Identifikacija osobe temeljena na TensorFlow biblioteci za strojno učenje

Bračun, Juraj

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

Juraj Bračun

Zagreb, 2022.

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

Mentori:

Doc. dr. sc. Tomislav Stipančić, dipl. ing.

Student:

Juraj Bračun

Zagreb, 2022.

Izjavljujem da sam ovaj rad izradio samostalno koristeći znanja stečena tijekom studija i navedenu literaturu.

Posebno se zahvaljujem mentoru doc. dr. Tomislavu Stipančiću na pomoći i vremenu odvojenom kako bi se ovaj rad mogao uspješno napraviti.

Posebno hvala i prijateljima Doni, Goranu i Mateu na pomoći oko testiranja programa za prepoznavanje lica.

Hvala i svim ostalim prijateljima, kolegama i kolegicama s fakulteta na podršci i ugodnom druženju tijekom studiranja. Također, zahvaljujem svojoj obitelji te svima ostalima koji su imali povjerenja u mene i moje sposobnosti kako bi uspješno mogao završiti ovaj fakultet.

Juraj Bračun

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE
Središnje povjerenstvo za završne i diplomske ispite
Povjerenstvo za diplomske radove studija strojarstva za smjerove:
proizvodno inženjerstvo, računalno inženjerstvo, industrijsko inženjerstvo i menadžment,
inženjerstvo materijala te mehatronika i robotika

Sveučilište u Zagrebu Fakultet strojarstva i brodogradnje	
Datum:	Prilog:
Klasa: 602-04/22-6/1	
Ur. broj: 15-1703-22-	

DIPLOMSKI ZADATAK

Student: **JURAJ BRAČUN** Mat. br.: 0035210477

Naslov rada na hrvatskom jeziku: **Identifikacija osobe temeljena na TensorFlow biblioteci za strojno učenje**

Naslov rada na engleskom jeziku: **Person identification based on the TensorFlow machine learning library**

Opis zadatka:

Računalni modeli temeljeni na algoritmima umjetne inteligencije omogućavaju automatsko prepoznavanje objekata koristeći vizijske senzore smještene u stvarnu okolinu. Prilikom identifikacije osobe koristeći informacije s lica moguće je koristiti različite biblioteke koje u sebi sadržavaju potrebnu metodologiju. U radu je potrebno izraditi cjelovito softversko rješenje za identifikaciju osobe temeljeno na TensorFlow biblioteci strojnog učenja.

U okviru rada potrebno je:

- proučiti metode za detekciju i identifikaciju osobe koristeći lice u stvarnom vremenu,
- upoznati se s TensorFlow programskom bibliotekom te koristiti prikladne metode u svrhu prepoznavanja i identifikacije osobe,
- metodologiju za prepoznavanje temeljiti na tzv. prenesenom učenju (eng. Transfer Learning) i konvolucijskoj neuronskoj mreži,
- analizirati učinkovitost razvijenog modela u ovisnosti o kutu gledanja.

Dobiveno softversko rješenje je potrebno eksperimentalno evaluirati uključivši ljudske subjekte.
U radu je potrebno navesti korištenu literaturu te eventualno dobivenu pomoć.

Zadatak zadan:
5. svibnja 2022.

Rok predaje rada:
7. srpnja 2022.

Predviđeni datum obrane:
18. srpnja do 22. srpnja 2022.

Zadatak zadao:
doc. dr. sc. Tomislav Stipančić

Predsjednica Povjerenstva:
prof. dr. sc. Biserka Runje

SADRŽAJ

POPIS SLIKA

POPIS TABLICA

POPIS OZNAKA

Oznaka	Jedinica	Opis
bias	-	dodatni neuron na ulaznu u sloj čija je vrijednost uvijek 1
d	-	željeni izlaz mreže
E	-	funkcija cilja, iznos pogreške
G	-	iznos Gausove funkcije
G_x	-	derivacija funkcije intenziteta u smjeru osi x
G_y	-	derivacija funkcije intenziteta u smjeru osi y
H	-	iznos *Cross-Entropy* funkcije cilja
I	-	broj neurona ulaznog sloja mreže
J	-	broj neurona sakrivenog sloja mreže
K	-	broj neurona izlaznog sloja mreže
Kp	-	nagib linearne aktivacijske funkcije
N	-	broj parova ulazno-izlaznih vrijednosti seta učenja
n	-	broj ulaza u razmatrani neuron
net	-	vrijednost funkcije sume
net_j	-	vrijednost funkcije sume j-tog neurona
net_H	-	vrijednost funkcije sume j-tog neurona skrivenog sloja
net_{Ok}	-	vrijednost funkcije k-tog neurona izlaznog sloja
O_k	-	k-ti izlaz neuronske mreže
$p(x)$	-	vjerojatnost događaja x u distribuciji p
$q(x)$	-	vjerojatnost događaja x u distribuciji q
u	-	vrijednost ulaza u neuron
$\boldsymbol{V}$	-	matrica težina skrivenog sloja
$\boldsymbol{W}$	-	matrica težina izlaznog sloja
w_j	-	vrijednost težine j-tog ulaza u neuron
x	-	udaljenost od x osi, događaj u distribuciji vjerojatnosti
y	-	udaljenost od y osi, vrijednost izlaza neurona
Z	-	vrijednosti na ulaznim neuronima
α	-	koeficijent momentuma
$\Delta\vartheta$	-	promjena parametara učenja
δ	-	parametar algoritma povratnog prostiranja pogreške
δ_{Ok}	-	parametar algoritma povratnog prostiranja pogreške izlaznog sloja
θ	-	kut koji gradijent zatvara sa x osi
ϑ	-	parametar učenja, težinski koeficijent
η	-	koeficijent brzine učenja
σ	-	standardno odstupanje, širina Gaussove funkcije

SAŽETAK

Područje umjetne inteligencije u današnje se vrijeme koristi gotovo u svakoj grani tehnologije kao i u sve većem broju strojeva i predmeta kojima svaki dan rukujemo. Neki su se dijelovi umjetne inteligenciju razvili do stupnja gdje je čovjek postao gotovo ovisan o rezultatima koje takvi, pametni sustavi daju. Primjena takvih sustava uvelike je promijenila način na koji ljudi pristupaju određenim tehničkim problemima. Njihova je glavna prednost, mogućnost samostalnog i kontinuiranog učenja koje konstantno unaprjeđuje preciznost i točnost takvih sustava, čime se oponaša djelovanje ljudskog mozga koji na isti način funkcionira u realnom okruženju. Jedna vrsta takvih pametnih sustava koja je postala veoma popularna u zadnjih desetak ili više godina, svakako je sposobnost računala da prepozna različita ljudska lica. Iako je taj proces u ljudskom mozgu gotovo automatiziran te ne predstavlja nikakav poseban napor, za računalo to je ozbiljan izazov kojega nije bilo lako riješiti.

U ovome će se radu kreirati upravo jedan program čiji će zadatak biti prepoznavanje ljudskih lica u realnom vremenu. Da bi se to moglo ostvariti, potrebno je koristiti dva odvojena sustava, jedan zadužen za detekciju lica na slici, a drugi zadužen za prepoznavanje osobe čije se lice nalazi na slici. Korištenje oba sustava temelji se na tzv. prenesenom učenju, gdje se već kreirani sustavi prilagođavaju i modificiraju za neke specifične potrebe, u ovom slučaju za izradu sustava za prepoznavanje lica.

Ključne riječi: umjetna inteligencija, računalni vid, neuronske mreže, detekcija lica, prepoznavanje lica

SUMMARY

Today, artificial intelligence is being used in almost every aspect of technology, as well as in growing number of machines and gadgets that we use on daily basis. Some areas of artificial intelligence are so advanced, that people almost depend on the results which such smart systems give on their output. Application of those systems has greatly changed the way we approach different technical problems. Main advantage of modern, smart systems is the capability to continuously learn by themselves, which results in much higher accuracy and precision. This learning is based on imitating human brain that works the same way in real environment. One example of these smart systems is definitely computer ability to recognize human faces, which has become very popular in the last 10 or so years. Although, the process of recognizing faces is almost automated in human brain, it has been a real challenge to implement such thing in computers and machines.

In this paper, one such facial recognition system will be created. The main task of that system will be recognizing faces in real time using web camera. To accomplish that, it is necessary to use two separate subsystems, one for the face detection and the other for face recognition once face has been detected. Both subsystems are based on transfer learning, which means that they are already created and only need to be slightly modified for this special case of recognizing faces in real time.

Key words: artificial intelligence, computer vision, neural networks, face detection, face recognition

1. UVOD

Današnji je život teško zamisliti bez utjecaja tehnologije koja svojim eksponencijalnim rastom te sve većom razinom kompleksnosti sve više mijenja naš način kako živimo. Većina se radnji počela sustavno automatizirati čime su neki dijelovi naših života postali vrlo jednostavni ili vremenski slabo zahtjevni što ostavlja veću mogućnost obavljanja onih radnji koje iziskuju našu prisutnost i pažnju. Jedna od grana tehnologije koja se počela drastično razvijati u posljednjih nekoliko desetljeća, svakako je područje strojnog učenja. Strojno je učenje, jednostavno objašnjeno, sposobnost nekog stroja, tj. uređaja da pomoću nekih metoda umjetne inteligencije može samostalno učiti te kontinuirano poboljšavati vlastiti način rada čime se postižu precizniji i bolji rezultati. Strojno se učenje koristi za rješavanje vrlo velikog broja raznovrsnih problema koji koriste znatno drugačije parametre i dijelove programa, ovisno o njihovoj primjeni kao što su na primjer prepoznavanje govora, prepoznavanje objekata na slikama, razni algoritmi klasifikacije podataka i sl. Dio strojnog učenja na kojemu će ovdje biti fokus je računalni vid, tj. sposobnost računala da na temelju ulaznih podataka u vizualnom obliku razumije kontekst slike te objekte koji se na njoj nalaze. Računalni se vid prvenstveno koristi za procese koji mogu zamijeniti radnje koje bi ljudi inače trebali raditi, često duže vremena, a odnose se na percipiranje i analizu slika. Računalni se vid, naravno, uvelike razlikuje od onoga ljudskoga, čime pouzdanost takvih sustava nikada neće biti 100% pa je važno pronaći parametre koji će se što bliže približiti mogućnostima čovjeka, ako ne i nadmašiti ih u nekim slučajevima. Primjer uporabe računalnog vida, svakako je prepoznavanje ljudskog lica (*engl. face recognition*) na slici te određivanja osobe kojemu to lice pripada. Taj je primjer svuda oko nas, od čega je najčešća njegova primjena otključavanje mobitela, prepoznavanje odgovarajućih osoba na sigurnosnim kamerama te razne primjene na društvenim mrežama. Način na koji se jedan takav sustav može kreirati, testirati te provjeriti njegova točnost, bit će prikazan u ovome radu. Prije toga, potrebno je objasniti neke osnovne pojmove te način na koji oni funkcioniraju.

2. UMJETNA INTELIGENCIJA

Umjetna je inteligencija sposobnost računala ili računalno kontroliranog robota da izvodi zadatke obično povezane s inteligentnim bićima [1]. Ovo je samo jedna od mnogih definicija kojima se može opisati široko područje umjetne inteligencije. U pravilu, skoro svaka definicija uključuje računala ili robote koji imaju mogućnosti na neki način postupati kao ljudi ili ostala inteligentna živa bića. Tako se najčešće spominje sposobnost nekog sustava da na temelju danih podataka, iz okoline u kojoj se nalazi ili iz nekog drugog izvora, donese samostalnu odluku koja će rezultirati najpovoljnijem ishodu. Na sličan način i mi ljudi donosimo odluke dok se krećemo kroz svakodnevni život. Ovdje je bitno naglasiti kako sustav koji može reagirati ovisno u okolini u kojoj se nalazi ili ne temelju nekih drugih podataka mora imati određeni set pravila koji će slijediti te probati doći do rješenja koje će najviše zadovoljiti ta pravila.

Za razliku od sustava koje ne smatramo inteligentnima, ovi se sustavi mogu samostalno poboljšavati i unaprjeđivati svoj rad kako bi što brže i efikasnije ispunili zahtijevani zadatak. Drugim riječima, inteligentni su sustavi sposobni učiti te na taj način unaprijediti svoj rad. Najjednostavniji je oblik učenja metoda pokušaja i pogreške koja je opravdana u nekim jednostavnijim slučajevima ali ukoliko je potrebno obraditi puno podataka ili provjeriti puno mogućnosti, vrlo brzo se javlja potreba za razvijanjem nekog naprednijeg načina učenja. Na isti princip ljudi, razvijanjem svog mozga postaju sposobniji i bolji u rješavanju određenih problema te razvijaju bolje i pouzdanije metode kako pristupiti određenom problemu.

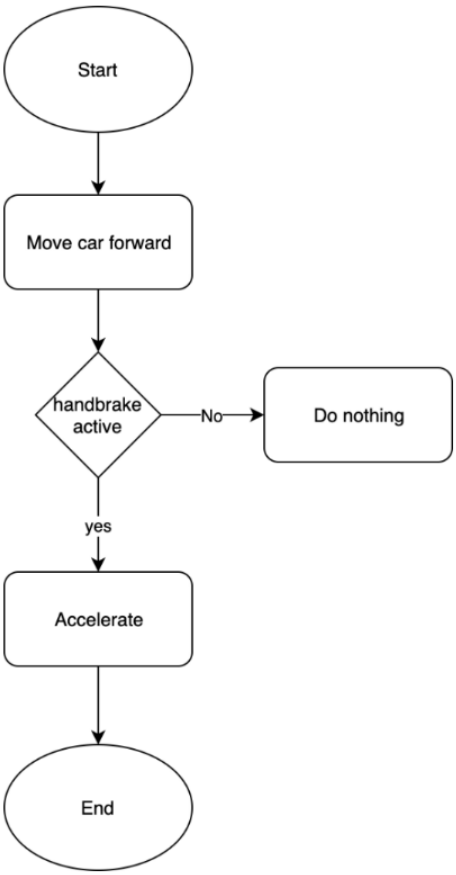
2.1. Povijest umjetne inteligencije

Koncept umjetne inteligencije nešto je o čemu je čovjek razmišljao već desetljećima. Prve su se ideje o kreiranju nečeg „živog“ tj. inteligentnog od nečega običnoga i materijalnog, razvile već u 19. stoljeću kada je Mary Shelley napisala poznato djelo Frankenstein; or, The Modern Prometheus. U toj se noveli radi o znanstveniku Victoru Frankensteina čija je ideja upravo bila kreacija potpuno novog inteligentnog bića od dijelova preminulih ljudi. Nešto kasnije, na početku 20. stoljeća, Karel Čapek, češki pisac, objavljuje svoje djelo R.U.R. u kojemu se prvi put spominje riječ robot čije je značenje i upotreba postalo univerzalno u svim jezicima. U tom se djelu, roboti prikazuju kao inteligentna bića, nalik ljudima, što po vanjskim, što po

unutarnjim značajkama, koji mogu misliti za sebe te praktički funkcionirati kao i ljudi. Kasnije će se takvu vrstu robota nazivati androidima [2], a problem koji Čapek predstavlja u ovom djelu svodi se na pobunu robota koji su radili za ljude te doveli do nestanka ljudske populacije, što je i danas jedna od tema koju ćemo morati riješiti u budućnosti, te je usko vezana uz etička pitanja koja se nameću u području umjetne inteligencije.

Kasnijim razvojem matematičkih i logičkih alata za rješavanje pojedinih problema, Alan Turing je teorijom o računanju došao do zaključka da je moguće simulirati svaki model matematičke dedukcije, pomoću stroja, jednostavno izmjenjujući nule i jedinice [3]. To je otkriće zajedno s razvojem u neurobiologiji, informacijskoj teoriji i kibernetici dovelo do ideje o potencijalnom kreiranju električnog mozga [3]. 1950. dvije ideje su se počele isticati kako bi se mogla ostvariti umjetna inteligencija u stroju odnosno računalu.

Prva je ideja o tzv. simboličkoj umjetnoj inteligenciji koja se bazira na visokoj razini simboličke reprezentacije problema, logike i pretraživanja [4]. Ideja se temelji na simbolima, odnosno definiranju setova pravila zakona koji opisuju odnose između tih simbola. Ljudi percipiraju gotovo sve preko simbola, a najviše se to odražava u vizualizaciji objekata u prostoru. Primjerice, ako netko kaže da je vidio crveni auto, svatko od nas imat će generalnu predodžbu o tome što je ta osoba vidjela, uz naravno veće ili manje odstupanje. Također, simboli mogu biti i neki apstraktni koncepti poput bankovnih transakcija ili pak neke radnje poput hodanja, trčanja ili slično. Problem se javlja kada se ti simboli trebaju definirati u računalu na način da ono „shvati“ što koji simbol predstavlja i koje su neke karakteristike i osobine koje ga opisuju. Neke je simbole potrebno opisati drugim simbolima ili pak rastaviti na simbole od kojih su sastavljeni. Sve te stvari nama ne predstavljaju problem, ali napraviti takvu složenu mrežu informacija i znanja o simbolima koji mogu predstavljati gotovo bilo što, nije nimalo jednostavno učiniti na računalu. Stoga se taj način pristupa umjetnoj inteligenciji postepeno prestao razvijati te se sve veći udio okreće drugoj ideji kako pristupiti kreiranju umjetne inteligencije. No, ipak u ranim fazama razvoja ove metode, stvari se nisu činile toliko kompliciranima. Vrlo je jednostavno vizualizirati logički slijed operacija te pravila koje ovakav sustav umjetne inteligencije prati. Ako su pravila jasno definirana i ulazni podaci koje treba obraditi precizno zadani, slijed izvođenja bit će vrlo jednostavan i uvijek raditi po zadanim pravilima. Na slici 1. prikazan je jedan jednostavan primjer kako bi simbolički pristup umjetnoj inteligenciji mogao jasno odrediti ishod na temelju zadanih pravila.



Slika 1. Primjer simboličkog pristupa umjetnoj inteligenciji [5]

Nedvojbeno je jasno kako će se ponašati sustav opisan na slici 1. Ako tako zamislimo svaki proces, na prvi bi se pogled činilo kako bi mogli riješiti gotovo svaki problem nadograđivanjem dodatnih uvjeta i operacija na ovaj postojeći. No, problem kod ovakvoga jednostavnog sustava javlja se pri implementaciji u stvarnom svijetu. Varijacija istih simbola u realnom svijetu ima praktički beskonačno, što znači da bi za svaku takvu varijaciju trebali dodati nova pravila i nekako ju definirati na ulazu. To u praksi nije nikako pogodno ni optimalno koristiti jer je realnost nepredvidiva, puna raznih šumova i smetnji koje uvijek mogu uzrokovati situaciju u kojoj sustav neće naći pravi put do konačne odluke. Drugi je problem i samo definiranje simbola te što oni predstavljaju. Ako npr. sustav treba pomoću kamere prepoznati automobil, on će to moći napraviti jedino ako mu se na ulazu postavi točno ta slika tog automobila. U suprotnom, sustav uopće neće moći spoznati da se radi o automobilu. Moguće bi rješenje bilo iskoristiti slike iz različitih kutova i pogleda na automobil i kreirati pravila za svaku takvu sliku, no ne postoji tehnički dovoljan broj slika kojima bi mogli opisati apsolutni svaku poziciju i pogled automobila, a kamoli još napraviti toliki broj posebnih pravila i radnji što bi sustav trebao napraviti za svaku pojedinu sliku.

Zbog tih navedenih razloga a i još nekih, kao što je i ranije rečeno, proučavanje i uporaba ovakvoga se pristupa u zadnjih par desetljeća znatno smanjila te se sve više sustava umjetne inteligencije okreće drugoj ideji koju je ponajviše zastupao Frank Rosenbaltt.

Iako je isprva bila odbačena i slabo razvijana, ideja o konekcionizmu doživjela je veliki napredak u nedavnoj prošlosti. Konekcionizam kao torija bazira se simultanoj aktivnosti distribuiranih signala pomoću veza koje mogu biti reprezentirane numerički, pri čemu se učenje odvija modificiranjem jakosti veza na temelju iskustva [6]. Ovaj se pristup temelji na radu ljudskog mozga gdje međusobno povezani neuroni prenose signale s jednog na drugi sve do krajnjeg neurona na kojemu se očituje konačna odluka. Najpoznatiji i najznačajniji predstavnik ovoga pristupa umjetnoj inteligenciji su neuronske mreže. Njima se pokušavaju imitirati događaji koji se odvijaju u ljudskom mozgu kada dođe do nekog podražaja. Povezanost neurona u mozgu puno je kompleksnija i brojčano nadmoćnija nego u umjetnim neuronskim mrežama, ali za većinu problema koji iziskuju umjetnu inteligenciju, moguće je kreirati relativno dobre i pouzdane neuronske mreže koje će davati dobre krajnje rezultate. Koncept povezanosti neurona u ljudskom mozgu prikazan je na slici 2.

Slika 2. Koncept povezanosti neurona u ljudskom mozgu [7]

Kao što je već spomenuto, ovaj je dio umjetne inteligencije postao popularan u zadnjih nekoliko desetljeća, razvojem bržih i procesorski jačih računala kao i korištenjem baza većeg broja podataka. Ovim se pristupom otklanjaju nedostaci simboličke umjetne inteligencije jer se više ne upravlja simbolima nego se pokušava optimizirati jakost veza između pojedinih neurona do zadane točnosti, što je u pozadini u stvari čista matematika.

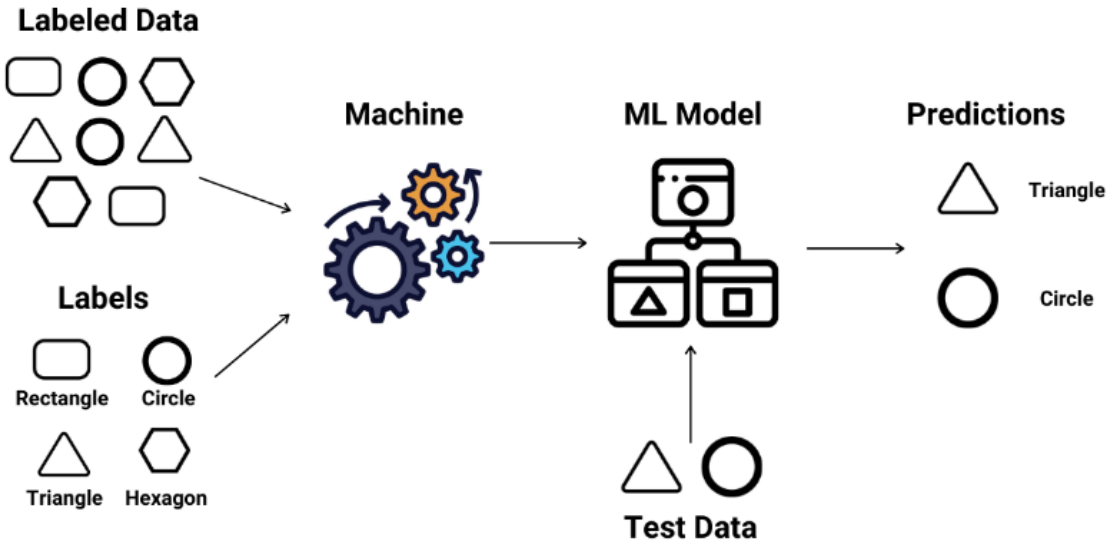
3. STROJNO UČENJE

Nakon što je ukratko objašnjen krovni pojam umjetne inteligencije, spustivši se razinu niže, možemo definirati pojam strojnog učenja kao dijela umjetne inteligencije. Kako bi imali jasniji i bolji uvid što se točno misli pod tim pojmom, najbolje je krenuti od same definicije. Strojno učenje je nauka o računalnim algoritmima koji se mogu samostalno unaprjeđivati kroz iskustvo i pomoću korištenja podataka [8]. Dakle, osnovna je zadaća svakog sustava strojnog učenja samostalno poboljšavanje na temelju nekih ulaznih podataka kako bi se ostvarili željeni rezultati. Prednost takvih sustava u odnosu na one tradicionalne, je u tome što se oni ne moraju eksplicitno programirati, nego se često koriste na način da računalo samo kreira algoritam prema kojem će raditi. To ujedno može biti i mana takvog sustava, jer ukoliko se željeni rezultati ne ostvare, ne može se jednostavno odrediti i detektirati gdje se nalazi greška te ju probati otkloniti.

Gledajući sa povijesnog aspekta, pojam strojno učenje, prvi je upotrijebio američki IBM-ovac Arthur Samuel, 1959. godine [8]. Tijekom 60-ih i 70-ih godina prošlog stoljeća, strojno se učenje bavilo pretežito problematikom prepoznavanja uzoraka, a 1981. objavljeno je istraživanje o strategijama učenja neuronske mreže koja bi naučila prepoznati 40 znakova sa računalnog terminala [8]. Danas se pojam strojnog učenja generalno može podijeliti u tri skupine, ovisno o načinu na koji je proces učenja ostvaren.

3.1. Nadzirano učenje

Pojam nadziranog učenja označava vrstu strojnog učenja prilikom kojeg se pomoću zadanih ulazno-izlaznih parova pokušava dobiti odgovarajuća funkcija koja će moći preslikavati ulazne podatke u one izlazne. Dakle, pomoću unaprijed definiranih ulaznih podataka kojima smo pridružili odgovarajući izlaz koji se na kraju treba ostvariti, pokušavamo doći do algoritma koji će moći prepoznavati i sukladno tome davati željene izlaze. Nakon što se proces učenja završi, algoritam se može testirati na novim podacima, tj. onima koje nismo koristili prilikom učenja te odrediti koliko će algoritam biti precizan u tom slučaju. Ovo je vrlo česta metoda strojnog učenja koja se koristi u algoritmima kao što su linearna regresija, logistička regresija, klasifikacije, neuronske mreže i drugi. Na slici 3. nalazi se pojednostavljeni grafički prikaz nadziranog strojnog učenja.

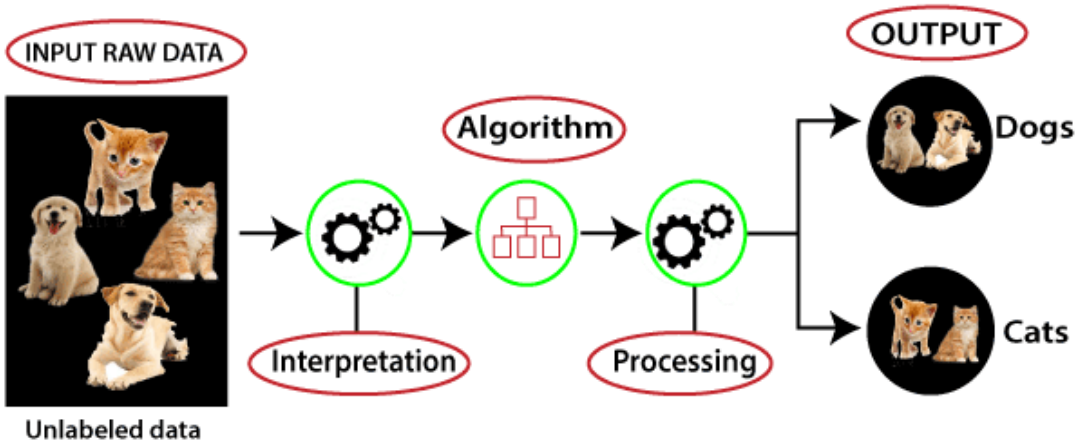


Slika 3. Pojednostavljeni prikaz nadgledanog učenja [9]

Kao što je prikazano na slici 3. te maloprije objašnjeno, svim ulaznim podacima pridružuje se željeni izlaz, najčešće u obliku oznaka (*engl. labels*), te sa tako označenim podacima, nakon faze učenja, možemo testirati algoritam na novim podacima očekujući da ti podaci pripadaju jednoj ili više zadanih oznaka.

3.2. Nenadzirano učenje

Drugi je tip strojnog učenja nenadzirano učenje kojem je cilj na temelju ulaznih podataka pronaći odgovarajući, nepoznati izlaz na temelju prepoznavanja uzoraka među podacima. Ovaj je pristup učenju mnogo složeniji od nadziranog učenja, jer nije jednostavno odrediti preciznost rada algoritma bez poznatih izlaznih podataka. Pošto ne postoje oznake kojima se ulazni podaci mogu označiti, oslanjamo se na sposobnost algoritma da će pomoću raznih uzoraka u ulaznim podacima odrediti neke strukturalne karakteristike prema kojima će se generirati izlazi. Zbog takvog načina rada algoritma, ovaj se tip učenja često koristi prilikom grupiranja podataka, određivanja skupina, smanjenja dimenzionalnosti i procjenom gustoće. Na slici 4. nalazi se grafički prikaz ovog tipa učenja.

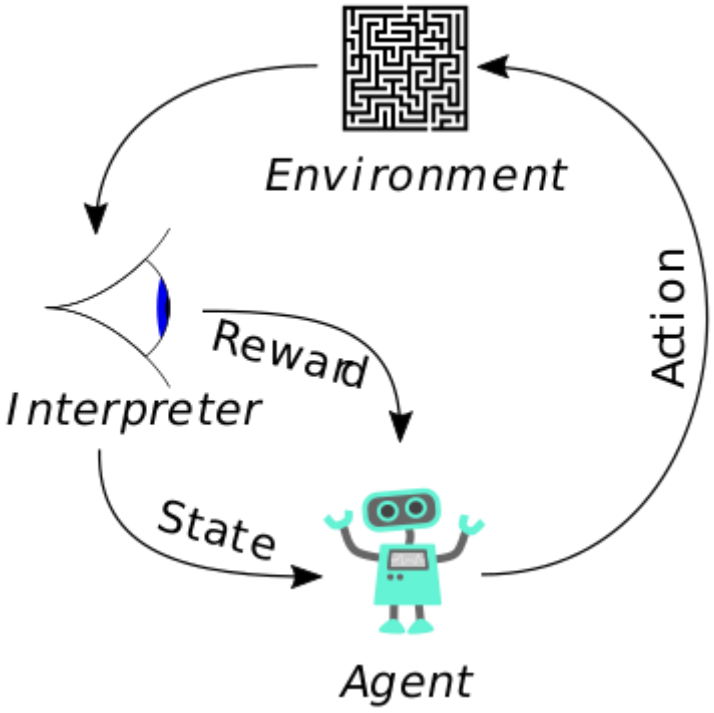


Slika 4. Primjer nenadziranog učenja [10]

Na slici 4. prikazan je jednostavan, ali načelno jasan primjer rada algoritma nenadziranog učenja. Naravno, kompleksnost problema te veći broj potrebnih klasifikacija, uvelike će utjecati na ovakav algoritam, pa nije uvijek moguće ovako jednostavno odrediti daje li model dobre rezultate ili ne.

3.3. Podržano učenje

Treći tip strojnog učenja naziva se podržano učenje. Podržano učenje podrazumijeva uporabu agenta koji samostalno poduzima neke radnje u zadanoj okolini kako bi maksimizirao iznos nagrade. Jednostavnije rečeno, virtualni agent u virtualnoj okolini ima mogućnost obavljanja određenih radnji pri čemu je definiran neki sustav koji daje nagradu ili kažnjava agenta ovisno o radnjama koje agent obavlja. Ako agent napravi radnju koja rezultira nagradom, algoritam će to memorirati te uspoređivati sa ostalim radnjama koje je agent mogao napraviti u tom trenutku. Zatim se pokušava pronaći optimalan slijed radnji koji će rezultirati maksimalnim iznosom nagrade. Za brži i učinkovitiji proces učenja, moguće je odjednom provesti simulaciju većeg broja agenata kako bi se ranije došlo do željenog iznosa maksimalne nagrade. Ovakav je tip učenja dosta čest u disciplinama poput teorije igara, teorije informacija, optimizacije na temelju simulacija i sl. Na slici 5. nalazi se primjer podržanog učenja.



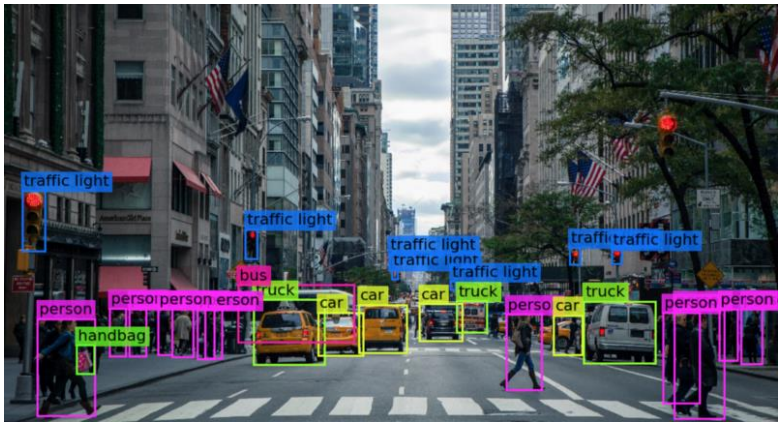
Slika 5. Primjer podržanoga učenja [11]

Primjer sa slike 5. jasnije ilustrira na koji princip model podržanog učenja radi. Ako se osvrnemo na taj primjer, možemo vidjeti kako agent ima zadaću pronaći izlaz iz zadane okoline, u ovom slučaju neke vrste labirinta, prilikom čega je očito povećavanje nagrade što se više agent približava izlazu. Agent se u početku kreće nasumično te nema neku percepciju nalazi li se na dobrome putu ili ne. Kako se s vremenom agent počinje kretati u željenom pravcu, ukupni iznos nagrade raste. Problem može nastati na mjestima gdje agent ima više opcija kojim putem krenuti, tj. gdje se iznos nagrade može povećavati brže na putevima koji ne vode do cilja nego na onima koji do njega vode. Takav bi slučaj bio klasični problem lokalnog maksimuma, gdje bi agent dosegao neku razinu nagrade koju bi smatrao maksimalnom, ali ta razina ne bi bila jednaka maksimalnom iznosu nagrade u globalnom smislu. Zato je korisno koristiti više agenata u istoj okolini od kojih bi svaki krenuo različitim putem, te na kraju onaj agent koji postigne maksimalan iznos nagrade, u ovom slučaju izađe iz labirinta, ujedno bi postigao željeni rezultat ovakvog tipa učenja.

4. RAČUNALNI VID

Kao granu strojnog učenja koje se bavi procesuiranjem i obradom slike, potrebno je definirati pojam računalnog vida i njegov kontekst u svijetu umjetne inteligencije. Računalni je vid interdisciplinarno znanstveno područje koje se bavi načinom na koji računala mogu postići visoku razinu razumijevanja digitalnih fotografija ili videa [12]. Cilj je svakog sustava koji se koristi računalnim vidom pretvaranje vizualnih podataka u one razumljive računalu kako bi se na temelju njih mogli postići određeni zaključci o tim podacima.

Povijest računalnog vida počinje teći kasnih 60-ih godina prošlog stoljeća na sveučilištima koja su prednjačila u području umjetne inteligencije. Cilj kreiranja računalnog vida bio je oponašanje ljudskog vizualnog sustava, kako bi roboti mogli postići inteligentno ponašanje [12]. Problem naravno, nije bio tako jednostavan jer se razlikovao od već postojeće tehnologije obrade slika po tome što se od računalnog vida očekivalo potpuno razumijevanje onoga što se nalazi na slici. U nadolazećim godinama i desetljećima, bavljenje ovim problemom, dovelo je do razvijanja nekih značajnih tehnika u ovom području kao što su detekcija rubova, segmentacija slika, bolje razumijevanje kalibracije kamere i sl. Danas je većina sustava računalnog vida povezana sa nekom vrstom dubokog učenja, najčešće u obliku neuronskih mreža koje u pozadini analiziraju dobivene vizualne podatke te donose konkretne zaključke o njima. Također, uporaba računalnog vida postala je neizostavan dio svake modernizirane grane industrije te svakodnevnog života. Najistaknutiji primjeri toga su automatizirana inspekcija u proizvodnji, navigacija autonomnih vozila, detekcija raznih objekata ili ljudi na slici, identifikacija ljudi i još mnogo drugih.



Slika 6. Sustav računalnog vida [13]

Na slici 6. prikazan je jedan od vrlo čestih slučajeva uporabe računalnog vida koji raspoznaje objekte na slici te im dodjeljuje pripadajuće značenje. Da bi se došlo do takve razine raspoznavanja, potrebno je dobro strukturirati program koji se nalazi u pozadini cijele priče. Najprije je potrebno sakupiti vrlo veliku količinu podataka koje je potrebno označiti kao što je to slučaj u tipu nadziranoga učenja. Nakon što se svaki ulazni podatak označi onime što on predstavlja, provodi se faza treniranja sustava koji će na temelju mnoštva ulaznih podataka moći razlučivati željeni objekt od ostatka slike. Primjerice sa slike 6., ako želimo da program može napraviti razliku između običnog automobila ili kombi vozila, potrebno je imati jako velik uzorak slika koje predstavljaju oba objekta. Također, položaj, boja i oblik svakog od objekata može se drastično razlikovati te dodatno zakomplicirati raspoznavanje između ta dva objekta. Još se jedan problem može javiti prilikom detekcije objekata, a to je izdvajanje tog objekta iz pozadine slike. Ljudskom je oku jednostavno odrediti granicu gdje automobil počinje te gdje završava, ali računalni vid koji koristi samo vrijednosti piksela sa slike mora biti dobro istreniran kako bi to mogao odrediti, pogotovo ukoliko se radi o analizi videa gdje se svaka slika obrađuje nekoliko puta u sekundi. Slično tome je i detekcija objekata koji nisu u potpunosti prikazani na slici. Čovjek može vrlo lako zaključiti da se na slici nalazi automobil iako je prikazana samo njegova četvrtina, dok računalni sustav koji bi ga trebao detektirati neće uvijek moći lako zaključiti da se radi o automobilu. Zato se vrlo veliki napori i sredstva ulažu u razvijanje ove grane kako bi imali što preciznije vizualne sustave, slične onima koje posjeduju ljudi.

5. NEURONSKE MREŽE

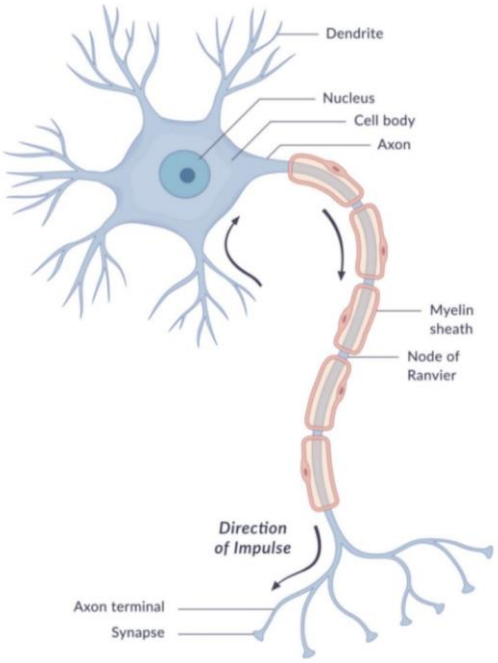
Kako bi mogli kreirati jedan sustav koji će moći iskoristiti sve benefite računalnoga vida, potrebno je koristiti odgovarajući „alat“. U ovome će slučaju to biti primjena neuronskih mreža kao univerzalno prihvaćenog načina rješavanja problema u tom području. No najprije se treba definirati što zapravo jesu neuronske mreže te na koji način rade te kako se pomoću njih mogu postići rezultati koji su prije ne toliko mnogo godina bili gotovo nezamislivi.

Već djelomično opisan u poglavlju 2.1., pristup umjetnoj inteligenciji koji se temelji na konekcionizmu češći je i prihvatljiviji način rješavanja problema koji u današnje vrijeme zahtijevaju uporabu umjetne inteligencije. Pristup je temeljen na povezanosti neurona u ljudskom mozgu te njihovom međusobnom interakcijom tj. prolaskom signala kroz pojedine neurone sve do onog posljednjeg. Rezultat toga pristupa, kreiranje je umjetnih neuronskih mreža koje se sastoje od nekolicine slojeva te određenog broja neurona koji su međusobno povezani vezama s odgovarajućim težinama svake pripadajuće veze. Za bolju analogiju kako je ljudski mozak inspirirao ovaj pristup, najbolje bi bilo krenuti sa objašnjenjem što su pravi biološki neuroni u mozgu te na koji se način ostvaruje prijenos signala sa jednog neurona na drugi.

5.1. Biološke neuronske mreže

Ljudski je mozak kao i u ostalih živih bića sastoji od dvije vrste stanica. Prvu vrstu stanica čine glija stanice koje čine oko 50% ukupnog volumena moždanog tkiva [14]. Njihova uloga u mozgu nije funkcionalna osnova živčanog sustava, već su one zadužene za kontroliranje i zaštitu rada druge vrste stanica u mozgu, neurona. Neuroni su osnovne jedinice koje tvore živčani sustav u ljudskom tijelu. Iako se donedavno smatralo kako ljudski mozak sadrži oko 100 milijardi neurona, novija istraživanja ipak smanjuju taj broj na otprilike 86 milijardi. Biološki se neuroni sastoje od tijela neurona, dendrita i aksona. Svaki dio ima soju ulogu, pa tako tijelo neurona tj. soma, sadrži jezgru u kojoj se nalaze RNK i DNK koje predstavljaju „pamćenje“ genetskog koda, zbog čega su iznimno bitne u svakoj stanici organizma. Dendriti su razgranati članci na tijelu neurona koji se vežu za druge neurone. Preko njih se prenose elektrokemijski signali sa kraja jednog neurona do tijela drugog. Akson je tanak i dug produžetak tijela čiji je kraj razgranat kako bi se mogao vezati za dendrite drugog neurona.

Dakle, signal se u biološkim neuronskim mrežama prenosi međusobno povezanim neuronima koji reagiraju na elektrokemijske podražaje. Kada signal s aksona jednog neurona prijeđe na dendrite drugog neurona na mjestu njihovog spajanja, sinapsi, dolazi do prijenosa informacija u mozgu. Nisu svi elektrokemijski signali dovoljno „jaki“ kako bi mogli odmah prijeći na drugi neuron, pa je ponekad potrebna uporaba neurotransmitera koji mogu prenijeti informaciju preko sinapsi do sljedećeg neurona. Ovaj se proces slanja informacija dešava konstantno u ljudskom mozgu čime se podražuje milijune neurona koji neumorno primaju i šalju elektrokemijske signale kako bi mi mogli normalno razmišljati, učiti, primati informacije iz okoline i slično. Na slici 7. prikazan je primjer jednog neurona sa svim navedenim dijelovima.

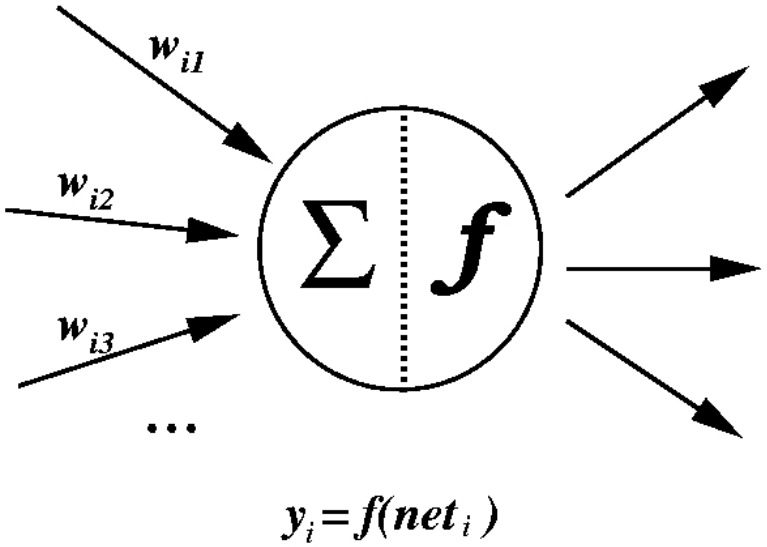


Slika 7. Neuron u ljudskom mozgu [15]

Razumijevanjem ovog naizgled jednostavnog, ali zapravo vrlo kompleksnog sustava koji je još jako daleko od potpunog definiranja na koji način naš mozak zapravo sprema te signale, i koristi ih kasnije po potrebi odnosno kako se tim signalima pridodaje neki smisao kojeg mi razumijemo i vrlo jednostavno možemo interpretirati iako je to samo skup elektrokemijskih reakcija koji putuje našim neuronima. Uz sve to rečeno, lako je shvatiti inspiraciju koju su znanstvenici sredinom prošlog stoljeća pronašli u ovim procesima te pokušali na isti način kreirati umjetne sustave koji bi imitirali rad ljudskoga mozga na ovome principu.

5.2. Umjetne neuronske mreže

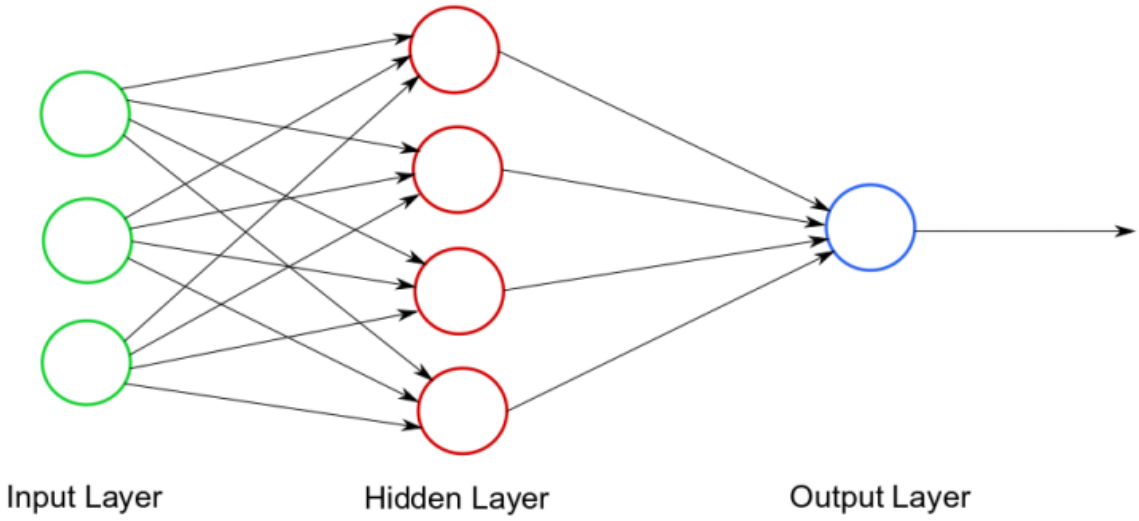
Umjetne su neuronske mreže, kao što je već rečeno, zapravo pokušaj repliciranja rada ljudskog mozga pomoću računala ili nekog računalno upravljanog sustava. Neuroni koji se nalaze u mozgu bioloških organizama, u neuronskim su mrežama zamijenjeni umjetnim neuronima koji zapravo obavljaju iste radnje kao oni biološki. Broj neurona u neuronskim mrežama također je znatno manji nego u biološkim neuronskim mrežama zbog čega je potrebno nekoliko puta provući ulazne podatke kroz umjetnu neuronsku mrežu kako bi se dobila željena izlazna vrijednost na kraju mreže. Na slici 8. prikazana je shema umjetnog neurona u neuronskoj mreži.



Slika 8. Umjetni neuron [16]

Analogno biološkom neuronu, umjetni su neuroni na svom ulazu povezani sa prethodnim neuronima pomoću veza koje oponašaju dendrite bioloških neurona. Jakost svake veze koja ulazi u neuron određena je pripadajućom težinom koje se konstantno mijenjaju kako bi što preciznije postigli željeni izlaz. Sve veze koje ulaze u neuron zbrajaju se te se iznos tog zbroja provlači kroz neku aktivacijsku funkciju koja preslikava zbroj ulaznih težina u neku novu vrijednost koja će se dalje poslati kroz mrežu do sljedećeg sloja neurona na isti način kako se preko krajeva aksona preko sinapsi prenose signali u mozgu.

Kada generalno govorimo o umjetnim neuronski mrežama, njihova je podjela izuzetno široka te često postoji nekoliko vrsta mreža koje se bave isključivo rješavanjem jednog problema te optimizacijom pronalaska rješenja ili pokušaja unaprjeđivanja svoga rada. Tako se primjerice neuronske mreže mogu podijeliti prema broju slojeva, načinu i algoritmu po kojemu uče, dinamičnosti ili statičnosti neurona, protoku signala među neuronima i sl. Kako bi se sačuvala jasna i pregledna slika o temeljnim principima rada neuronskih mreža, najbolje će biti krenuti od jednostavnijeg primjera jednoslojne mreže koja ima samo jedan skriveni sloj neurona te jedan izlazni neuron. Sve jednadžbe i pravila koja se kod takvih mreža koriste mogu se analogno primijeniti na kompleksnije i složenije mreže kojima se rješavaju problemi koji su izvan opsega mogućnosti običnih statičkih jednoslojnih neuronskih mreža. Kao referentnu sliku koja će poslužiti za lakšu vizualizaciju i uvid u strukturu neuronskih mreža, koristit ćemo sliku 9.



Slika 9. Primjer jednostavne jednoslojne neuronske mreže [17]

Kao što je prikazano na slici 9. i spomenuto maloprije, najprije je potrebno proučiti rad jedne jednostavne statičke neuronske mreže kako bi se kasnije mogao shvatiti rad kompleksnijih mreža koje će se koristiti kao temeljni alat za rješavanje problema u ovome radu. Pojam statička neuronska mreža odnosi se na činjenicu da je struktura slojeva u mreži statička tj. neuroni u skrivenom sloju ne prolaze dodatne dinamičke operacije koje mogu mijenjati njihove utjecaje čak i nakon postupka treniranja neuronske mreže. Za početak, na ulazu u samu neuronsku mrežu, prema slici 9. nalaze se tri neurona pri čemu svaki od njih ima neku vrijednost. Vrijednosti ulaznih neurona uvijek ovise o problemu koji pokušavamo riješiti,

odnosno o tome što želimo naučiti neuronsku mrežu. Ako je to na primjer zbrajanje tri proizvoljna pozitivna cijela broja koja su manja od 10, onda će se na ulaznim neuronima nalaziti sve vrijednosti i kombinacije tih brojeva. Izlaz koji će biti pridružen svakom setu ulaznih brojeva iznosit će njihov zbroj i to je ono čemu će neuronska mreža u takvom primjeru težiti. Dakle, potrebno je napraviti setove ulaznih podataka sa odgovarajućim i željenim izlazom koji pokrivaju sve kombinacije tih ulaznih brojeva. To odgovara nadgledanom tipu strojnog učenja opisanom u poglavlju 3.1. gdje je za zadani ulaz poznat iznos izlaza koji se treba ostvariti. Takvi se sustavi obično koriste za interpolaciju ili ekstrapolaciju podataka koji nisu prošli fazu učenja mreže. U navedenom primjeru zbrajanja tri cjelobrojna broja, to bi moglo odgovarati zbrajanju ne cjelobrojnih brojeva manjih od 10 ili zbrajanju brojeva većih od 10 te bi se moglo odrediti koliko dobro mreža radi u takvoj situaciji.

Nakon što su ulazni podaci ili setovi podataka zadani, iz svakog je neurona napravljena po jedna veza sa svakim neuronom iz sljedećeg, skrivenog, sloja. Te su veze opisane nekim težinskim faktorom koji se može protumačiti kao jačina utjecaja pojedinog neurona na krajnji izlaz. Uzevši ponovno ovaj primjer sa zbrajanjem triju brojeva na ulazu, može se zaključiti kako će veći brojevi svakako imati veći utjecaj na krajnji rezultat. Ukoliko se takvi zaključci ne mogu jednostavno donijeti ili se njih upravo treba potvrditi, iznosi svih težina uzimaju se proizvoljno, najčešće u iznosu između nula i jedan. Sada je moguće izračunati sumu na svakom pojedinom neuronu skrivenog sloja, često označavanu *net*. Njen će se iznos jednostavno odrediti jednadžbom (1):

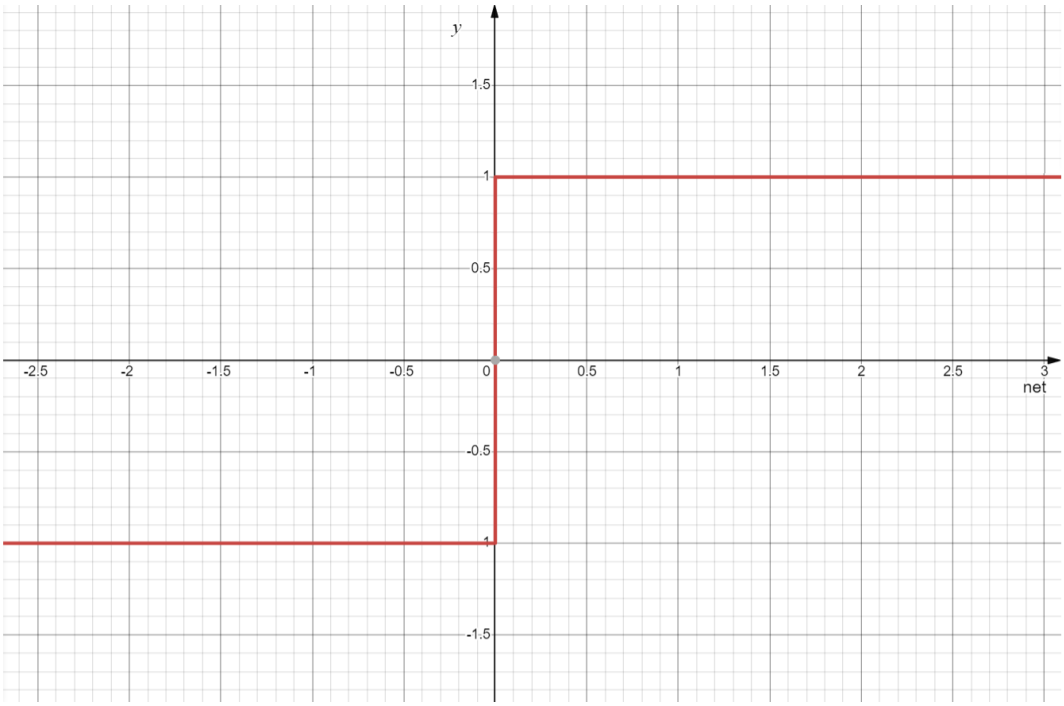
$$net = \sum_{j=1}^{n} w_j u_j, \tag{1}$$

pri čemu je *n* broj ulaza u razmatrani neuron, *w* je iznos težina dodijeljenih vezama na ulazu u neuron, a *u* je vrijednost pojedinog ulaza u neuron. Prema slici 9. *n* bi iznosio tri, a svaki od neurona u skrivenom sloju bio bi dakle zbroj svakog ulaznog neurona pomnoženog sa pripadnom težinom. Ovdje je također važno napomenuti, iako to nije grafički prikazano, ulogu dodatnog neurona čija je vrijednost uvijek jednaka jedan, zvanog *bias*. Ukratko, taj se dodatni neuron dodaje kako bi se aktivacijska funkcija mogla „pomicati“ lijevo ili desno, odnosno mijenjati iznose pri kojima će se za određeni iznos sume, *net* neuron aktivirati ili ne. Kada su svi iznosi suma na neuronima izračunati, te je sume potrebno provući kroz aktivacijsku funkciju tog sloja. Aktivacijske funkcije mogu biti vrlo različite ovisno o problemu koji se rješava ali neki od primjera često upotrebljavanih aktivacijskih funkcija su:

a) Signum funkcija – neparna funkcija koja uzima predznak svakog realnog broja i pridružuje mu vrijednost 1 ukoliko je predznak pozitivan, odnosno -1 ako je predznak negativan. Vrijednost funkcije u nuli jednaka je nula. Jednadžba koja opisuje ovu funkciju je:

$$y = sign(net), \tag{2}$$

a pripadajući graf nalazi se na slici 10.

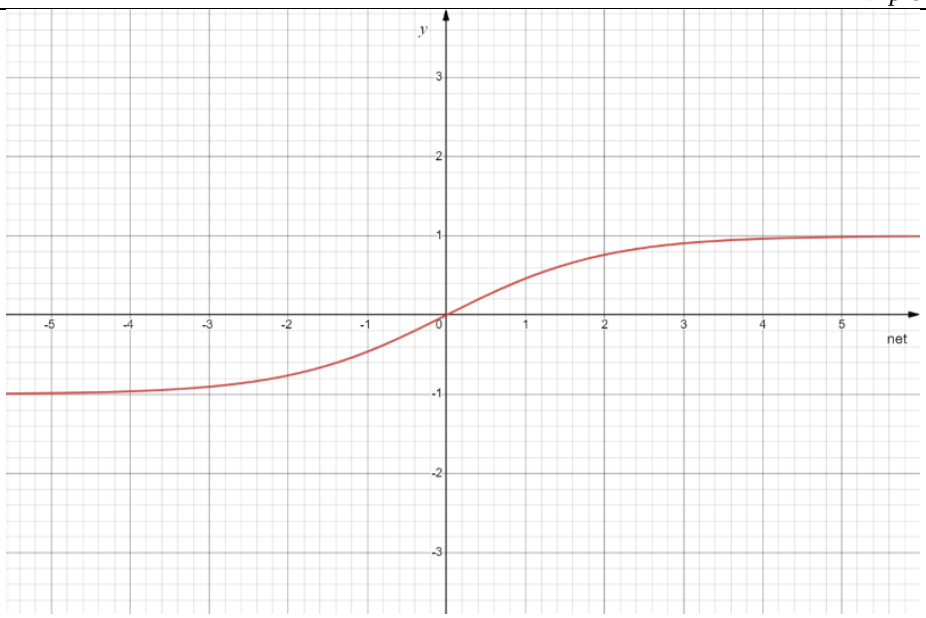


Slika 10. Signum aktivacijska funkcija

b) Nelinearna sigmoidalna funkcija – funkcija koja poprima oblik slova S kada se prikaže grafički te praktički predstavlja glatku signum funkciju, što znači da postoji derivacija u svakoj točki što je vrlo značajno za umjetne neuronske mreže. Sigmoidalnih funkcija ima vrlo velik broj i definiraju se raznim jednadžbama, ali kao primjer uzet će se funkcija definirana jednadžbom:

$$y = \frac{2}{1+e^{-net}} - 1, \tag{3}$$

čiji je graf prikazan na slici 11. Vrijednosti ove funkcije također se kreću između -1 i 1, ali će se vrijednosti sume na neuronu, *net*, postepeno preslikavati i težiti broju jedan što je ta suma veća, odnosno udaljenija od y osi. Ovdje se može i vidjeti uloga koju *bias* ima u neuronskim mrežama, koji može translatirati aktivacijsku funkciju lijevo ili desno po x osi.

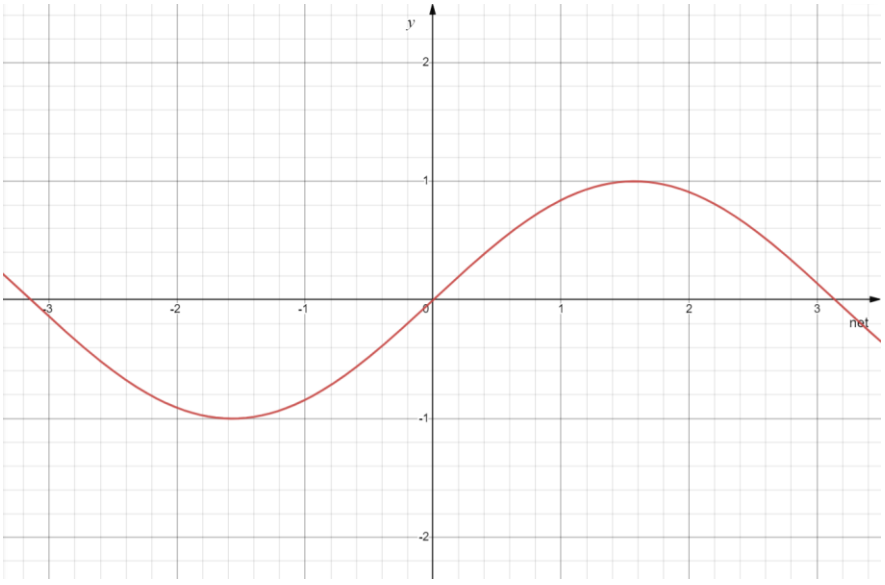


Slika 11. Primjer sigmoidalne aktivacijske funkcije

c) Trigonometrijske funkcije – funkcije poput sinusa ili kosinusa često su se koristile kao aktivacijske funkcije, no problem je u tome što su one periodičke, što znači da će različita vrijednost sume jednako „pobuditi“ neuron iako se možda nalazi vrlo daleko od y osi. Zbog tog se razloga, u novije doba, jednostavne trigonometrijske funkcije manje koriste u kompleksnijim neuronskim mrežama, pogotovo kada se iznosi suma mogu jako razlikovati. Primjer jedne standardne trigonometrijske aktivacijske funkcije dan je jednadžbom:

$$y = \sin(net), \tag{4}$$

dok se odgovarajući graf nalazi na slici 12.



Slika 12. Sinusna aktivacijska funkcija

Pored ovih navedenih primjera postoji još mnogo funkcija koje se mogu koristiti kao aktivacijske funkcije. Najčešće korištena aktivacijska funkcija neurona je nelinearna bipolarna sigmoidalna funkcija [18], prikazana na slici 11.

Kada se odredi aktivacijska funkcija svih skrivenih i izlaznih slojeva mreže, u primjeru sa slike 9. to je samo jedan sloj, te se posloži čitava struktura, odnosno broj svih neurona na ulaznom, skrivenom te izlaznom sloju, može se krenuti u proces učenja umjetne neuronske mreže. Faza u kojoj mreža uči, odvijat će se u dvoje faze, a to su unaprijedna faza učenja i povratna faza. Učenje svake mreže, tj. promjena iznosa težina, može se ostvariti na dva načina. Prvi je način tzv. „batch" procedura pri kojoj se težine mijenjaju nakon što cijeli ulazni skup podataka prođe kroz mrežu, a promjena se odvija nekom srednjom pogreškom proizvedenom tim ulaznim skupom podataka [18]. U drugom se načinu iznosi težina mijenjaju nakon svakog ulazno-izlaznog para skupa učenja, a takav se način naziva „pattern" ili „stochastic" procedura. Kako bi se lakše objasnile spomenute faze učenja mreže na primjeru sa slike 9., koristit će se drugi način na koji se to učenje ostvaruje.

5.2.1. Unaprijedna faza učenja

Prije nego samo učenje mreže započne, dobra je praksa jednom proći kroz cijelu mrežu te vidjeti kakve ćemo rezultate dobiti slučajno generiranim težinama skrivenog sloja, označenim $\boldsymbol{V}$ i isto tako slučajno generiranim težinama izlaznog sloja, označenim $\boldsymbol{W}$. U matematičkom smislu težine se obično predstavljaju u obliku vektora ili matrica ovisno o strukturi mreže. Analogno jednadžbi (1) možemo najprije izračunati pojedine iznose suma na skrivenom sloju neurona, *H*:

$$net_{Hj} = \sum_{i=1}^{I} \mathbf{v}_{ji} Z_i\,, \quad j = 1,2,\dots,J-1, \quad i = 1,2,\dots,I, \tag{5}$$

pri čemu net_{Hj} označava sumu na *j*-tom neuronu skrivenog sloja *H*, *I* označava broj ulaznih neurona plus dodatni neuron, *bias*, $\mathbf{v}_j$ su težine *j*-tog neurona skrivenog sloja, *J* je broj neurona u skrivenom sloju uvećan za jedan, *bias*, a *Z* predstavlja vrijednosti na ulaznim neuronima. U primjeru sa slike 9. vrijednost *I* bila bi jednaka 4, dok bi vrijednost *J* iznosila 5. Sa dobivenim vrijednostima net_{Hj} prolazimo kroz aktivacijsku funkciju u skrivenom sloju. Kao što je ranije spomenuto, koristit će se nelinearna bipolarna sigmoidalna funkcija prikazana na slici 11. i opisana jednadžbom (3). Uvrštavanjem dobivenih iznosa net_{Hj} u jednadžbu (3), dobivamo:

$$y_j = \frac{2}{1+e^{-net_{Hj}}} - 1, \;\; j = 1,2, \ldots . J-1, \tag{6}$$

$$y_J = 1, bias, \tag{7}$$

gdje je y_j vrijednost na izlazu neurona skrivenog sloja. Sve su vrijednosti dobivene jednadžbom (6), uključeno sa dodatnim neuronom, *bias*-om, povezano do sljedećeg, izlaznog sloja neurona preko težinskih faktora w_{kj} [18].

Nakon određivanja y_j potrebno je izračunati sumu na izlaznim neuronima, označenu sa net_{Ok}, gdje *O* predstavlja neurone izlaznog sloja. Ta se suma može odrediti na sličan način kao i u jednadžbi (5):

$$net_{Ok} = \sum_{j=1}^{J} \mathbf{w}_{kj} y_j \,, \;\; k = 1,2, \ldots K, \tag{8}$$

gdje je K broj neurona izlaznog sloja, odnosno broj izlaza mreže, a w_{kj} težinski koeficijent veze između *k*-tog neurona izlaznog sloja i *j*-tog neurona skrivenog sloja. Dobivena se suma opet može provući kroz aktivacijsku funkciju izlaznog sloja, najčešće linearnu funkciju kako bi se dobile izlazne vrijednosti veće od 1. Iznos izlaza na *k*-tom neuronu tada je jednaka:

$$O_k = K_p net_{Ok}, \;\; k = 1,2, \ldots K, \tag{9}$$

pri čemu je O_k krajnja vrijednost izlaza *k*-tog neurona izlaznog sloja, a K_p nagib linearne aktivacijske funkcije. Radi jednostavnosti i preglednosti primjera sa slike 9., u nastavku će se koristiti vrijednost $K_p = 1$.

Vrijednosti izlaznog sloja O_k krajnji su izlaz koji će mreža dati kao rješenje. Ako se osvrnemo na ranije spomenuti primjer zbrajanja triju cijelih brojeva manjih od 10 prema strukturi mreže na slici 9., rezultat koji će mreža davati bit će zbroj ta tri broja, odnosno mi ćemo pokušati „natjerati“ mrežu da daje rješenja koja odgovaraju tom zbroju iako sama mreža neće imati jasan kontekst da se radi o zbrajanju. Zato je uvijek zanimljivo prvo napraviti tzv. nulti korak učenja koji će dati rezultat na temelju slučajno generiranih težina skrivenog i izlaznog sloja mreže. Tada već možemo vidjeti koliko je mreža blizu ili daleko ostvarivanju željenog izlaza što nam daje nekakav približni dojam koliko će zapravo trajati postupak učenja mreže, odnosno koliko će otprilike koraka biti potrebno kako bi se postigli takvi iznosi težina pri kojima će mreža davati zadovoljavajuće rezultate.

5.2.2. Povratna faza učenja

Nakon što je mreža prošla nulti korak učenja, što je u biti proizvoljno, možemo prijeći na konkretni postupak kako se težine mijenjaju tijekom povratne faze učenja. Za početak se, na temelju dobivenog izlaza u unaprijednoj fazi učenja, računa iznos pogreške, tj. odstupanja dobivenog rješenja od onog traženog, odnosno zadanog. Mijenjanjem težina u ovoj fazi učenja, nakon svakog ulazno-izlaznog para, dobiva se nova vrijednost pogreške. Proces se ponavlja sve dok se ne postigne iznos pogreške koji je manji ili jednak iznosu dozvoljene pogreške. Najčešća statistička metoda regresijske analize, suma kvadrata pogreške kao mjera odstupanja ostvarenog izlaza od onog željenog je uobičajena funkcija cilja [18]. Iznos pogreške prema metodi najmanjih kvadrata iznosi:

$$E = \frac{1}{2}\sum_{n=1}^{N}(d_n - O_n)^2, \tag{10}$$

gdje je E iznos pogreške, N broj elemenata u skupu za učenje, d_n željeni izlaz iz mreže, a O_n ostvareni izlaz mreže. Faktor jedne polovine prije izraza pojednostavljuje potrebite derivacije funkcije cilja [18]. Cilj će, dakle, biti minimizirati funkciju cilja E, mijenjanjem težina skrivenog i izlaznog sloja. Za odabranu funkciju cilja, primjenjuje se neki od algoritama nelinearnog optimiranja. Poznata forma promjene parametara učenja ϑ, dana je jednadžbom:

$$\vartheta(n+1) = \vartheta(n) + \Delta\vartheta(n), \tag{11}$$

pri čemu je n trenutni korak učenja, a $\Delta\vartheta(n)$ veličina promjene parametara učenja. Za izlazni sloj u neuronskoj mreži ϑ je jednaka w, odnosno za skriveni sloj ϑ je v. $\vartheta(n+1)$ biti će nova vrijednost parametara učenja u sljedećem koraku. Pogrešku $E(\vartheta)$ moguće je aproksimirati u blizini točke ϑ pomoću prva dva člana Taylorovog reda:

$$E(\vartheta + \Delta\vartheta) \approx E(\vartheta) + \Delta E(\vartheta), \tag{12}$$

$$\Delta E(\vartheta) = \Delta\vartheta^T \cdot \nabla E(\vartheta), \tag{13}$$

$$\nabla E(\vartheta) = \frac{\partial E(\vartheta)}{\partial \vartheta}. \tag{14}$$

Jednadžba (14) predstavlja gradijent pogreške. Cilj je neuronske mreže da se pogreška smanjuje najvećim mogućim iznosom, pa je potrebno odrediti $\Delta\vartheta$ za koji će promjena pogreške $\Delta E(\vartheta)$ imati najveći negativni iznos, što je ostvareno uz uvjet:

$$\Delta\vartheta = -\eta\nabla E(\vartheta), \tag{15}$$

gdje je η mjera promjene pogreške, često nazivana koeficijentom brzine učenja. Taj se koeficijent izabire proizvoljno, a najbolje vrijednosti tog koeficijenta pokazale su se u rasponu od 10^{-3} do 10. Uvrsti li se izraz (15) u izraz (11), dobiva se:

$$\vartheta(n+1) = \vartheta(n) - \eta \nabla E(\vartheta(n)). \tag{16}$$

Izraz (16) predstavlja algoritam najstrmijeg pada (*engl. steepest descent*), koji se u području umjetnih neuronskih mreža naziva algoritam povratnog prostiranja pogreške (*engl. Error back-propagation algorithm*) [18]. Ovaj je algoritam najpoznatiji i najčešće primjenjivani algoritam promjene parametara učenja. Mana ovog algoritma je velik broj iteracija koji je potreban kako bi se ostvario željeni iznos pogreške, pa se često koriste neke modifikacije koje bi smanjile broj tih iteracije i ubrzale proces. Jedna od češćih metoda za tu primjenu je ubacivanje momentuma u izraz (16), čime se dobiva izraz:

$$\vartheta(n+1) = \vartheta(n) - \eta \nabla E(\vartheta(n)) + \alpha \Delta \vartheta(n-1), \tag{17}$$

gdje α predstavlja koeficijent momentuma, a često se kreće između vrijednosti 0,1 i 0,9 [18].

Nakon definiranja algoritma kojim se mijenjaju težine u neuronskoj mreži, potrebno ga je primijeniti u odgovarajućem sloju mreže. Kako samo ime algoritma nalaže, promjena parametara učenja, odnosno iznosi težina, mijenjaju se od izlaznog sloja prema ulaznom. Uz modifikaciju izraza (17), dobivamo izraz kojim se mijenjaju težine izlaznog sloja:

$$\mathbf{w}_{kj}(n+1) = \mathbf{w}_{kj}(n) - \eta \nabla E(n) + \alpha \Delta \mathbf{w}_{kj}(n-1). \tag{18}$$

Gradijent se pogreške kod težina izlaznog sloja w_{kj} može računati prema izrazu (14), čime se dobije:

$$\nabla E(n) = \frac{\partial E(n)}{\partial \mathbf{w}_{kj}}. \tag{19}$$

Iz gornjeg je izraza vidljivo kako je potrebno odrediti iznos pripadajućeg gradijenta pogreške, što je u biti osnovni zadatak čitavog postupka. Primjenom uzastopnih parcijalnih derivacija, može se dobiti izraz:

$$\frac{\partial E(n)}{\partial \mathbf{w}_{kj}} = \frac{\partial E(n)}{\partial O_k} \frac{\partial O_k}{\partial net_{Ok}} \frac{\partial net_{Ok}}{\partial \mathbf{w}_{kj}}. \tag{20}$$

Pokazano je prema [19] da je karakteristična vrijednost algoritma povratnog prostiranja pogreške, δ izračunata za izlazni sloj jednaka:

$$\delta = -\frac{\partial E(n)}{\partial net}. \tag{21}$$

Sređivanjem izraza (20) uz jednakost koja vrijedi u izrazu (21), može se dobiti konačni algoritam promjene težina izlaznog sloja mreže:

$$\mathbf{w}_{kj}(n+1) = \mathbf{w}_{kj}(n) + \eta\delta_{Ok}y_j + \alpha\Delta\mathbf{w}_{kj}(n-1), \tag{22}$$

gdje δ_{Ok} označava parametar algoritma povratnog prostiranja greške u izlaznom sloju. Važno je napomenuti kako će se krajnji izraz δ_{Ok} mijenjati u ovisnosti o strukturi mreže koju koristimo. Isto će tako parcijalne derivacije u izrazu (20) biti drugačije za drugačiju konfiguraciju mreže.

Kada se odredio izraz po kojem se mogu mijenjati težine $\mathbf{w}_{kj}$ izlaznog sloja neurona, potrebno je još pokazati na koji će se način mijenjati težine u skrivenom sloju. Analogno jednadžbi (18), primijenjenoj za težine skrivenog sloja, $\mathbf{v}_{ji}$, dobiva se:

$$\mathbf{v}_{ji}(n+1) = \mathbf{v}_{ji}(n) - \eta\nabla E(n) + \alpha\Delta\mathbf{v}_{ji}(n-1). \tag{23}$$

Slično kao i kod težina izlaznog sloja, i ovdje je glavni problem odrediti gradijent greške $\nabla E(n)$. Primjenom uzastopnog parcijalnog deriviranja, može se dobiti:

$$\frac{\partial E(n)}{\partial \mathbf{v}_{ji}} = \frac{\partial E(n)}{\partial y_j}\frac{\partial y_j}{\partial net_{Hj}}\frac{\partial net_{Hj}}{\partial \mathbf{v}_{ji}}. \tag{24}$$

Ponovnim sređivanjem izraza (24) te raspisivanjem svakog pojedinog razlomka, početni izraz za promjenu težina skrivenog sloja postaje jednak:

$$\mathbf{v}_{ji}(n+1) = \mathbf{v}_{ji}(n) + \frac{1}{2}\eta(1-{y_j}^2)Z_i\left(\sum_{k=1}^{K}\delta_{Ok}\mathbf{w}_{kj}\right) + \alpha\Delta\mathbf{v}_{ji}(n-1). \tag{25}$$

Sada smo u potpunosti definirali sve što je potrebno kako bi se mogao provesti postupak učenja umjetne neuronske mreže sa spomenutim karakteristikama i strukturom. Svakim se novim korakom iznosi težina mijenjaju prema jednadžbama (22) i (25) te se tako postepeno smanjuje iznos greške izračunat metodom najmanjih kvadrata prema jednadžbi (10). Zadovoljavajući iznosi pogrešaka ovisit će o primjeni za koju se neuronska mreža upotrebljava, no ukoliko se tražena preciznost zada kao iznimno mali broj, moguće je da će proces učenja mreže trajati jako dugo, pa se stoga često definiraju parametri koji će prekinuti učenje iako se nije postigao zadani iznos greške. Tako se npr. može ograničiti broj koraka kroz koji će mreža učiti ili postaviti uvjet koji će prekinuti učenje mreže ukoliko se iznos greške smanjuje jako sporo, što bi zapravo značilo da ta mreža niti ne može postići traženu preciznost ili se dogodio problem lokalnog minimuma. Podešavanjem parametara kao što su koeficijent učenja ili koeficijent momentuma, može se ubrzati i poboljšati učenje mreže.

6. METODE ZA DETEKCIJU I IDENTIFIKACIJU OSOBE

Nakon što je ukratko objašnjeno što je računalni vid u kontekstu umjetne inteligencije, kojim područjem primjene se bavi te kako se pomoću umjetnih neuronskih mreža kao algoritma može ostvariti proces učenja koji za posljedicu ima mogućnost aproksimacije ili ekstrapolacije novih podataka, možemo proučiti konkretne primjere već razvijenih sustava računalnog vida koji se koristi kao identifikator osoba na temelju njihovih lica. Za početak, nekakva osnovna definicija sustava za prepoznavanje lica ukratko opisuje što to zapravo jest. Sustav prepoznavanja lica je tehnologija koja ima mogućnost uparivanja ljudskog lica sa digitalne fotografije ili videa sa onim u bazi podataka, što se često koristi za identifikaciju osoba kroz razne verifikacijske sustave temeljem uočavanja i mjerenja značajki na danoj slici ljudskog lica [20]. Jednostavnije rečeno, sustav uzima slike ili sličice videa kao ulazne podatke te prepoznaje karakteristike i značajke svakog ljudskog lica koje uspoređuje sa nekakvim otprije poznatim podacima te donosi zaključak o kojoj se osobi na slici radi. Ovakvi su se sustavi relativno nedavno razvili te jako napredovali u zadnjih nekoliko godina zbog vrlo brzih i jeftinih te vrlo moćnih procesorskih i grafičkih komponenti koje omogućuju obradu slika u vrlo brzom vremenu pri visokim rezolucijama.

6.1. Povijest sustava za prepoznavanje lica

Ideje o automatiziranom sustavu prepoznavanja osoba na temelju lica počele su se razvijati u 60-im godinama prošlog stoljeća. U početku su ljudi morali ručno unositi koordinate pojedinih značajki ljudskog lica na slici, kao npr. središte zjenice oka, unutarnje i vanjske kutove očiju i mjesta gdje počinje rasti kosa. Te su koordinate služile kako bi se moglo izračunati 20-ak udaljenosti na licu, kao što su širina usta i razmak između očiju [20]. Obrađene su se slike spremale u bazu podataka, a prilikom testiranja nove slike, sustav bi uspoređivao izračunate udaljenosti nove slike sa udaljenostima na slikama iz baze podataka te davao potencijalna rješenja koja odgovaraju toj osobi.

Takeo Kanade je 1970. javno demonstrirao sustav prepoznavanja lica koji je mogao locirati značajke poput brade i izračunati omjer udaljenosti među ostalim značajkama ljudskog lica bez potrebe za intervencijom čovjeka [20]. Sustav ipak nije bio dovoljno pouzdan u široj primjeni ali je svejedno zainteresirao mnoge koji su se bavili tim područjem. Na slici 13.

nalazi se shematski prikaz kako su se mjerile i opisivale značajke ljudskog lica u početcima razvoja ove grane tehnologije.

Slika 13. Početci razvoja sustava za prepoznavanje lica [21]

90-tih godina američka je vojna organizacija DARPA počela razvijati vlastitu tehnologiju prepoznavanja lica predvođenu programom FERET [20]. Taj je program trebao omogućiti potporu u svakodnevnom životu i radu policije, zaštitara, tajnih službi i ostalih predstavnika sigurnosti. Provedbom programa, utvrdilo se da većina sustava prepoznavanja lica nije dovoljno pouzdana, no dio je razvijenih sustava ipak mogao pouzdano prepoznavati lica na fotografijama sa kontroliranom okolinom. FERET je također uzrokovao osnutak nekoliko kompanija koje su se bavile ovom tematikom, kao što su Vision Corporation, Miros Inc i Viisage Technology. Nakon toga, sustavi prepoznavanja lica počeli su se šire primjenjivati. Američki Zavod za motorna vozila (*engl. Department of motor vehicles, DMV*) bio je među prvim institucijama koje su uvele ovu tehnologiju kako bi se spriječilo izdavanje više vozačkih dozvola pod drugim lažnim imenima. Zatvorski je sustav, također imao koristi od razvitka ove tehnologije jer se pomoću sustava prepoznavanja lica, ugrađenog u bazu slika zatvorenika prilikom njihovog uhićenja, moglo pratiti kriminalce po cijelom području SAD-a.

Sustavi temeljeni isključivo na značajkama lica, postali su manje korišteni krajem prošlog stoljeća, razvitkom metode elastičnog podudaranja koju je razvio Christoph von der Malsburg i njegov tim na sveučilištu u Ruhru pomoću koje se moglo detektirati lice na slici

korištenjem segmentacije kože [20]. Ta je metoda pokazivala znatno bolje rezultate od prijašnjih, pa je otprilike u to doba počela komercijalna prodaja sustava za prepoznavanje lica.

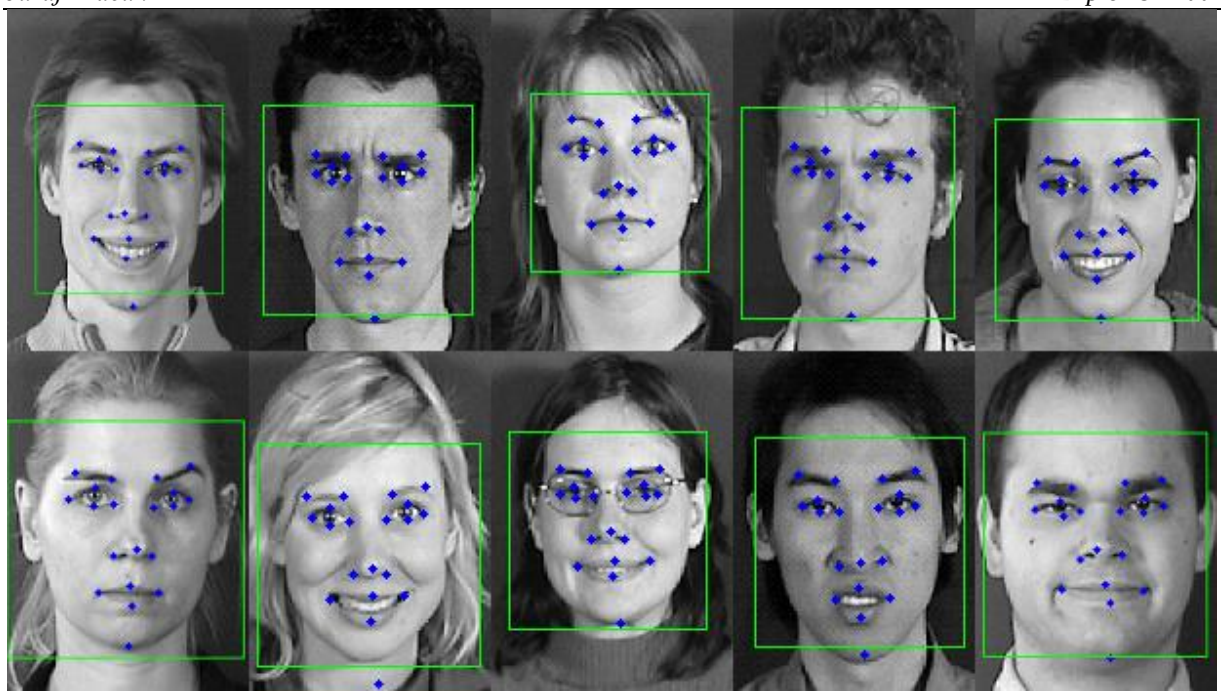
Prepoznavanje lica na videu u stvarnom vremenu postalo je moguće tek 2001. pomoću Viola-Jones sustava za detekciju objekata [20]. Paul Viola i Michael Jones zajedničkim su radom i kombinacijom vlastitih metoda prepoznavanja objekata na slici razvili AdaBoost algoritam koji je bio prvi sustav prepoznavanja prednje strane lica u stvarnom vremenu. Taj je algoritam i danas osnovni temelj nadogradnje raznih sustava koji koriste ovu tehnologiju.

6.2. Algoritmi korišteni pri prepoznavanju lica

Današnji se sustavi prepoznavanja lica temelje na različitim pozadinskim načinima na koje se izvlače značajke sa slike i ljudskog lica. Ljudskom mozgu to ne predstavlja neki preveliki problem, ali u svijetu nula i jedinica koje računalo treba interpretirati, to može biti značajan izazov koji se temelji da dobrom prepoznavanju uzoraka i njihovoj obradi. Generalno, tijek prepoznavanja lica i odgovarajuće osobe kojoj to lice pripada, može se podijeliti u četiri koraka. U prvom se koraku treba detektirati ljudsko lice na slici te izdvojiti od ostatka okoline. Zatim se za izdvojeno lice određuje položaj i udaljenost zajedno s ostalim faktorima kao što su svijetlost i kut gledanja. Treći je korak odrediti i izmjeriti pojedine značajke lica poput očiju, usta i nosa te ih precizno lokalizirati. Zadnji je korak, usporedba dobivenih značajki lica tj. slike lica sa bazom podataka te određivanje kome to lice pripada [20].

6.2.1. Tradicionalni algoritmi prepoznavanja lica

Kao što je već objašnjeno u ovom poglavlju, najčešći i najjednostavniji je način prepoznavanja lica pronalazak odgovarajućih značajki čijom se interpretacijom zaključuje o kojoj se osobi iz baze podataka radi. Neki algoritmi koriste i razne operacije na slikama prije njihovog analiziranja pri čemu se pokušavaju spremiti informacije o slici koje su značajne samo za prepoznavanje lica. Takve se modificirane slike onda uspoređuju sa bazom podataka te se donose pripadajuće odluke. Ovi se algoritmi generalno mogu podijeliti u dvije veće skupine ovisno kako interpretiraju sliku lica. Prvu skupinu čine holistički algoritmi koji pokušavaju prepoznati cijelo lice osobe, dok druga skupina koristi podjelu lica na značajke pomoću kojih se to lice identificira. Primjer slika na kojima algoritam dijeli lice na značajke kako bi se odredila pripadajuća osoba, prikazan je na slici 14.



Slika 14. Algoritmom detektirane značajke lica [22]

6.2.2. Prepoznavanje lica s udaljenosti

Za razliku od klasičnih sustava prepoznavanja lica, problemi koji se često javljaju kod sustava nadzornih kamera ili sličnih sigurnosnih sistema, su problemi niske rezolucije slika koje bi sustav trebao obraditi i analizirati. Naravno, što su osobe udaljenije od same kamere, to će biti manje piksela dostupno sustavu za obradu njihovih lica. Zbog takvih se problema, često koriste dodatne operacije izoštravanja slika, odnosno povećanje rezolucije a da se pritom sačuvaju sve vrijedne značajke. Postoje razni filteri i algoritmi koji prolaze kroz sliku, zapisanu u matričnom obliku i obrađuju okolinu oko svakog piksela te generiraju novu sliku koja će imati pripadajuće posebne značajke ovisno o primijenjenom filteru.

Primjerice, jedan od vrlo čestih koraka u svakoj analizi slika je određivanje, tj. detekcija rubova. Rubovi su na slici zapravo, jednostavno rečeno, nagle promjene kontinuiteta poput promjene boje, osvjetljenja, hrapavosti površine i sl. Njihova se važnost očituje skoro u svakom području računalnoga vida, a najviše upravo prilikom prepoznavanja nekakvih objekata ili posebnih značajki na slici. No, problem detekcije rubova nije tako jednostavan. Ako zapišemo vrijednosti piksela na realnim slikama u obliku njihovog intenziteta, često dobivamo velike šumove gdje nije jednoznačno određeno nalazi li se tu zaista rub ili ne. Stoga je potrebno koristiti odgovarajuće, već razvijene algoritme koji mogu s velikom preciznošću

detektirati sve rubove na slici. Najpoznatiji i najkorišteniji takav alat detekcije rubova je *Canny edge detector*. Postupak primjene ovog filtera može se svesti na pet koraka kojima se mogu detektirati rubovi. Prvi je korak primjena *Gaussian* filtera. Njegova je uloga u ovom postupku smanjenje buke koja se javlja na slici. To se može postići kreiranjem konvolucijske matrice, izračunavanjem Gausove funkcije u dvodimenzionalnom prostoru koji tvore pikseli slike. Jednadžba Gausove funkcije za takav slučaj glasi:

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}, \tag{26}$$

pri čemu su x i y udaljenosti od ishodišta, u ovom slučaju udaljenosti od središnjeg piksela a σ standardna devijacija Gausove distribucije. Izračunavanjem vrijednosti prema izrazu (26) u okolini središnjeg piksela, dobiva se matrica konvolucijska matrica koja će biti primijenjena na svaki piksel slike. Te su matrice uvijek simetrične i obično neparnog broja redaka i stupaca kako bi se upravo mogao isticati središnji piksel koji se mijenja. Primjenom ovako generirane matrice dobiva se lagano zamućenje slike koje eliminira slabe izvore buke, pri čemu se rubovi i diskontinuiteti ipak mogu očuvati. Primjer prolaska *Gaussian* filtera kroz sliku i krajnji rezultat te primjene prikazan je na slici 15.

Slika 15. Primjena *Gaussian* filtera [23]

Na slici 15. prikazana je primjena *Gaussian* filtera pomoću MATLAB-ove ugrađene funkcije *imgaussfit*, čiji je krajnji rezultat prikazan na desnoj strani. Nakon primjene *Gaussian* filtera, drugi je korak u postupku detekcije rubova pomoću *Canny edge detector*-a, pronalazak gradijenata intenziteta na slici kako bi se otkrili potencijalni rubovi na slici. U ovom se koraku koriste neki od postojećih operatora detekcije rubova, poput Roberts, Prewitt ili Sobel, koji

mogu detektirati vertikalne, horizontalne i dijagonalne rubove. Ti operatori vraćaju prvu derivaciju funkcije intenziteta u x i y smjerovima [24]. Na temelju tih derivacija, može se odrediti smjer i iznos gradijenta prema izrazima:

$$G = \sqrt{{G_x}^2 + {G_y}^2}, \tag{27}$$

$$\theta = atan2(G_x, G_y), \tag{28}$$

gdje su G_x i G_y derivacije funkcije intenziteta u smjeru x i y, a θ kut koji gradijent zatvara sa x osi. Treći je korak, primjena supresije donjom granicom, ispod koje se nalaze „slabi" rubovi. Dakle, trenutni piksel uspoređuje se sa pikselima u pozitivnom i negativnom smjeru gradijenta te ukoliko je taj piksel veći od ostalih, njegova se vrijednost ne mijenja, dok je u suprotnom ta vrijednost potisnuta [24]. Tim se procesom miču tanki i neznačajni rubovi na slici.

Četvrti je korak ovog procesa, primjena tzv. *double threshold* metode. Nakon prethodnog koraka, moguće je još uvijek naći rubove na slici koji zapravo nisu tamo, nego su samo nastali zbog nagle promjene boje ili šumova pa je potrebno odrediti koji od tih rubova spadaju u jake a koji u slabe rubove. Najprije se određuju dvije vrijednosti koje predstavljaju donju i gornju granicu. Zatim se iznosi gradijenta na rubovima uspoređuju s vrijednostima tih granica. Ukoliko je vrijednost gradijenta veća od gornje granice, taj se rub može označiti kao jaki rub i on ostaje nepromijenjen. Ako je vrijednost gradijenta manja od gornje granice, a veća od donje granice, taj će se rub klasificirati kao slab, a ukoliko je gradijent manji od donje granice, tada će taj rub biti potisnut [24].

U posljednjem se koraku provjeravaju rubovi koji su bili svrstani u slabe rubove u prethodnom koraku. Većina je slabih rubova koji uistinu predstavljaju rubove na slici, barem djelomično povezana sa jakim rubovima. Tako se gleda okolina svakog slabog rube te ukoliko se unutar te okoline nalazi jaki rub, slabi rub se ostavlja na slici, dok se u suprotnom potiskuje. Na taj bi način dobivena slika trebala predstavljati sve značajne rubove koji se nalaze na izvornoj slici. Ukoliko krajnja slika ne predstavlja dovoljno dobro tražene rubove, moguće je mijenjati parametre koji će utjecati na rezultat poput iznosa donje i gornje granice, dimenzije konvolucijske matrice u prvom koraku i sl. Primjer kako izgleda krajnji rezultat *Canny edge detector*-a u usporedbi sa originalnom slikom, prikazan je na slici 16.

Slika 16. ***Canny edge detector*** **[25]**

Ovakvi i puno kompleksniji algoritmi koriste se za razne analize piksela slike koje mogu rezultirati vrlo dobrim rezultatima. Tako se u području detektiranja lica na većim udaljenostima, npr. koriste postupci povećavanja rezolucije čime se dobiva oštrija i bogatija slika koju je kasnije lakše analizirati. Ti se postupci temelje na sličnim principima i pretvorbama piksela kao i navedeni *Canny edge detector*.

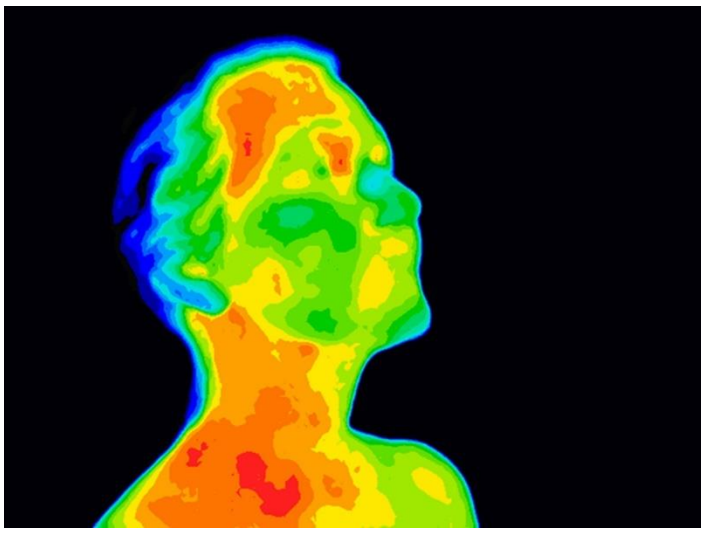
6.2.3. Trodimenzionalno prepoznavanje lica

Iako dosta rjeđi i kompleksniji proces od tradicionalnog sustava za prepoznavanje lica, trodimenzionalni sustavi imaju nekoliko značajnih prednosti. Potrebno je pozicionirati mnogo raznih 3D senzora kako bi se kreirao model ljudskog lica koji sadrži trodimenzionalne karakteristike tog lica. Prednost je u tome što jednom tako kreirani model lica može prepoznati tu osobu iz vrlo širokog raspona kutova gledanja. Također, utjecaj svijetlosti uvelike je smanjen kod ovakvih sustava. Unatoč relativno mladom području primjene i uporabe, velika se sredstva ulažu u daljnji razvoj ove tehnologije, pogotovo u primjeni kod detekcije emocija na ljudskom licu u realnom vremenu. Primjer modela generiranog pomoću navedenih senzora nalazi se na slici 17.

Slika 17. 3D model za prepoznavanje ljudskog lica [20]

6.2.4. Prepoznavanje lica pomoću termalnih kamera

Ovaj način prepoznavanja lica daje najpreciznije rezultate kada je dio lica prekriven šminkom, naočalama, brkovima ili bradom i sl. Termalne kamere mogu snimiti oblik i konturu lica bez obzira na spomenute dodatke koji će možda imati znatan utjecaj na rezultate prije navedenih sustava. Svijetlost, također nije bitan faktor koji će imati znatan utjecaj na krajnji rezultat. Pored toga, moguće je i istovremeno mjerenje temperature osobe, što je bilo od vrlo velike važnosti tijekom pandemije. No, njihov su limitirajući faktor baza podataka kao i slaba točnost pri analizi slika termalnom kamerom koja se ne nalazi u zatvorenom prostoru. Postoje mnogi pokušaji da se ovakvi sustavi spoje s onim tradicionalnim tako da se iz oba sustava iskoristi ono pozitivno, no ti su pokušaji još većinom u razvojnim stadijima [20]. Na slici 18. prikazano je ljudsko lice uslikano termalnom kamerom.



Slika 18. Termalni prikaz ljudskog lica [26]

6.3. Primjena sustava za prepoznavanje lica

Sustave prepoznavanja lica danas možemo vrlo često susresti u raznim područjima koja se bave sustavnim identificiranjem pojedine osobe ili jednostavno kao izvor zabave na društvenim mrežama. U nastavku su navedeni neki od najkorištenijih i najpoznatijih primjera uporabe tih sustava.

6.3.1. Prepoznavanje lica u sigurnosnim sustavima

Vjerojatno najbolji i najprecizniji način detekcije pojedine osobe i njezine identifikacije u svrhu sigurnosnih obveza, upravo je tehnologija prepoznavanja lica pomoću umjetnih neuronskih mreža. Gotovo sve veće zemlje u cijelom svijetu već su dobrim dijelom uvele umjetnu inteligenciju u svoje sustave nadzornih kamera, pogotovo na graničnim prijelazima ili aerodromima. Tako primjerice, SAD koristi jedan od najvećih sustava za prepoznavanje lica u cijelome svijetu, pri čemu se u bazi podataka nalaze slike gotovo 117 milijuna odraslih Amerikanaca koje su dobivene iz vozačkih dozvola i ostalih državnih dokumenata [20]. Taj je sustav koristio FBI za istraživačke svrhe, a policija u San Diegu i Los Angelesu kako bi identificirali osobu na slici koju su oni uslikali. Američka granična policija, također koristi umjetnu inteligenciju kako bi pronašla ilegalne migrante koji su uspjeli prijeći granicu bez valjane dozvole.

U Kini se, s druge strane, već 2006. uveo Skynet Project kojeg je inicirala tamošnja vlada kako bi se prepoznavanje lica uvelo u sustav javnih nadzornih kamera širom zemlje, koje mogu u realnom vremenu pratiti i identificirati svaku osobu, a čiji je broj 2018. iznosio oko 20 milijuna [20]. Takav sustav korišten je u mnogim gradovima kako bi se pronašle osobe za kojima policija ili neka druga služba traga već nekoliko godina. Pored toga, u mnogim se turističkim destinacijama u Kini koristi sličan sustav kako bi se mogli pratiti i dočekati svi posjetitelji. Naravno, sa nadzornim kamerama postavljenim na skoro svakih 100 metara [20], dolazi do velike zabrinutosti za privatnost građana i njihovih svakodnevica, što je rezultiralo sa nekoliko tužbi protiv pojedinih kineskih organizacija.

Granični prijelazi između Australije i Novog Zelanda koriste automatizirani sustav SmartGate koji uspoređuje slike putnika sa nadzornih kamera i podatke u mikročipovima e-putovnica

[20]. Sličan se sustav koristi i u internacionalnim zračnim lukama u Kanadi gdje se također uspoređuje lice osobe sa nadzorne kamere sa slikom iz putovnice. Taj je sustav prvi uveden u Vancouveru 2017. godine, a u ostatku Kanade u 2018. i 2019. [20]. Većina je i ostalih zemalja već uvela ili bar planira uvesti slične sustave koji će se koristiti kao poboljšanje sigurnosti i olakšanje trenutnim policijskim poslovima kako bi se smanjila stopa kriminala. Primjer sustava za sigurnosnu primjenu u zračnim lukama, prikazan je na slici 19.

Slika 19. Prepoznavanje lica u zračnoj luci [30]

6.3.2. Face ID

Jedan od poznatijih primjera korištenih prilikom autentifikacije osobe, svakako je sustav prepoznavanja lica *Face ID* kojega je razvila tvrtka Apple. Pomoću ovoga je sustava moguća biometrijska autentifikacija osobe kako bi se moglo otključavati određeni uređaj, ostvarivati plaćanja i transakcije, omogućiti pristup senzitivnim podacima te još mnoštvo drugih mogućnosti. Rad ovoga sustava omogućuju tri komponente. Prva je projektor točaka, koji projektira mrežu sitnih infracrvenih točaka na korisnikovo lice. Druga je komponenta zadužena za obasjavanje korisnikovog lica infracrvenom svjetlosti, a pomoću treće se komponente uzimaju slike sa infracrvenom kamerom te se dobiveni podaci i uzorci očitavaju te se formira 3D model lica korisnika [31]. Taj se model uspoređuje sa spremljenim modelom kojeg je korisnik zadao te se određuje odgovara li to korisniku ili ne. Sustav je također kreiran s namjerom prepoznavanja odgovarajuće osobe sa naočalama, šminkom, odjećom, brkovima ili bradom te ostalim faktorima koji utječu na lice osobe uključujući i promjene uzrokovane starenjem. Sustav se prvi put pojavio kao sastavni dio iPhone X serije mobitela u rujnu 2017.

te se koristi u svim novijim modelima Appleovih mobitela i tableta [31]. Kritike na račun ovoga softvera najviše se tiču nekonzistentnog prepoznavanja jednojajčanih blizanaca ili bliske rodbine samog korisnika, potencijalne mogućnosti otključavanja uređaja jednostavnim pokazivanjem na lice korisnika bez njegovog ili njenog pristanka, mogućom uporabom maske koja bi predstavljala korisnika te bi tako netko drugi mogao imati pristup uređaju, odobravanje pristupa podataka stranim aplikacijama koje koriste ovaj sustav te korištenju medicinskih maski za lice koje su u početku pandemije predstavljale problem, no kasnije je ipak nađeno rješenje koje može prepoznati korisnika bez obzira na masku za lice. Uporaba *Face ID* sustava prikazana je na slici 20.

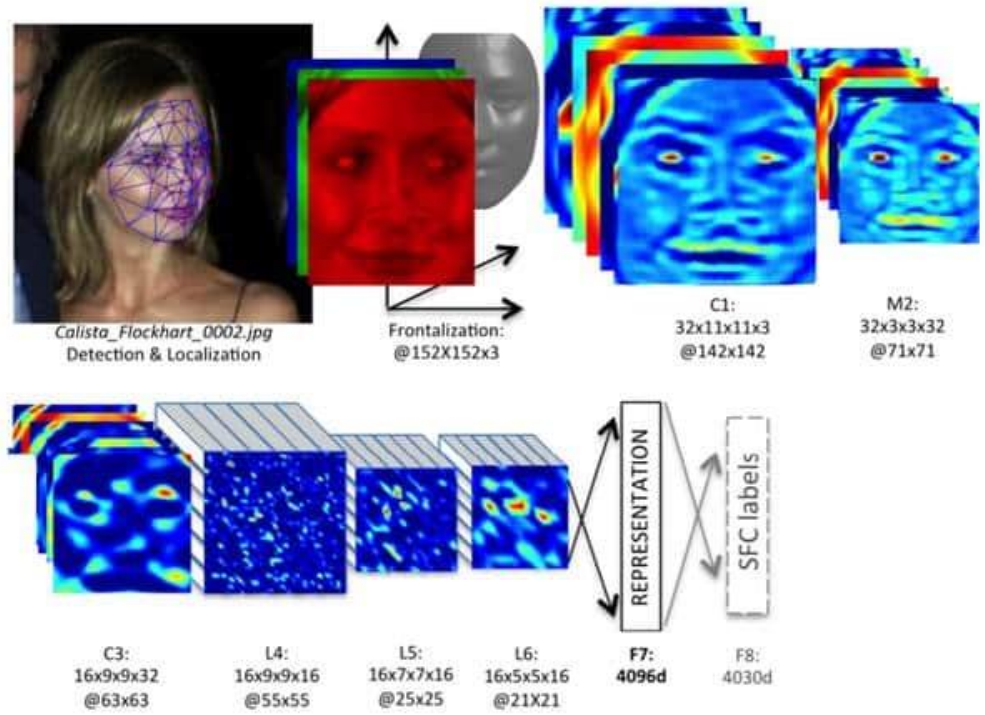
Slika 20. *Face ID* sustav [32]

6.3.3. Primjena u društvenim mrežama

Društvene su mreže veoma profitirale koristeći tehnologiju prepoznavanja lica putem umjetne inteligencije. Već je 2014. tvrtka Looksery, nakon prikupljenih novčanih sredstava putem donacija, plasirala svoju aplikaciju koja je korisnicima pružala video pozive sa drugim korisnicima pri čemu je postojala mogućnost primjene raznih filtera koji bi mijenjali izgled korisnikovog lica [20]. Kompaniju je kasnije kupila tvrtka SnapChat koja je navedene filtere ugradila u svoju aplikaciju što je praktički postao njen prepoznatljivi čimbenik.

Facebook je, također, razvio svoj sustav prepoznavanja lica pomoću dubokog učenja koji je nazvan *DeepFace*. Sustav se sastoji od neuronske mreže sa 9 slojeva te više od 120 milijuna

težina koje povezuju neurone, a bazu podataka za treniranje mreže čini više od 4 milijuna slika koje su korisnici postavili na Facebook [20]. Sustav se koristi za prepoznavanje osoba na slikama, a njegova je navodna točnost 97%, što je više od sustava kojega koristi FBI čija je točnost 85% [20]. Primjer rada *DeepFace* sustava te načina analiziranja slike prikazan je na slici 21.



Slika 21. *DeepFace* sustav za prepoznavanje lica [34]

Popularna aplikacija TikTok, koristi se umjetnom inteligencijom kako bi predvidjela sadržaj prilagođen svakom pojedinom korisniku. Iako su u kompaniji 2020. objavili kako ne koriste sustave prepoznavanja lica prilikom određivanja što bi korisnik htio gledati, u veljači 2021. TikTok je pristao na dogovor u iznosu od 92 milijuna američkih dolara u tužbi koja je govorila o korištenju sustava za prepoznavanje lica u korisničkim videima i svom algoritmu kako bi se odredila dob, spol i etnička pripadnost korisnika [33].

7. TENSORFLOW PROGRAMSKA BIBLIOTEKA

Za realizaciju već spomenutih algoritama za strojno učenje, potreban je neki oblik računalne podrške koja će to omogućiti. U današnje je vrijeme stvarno dostupno mnogo dobro razvijenih programskih jezika koji se sastoje od velikog broja različitih modula specijaliziranih za neke posebne namjene. Programski jezik koji je danas možda i najpoznatiji i najčešće primjenjivani jezik je Python. Python je interpretirani, objektno orijentirani programski jezik visoke razine sa dinamičkom semantikom [27]. Kao veoma dobro prilagođen korisnicima i lako razumljiv jezik, Python je postao vrlo česta pojava u svijetu programiranja ne samo kao kućni hobi nego kao i ozbiljan program koji se koristi u raznim granama tehnologije. Postoje cijeli industrijski programi napisani Python jezikom koji se redovno upotrebljavaju u mnogim inženjerskim poslovima. Prednost Python-a, također se odražava kroz mogućnosti korištenja raznih otvorenih biblioteka koje nisu osnovni dio samog Python programa, nego se mogu vrlo jednostavno i lako dodati u program što omogućuje veoma široku primjenu u raznim područjima informacijskih tehnologija.

Kao što je maloprije rečeno, Python se može nadograđivati raznim modulima razvijenima za različite svrhe. Jedan od takvih modula, korišten i u ovom radu, je TensorFlow. Jednostavnim riječima, TensorFlow je platforma otvorenog koda za strojno učenje [28]. Dakle, ova je platforma predviđena za razne probleme strojnog učenja koji se javljaju u istraživačkim i praktičnim primjenama. Pomoću nje je moguće kreirati vlastite modele neuronskih mreža te vrlo jednostavno i pregledno definirati svaki pojedini sloj u mreži. Sve one jednadžbe i izvodi napisani u poglavlju 5.2, ovdje su već definirani unutar odgovarajućih funkcija. Na taj se način omogućuje korisnicima, koji se tek upoznaju s područjem strojnog učenja i umjetnih neuronskih mreža, slobodnu izradu vlastitog sustava umjetne inteligencije. Naravno, složenije strukture i vrste neuronskih mreža, zahtijevat će od korisnika bolje poznavanje tematike strojnog učenja, pogotovo ukoliko se želi postići vrlo precizan i točan rezultat zadanog problema. U TensorFlow-u, također postoje i takve, specijalno definirane funkcije koje vrlo lako mogu odraditi složenije procese sa puno većim brojem neurona i veza u neuronskim mrežama. Primjerice, cijeli slojevi sa proizvoljno definiranim brojem neurona i aktivacijskim funkcijama mogu se zamijeniti sa samo nekoliko redova koda koji će se automatski povezati

sa neuronima prethodnog sloja i provesti sve potrebne radnje, kao što se dodjeljivanje težina vezama, određivanje na koji će način mreža učiti i sl. Ti se svi parametri mogu i zasebno definirati ako korisniku ne odgovaraju unaprijed zadane postavke.

Sve operacije nad neuronskom mrežom koja je središnji dio ovog rada, također su napravljene pomoću TensorFlow modula u Python programskom jeziku. Pored toga, sve potrebne pripreme i manipulacije bazama podataka, odnosno slikama osoba čije će se lice prepoznavati, te njihovom obradom prije nego što se ubace na ulaz neuronske mreže, napravljene su isto tako ovom programskom bibliotekom. Mogućnosti koje TensorFlow pruža u području strojnog učenja, stvarno su velike. Tako nije ni čudo što kompanije kao što su Airbnb, Airbus, Coca-Cola, Intel i mnoge druge upravo koriste TensorFlow kako bi kreirali vlastite modele za strojno učenje, prilagođene njihovim potrebama. Čak i Google koristi TensorFlow kako bi implementirao strojno učenje u rezultatima internetske tražilice, sustavu elektroničke pošte, Gmail i sustavu za prijevod s jednog jezika na drugi, Google Translate, kako bi pomogao istraživačima u novim područjima te omogućio napredak u ljudskim i prirodnim izazovima [29].

8. PROGRAM ZA IDENTIFIKACIJU OSOBE

Nakon što su definirani temeljni pojmovi poput neuronskih mreža, računalnog vida, TensorFlow biblioteke i ostalih potrebnih stvari, konačno je moguće kreirati vlastiti sustav za prepoznavanje lica osobe u realnom vremenu, što je i sami cilj ovoga rada. U ovome će se poglavlju objasniti kompletan postupak izrade te opisati sve značajke i sposobnosti, zaključno sa rezultatima testa na realnim osobama, cijelog programa.

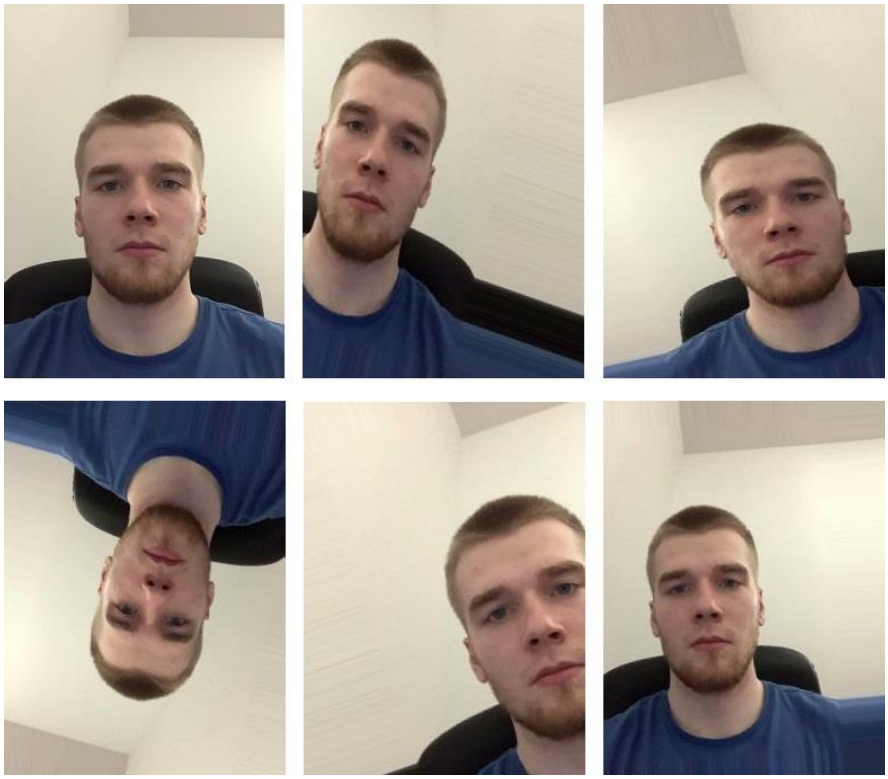
Generalno, program se može podijeliti na dvije glavne zadaće koje treba napraviti. Prva je treniranje modela neuronske mreže za klasifikaciju lica osobe koju je potrebno prepoznati. Druga je zadaća programa, prilagoditi model na način da se u realnom vremenu, pomoću web kamere može pratiti i prepoznati odgovarajuća osoba. Da bi se ove dvije zadaće mogle uspješno ostvariti, potrebno je koristiti valjane ulazne podatke, dobru strukturu i parametre neuronske mreže te omogućiti rad u realnom vremenu gdje se nekoliko desetaka slika treba obraditi svake sekunde. Za početak, potrebno je definirati i opisati ulazne podatke koji će se koristiti u samome programu, kao i na koji se način oni mogu dobiti.

8.1. Ulazni skup podataka

Prvi je korak u samome procesu kreiranja jednog ovakvoga sustava, prikupljanje dobrog i raznovrsnog skupa podataka, u ovome slučaju slika osoba koje će se prepoznavati. Najjednostavniji je način korištenje slika pojedine osobe na kojima se nalazi prikaz lica u što raznovrsnijim položajima i udaljenostima u odnosu na kameru. Takav bi pristup vjerojatno imao najbolji i najprecizniji rezultat prilikom testiranja samog programa, no postoji nekoliko prepreka koje sprječavaju takav pristup ovome načinu definiranja ulaznih podataka. Broj potrebnih slika trebao bi biti što veći kako bi mreža mogla dobro razlikovati pojedinu osobu u odnosu na druge, što povlači za sobom velik broj slika svake osobe koja se želi identificirati pomoću ovoga programa. Za malen broj osoba uz ne preveliku točnost sustava, to bi i moglo biti ostvarivo, no ukoliko želimo kreirati uistinu robustan, fleksibilan i precizan sustav, baza podataka sa slikama svih osoba bila bi jednostavno prevelika te bi uvelike ovisila o dostupnim slikama svake osobe, što nije uvijek na raspolaganju. Dakle, kako bi se spriječio problem potrebe za velikim brojem slika svih osoba koje nisu uvijek dostupne, može se kreirati neko

drugačije rješenje kako dobro istrenirati sustav, a da se pri tome što jednostavnije prođe kroz proces prikupljanja ulaznih podataka. Ovdje će biti navedena konkretna dva načina kako se ulazni podaci mogu automatizirano prikupljati ili modificirati kako bi mreža mogla iskoristiti što više informacija sa svake slike, pri čemu je ljudska potreba za ručnim prikupljanjem podataka što manja.

Prvi je način, modifikacija već postojećeg skupa podataka, odnosno slika na način da se na svakoj slici primjeni neka posebna operacija koja će promijeniti njen prvotni izgled. Ta se tehnika obično naziva povećanje podataka, *engl. Dana augmentation* i u kontekstu programa za prepoznavanje lica označava primjenu nekih od modifikacija izvorne slike poput horizontalne ili vertikalne refleksije, rotacije, zumiranja, promjene boje i sl. Na taj se način na temelju jedne slike može generirati nekoliko desetaka sličnih slika koje sadrže sve bitne značajke kao i izvorna slika, a ipak predstavljaju razlike u ulaznom skupu podataka koji će poboljšati preciznost rada neuronske mreže. Primjenom ovakvoga pristupa, otklanjamo potrebu za korištenjem mnogo različitih slika kako bi proširili raznovrsnost slika za učenje mreže. Stoga je za svaku osobu dovoljan manji ulazni skup podataka, koji će kasnije biti proširen primjenom ove tehnike. Primjer slika generiranih ovim načinom, kao i sama izvorna slika, nalazi se na slici 22.



Slika 22. Primjer umjetnog povećanja ulaznog skupa podataka

U gornjem lijevom kutu slike 22. nalazi se izvorna slika, dok su oko nje prikazane umjetno generirane slike gore navedenom tehnikom. Ovim se jednostavnim načinom početni skup slika može povećati do nekoliko desetaka puta čime nastaje puno raznovrsnija baza podataka, ali se usporava proces učenja mreže, pa treba o tome voditi računa ukoliko je taj faktor bitan. Također, važno je napomenuti kako se sve ove pretvorbe mogu jednostavno kreirati pomoću TensorFlow biblioteke opisane u poglavlju 7. u kojoj se već nalaze gotove funkcije koje odrađuju sve operacije ovisno o ulaznim parametrima koje je korisnik postavio. Jedan od nedostataka ove metode, korištenje je slika različitih dimenzija, koje kada se transformiraju u dimenzije potrebne neuronskoj mreži, mogu biti vrlo deformirane i prikazivati nerealne proporcije ljudskog lica što nije pogodno za sami program.

Iako je ovaj način vrlo jednostavan i dosta pogodan za implementaciju, u samome programu neće se koristiti samo ova metoda prikupljanja odgovarajućeg broja slika pojedine osobe. Najveći je nedostatak prve metode to što programu i dalje trebaju slike osobe koju je potrebno prepoznati prije samog početka rada programa. Stoga se, u drugoj metodi koristi sama web kamera kao izvor ulaznog skupa podataka. Prilikom pokretanja programa, uključuje se ista kamera koja će se koristiti za prepoznavanje osobe u realnom vremenu te se pomoću nje uzimaju slike svakih nekoliko sličica videa koji ona snima, te se takve slike automatski spremaju u datoteku koja sadrži ulazne podatke. Kada se prikupi dovoljan broj slika, kamera se gasi te se nastavlja proces učenja neuronske mreže. Ovakvim je pristupom moguće prikupljanje slika željene osobe bez potrebe za posjedovanjem slika te osobe prije nego se sustav može testirati. Na sličan se način i u ostalim programima za prepoznavanje lica, poput navedenog *Face ID-a*, prikupljaju slike korisnika čije se lice na početku snima te tako započinje proces učenja neuronske mreže. Ovdje je vrlo bitno napomenuti, kako će se najbolji rezultati ostvariti ukoliko se osoba prilikom snimanja lagano pomiče u odnosu na kameru kako bi se dobila što veća raznovrsnost ulaznog skupa podataka.

Ako se iskoriste prednosti obiju navedenih metoda, moguće je kreirati uistinu dobar i pouzdan sustav koji će zaista imati na raspolaganju vrlo širok asortiman slika svake osobe koju treba prepoznati. Dakle, u samome programu, koristit će se kombinacija ovih metoda na način da se slike pojedine osobe, uzete pomoću web kamere, umjetno modificiraju te povećaju raznovrsnost ulaznog skupa kao što je to na početku poglavlja 8.1. opisano. Pored toga, kako bi se spriječilo uzimanje mutnih slika ili slika dok osoba nije u kadru kamere, slike će se

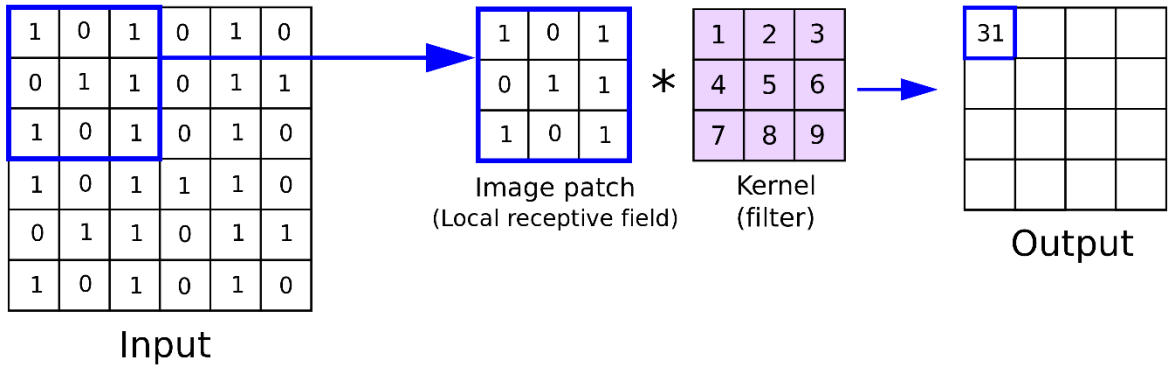
uzimati isključivo kad program prepozna da se na slici uistinu nalazi osoba. Ovaj je dio programa detaljnije opisan u poglavlju 8.3. Ovdje je potrebno razlikovati termine kao što su detekcija osobe, koji označava da se na zadanoj slici nalazi neka osoba, odnosno njeno lice i pojam identifikacije ili prepoznavanja osobe, čime se označava da se detektiranoj osobi dodjeljuje odgovarajuća oznaka iz baze podataka. Ova distinkcija vrlo je bitna jer ta dva sustava mogu raditi različite stvari i različito se koristiti, a njihovim se spajanjem može napraviti sustav koji će moći detektirati i prepoznati osobu na slici ili videu. Kada su ulazni podaci dobiveni, možemo prijeći na samu strukturu i parametre neuronske mreže koja će se koristiti u ovome programu.

8.2. Struktura i parametri neuronske mreže

Za razliku od vrlo jednostavnog primjera neuronske mreže opisane u poglavlju 5.2., za potrebe sustava prepoznavanja lica, a samim time i ovoga rada, ipak je potrebno koristiti nešto složeniju strukturu neuronske mreže koja će moći izvršavati puno kompleksnije zadatke. Takve se mreže sastoje od velikog broja slojeva koje bi ručno bilo vrlo zahtjevno sustavno i strukturirano posložiti na način da se svi slojevi ispravno povezuju i daju dobar krajnji rezultat. Potrebne aktivacijske funkcije svakog pojedinog sloja, također bi trebalo vrlo pažljivo ispitati i vidjeti kakav je njihov krajnji utjecaj na izlazu iz mreže. Pored toga, i dalje bi nam trebao veoma velik skup ulaznih podataka koje bi trebalo obraditi kako bi se mreža naučila detektirati i prepoznavati ljudska lica.

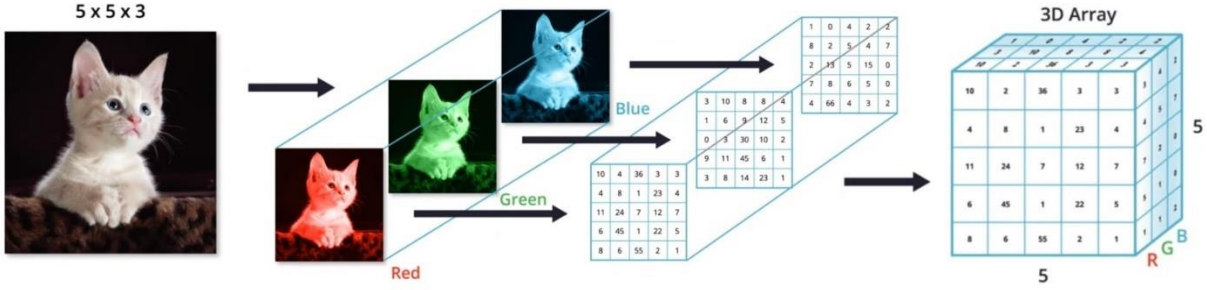
Iz tih se razloga, vrlo često koriste modeli mreža koji su već razvijeni za određene potrebe, kao što je analiza slike u području računalnog vida. U prvoj se fazi ovoga programa trenira neuronska mreža koja je zadužena za identifikaciju osobe u realnom vremenu. Korištena je već strukturirana neuronska mreža prilagođena prepoznavanju objekata na slici, *ResNet50*. *ResNet50* je varijanta *ResNet* modela neuronskih mreža koja se sastoji od 48 konvolucijskih slojeva, jednog *Max Pool* sloja i jednog *Average Pool* sloja [35]. Kako bi mogli potpuno shvatiti prethodnu definiciju, potrebno je definirati neke spomenute pojmove. Najprije ćemo definirati što označavaju *ResNet* modeli neuronskih mreža te koje su njihove karakteristike. Ti modeli spadaju u grupu dubokih konvolucijskih neuronskih mreža koje koriste metode rezidualnog učenja kako bi ostvarile potrebne rezultate. Pojam konvolucijskih neuronskih mreža označava skupinu umjetnih neuronskih mreža koja se često koristi u analizama slika, a temelji se na matematičkim operacijama konvolucije, koja u kontekstu područja neuronskih

mreža označava linearnu operaciju koja uključuje množenje matrica ulaznih podataka sa matricama vrijednosti težina. Jednostavnije rečeno, konvolucijske su mreže dio neuronskih mreža koje služe za analizu slike i imaju barem jedan konvolucijski sloj. Konvolucijskim se slojevima kreira tzv. mapa značajki koja se dobiva spomenutim množenjem filtera, često nazivanog i *kernel* (matrica vrijednosti težina) sa matricom na ulazu. Nakon završenog množenja dobivamo novu matricu, tj. mapu značajki koju dalje provlačimo kroz aktivacijsku funkciju. Na slici 23. prikazan je primjer načina rada konvolucijskog sloja neuronske mreže čiji je izlaz mapa značajki.



Slika 23. Operacija konvolucije u konvolucijskom sloju neuronske mreže [36]

Ovakvi se slojevi često koriste u obradama slika jer se slike jednostavno mogu reprezentirati u obliku matrica ili tenzora drugog reda u kojima su zapisane vrijednosti svih piksela na slici. Ukoliko je slika crno-bijele boje, tada se ona može zapisati kao matrica gdje svaki piksel predstavlja vrijednost od 0 do 255 pri čemu se često vrijednosti 0 pridružuje potpuno crna boja, a vrijednosti 255 potpuno bijela boja. Na analogan se način mogu zapisati slike koje imaju tri kanala, crveni, zeleni i plavi. Svaki će piksel imati vrijednosti 0 do 255 koje će predstavljati jačinu pojedinog kanala te povezivanjem sva tri kanala, dobiva se izvorna vrijednost boje piksela. To je najjednostavnije interpretirati kao tri matrice koje predstavljaju vrijednosti crvene, zelene i plave boje na slici, povezane u trećoj dimenziji, tvoreći tenzor drugog reda. Dimenzije takvog tenzora bit će jednaki visini slike × širini slike × 3, pri čemu broj 3 predstavlja broj kanala na slici. Jednostavniji prikaz vizualizacije ovakvog zapisa slika, prikazan je na slici 24.

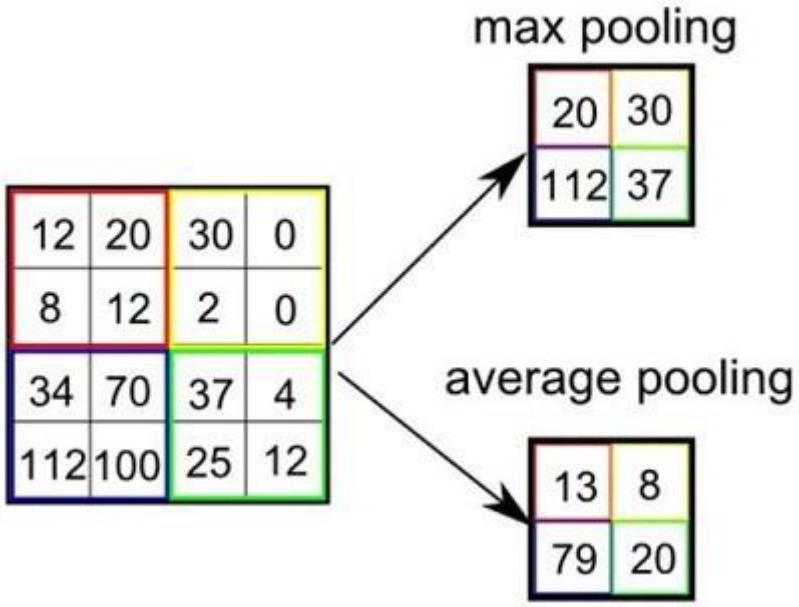


Slika 24. Vizualizacija zapisa slike pomoću RGB kanala [37]

Sada je jasnije vidljivo zašto se konvolucijski slojevi neuronskih mreža koriste za obrađivanje i analizu slika. Primjenom spomenute konvolucije, prikazane na slici 23., moguće je iz početne slike, uporabom odgovarajuće aktivacijske funkcije, izolirati pojedine značajke na slici pomoću kojih se može dobiti neka vrsta predodžbe što se na toj slici nalazi. Ovakvih je slojeva u konvolucijskim mrežama najčešće nekoliko, u ovoj konkretnoj mreži programa 48, koji se mogu ulančano povezati te tako imati vrlo veliku moć razlikovanja i detekcije pojedinih značajki na slici.

Pored konvolucijskih slojeva i ulaznog sloja, koji u biti predstavlja sliku zapisanu na način prikazan na slici 24., *ResNet50* model neuronske mreže sastoji se još od dva sloja. Prvi je spomenuti sloj *Max Pool* ili *Max Pooling* sloj. *Pooling* slojevi, slično kao i konvolucijski slojevi, uzimaju matricu na ulazu te ju transformiraju prema nekom kriteriju kako bi se bolje istaknule značajke slike. Generalno, u ovakvim se slojevima uzima okolina nekog piksela slike te se primjenjuje neka transformacija ovisno o kojoj vrsti *Pooling* sloja se radi. Pa tako, prema samom nazivu, *Max Pooling* sloj uzima nekoliko piksela u okolini trenutnog piksela te kao izlaz postavlja najveću vrijednost piksela u toj okolini [38]. Ovim se slojem, također, postiže smanjenje šuma na slici te se reducira dimenzija same slike što ubrzava proces obrade slike u slijedećem sloju mreže.

Slično kao i *Max Pooling* sloj, *Average Pooling* uzima okolinu trenutnog piksela te računa prosječnu vrijednost svih piksela u toj okolini te ju postavlja na mjesto izlaznog piksela. Iako su ova dva sloja vrlo slična, *Average Pooling* sloj ne postiže smanjenje šumova na slici kao što je to slučaj kod *Max Pooling* sloja. Stoga, možemo reći kako je *Max Pooling* sloj puno bolja opcija kada se uspoređuju ova dva sloja [38]. Vizualni prikaz operacija ovih slojeva nad ulaznom matricom neuronske mreže, prikazan je na slici 25.



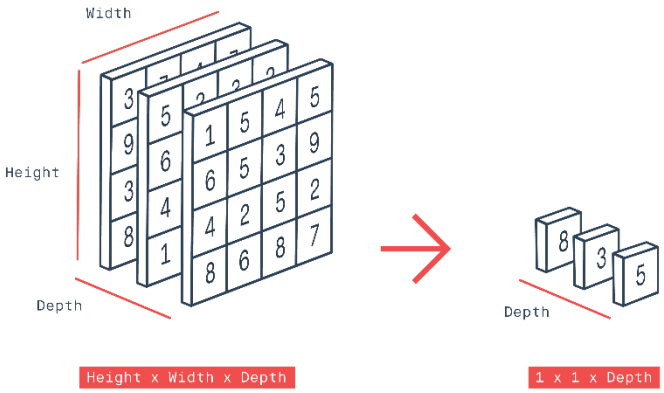
Slika 25. Operacije *Max Pooling* i *Average Pooling* slojeva [39]

Jedina je preostala stvar pri potpunom definiranju *ResNet50* modela, definiranje pojma rezidualnog učenja koje ovaj model koristi. Za razliku od klasičnih konvolucijskih neuronskih mreža, mreže koje koriste rezidualno učenje, umjesto pokušavanja učenja određenih značajki na slici, pokušavaju naučiti neke rezidue. Te se rezidue mogu jednostavno shvatiti kao oduzimanje značajki naučenih na ulazu u sloj [40]. To se ostvaruje u ovome modelu korištenjem veza koje su direktno povezane sa nekim, nekoliko slojeva udaljenim, slojem u mreži. Na taj se način ostvaruje lakše učenje u usporedbi sa klasičnim konvolucijskim mrežama te se uklanja problem smanjenja preciznosti povećanjem dubine neuronske mreže. Taj je problem posebno izražen u vrlo dubokim konvolucijskim neuronskim mrežama jer se povećanjem broja slojeva mreže povećava preciznost ali se jačina signala ostvarenog na kraju mreže, kojim se mijenjaju iznosi težina pojedinog sloja, veoma smanjuje na ranijim slojevima mreže. To znači da raniji slojevi gotovo zanemarivo uče što se naziva nestajanje gradijenta [40].

Pored same kompletne strukture *ResNet50* modela neuronske mreže, važno je istaknuti primjenu tog modela isto kao i bazu podataka koja će biti korištena kako bi se kreirao jedan takav kompleksan i pouzdan model. Baza se podataka pomoću koje će se ovaj model trenirati, naziva *ImageNet*. *ImageNet* je baza slika organiziranih prema *WordNet* hijerarhiji, trenutno samo imenice, gdje je svaki čvor hijerarhije opisan stotinama i tisućama slika [41]. *WordNet*

je pak, golema baza engleskih riječi, koje su opisane na način da sve imenice prije ili kasnije „dođu“ do svog temeljnog čvora. Dakle, sve su imenice zapravo instanca ili sinonim neke druge krovne imenice. Te su imenice, odnosno čvorovi u bazi *ImageNet* opisani slikama sa pripadajućim oznakama što one predstavljaju. Ta se baza sastoji od preko 14 milijuna slika te više od 21.000 setova sinonima [41]. Za potrebe ovoga programa, koristit će se dio ove baze koji se sastoji od 1000 različitih objektnih klasa koristeći 1,2 milijuna slika za treniranje, 50.000 slika za validaciju i 100.000 slika za testiranje modela [42].

Kada je konačno u potpunosti definiran model neuronske mreže koji će se koristiti za prepoznavanje lica pojedine osobe, potrebno ga je prilagoditi potrebama ovog programa. To je najlakše ostvariti „zamrzavanjem“ dijela težina u modelu, dodavanjem izlaznih slojeva koji će moći riješiti konkretan problem prepoznavanja lica i treniranjem modela pomoću ulaznih podataka opisanih u poglavlju 8.1. Dakle, najprije je potrebno vrijednosti težina „zamrznuti“ što označava da se njihov iznos ne može mijenjati tijekom procesa treniranja mreže. Ne želimo mijenjati sve težine u već istreniranom modelu mreže, nego samo one kojima smo prilagodili mrežu našim potrebama. Učitavanjem modela u program te „zamrzavanjem“ težina, dodajemo dva potrebna sloja na kraju mreže. Predzadnji sloj u mreži bit će *Global Average Pooling* sloj kojim se slika zapisana u obliku tenzora drugog reda transformira u vektor na način da se svaka matrica RGB kanala boje zamijeni prosječnom vrijednosti te matrice za svaku sliku u skupu ulaznih podataka. Taj je proces vizualno prikazan na slici 26.



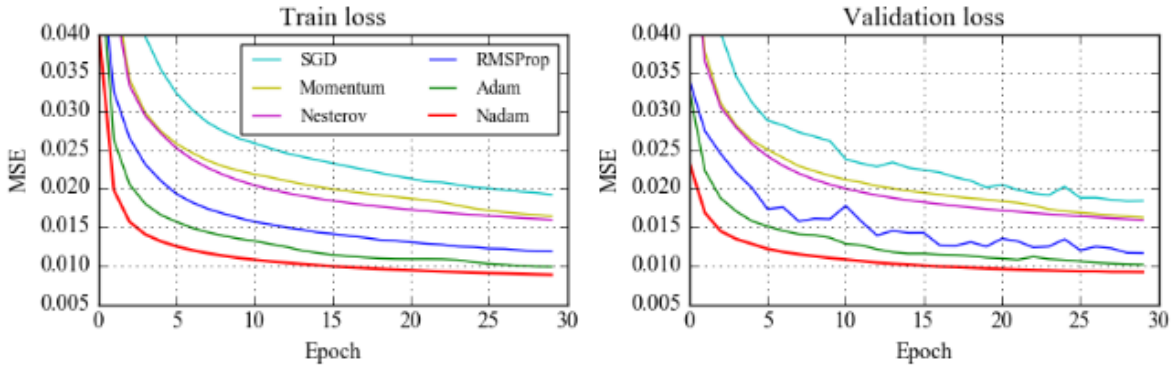
Slika 26. *Global Average Pooling* sloj [43]

Ovim se slojem povezuju slojevi konvolucijskog dijela neuronske mreže sa klasičnim slojevima koji barataju vektorima, kao što je opisano u poglavlju 5.2. Posljednji sloj u mreži

ovisit će o broju osoba koje želimo prepoznati na slici. Taj će sloj imati broj neurona jednak broju spomenutih osoba a njegova se struktura naziva potpuno povezani sloj mreže. Sami naziv označava da će svi neuroni posljednjeg sloja biti povezani sa svim neuronima prethodnog sloja. Kao aktivacijska funkcija na izlaznom sloju mreže, postavljena je sigmoidalna nelinearna funkcija, slična onoj prikazanoj na slici 11.

Dodatne modifikacije učitanog *ResNet50* modela odnose se na nekoliko stvari poput dodavanja automatskog spremanja modela svakih nekoliko koraka učenja, kako se proces ne bi morao ponavljati iz početka ukoliko se program iz nekog razloga naglo prekine ili nešto pođe po zlu, što se u TensorFlow biblioteci naziva *ModelCheckpoint* objektom. Potrebno je još postaviti način optimizacije mreže, tj. tehniku kojom će se mijenjati težine u procesu učenja. Za to će se koristiti optimizacijski algoritam *Adam*. *Adam* (*Adaptive Movement Estimation*) je optimizacijski algoritam koji se može koristiti umjesto klasičnog stohastičkog pada gradijenta pogreške za iterativnu promjenu težina mreže prilikom učenja [44]. Klasični algoritmi pada gradijenta pogreške računaju gradijent funkcije cilja s obzirom na vrijednosti ulaznih varijabli te mijenjaju težine na način da se pogreška mreže smanjuje u suprotnom smjeru od izračunatog gradijenta. Na taj se način ostvaruje najbrži mogući pad pogreške, a samim time se i najbrže dolazi do željene točnosti. Parametar η, odnosno stopa učenja, označava koliko će se mijenjati ulazne varijable u odnosu na izračunati gradijent. Prilagođena aproksimacija pada gradijenta pogreške, u kojoj se u svakom koraku učenja računa gradijent u jednoj nasumičnoj točki, a ne u cijelom skupu podataka, naziva se stohastički gradijent pada pogreške. Pojam stohastički, označava da se radi o veličini čiji iznos ovisi o vjerojatnosti distribucije skupa u kojem se računa gradijent.

Adam algoritam nadogradnja je stohastičkog gradijenta pada pogreške. U poglavlju 5.2.2. ukratko je spomenut pojam momentuma te njegova uloga u procesu optimiranja učenja mreže. Momentum se koristi kako bi se smanjio potrebni broj koraka prilikom treniranja mreže pri čemu se vrijednost promjene parametra učenja iz prethodnog koraka množi sa koeficijentom momentuma α. *Adam* algoritam koristi vrijednosti prvog i drugog momentuma te pomoću njih prilagođava promjenjivu stopu učenja η za svaku težinu u neuronskoj mreži. Rezultati dobiveni korištenjem ovoga optimizacijskog algoritma pokazuju njegovu vrlo dobru točnost i brzu konvergenciju ka globalnom maksimumu funkcije cilja. Usporedba *Adam* algoritma za optimizaciju i nekih drugih poznatih algoritama prikazana je na slici 27.



Slika 27. Usporedba različitih optimizacijskih algoritama [46]

Jedna od najčešćih funkcija cilja, odnosno funkcija prema kojoj se računa iznos pogreške na izlazu iz mreže, je već spomenuta suma najmanjih kvadrata definirana u poglavlju 5.2.2. izrazom (10) korištena za probleme regresije. Kod problema klasifikacije, kao što je to slučaju u programu ovoga rada, češće se koristi *Cross-Entropy* kao funkcija cilja koju je potrebno minimizirati. *Cross-Entropy* je mjera razlike dvije distribucije vjerojatnosti za dane varijable ili događaje [45]. Ona je definirana izrazom:

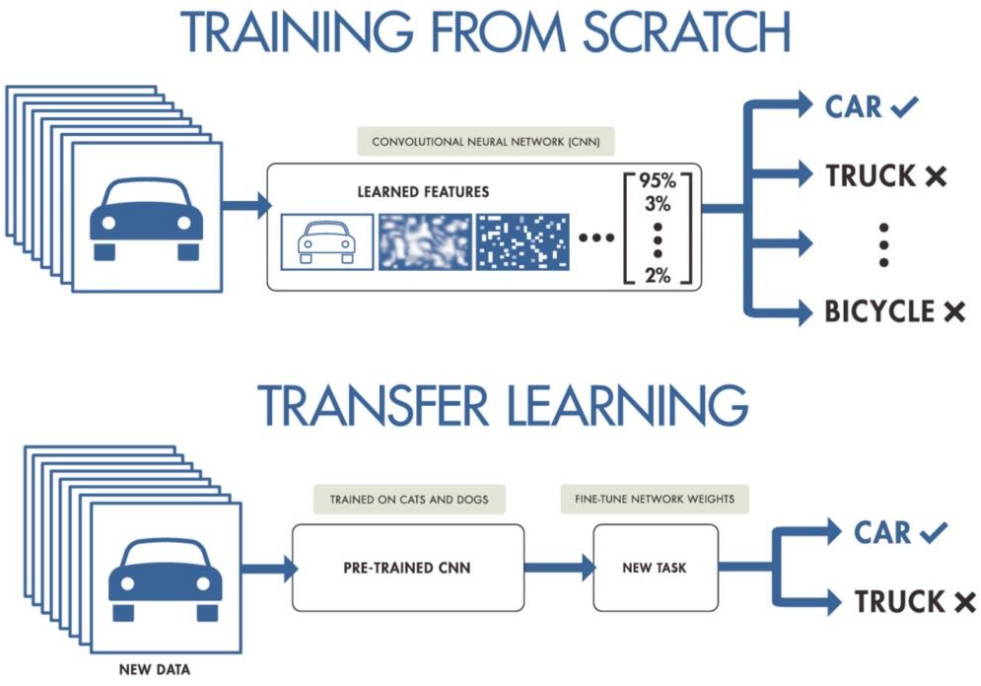
$$H(p,q) = -\sum_{x \in X} p(x) \log q(x), \tag{29}$$

pri čemu je $H()$ iznos funkcije cilja, $p(x)$ vjerojatnost događaja x u distribuciji p, a $q(x)$ vjerojatnost događaja x u distribuciji q. Pomoću ovoga se izraza zapravo računa prosječan broj bitova koji je potreban za identifikaciju događaja x iz skupa X za slučaj da se taj skup optimizira distribucijom q, umjesto prave distribucije p. Za minimizaciju ove funkcije koristit će se spomenuti *Adam* algoritam sa stopom učenja koja iznosi 0,01.

Kao mjera koja će biti pokazatelj koliko brzo spomenuta *ResNet50* neuronska mreža uči, koristit će se iznos pogreške prilikom validacije. Skup podataka za validaciju iznosi 20% sveukupne baze podataka kojima se trenira mreža. Dakle, 20% slika svake osobe prilikom treniranja mreže služit će kao validacija mreže i njezinoga rada. Iznos pogreške prilikom validacije trebao bi padati kako mreža napreduje kroz proces učenja. Za ovaj je program definiran prestanak faze učenja kada se iznos pogreške pri validaciji nije značajno promijenio 3 epohe za redom, što bi označavalo da iznos te pogreške vrlo slabo pada te više nije potrebno trenirati mrežu. U TensorFlow biblioteci ta je naredba označena *EarlyStopping* integriranom funkcijom čiji su parametri definirani na način da predstavljaju maloprije opisan način

prestanka treniranja neuronske mreže. Sada je *ResNet50* model mreže u potpunosti prilagođen potrebama i načinu rada ovoga programa.

Ovakav način korištenja već kreiranog i provjerenog modela neuronske mreže te njegovu adaptaciju vlastitim potrebama kako bi se riješio neki specifični problem, često se naziva proces prenesenog učenja (*engl. Transfer Learning*). Većina je današnjih sustava temeljena na ovome principu gdje nije potrebno kreirati cijeli sustav od nule, već se možemo poslužiti nekom javno dostupnom sustavu za kojeg smo sigurni da je sposoban riješiti naš problem. To nije slučaj samo u području umjetne inteligencije već i u ostalim granama tehnologije u kojima se na neku osnovnu bazu dodaju i modificiraju potrebni parametri pomoću kojih se dobiva neka nova, prilagođena struktura. Takav je proces puno lakše izvesti nego pokušavati napraviti neki novi sustav i optimizirati sve njegove parametre. Jednostavna vizualizacija procesa prenesenog učenja u području prepoznavanja objekata na slici, što je i slučaj u ovome programu, prikazana je na slici 28.



Slika 28. Proces prenesenog učenja [47]

Sa slike 28. vrlo se jasno vidi princip korištenja prenesenog učenja kao bolje i učinkovitije metode uporabe neuronskih mreža u usporedbi sa potpunim kreiranjem i treniranjem vlastitog sustava. Kao što je već spomenuto u ranijem dijelu ovog poglavlja, preneseno se učenje, za ovaj program, ostvaruje „zamrzavanjem“ dijela težina postojeće strukture na koju se dodaju

odgovarajući slojevi prilagođeni za prepoznavanje osoba na temelju njihovih lica. Tako se puno kraće i jednostavnije mogu mijenjati samo težine tih dodanih slojeva što ubrzava cijeli proces učenja mreže. Najbolji je primjer toga, broj parametara koji tvore *ResNet50* model neuronske mreže. Ukupan broj parametara koje moguće naučiti i mijenjati u cijeloj strukturi mreže iznosi 23.591.810, od kojih je primjenom prenesenog učenja potrebno trenirati samo 4.098. To je jasan pokazatelj koliko bi zapravo proces treniranja ove mreže iz početka trajao te koliko bi to bilo zahtjevno za obično računalo.

Sada možemo još nešto reći o ostalim parametrima koji će se koristi pri treniranju ovog, prilagođenog *ResNet50* modela mreže. Najprije je definiran način kako mreža prolazi kroz ulazni skup podataka. U poglavlju 5.2.2. opisan je jedan od načina na koji mreža mijenja svoje parametre učenja, odnosno težine. U tamo opisanim jednadžbama, neki su izrazi ovisili o pojedinom koraku učenja, tj. iznosi težina u slijedećem koraku računali su se na temelju iznosa težina u trenutnom i prošlom koraku učenja. Ti koraci predstavljaju promjenu težina u mreži, a dešavaju se ovisno o tome nakon koliko ulaznih podataka prošlih kroz mrežu, želimo da se ti iznosi težina promjene. Dakle, moguće je mijenjati težine nakon svakog ulaznog podatka ili nakon svakih nekoliko podataka, ovisno o zadanome problemu. U ovome je programu zadano da se nakon svakih 16 slika koje prođu kroz mrežu, promijene težine u zadnja dva sloja. Takav se način treniranja obično naziva treniranje mini-setom podataka (*engl. mini-batch*), a hiperparametar kojim se definira broj podataka u mini-setu naziva se *batch size*. Standardne su vrijednosti tog hiperparametra 16, 32, 64 ili 128, a njegova je vrijednost u ovome je programu, kao što je već spomenuto, jednaka 16.

Pored količine ulaznih podataka nakon kojih će se mijenjati iznosi težina potrebno je i odrediti koliko puta cijeli skup ulaznih podataka mora proći kroz mrežu kako bi se postigla željena točnost mreže. Taj se hiperparametar naziva broj epoha. Epoha, dakle, predstavlja jedan prolazak kompletnog skupa podataka kroz mrežu. Obično je potrebno nekoliko epoha kako bi mreža ispunila zahtjeve točnosti, ovisno o tome koji se algoritmi optimizacije koriste. Nekakve su standardne vrijednosti ovog hiperparametra 10, 50, 100, 500 ili 1000 i više. U ovome programu zadani broj epoha iznosi 50. U fazi učenja, često neće biti potrebno proći kroz svih 50 epoha učenja jer će vrijednosti pogreške pri validaciji relativno brzo konvergirati u minimum te će se proces učenja prekinuti. Ukoliko sustav ne daje dovoljno dobre rezultate, uvijek je moguće povećati traženu točnost, odnosno smanjiti dopušteni iznos pogreške.

Definiranjem spomenutih hiperparametara, konačno je moguće započeti proces treniranja mreže. Trajanje ovoga procesa, naravno ovisi o broju osoba koje će trebati prepoznati i razlikovati. Teoretski bi sustav mogao prepoznati velik broj osoba, no u praktičnoj primjeni povećanjem broja osoba povećava se kompleksnost sustava te broj parametara koje je potrebno naučiti mrežu, a samim time i mogućnost da se ne zadovolji željena točnost kroz svih 50 epoha faze treniranja mreže. Stoga je, u ovome radu, broj osoba koje će se prepoznavati ograničen na tri. Po završetku faze učenja, model mreže sprema se u obliku h5 datoteke. H5 je vrsta datoteka spremljenih u hijerarhijskom formatu koje sadrže višedimenzionalne tenzore spremljenih podataka. Tako su zapisani iznosi svih težina u cijeloj strukturi korištenog modela mreže. Slično se zapisuju podaci u područjima astronomije, financija, fizike, elektrotehnike i sl.

8.3. Prepoznavanje lica u realnom vremenu

U ovom će poglavlju biti opisan proces detekcije i prepoznavanja lica u realnom vremenu pomoću web kamere ili nekog drugog izvora. Temeljna biblioteka potrebna za gotovo sve operacije na slikama ili sličicama videa u Pythonu, naziva se OpenCV. OpenCV (*Open Source Computer Vision Library*) je softverska biblioteka otvorenoga koda za strojno učenje i računalni vid [48]. Čitava biblioteka sastoji se od preko 2500 optimiziranih algoritama u područjima računalnoga vida i strojnog učenja koji se mogu koristiti u raznim primjenama poput praćenja objekata koji se kreću, prepoznavanja i detekcije lica, detekcije pojedinih objekata na slici, kreiranja 3D oblaka točaka pomoću stereo kamera i sl. Otvoreni pristup koda omogućuje vrlo laku prilagodbu programa korisnikovim potrebama, što je dovelo do velike komercijalne i industrijske primjene ove biblioteke. Slično kao i TensorFlow, OpenCV je vrlo jednostavno integrirati u programski jezik Python, čime se dobiva pristup svim navedenim algoritmima koji se mogu iskoristiti ili prilagoditi vlastitim potrebama.

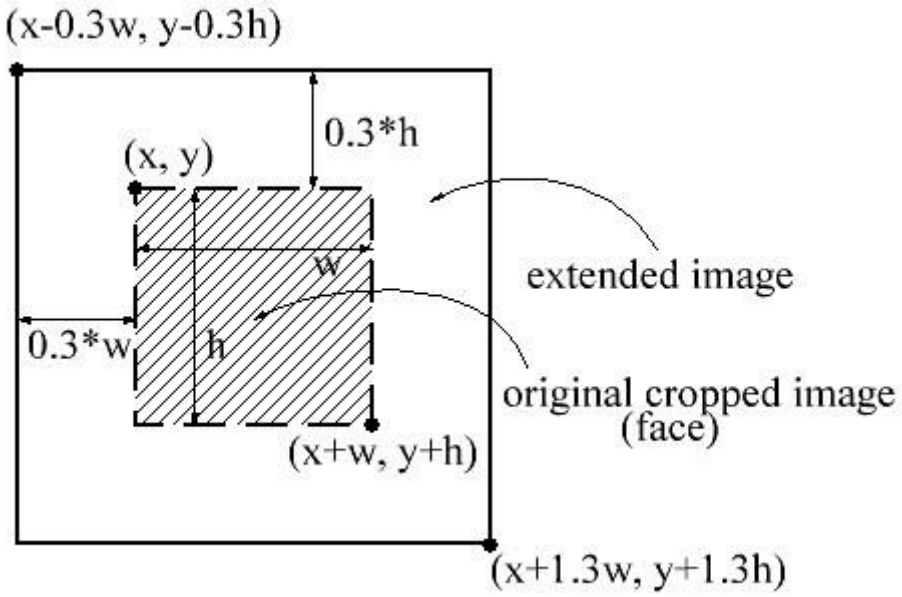
Pored spomenutog *ResNet50* modela neuronske mreže koji smo koristili kao alat koji će moći prepoznati lica različitih osoba na temelju dane slike na ulazu, korištenjem OpenCV biblioteke dobivamo mogućnost uporabe već definiranog sustava detekcije lica. U ovoj se biblioteci taj algoritam naziva *Cascade Classifier*, a služi kao algoritam za detekciju objekata na temelju značajki koje se nalaze na slici. Ovaj se algoritam temelji na tzv. *Haar* značajkama koje predstavljaju sume pojedinih piksela u određenom području, a rade na principu sličnom

kao kreiranje mape značajki kod konvolucijskih slojeva neuronske mreže. Takav način detekcije značajki generira i do nekoliko stotina tisuća značajki na pojedinoj slici, ovisno o njenim dimenzijama. Za sustav detekcije lica, sve je takve značajke potrebno računati na svim slikama u fazi treniranja ovakvoga algoritma. Zatim se odabiru značajke koje su davale najmanje iznose pogreške, odnosno koje su relativno precizno mogle klasificirati nalazi li se na slici zapravo lice ili ne. Tako odabranih, u ovom, konkretnom primjeru, gotovo 6000 značajki tvori algoritam za klasifikaciju lica. Te se značajke računaju na svakom dijelu dane slike te ukoliko zadovolje određeni prag, zaključuje se kako se na tom dijelu slike nalazi lice. No, ovdje je problem preveliki broj računskih operacija koje treba izvršiti, pa se stoga uvodi pojam kaskadnih klasifikacija. Umjesto da se sve značajke računaju na svakom dijelu slike, jednostavnije je prvo odrediti potencijalna mjesta gdje bi se lice moglo nalaziti pa tek onda računati nalazi li se ono tamo ili ne. To se ostvaruje tako da se značajke poslože kao u slojeve te se računaju sloj po sloj na svakom dijelu slike. Ukoliko neki raniji sloj ne zadovolji određeni prag, odmah se odbacuje mogućnost da se na tom dijelu slike nalazi lice i prelazi se na slijedeći dio slike. Na taj je način broj matematičkih operacija mnogo manji. Spomenuti algoritam za klasifikaciju lica koji se sastoji od 6000 značajki, podijeljen je u 38 slojeva, kaskada, gdje prvih 5 slojeva sadrži 1, 10, 25, 25 i 50 značajki [49].

Izlaz koji će, gore spomenuti, algoritam dati jednom kada ustvrdi da se na slici nalazi lice, je zapravo pravokutnik kojim će to lice biti omeđeno. Preciznije, algoritam će na izlazu davati 4 parametra koji će jednoznačno određivati mjesto na kojem se taj pravokutnik nalazi te njegove dimenzije. Ti su parametri redom, x koordinata gornjeg lijevog kuta pravokutnika, y koordinata gornjeg lijevog kuta pravokutnika, širina pravokutnika te visina pravokutnika. Na temelju ovih parametara, kasnije ćemo moći napraviti odgovarajući pravokutnik oko osobe čije je lice prepoznato te ga prikazati na krajnjem videu.

Prije nego se spremljeni model koji prepoznaje različita lica može ubaciti u algoritam koji ta lica detektira, potrebno je malo modificirati slike koje će naš model primati. Kako je navedeno, algoritam detekcije lica daje nam pravokutnik koji omeđuje detektirano lice. Unutar toga pravokutnika nalazi se isključivo detektirano lice bez ikakve okoline, što se razlikuje od slika kojima smo trenirali model mreže kako bi mogao razlikovati pojedina lica. Stoga je potrebno proširiti sliku lica kako bi model mogao dobro prepoznavati o kojoj se osobi zapravo radi. To se može ostvariti kreiranjem funkcije u Python-u koja će uzimati

parametre pravokutnika koji omeđuje lice, cijelu izvornu sliku web kamere te parametar *k* koji će predstavljati faktor koliko želimo proširiti sliku. Prikaz ove operacije proširivanja slike lica ovisno o zadanom faktoru *k* nalazi se na slici 29.



Slika 29. Proširivanje slike radi boljeg prepoznavanja lica [50]

Na slici 29. nalazi se prikaz kako će slika lica biti proširena u svim dimenzijama kako bi bolje odgovarala slikama kojima smo trenirali neuronsku mrežu za prepoznavanje lica. Konkretna vrijednost faktora *k* na slici 29. iznosi 0,3 što odgovara proširenju slike za 60% u smjeru širine i 60% u smjeru visine slike. Koeficijent *k* trebalo bi prilagoditi ovisno o ulaznim podacima sa kojima se trenirala modificirana *ResNet50* neuronska mreža.

Ovaj se postupak modificiranja slike, mogao i izbjeći drugačijem pristupu uzimanja podataka od onoga opisanog u poglavlju 8.1. Umjesto spremanja cijelih slika pomoću web kamere svakih nekoliko sličica u sekundi kada je lice na slici detektirano, moguće je spremiti samo dio slike koji će biti omeđen pravokutnikom kojeg je algoritam za detekciju lica dao na svom izlazu. Tako bi se mogle spremiti samo slike lica, a ne cijele slike web kamere čime bi se izbjegla potreba za proširivanjem slika prilikom detekcije i prepoznavanja u realnom vremenu. No, tu bi se mogao javiti problem umjetnog povećanja ulaznog skupa podataka, koji bi mogao previše promijeniti ili deformirati sliku lica zbog nedostatka margina na slici, tj. nedostatka slobodnog prostora gdje bi lice moglo biti pomaknuto ili zaokrenuto. Stoga se ovaj

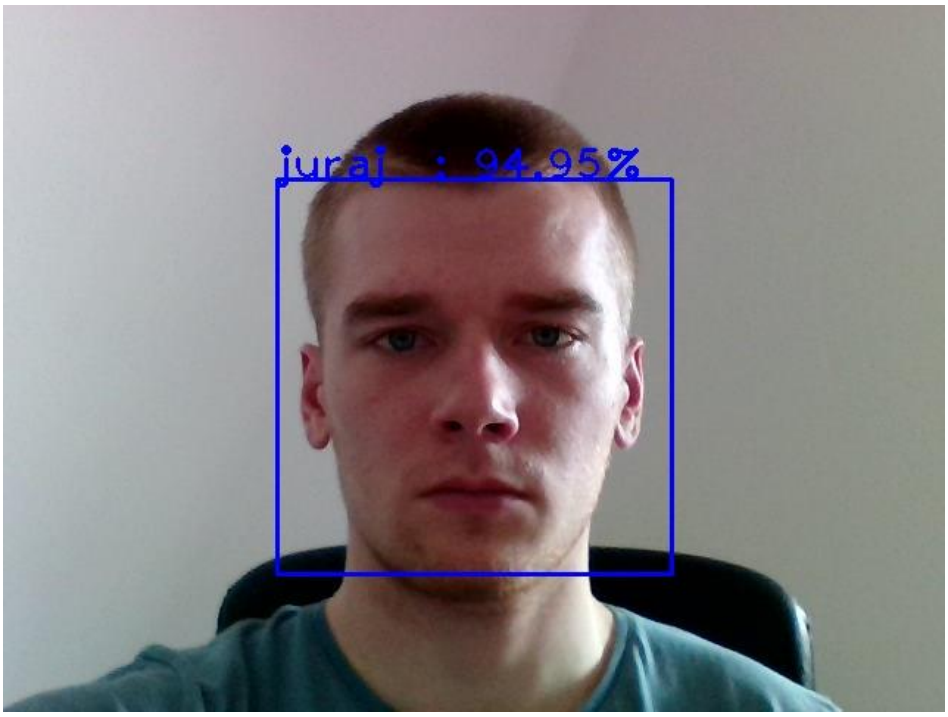
pristup uzimanju slika osoba nije koristio u ovome radu, iako je moguće i tako kreirati dobar ulazni skup podataka.

Sada se konačno može implementirati neuronska mreža za prepoznavanje lica u algoritam za detekciju lica u realnom vremenu. Radi bržeg rada sustava koji treba obrađivati 30 slika u sekundi, ulazne slike prebačene su iz RGB zapisa u *Greyscale* format, odnosno iz slika zapisanih sa kanalima crvene, zelene i plave boje u slike koje sadrže samo nijanse sive boje. Dimenzije su slika također relativno male te iznose 250 × 250 piksela kako bi se ubrzao cijeli proces te ne bi došlo do kašnjenja programa.

Preostalo je još prikazati izlaz koji model za prepoznavanje lica daje. Nakon što se lice sa web kamere prepoznalo te se ta sličica videa provukla kroz neuronsku mrežu, dobivamo dva podatka o krajnjem rezultatu. Prvi je podatak vektor vrijednosti izlaznog sloja neurona u mreži koji zapravo predstavlja mjeru koliko je mreža „sigurna“ da detektirano lice pripada kojoj osobi. Pošto je ta vrijednost zapisana kao neki broj od 0 do 1, možemo ga protumačiti koliko posto to detektirano lice sliči svakom licu osoba na kojima je mreža trenirana. Drugi je podatak, ime osobe na čijem se izlaznom neuronu ostvario najveći broj. Odnosno, ime osobe, ili neka druga njena oznaka, za koju je neuronska mreža dala najveću vjerojatnost da sliči na detektiranu osobu na slici. Na temelju ta dva podatka, možemo prikazati rezultat prepoznavanja na način da prvo prikažemo dobiveni pravokutnik koji omeđuje lice dobiven detekcijom lica te oko njega napišemo ime osobe čije je lice u tom pravokutniku prepoznato, kao i postotak sigurnosti mreže da to lice zaista pripada toj osobi.

Ovdje je još postavljena vrijednost praga sigurnosti koja mora biti zadovoljena kako bi zaista bili sigurni da je program prepoznao pravu osobu. Ukoliko taj prag nije zadovoljen a lice je detektirano na slici, prikazat će se samo crveni pravokutnik bez imena osobe i postotka sigurnosti mreže. To je korisno u slučajevima kada je mreža „nesigurna“ ili se na slici pojavi osoba čije slike lica nisu bile u ulaznom skupu podataka za treniranje. Tada će mreža vjerojatno davati vrijednosti koje ne želimo prikazati kao ispravnima. Primjerice, ukoliko koristimo 5 različitih osoba koje bi program trebao detektirati i razlikovati, pojavom neke nove osobe, koja će otprilike jednako pobuditi neuronsku mrežu na svim izlaznim neuronima, tj. sigurnost mreže za svaku osobu biti će približno jednaka te iznositi oko 20%, ne želimo takav rezultat smatrati dobrim prepoznavanjem ijedne od 5 osoba, što bi bio slučaj kada bi

jednostavno uzimali maksimalnu vrijednost izlaznih neurona. Samu vrijednost praga potrebno je prilagoditi primjeni za koju se sustav prepoznavanja lica koristi. Sustavi opisani u poglavlju 6.3. sigurno koriste vrlo visoke pragove kako bi bili zaista sigurni da se radi o ispravnoj osobi na slici. Za potrebe ovoga programa, taj je prag postavljen na vrijednost 0,8 što odgovara sigurnosti mreže od 80% da se na slici nalazi ispravna osoba. Primjer krajnjeg izlaza gotovog programa nalazi se na slici 30.

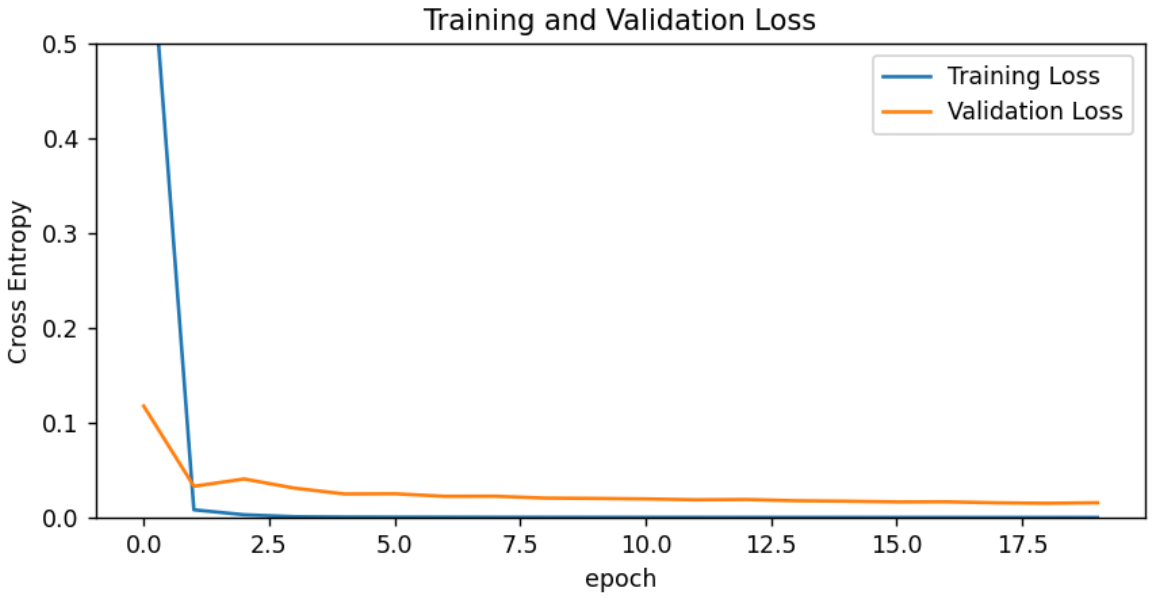


Slika 30. Primjer krajnjeg izlaza programa

9. TESTIRANJE PROGRAMA I REZULTATI

Nakon potpunog opisa cijelog programa te načina na koji program radi, možemo testirati kolika je njegova točnost u realnoj primjeni. U zadatku ovoga rada, zadano je kako program treba biti testiran na realnim osobama te provjeriti njegovu učinkovitost ovisno o kutu gledanja, odnosno o kutovima za koje je lice okrenuto u odnosu na kameru. Na kraju poglavlja 8.2., spomenuto je kako će broj osoba na kojima će se testirati navedena učinkovitost iznositi 3, zbog razloga navedenih u tom poglavlju.

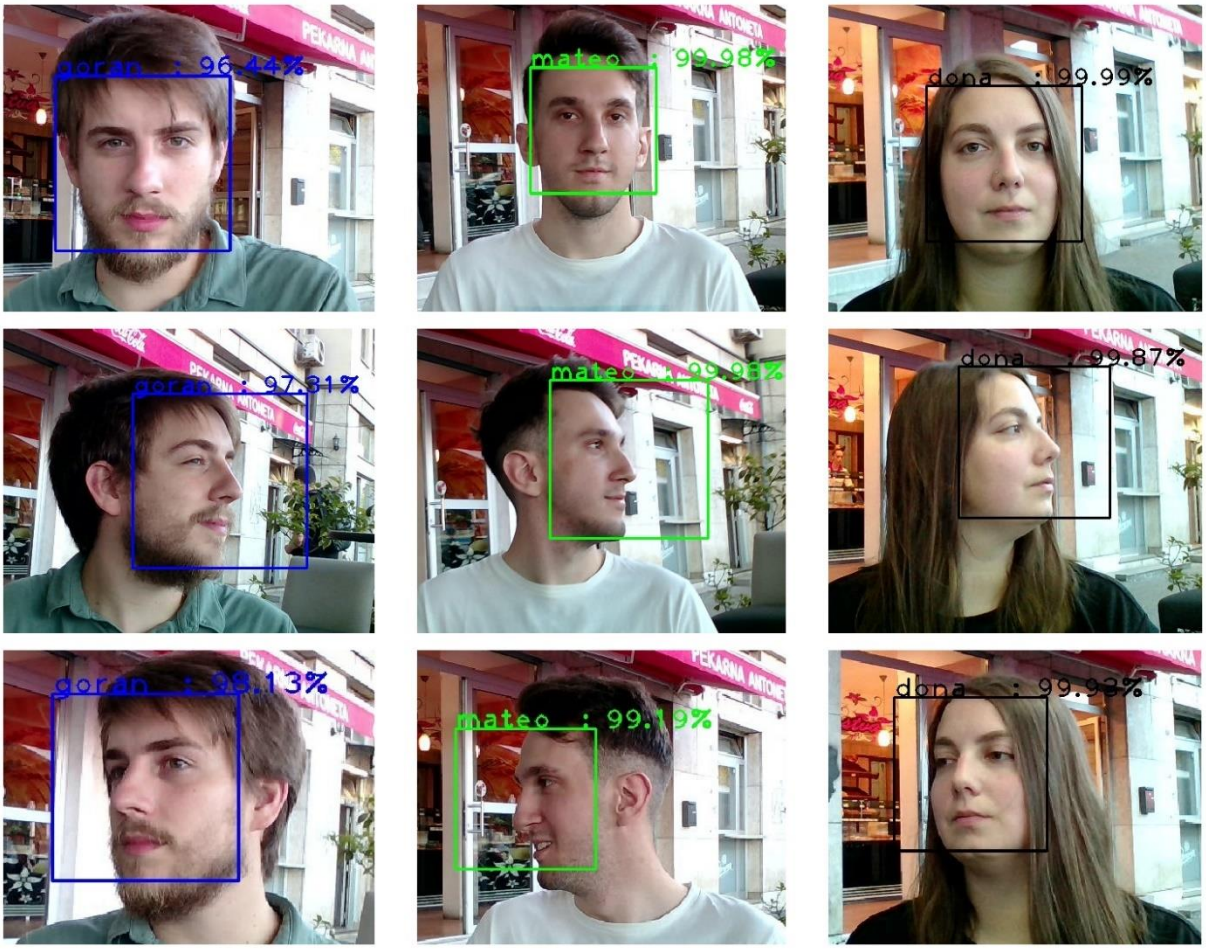
Za početak možemo proučiti koliko program dobro prepoznaje osobe ako im je lice zakrenuto u odnosu na web kameru koja ih snima. Osobe na kojima je testiran sustav zovu se redom Goran, Mateo i Dona. Njihove slike za učenje neuronske mreže dobivene su tako da se web kamerom uslikalo 50 slika za treniranje i 15 za testiranje mreže. Slike za treniranje i testiranje mreže zatim su umjetno modificirane na način opisan u poglavlju 8.1. čime se dobiva konačan broj slika po osobi koji iznosi 200 slika za treniranje i 45 slika za testiranje mreže. Proces treniranja mreže, konkretno za ove tri osobe, odnosno prikaz smanjenja iznosa pogreške prilikom treniranja i validacije mreže za zadanu *Cross-Entropy* funkciju cilja, grafički je prikazan na slici 31.

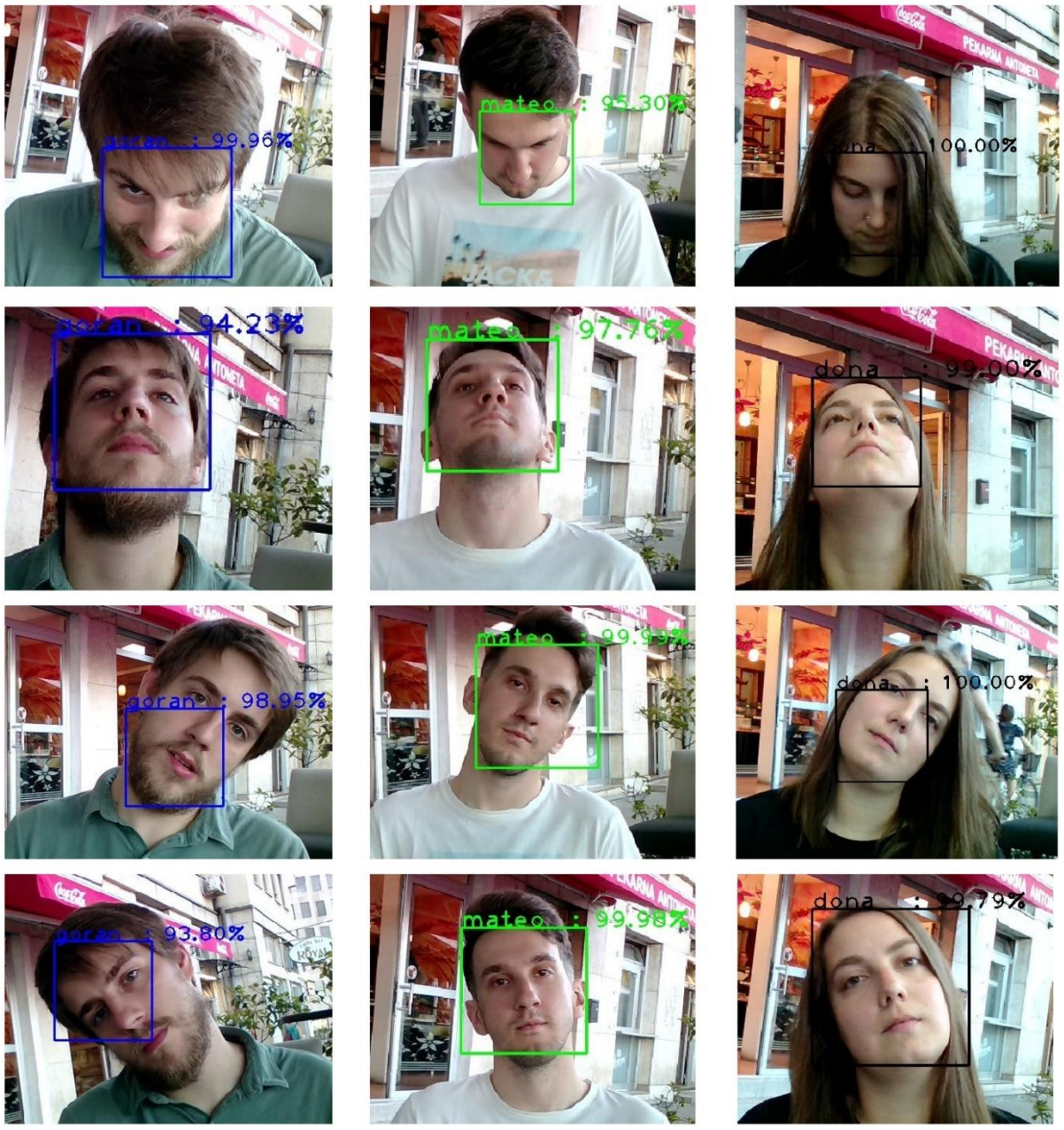


Slika 31. Treniranje neuronske mreže

Sa slike 31. vidljivo je kako se mjera kojom se prati treniranje mreže, pogreška pri validaciji, u prvih nekoliko epoha dosta smanjuje, dok pogreška skupa za treniranje još brže pada do iznosa manjeg od 10^{-3}. Takvo je ponašanje mreže očekivano, odnosno može se zaključiti kako bi mreža mogla davati dobre rezultate. Proces je završio nakon 20 epoha, a krajnji iznosi pogrešaka su 0,0157 prilikom validacije i 0,0002122 prilikom treniranja.

Nakon treniranja, možemo provjeriti sigurnost mreže u ovisnosti o kutu za koji je lice okrenuto. Kao i sva trodimenzionalna tijela u prostoru, tako je i ljudsko lice određeno sa tri prostorna kuta koje može zatvarati u odnosu na kameru. Ti kutovi u ovom slučaju predstavljaju zakretanje glave u lijevu i desnu stranu, pomak glave prema gore i prema dolje te naginjanje glave u lijevu i desnu stranu. Na slici 32. prikazana su lica navedenih osoba, u svim krajnjim kutovima u kojima ih je program mogao prepoznati kao i u početnom, neutralnom položaju kada lice nije pod nekim kutom u odnosu na kameru.





Slika 32. Krajnji položaji lica pri kojima sustav može detektirati lice

Slika 32. prikazuje sve zakrete lica navedenih osoba u graničnim položajima gdje je program još uvijek mogao detektirati lice na slici. Radi malih dimenzija slika kako bi se mogao prikazati usporedni položaj svih osoba, podaci o sigurnosti mreže koji se možda ne vide dobro, prikazani su i u tablici 1.

Tablica 1. Sigurnost mreže pri različitim zakretima lica

	Goran	Mateo	Dona
	Sigurnost mreže [%]		
Normalan pogled, bez kuta zakreta	96,44	99,95	99,99
Zakret glave u lijevo	97,31	99,98	99,87
Zakret glave u desno	98,13	99,19	99,93
Zakret glave prema dolje	99,96	95,30	100,00
Zakret glave prema gore	94,23	97,76	99,00
Naginjanje glave u lijevu stranu	98,95	99,99	100,00
Naginjanje glave u desnu stranu	93,80	99,98	99,79

Kao što je vidljivo sa slike 32. i iz tablice 1., vrijednosti sigurnosti mreže vrlo su visoke za skoro svaki pogled. Važno je istaknuti kako ti podaci predstavljaju rezultate mreže tek kada program detektira da se na slici uopće nalazi lice. Dakle, ukoliko bi osoba više zakrenula glavu nego što je prikazano, program u većini slučajeva ne bi prepoznao da je na slici prisutno lice, pa samim time ni mreža ne bi mogla predvidjeti kojoj osobi to lice odgovara. Na temelju toga, možemo zaključiti kako je program zapravo limitiran sustavom detektiranja lica, a ne sustavom pridruživanja lica određenoj osobi. Također je vidljivo, kako je sustav, od svih osoba, mogao najbolje prepoznavati Donu, pri čemu nijedna vrijednost sigurnosti mreže nije pala ispod 99%.

No, važno je ne zavarati se ovako dobrim rezultatima, jer ako pomnije proučimo sliku 32., možemo vidjeti kako kutovi za koje su sva lica zakrenuto nisu baš veliki. Najbolji je primjer toga, naginjanje glave u desnu stranu, gdje su granični slučajevi dok sustav još uvijek može detektirati lice vrlo blizu položaja gdje se lice ne nalazi pod nekim kutom. Nekakvom brzinskom procjenom, vidljivo je da ovaj program zapravo ne pokriva dosta velik raspon položaja u koje ljudi mogu dovesti svoju glavu, ali jednom kada se lice na slici detektira, sustav će ga vrlo dobro klasificirati. Naravno, uz povećanje broja osoba, vjerojatno će i padati ta preciznost sustava, no i ovako, sa samo 3 osobe, program zahtijeva vrlo moćno računalo kako bi mogao obraditi svaku sličicu videa tijekom testiranja, što je zapravo najutjecajniji

faktor koji limitira testiranje na nekom većem skupu ljudi. Također, na slici 32. prikazani su samo krajnji položaji kada se lice pojedine osobe zakrenulo samo za jedan kut u odnosu na kameru, što je rijetkost u realnoj situaciji gdje je čovjek često orijentiran pod nekim većim iznosima sva tri kuta. Sustavi koji mogu prepoznavati ljudska lica u takvim situacijama te još pri tomu detektirati lica na većim udaljenostima puno su složenija od ovoga programa, pa se i koriste u puno ozbiljnije svrhe.

Do sada se razmatralo prepoznavanje lica samo jedne osobe, pa je također potrebno spomenuti slučaj kada se na slici nalaze dva ili više lica. Algoritam iz OpenCV biblioteke, spomenut na početku poglavlja 8.3., koji se koristi za detekciju lica na slici, posjeduje sposobnost detekcije većeg broja ljudi na svakoj pojedinoj slici. Kao što je već rečeno, algoritam analizira svaki dio slike te pomoću izračunatih značajki određuje nalazi li se na tom dijelu slike lice ili ne. Veći će broj osoba na slici samo značiti da će izlaz na kraju algoritma biti matrica u kojoj će svaki redak sadržavati parametre o poziciji tog lica na slici, tj. informacije o pravokutniku koji omeđuje to lice, a ne jedan vektor redak kao što bi to bio slučaj ukoliko se samo jedna osoba nalazi na slici. Stoga, veći broj lica na slici zasigurno neće predstavljati problem ovome algoritmu. Vremenski će proces svakako biti zahtjevniji, ali povećanje broja osoba na slici neće imati značajnog utjecaja na preciznost i točnost krajnjeg rezultata. Ostatak je programa isto tako neovisan o broju osoba, pošto svako detektirano lice prolazi kroz neuronsku mrežu koja ga pokušava klasificirati, odnosno dodijeliti odgovarajućoj osobi. Dakle, u svakoj će sličici videa, za vrijeme rada programa u realnom vremenu, broj ulaznih podataka u neuronsku mrežu zapravo biti jednak broju osoba čije je lice bilo detektirano pomoću algoritma za detekciju lica. Sama slika koja ulazi u neuronsku mrežu predstavlja samo dio cijele sličice videa na kojem se nalazi veći broj lica, proširen za neki iznos piksela ovisno o faktoru k, kako je to prikazano na slici 29. Prema tome, ukoliko se na slici nalazi veći broj osoba, neće biti razlike u tome koliko dobro program prepoznaje pojedinu osobu, odnosno koliko je neuronska mreža sigurna da pojedino lice pripada upravo toj osobi, već će jedina razlika biti u tome što će sustav imati još manje vremena za analizu pojedine slike. Dodatno će se vrijeme potrebno za obradu, u ovome slučaju, javiti prilikom rada algoritma za detekciju lica, kao i prilikom prepoznavanja lica jer će se broj podataka koje neuronska mreža obrađuje povećati za najmanje dva puta. To povećanje može vrlo lako uzrokovati nemogućnost računala da uspije obraditi svaku sličicu na vrijeme, što potencijalno dovodi do nepoželjnog kašnjenja programa, a samim time i do izlaza koji neće izgledati kao

video sa detektiranim i prepoznatim licima nego više kao malo brža izmjena sličica. Iz tog je razloga potrebno još više računa voditi o sposobnostima računala koje pokreće ovaj program, odnosno o brzini procesora, kako ovaj problem ne bi postao glavni ograničavajući faktor prilikom uporabe jednog ovakvog programa.

10. ZAKLJUČAK

Na temelju prikazanih rezultata u prethodnom poglavlju, može se zaključiti kako je sustav dosta precizan jednom kada se na slici detektira prisutnost lica. Sustav koji se pokazao kao ograničavajući faktor, zapravo je sustav detekcije lica, koji se implementirao pomoću OpenCV biblioteke za računalni vid. Drugo najveće ograničenje je zahtijevana procesorska snaga samoga računala koje mora obrađivati 30 slika u sekundi kako bi sustav mogao pratiti lica različitih osoba u realnom vremenu. Te bi probleme mogli riješiti uporabom nekog drugog sustava za detekciju objekata na slici iz neke druge biblioteke koji bi možda davao bolje rezultate prilikom testiranja, te korištenjem boljeg procesora koji bi mogao obrađivati znatno veću količinu podataka u sekundi. No, za potrebe izrade jednog ovakvog rada, mislim da ovaj sustav dosta dobro funkcionira u cijelosti. Postignute vrijednosti kojima se opisuje koliko je sustav siguran da pojedino lice pripada nekoj osobi, dosta su visoke, iako sami ulazni skupovi podataka sadrže nešto više od 200 slika što u području konvolucijskih neuronskih mreža zapravo uopće nije velik broj. Proširivanjem sustava i finim podešavanjem pojedinih parametara neuronske mreže, mislim kako bi ovaj sustav zaista mogao dostići vrlo visoku razinu funkcionalnosti, čak i izvan istraživačkih područja, što pokazuje koliko je zapravo napredovala tehnologija da gotovo svatko koga zanima ovo područje umjetne inteligencije, može samostalno kreirati jedan ovakav sustav te ga pokušati prilagoditi svojim potrebama.

LITERATURA

[1] Britannica – artificial intelligence: https://www.britannica.com/technology/artificial-intelligence, Pristupljeno 07. lipnja 2022.

[2] R.U.R.: https://en.wikipedia.org/wiki/R.U.R., Pristupljeno 07. lipnja 2022.

[3] Umjetna inteligencija: https://en.wikipedia.org/wiki/Artificial_intelligence, Pristupljeno 07. lipnja 2022.

[4] Simbolička umjetna inteligencija: https://en.wikipedia.org/wiki/Symbolic_artificial_ intelligence, Pristupljeno 07. lipnja 2022.

[5] Ben Dickson, What is symbolic artificial intelligence, 18. studeni 2019. : https:// bdtechtalks.com/2019/11/18/what-is-symbolic-artificial-intelligence/, Pristupljeno 07. lipnja 2022.

[6] Konekcionizam: https://en.wikipedia.org/wiki/Connectionism, Pristupljeno 07. lipnja 2022.

[7] Kendra Cherry, How many neurons are in the brain?, 10. travanj 2020. : https://www. verywellmind.com/how-many-neurons-are-in-the-brain-2794889, Pristupljeno 08. lipnja 2022.

[8] Strojno učenje: https://en.wikipedia.org/wiki/Machine_learning, Pristupljeno 08. lipnja 2022.

[9] Metehan Kozan, Supervised and unsupervised learning (an intuitive approach), 01. rujan 2021. : https://medium.com/@metehankozan/supervised-and-unsupervised-learning-an-intuitive-approach-cd8f8f64b644, Pristupljeno 08. lipnja 2022.

[10] Unsupervised machine learning : https://www.javatpoint.com/unsupervised-machine-learning, Pristupljeno 08. lipnja 2022.

[11] Reinforcement learning: https://en.wikipedia.org/wiki/Reinforcement_learning, Pristupljeno 08. lipnja 2022.

[12] Computer vision, https://en.wikipedia.org/wiki/Computer_vision, Pristupljeno 08. lipnja, 2022.

[13] Ilija Mihajlović, Everything you ever wanted to know about computer vision, 25. travanj. 2019, https://towardsdatascience.com/everything-you-ever-wanted-to-know-about-computer-vision-heres-a-look-why-it-s-so-awesome-e8a58dfb641e, Pristupljeno 08. lipnja 2022.

[14] Glija: https://hr.wikipedia.org/wiki/Glija, Pristupljeno 08. lipnja 2022.

[15] Regina Bailey, Neuron anatomy, nerve impulses, and classifications, 10. srpanj 2019. : https://www.thoughtco.com/neurons-373486, Pristupljeno 08. lipnja 2022.

[16] Artificial neuron models: https://cnl.salk.edu/~schraudo/teach/NNcourse/ann-overview .html, Pristupljeno 08. lipnja 2022.

[17] Matthew Mayo, Neural network foundations, explained: activation function, 13. rujan 2017. : https://www.kdnuggets.com/2017/09/neural-network-foundations-explained-activation-function.html, Pristupljeno 08. lipnja 2022.

[18] Novaković, B., Majetić, D., Široki, M. : Umjetne neuronske mreže, Zagreb 2011.

[19] Zurada. J. M. : Artificial neural systems, W. P. Company, USA, 1992.

[20] Facial recognition system: https://en.wikipedia.org/wiki/Facial_recognition_system, Pristupljeno 11. lipnja 2022.

[21] Samuel Brice, A short history of facial recognition, 07. studeni 2020. : https://samdbrice.medium.com/4ecd290aaab1, Pristupljeno 11. lipnja 2022.

[22] Detection of static geometric facial features: https://ibug.doc.ic.ac.uk/research/ detection-static-geometric-facial-features/, Pristupljeno 11. lipnja 2022.

[23] MathWorks, Imgaussfit: https://www.mathworks.com/help/images/ref/imgaussfilt.html, Pristupljeno 14. lipnja 2022.

[24] Canny edge detector: https://en.wikipedia.org/wiki/Canny_edge_detector, Pristupljeno 14. lipnja 2022.

[25] Sofiane Sahir, Canny edge detection step by step in Python – Computer vision, 25. siječanj 2019. : https://towardsdatascience.com/canny-edge-detection-step-by-step-in-python-computer-vision-b49c3a2d8123, Pristupljeno 14. lipnja 2022.

[26] Ryan Daws, Intel examines wheter AI can recognise faces using themral imaging, 10. siječanj 2020. : https://www.artificialintelligence-news.com/2020/01/10/intel-examines-ai-recognise-faces-thermal-imaging/, Pristupljeno 14. lipnja 2022.

[27] What is Python? Executive summary: https://www.python.org/doc/essays/blurb/, pristupljeno 14. lipnja 2022.

[28] Tensorflow: https://www.tensorflow.org, Pristupljeno 14. lipnja 2022.

[29] Learn how TensorFlow solves real, everyday machine learning problems: https://www. tensorflow.org/about/case-studies, Pristupljeno 14. lipnja 2022.

[30] Leo Kelion, Gatwick Airport commits to facial recognition tech at boarding, 17. rujan 2019. : https://www.bbc.com/news/technology-49728301, Pristupljeno 19. lipnja 2022.

[31] Face ID: https://en.wikipedia.org/wiki/Face_ID, Pristupljeno 19. lipnja 2022.

[32] Apple's Face ID technology security possibilities: https://www.veprof.com/face-id-technology-security-possibilities.html, Pristupljeno 19. lipnja 2022.

[33] TikTok agrees legal payout over facial recognition: https://web.archive.org/web/20210226160803/https://www.bbc.com/news/technology-56210052, Pristupljeno 19. lipnja 2022.

[34] Graham Cluley, Facebook is developing creepy technology that can recognise faces almost as well as humans, 18. ožujak 2014. : https://grahamcluley.com/facebook-facial-recognition/, Pristupljeno 19. lipnja 2022.

[35] Understanding ResNet50 architecture: https://iq.opengenus.org/resnet50-architecture/, Pristupljeno 20. lipnja 2022.

[36] Ann H. Reynolds, Convolutional neural networks (CNNs), 2019. : https://anhreynolds.com/blogs/cnn.html, Pristupljeno 20. lipnja 2022.

[37] Sandeep Balachandran, Machine learning – Going Furthur with CNN part 2, 17. ožujak 2020. : https://dev.to/sandeepbalachandran/machine-learning-going-furthur-with-cnn-part-2-41km, Pristupljeno 20. lipnja 2022.

[38] Sumit Saha, A comprehensive guide to convolutional neural networks – the ELI5 way, 15. prosinac 2018. : https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53, Pristupljeno 20. lipnja 2022.

[39] Pooja Mahajan, Max Pooling, 05. srpanj 2020. : https://poojamahajan5131.medium.com/max-pooling-210fc94c4f11, Pristupljeno 20. lipnja 2022.

[40] Residual Network (ResNet): https://iq.opengenus.org/resnet/, Pristupljeno 20. lipnja 2022.

[41] ImageNet: https://www.image-net.org, Pristupljeno 20. lipnja 2022.

[42] Download ImageNet data: https://www.image-net.org/download.php, Pristupljeno 20. lipnja 2022.

[43] Global Average Pooling 2D: https://peltarion.com/knowledge-center/documentation/modeling-view/build-an-ai-model/blocks/global-average-pooling-2d, Pristupljeno 20. lipnja 2022.

[44] Jason Brownlee, Gentle Introduction to the Adam Optimization Algorithm for Deep Learning, 03. srpanj 2017. : https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/, Pristupljeno 21. lipnja 2022.

[45] Jason Brownlee, Gentle Introduction to Cross-Entropy for Machine Learning, 22. prosinac 2020. : https://machinelearningmastery.com/cross-entropy-for-machine-learning, Pristupljeno 21. lipnja 2022.

[46] Explain about Adam Optimization Function?, 09. rujan 2019. : https://www.i2tutorials .com/explain-about-adam-optimization-function/, Pristupljeno 21. lipnja 2022.

[47] Harley Davidson Regua, Introduction Transfer Learning as Your Next Engine to Drive Future Innovations, 27. veljače 2020. : https://medium.datadriveninvestor.com/introduc ing-transfer-learning-as-your-next-engine-to-drive-future-innovations-5e81a15bb567, Pristupljeno 21. lipnja 2022.

[48] OpenCV: https://opencv.org/about/, Pristupljeno 22. lipnja 2022.

[49] Cascade Classifier: https://docs.opencv.org/3.4/db/d28/tutorial_cascade_classifier.html, Pristupljeno 22. lipnja 2022.

[50] Dmytro Nikolaiev, How to create a Real-time Face Detecor, 20. rujan 2021. : https://to wardsdatascience.com/how-to-create-real-time-face-detector-ff0e1f81925f, Pristupljeno 23. lipnja 2022.

PRILOZI

I. Skripta za slikanje osoba i umjetno proširivanje skupa slika

```
import cv2
import os
import numpy as np
from tensorflow import keras
from keras.preprocessing.image import ImageDataGenerator

name = 'name1'
i = 0
j = 0
path1 = os.path.join('Data', 'Train', name)
if not os.path.exists(path1):
    os.makedirs(path1)
path2 = os.path.join('Data', 'Test', name)
if not os.path.exists(path2):
    os.makedirs(path2)

cap = cv2.VideoCapture(0)
cap.set(3, 1280)
cap.set(4, 720)
dim = (250, 250)
face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades +
'haarcascade_frontalface_default.xml')
while True:
    ret, frame = cap.read()
    if not ret:
        break
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    faces = face_cascade.detectMultiScale(
        gray,
        scaleFactor=1.1,
        minNeighbors=3,
        minSize=(150, 150)
    )
    for (x, y, w, h) in faces:
        face_img_gray = gray[y:y + h, x:x + w]
        laplacian_var = cv2.Laplacian(face_img_gray, cv2.CV_64F).var()
        if laplacian_var > 20 and j % 10 == 0:
            img_name = f'Frame_{i}.jpg'
            if i < 50:
                center = frame[110:610, 390:890]
                resized = cv2.resize(center, dim)
                cv2.imwrite(os.path.join(path1, img_name), resized)
            if i >= 50:
```

```
            center = frame[110:610, 390:890]
            resized = cv2.resize(center, dim)
            cv2.imwrite(os.path.join(path2, img_name), resized)
         if i > 63:
            cap.release()
            cv2.destroyAllWindows()
         print(i, laplacian_var)
         i = i + 1
   j = j + 1
   frame_rec = cv2.rectangle(frame, (390, 110), (890, 610), (0, 0, 255), 2)
   cv2.imshow("Web cam", frame_rec)
   c = cv2.waitKey(1)
   if c == 27:
      break
cap.release()
cv2.destroyAllWindows()

img_height, img_width = 250, 250
train_datagen = ImageDataGenerator(
   rotation_range=20,
   width_shift_range=0.2,
   height_shift_range=0.2,
   brightness_range=(0.7, 1),
   shear_range=0.2,
   zoom_range=0.2,
   horizontal_flip=True,
   vertical_flip=False,
   fill_mode='nearest')
n = 3
i2 = 0
total = len(os.listdir(path1))
for filename in os.listdir(path1):
   print("Step {} of {}".format(i2+1, total))
   image_path = os.path.join(path1, filename)
   image = keras.preprocessing.image.load_img(
      image_path, target_size=(img_height, img_width, 3))
   image = keras.preprocessing.image.img_to_array(image)
   image = np.expand_dims(image, axis=0)
   current_image_gen = train_datagen.flow(image,
                                          batch_size=1,
                                          save_to_dir=path1,
                                          save_prefix=filename[:len(filename) - 4],
                                          save_format="jpg")
   count = 0
   for image in current_image_gen:
      count += 1
      if count == n:
         break
   print('\tGenerate {} samples for file {}'.format(n, filename))
   i2 += 1
```

```
print("\nTotal number images generated = {}".format(n*total))

n = 2
i3 = 0
total = len(os.listdir(path2))
for filename in os.listdir(path2):
   print("Step {} of {}".format(i3+1, total))
   image_path = os.path.join(path2, filename)
   image = keras.preprocessing.image.load_img(
      image_path, target_size=(img_height, img_width, 3))
   image = keras.preprocessing.image.img_to_array(image)
   image = np.expand_dims(image, axis=0)
   current_image_gen = train_datagen.flow(image,
                                          batch_size=1,
                                          save_to_dir=path2,
                                          save_prefix=filename[:len(filename) - 4],
                                          save_format="jpg")
   count = 0
   for image in current_image_gen:
      count += 1
      if count == n:
         break
   print('\tGenerate {} samples for file {}'.format(n, filename))
   i3 += 1

print("\nTotal number images generated = {}".format(n*total))
```

II. Skripta za treniranje neuronske mreže

```
import os
import tensorflow as tf
from tensorflow import keras
from keras.callbacks import ModelCheckpoint
from keras.callbacks import EarlyStopping
import matplotlib.pyplot as plt


train_image_folder = os.path.join('Data', 'Train')
test_image_folder = os.path.join('Data', 'Test')
img_height, img_width = 250, 250
num_classes = len(os.listdir(train_image_folder))
validation_ratio = 0.2
batch_size = 16
AUTOTUNE = tf.data.AUTOTUNE
```

```
train_ds = keras.preprocessing.image_dataset_from_directory(
    train_image_folder,
    validation_split=validation_ratio,
    subset="training",
    seed=42,
    image_size=(img_height, img_width),
    label_mode='categorical',
    batch_size=batch_size,
    shuffle=True)
val_ds = keras.preprocessing.image_dataset_from_directory(
    train_image_folder,
    validation_split=validation_ratio,
    subset="validation",
    seed=42,
    image_size=(img_height, img_width),
    batch_size=batch_size,
    label_mode='categorical',
    shuffle=True)
test_ds = keras.preprocessing.image_dataset_from_directory(
    test_image_folder,
    image_size=(img_height, img_width),
    label_mode='categorical',
    shuffle=False)
class_names = test_ds.class_names
base_model = keras.applications.ResNet50(weights='imagenet',
                                         include_top=False,
                                         input_shape=(img_height, img_width, 3))
for layer in base_model.layers:
    layer.trainable = False
global_avg_pooling = keras.layers.GlobalAveragePooling2D()(base_model.output)
output = keras.layers.Dense(num_classes, activation='sigmoid')(global_avg_pooling)
face_classifier = keras.models.Model(inputs=base_model.input,
                                     outputs=output,
                                     name='ResNet50')
```

```
checkpoint = ModelCheckpoint("save/face_classifier.h5",
                             monitor="val_loss",
                             mode="min",
                             save_best_only=True,
                             verbose=1)
earlystop = EarlyStopping(monitor='val_loss',
                          restore_best_weights=True,
                          patience=2,
                          # min_delta=0.000001,
                          verbose=1)

callbacks = [earlystop, checkpoint]
face_classifier.compile(loss='categorical_crossentropy',
                        optimizer=keras.optimizers.Adam(learning_rate=0.01),
                        metrics=['accuracy'])
epochs = 50
history = face_classifier.fit(
    train_ds,
    epochs=epochs,
    callbacks=callbacks,
    validation_data=val_ds)
face_classifier.save("save/face_classifier.h5")

acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']
plt.figure(figsize=(8, 8))
plt.subplot(2, 1, 1)
plt.plot(acc, label='Training Accuracy')
plt.plot(val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.ylabel('Accuracy')
plt.ylim([min(plt.ylim()), 1.1])
```

```
plt.title('Training and Validation Accuracy')
plt.subplot(2, 1, 2)
plt.plot(loss, label='Training Loss')
plt.plot(val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.ylabel('Cross Entropy')
plt.ylim([0, 0.5])
plt.title('Training and Validation Loss')
plt.xlabel('epoch')
plt.show()
```

III. Skripta za prepoznavanje lica u realnom vremenu

```
import numpy as np
import tensorflow as tf
from tensorflow import keras
import cv2

RED = (0, 0, 255)
GREEN = (0, 255, 0)
BLUE = (255, 0, 0)
WHITE = (255, 255, 255)
color = RED

face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades +
'haarcascade_frontalface_default.xml')

model_name = 'face_classifier.h5'
face_classifier = keras.models.load_model(f'save/{model_name}')
class_names = ['name1', 'name2', 'name3']

def get_extended_image(img, x, y, w, h, k=0.1):
    if x - k * w > 0:
        start_x = int(x - k * w)
```

```
    else:
        start_x = x
    if y - k * h > 0:
        start_y = int(y - k * h)
    else:
        start_y = y

    end_x = int(x + (1 + k) * w)
    end_y = int(y + (1 + k) * h)

    face_image = img[start_y:end_y, start_x:end_x]
    face_image = tf.image.resize(face_image, [250, 250])
    face_image = np.expand_dims(face_image, axis=0)
    return face_image



video_capture = cv2.VideoCapture(0)
if not video_capture.isOpened():
    print("Unable to access the camera")
else:
    print("Access to the camera was successfully obtained")
print("Streaming started - to quit press ESC")
while True:
    ret, frame = video_capture.read()
    if not ret:
        print("Can't receive frame (stream end?). Exiting ...")
        break
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    faces = face_cascade.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=3)
    for (x, y, w, h) in faces:
        face_image = get_extended_image(frame, x, y, w, h, 0.5)
        result = face_classifier.predict(face_image)
        prediction = class_names[np.array(result[0]).argmax(axis=0)]
        confidence = np.array(result[0]).max(axis=0)
```

```
        print(prediction, result, confidence)
        if prediction == class_names[0]:
            color = BLUE
        if prediction == class_names[1]:
            color = GREEN
        if prediction == class_names[2]:
            color = WHITE
        if confidence > 0.8:
            cv2.rectangle(frame, (x, y), (x + w, y + h), color, 2)
            cv2.putText(frame, "{:6} : {:.2f}%".format(prediction, confidence * 100), (x, y),
cv2.FONT_HERSHEY_PLAIN, 2, color, 2)
        else:
            cv2.rectangle(frame, (x, y), (x + w, y + h), RED, 2)
    cv2.imshow("Face detector - to quit press ESC", frame)
    key = cv2.waitKey(1)
    if key % 256 == 27:  # ESC
        break

video_capture.release()
cv2.destroyAllWindows()
print("Streaming ended")
```