android operacijski sustav, koja omogućava korisniku autentifikaciju u sustav korištenjem biometrijske metode prepoznavanja potpisa. Korisnik ima opciju fotografirati potpis ili učitati fotografiju potpisa iz galerije na uređaju. Ukoliko nije u mogućnosti dati sliku postoji opcija prijave u sustav putem Google računa. Za programiranje aplikacije te *LBP* i *kNN* algoritama, korišten je *Kotlin* programski jezik. Za izgradnju sučelja korišten je *XML* jezik za označavanje.

Ova verzija aplikacije je samo prototip i napravljena je isključivo u svrhu demonstracije rješenja predstavljenog u diplomskom radu. Algoritmi odabrani za realizaciju aplikacije pokazali su zadovoljavajuće rezultate tijekom testiranja, ali budući da su za usporedbu korištena samo 3 klasifikatora i dvije metode za izdvajanje značajki, postoji mogućnost da se za rješenje ovog problema mogu postići bolji rezultati korištenjem nekih od klasifikatora ili metoda koje nisu uzete u obzir u ovom radu. Primjerice, korištenjem kompleksnijih metoda iz područja strojnog učenja, poput dubokih neuronskih mreža, mogli bi se postići mnogo bolji rezultati. Uvjet za to je da postoji veliki skup podataka za treniranje.

LITERATURA

[1] Ured vijeća za nacionalnu sigurnost, UVNS, Što je to informacijska sigurnost?, https://www.uvns.hr/hr/sto-je-to-informacijska-sigurnost [pristupljeno: 8. rujna 2019.]

[2] Panda mediacenter, 52% of users reuse their passwords, Panda security, https://www.pandasecurity.com/mediacenter/security/password-reuse/ [pristupljeno: 8.rujna 2019.]

[3] Ž. Radmilović, Biometrijska identifikacija, Policijska sigurnost, br. 3-4, str. 161, kolovoz 2008.

[4] F. Ennaama, K.Benhida, A. Boulahoual, Comparative and analysis study of biometric systems, Journal of Theoretical and Applied Information Technology, br. 12, sv. 97, str. 3466 – 3476, lipanj 2019.

[5] Leksikografski zavod Miroslav Krleža, Daktiloskopija, Hrvatska enciklopedija, http://www.enciklopedija.hr/Natuknica.aspx?ID=13709, [pristupljeno: 15. rujna 2019.]

[6] N. Gogoi, How In-display fingerprint sensors work on smartphones, Guiding Tech, 2018., https://www.guidingtech.com/how-in-display-fingerprint-scanners-work/ [pristupljeno: 18. kolovoza 2019.]

[7] Ell mobile, Eyes to the world, The new Honor 8, http://www.ellmobile.com/products/huawei-honor-8, [pristupljeno: 15.rujna 2019.]

[8] C. Marr, Retinal scans and fingerprint checks: High tech or high risk?, PHYS, 22. travnja 2016., https://phys.org/news/2016-04-retinal-scans-fingerprint-high-tech.html, [pristupljeno: 15. rujna 2019.]

[9] D. N. Parmar, B. B. Mehta, Face recognition methods and applications, International Journal of Computer Applications in Technology, br. 1, sv. 4, str. 84-86, 2014.

[10] F. Tepper, Face ID is replacing Touch ID on the new iPhone X, Tech Crunch, 12. rujna 2017, https://techcrunch.com/2017/09/12/face-id-is-replacing-touch-id-on-the-new-iphone-x/, [pristupljeno: 20. kolovoza 2019.]

[11] Australian cyber security magazine, Facial recognition 20. prosinca 2018., https://australiancybersecuritymagazine.com.au/facial-recognition/ [pristupljeno: 15. rujna 2019.]

[12] R. Triggs, Google Assistant guide: Make the most of your virtual assistant, Android authority, 14. ožujka 2018., https://www.androidauthority.com/google-assistant-838138/, [pristupljeno: 15. rujna 2019.]

[13] View sonic, https://www.viewsonic.com/me/products/pendisplay/PD1011.php, [pristupljeno: 5. veljače 2019.]

[14] Training and test sets: Splitting data, Google developers, 2019., https://developers.google.com/machine-learning/crash-course/training-and-test-sets/splitting-data, [pristupljeno: 20. kolovoza 2019.]

[15] Damien, How to split a dataset, Beyond the lines, 2016., https://www.beyondthelines.net/machine-learning/how-to-split-a-dataset/, [pristupljeno: 18. kolovoza 2019.]

[16] M. Ved, Feature Selection and Feature Extraction in Machine Learning: An Overview, 19. srpnja 2018., https://medium.com/@mehulved1503/feature-selection-and-feature-extraction-in-machine-learning-an-overview-57891c595e96, [pristupljeno: 15. rujna 2019.]

[17] G. Press, Cleaning Big Data: Most time consuming, least enjoyable data science task, survey says, Forbes, 2016., https://www.forbes.com/sites/gilpress/2016/03/23/data-preparation-most-time-consuming-least-enjoyable-data-science-task-survey-says/?source=post_page-----------------------#5b2b89346f63, [pristupljeno: 20. srpnja 2019.]

[18] W. Koehrsen, Automated Feature Engineering in Python, Medium, 2018., https://towardsdatascience.com/automated-feature-engineering-in-python-99baf11cc219, [pristupljeno: 21. kolovoza 2019.]

[19] A. Rosebrock, Local binary patterns with Python and OpenCv, Pyimagesearch, 2015., https://www.pyimagesearch.com/2015/12/07/local-binary-patterns-with-python-opencv/, [pristupljeno: 21. kolovoza 2019.]

[20] T. Ojala, M. Pietikäinen, T. Mäenpää, Gray Scale and Rotation Invariant Texture Classification with Local Binary Patterns. Lecture Notes in Computer Science, sv. 1842, str. 404-420, 2000.

[21] Computerphile, Faces & the Local Binary Pattern, Youtube, 21. listopada 2015., https://www.youtube.com/watch?v=wpAwdsubl1w, [pristupljeno: 24. kolovoza 2019.]

[22] Histograms – 1: Find, Plot, Analyze!, Open Source Computer Vision, https://docs.opencv.org/master/d1/db7/tutorial_py_histogram_begins.html, [pristupljeno: 15. rujna 2019.]

[23] OpenCV Python Program to analyze an image using Histogram, GeeksforGeeks, https://www.geeksforgeeks.org/opencv-python-program-analyze-image-using-histogram/, [pristupljeno: 15.rujna 2019.]

[24] N. Dalal, B. Triggs, Histograms of Oriented Gradients for Human Detection. International Conference on Computer Vision & Pattern Recognition. sv. 2, str. 886–893, lipanj 2005.

[25] A. Rosebrock, Histogram of oriented gradients, Pyimagesearch, https://gurus.pyimagesearch.com/lesson-sample-histogram-of-oriented-gradients-and-car-logo-recognition/#, [pristupljeno: 11. rujna 2019.]

[26] N. Bambrick, Support Vector Machines: A Simple Explanation, KDnuggets, 2016. https://www.kdnuggets.com/2016/07/support-vector-machines-simple-explanation.html, [pristupljeno: 11. rujna 2019.]

[27] Scikit-learn, Nearest Neighbors, scikit-learn, https://scikit-learn.org/stable/modules/neighbors.html , [pristupljeno: 11. rujna 2019.]

[28] Genesis, Pros and cons of K-Nearest Neighbors, Genesis, 25. rujna 2018. https://www.fromthegenesis.com/pros-and-cons-of-k-nearest-neighbors/, [pristupljeno: 11. rujna 2019.]

[29] Synced, How Random Forest Algorithm Works in Machine Learning, Synced, 24. listopada 2017., https://syncedreview.com/2017/10/24/how-random-forest-algorithm-works-in-machine-learning/, [pristupljeno: 11. rujna 2019.]

[30] M. Barnwal, Random forests explained intuitively, KDnuggets, 2019, https://www.kdnuggets.com/2019/01/random-forests-explained-intuitively.html, [pristupljeno: 11. rujna 2019.]

[31] Machine Learning crash course, Neural networks, Google developers, https://developers.google.com/machine-learning/crash-course/, [pristupljeno: 11 rujna 2019.]

[32] S. Kampakis, What deep learning is and isn't, The Data Scientist, 17. travnja 2018., https://thedatascientist.com/what-deep-learning-is-and-isnt/, [pristupljeno: 15. rujna 2019.]

[33] A.Mishra, Metrics to evaluate your machine learning algorithm, Medium, veljača 2018., https://towardsdatascience.com/metrics-to-evaluate-your-machine-learning-algorithm-f10ba6e38234, [pristupljeno: 12 rujna 2019.]

[34] Machine Learning crash course, precision and recall, Google developers, https://developers.google.com/machine-learning/crash-course/classification/precision-and-recall, [pristupljeno: 12. rujna 2019.]

[35] Precision, recall and F measures, scikit-learn, https://scikit-learn.org/stable/modules/model_evaluation.html#precision-recall-f-measure-metrics, [pristupljeno: 12. rujna 2019.]

[36] R. Joshi, Accuracy, precision, recall & F1 score: interpretation of performance measures, Exsilio, 9. rujna 2016, https://blog.exsilio.com/all/accuracy-precision-recall-f1-score-interpretation-of-performance-measures/ [pristupljeno: 12. rujna 2019.]

[37] S. Whalen, Google Pixel phones can be unlocked with a recording of a trusted voice by default, Sean the geek, 23. listopada 2016., https://seanthegeek.net/190/google-pixel-phones-unlocked-trusted-voice-recording/amp/, [pristupljeno: 15. rujna 2019.]

[38] J. Brownlee, Machine Learning Mastery, travanj 2014. https://machinelearningmastery.com/a-gentle-introduction-to-scikit-learn-a-python-machine-learning-library/, [pristupljeno: 12 rujna 2019.]

[39] Scikit-image, Overview, scikit-image, https://scikit-image.org/docs/stable/, [pristupljeno: 12. rujna 2019.]

[40] OpenCV Team, About, OpenCV https://opencv.org/about/, [pristupljeno: 12. rujna 2019.]

[41] NumPy developers, NumPy, NumPy, https://numpy.org/, [pristupljeno: 12. rujna 2019.]

[42] TensorFlow, TensorFlow Guide, TensorFlow, https://www.tensorflow.org/learn, [pristupljeno: 12. rujna 2019.]

SAŽETAK

U teorijskom dijelu ovog diplomskog rada opisuju se moderne metode raspoznavanja i autentifikacije korisnika u računalnim sustavima. Objašnjena je važnost i načini raspoznavanja korisnika. Obrađena je biometrijska identifikacija, neki njeni tipovi i njihove specifičnosti. Naglasak je stavljen na metode zasnovane na postupcima strojnog učenja i raspoznavanja uzoraka. Posebno je obrađena metoda raspoznavanja korisnika na temelju potpisa. Opisani su neki postupci strojnog učenja, algoritmi za klasifikaciju, dubinsko učenje i istražena su postojeća rješenja autentifikacije korisnika na Android platformi. U praktičnom dijelu rada ostvareno je programsko rješenje za raspoznavanje korisnika na mobilnoj platformi na temelju potpisa. Programsko rješenje izrađeno je za Android platformu i koristi fotoaparat za slikanje potpisa. Fotografija se prosljeđuje istreniranom modelu koji određuje je li potpis originalan. Ukoliko je uspješno autentificiran, korisnik je u mogućnosti pristupiti sadržaju aplikacije. U protivnom mu se pristup zabranjuje.

Ključne riječi: Android, autentifikacija, biometrija, klasifikacija, strojno učenje

ABSTRACT

Pattern recognition based user authentication on the Android platform

In the first part of this thesis, modern user authentication and recognition methods in computer systems are described. It is described how important user recognition can be and which ways of recognizing users exist. Biometric identification and its types are explained. Methods based on machine learning and classification methods are given emphasis in this paper, especially signature recognition methods. Furthermore, some important classification algorithms and machine learning techniques are explained and existing solutions to these topics are explored for Android platform. An Android application has been developed as a practical part of this thesis. Application is using phone's camera for capturing photos of user's signature and then forwards taken photo to machine learning model which determines whether the signature is original or fake. If successfully authenticated, user is able to access application's content.

Keywords: Android, authentication, biometrics, classification, machine learning

ŽIVOTOPIS

Luka Kordić rođen je 14. veljače 1995. godine u Požegi. Osnovnu školu pohađa u Kutjevu, a nakon završene osnovne škole upisuje srednju Tehničku školu u Požegi. 2013. godine diplomira na Tehničkoj školi te upisuje Fakultet elektrotehnike, računarstva i informacijskih tehnologija u Osijeku – smjer Računarstvo. 2016. godine završava preddiplomski sveučilišni studij i upisuje sveučilišni diplomski studij – smjer Programsko inženjerstvo. Tijekom studiranja odradio je stručnu praksu kao Android programer u tvrtki COBE d.o.o. gdje je sada i zaposlen. Radio je i kao demonstrator na laboratorijskim vježbama iz kolegija Razvoj mobilnih aplikacija. Tijekom ovih aktivnosti stekao je znanja o Java i Kotlin programskim jezicima, Android studio razvojnom okruženju, najboljim praksama u razvoju aplikacija za Android platformu te o raznim bibliotekama korištenim u razvoju aplikacija.

_____

Luka Kordić

Prilozi

- Autentifikacija uzoraka raspoznavanjem uzoraka na Android platformi u .docx formatu
- Autentifikacija uzoraka raspoznavanjem uzoraka na Android platformi u .pdf formatu
- Izvorni kod Android aplikacije
- Izvorni kod *Python* eksperimenta