Izrada 3D modela za strojno učenje kirurških instrumenata

Grdić, Matea

Sveučilište Jurja Dobrile u Puli

Fakultet informatike u Puli

MATEA GRDIĆ

Izrada 3D modela za strojno učenje kirurških instrumenata

Diplomski rad

Pula, 2021.

Sveučilište Jurja Dobrile u Puli

Fakultet informatike u Puli

MATEA GRDIĆ

Izrada 3D modela za strojno učenje kirurških instrumenata

Diplomski rad

JMBAG: 0303061533, redoviti student

Studijski smjer: Informatika

Predmet: Računalna grafika

Znanstveno područje: Društvene znanosti

Znanstveno polje: Informacijske i komunikacijske znanosti

Znanstvena grana: Informacijski sustavi i informatologija

Mentor: izv. prof. dr. sc. Sven Maričić

Pula, rujan 2021.

IZJAVA O AKADEMSKOJ ČESTITOSTI

Ja, dolje potpisani, **Matea Grdić** kandidat za magistra informatike ovime izjavljujem da je ovaj Diplomski rad rezultat isključivo mojega vlastitog rada, da se temelji na mojim istraživanjima te da se oslanja na objavljenu literaturu kao što to pokazuju korištene bilješke i bibliografija. Izjavljujem da niti jedan dio Diplomskog rada nije napisan na nedozvoljen način, odnosno da je prepisan iz kojega necitiranog rada, te da i koji dio rada krši bilo čija autorska prava. Izjavljujem, također, da nijedan dio rada nije iskorišten za koji drugi rad pri bilo kojoj drugoj visokoškolskoj, znanstvenoj ili radnoj ustanovi.

Student

U Puli, 29.09, 2021 godine

IZJAVA

o korištenju autorskog djela

Ja, **Matea Grdić** dajem odobrenje Sveučilištu Jurja Dobrile u Puli, kao nositelju prava iskorištavanja, da moj diplomski rad pod nazivom Izrada 3D modela za strojno učenje kirurških instrumenata koristi na način da gore navedeno autorsko djelo, kao cjeloviti tekst trajno objavi u javnoj internetskoj bazi Sveučilišne knjižnice Sveučilišta Jurja Dobrile u Puli te kopira u javnu internetsku bazu završnih radova Nacionalne i sveučilišne knjižnice (stavljanje na raspolaganje javnosti), sve u skladu s Zakonom o autorskom pravu i drugim srodnim pravima i dobrom akademskom praksom, a radi promicanja otvorenoga, slobodnoga pristupa znanstvenim informacijama.

Za korištenje autorskog djela na gore navedeni način ne potražujem naknadu.

U Puli, 29.09.2021

Potpis

ZAHVALA

Veliku zahvalnost dugujem mentoru izv. prof. dr. sc. Svenu Maričiću na nesebičnoj pomoći, brojnim savjetima, uloženom trudu, izdvojenom vremenu i potpori tijekom izrade rada. Voljela bi se zahvaliti na ukazanom povjerenju i mogućnosti sudjelovanja na projektu BRIGHT (Boosting the scientific excellence and innovation capacity of 3D printing methods in pandemic period) u sklopu kojega je diplomski rad izrađen.

Veliko hvala, Damiru Medvedu na izdvojenom vremenu, stručnoj pomoći i savjetima koji su bili od velike pomoći za realizaciju ovog rada.

Također, htjela bih se zahvaliti, Općoj bolnici Pula i dr. med. Ivici Fedelu na savjetima i ustupljenim informacijama potrebnih za izradu aplikacije.

Sadržaj

1 Uvod

Živimo u globalnom društvu gdje tehnologija, posebno informacijske i komunikacijske tehnologije, mijenjaju način na koji ljudi generiraju i bilježe vrijednost, mijena se načini kako i gdje radimo te kako se povezujemo i komuniciramo (Cascio i Montealegre, 2016). Tehnološki svijet se drastično razvija, u kratkom vremenskom razdoblju. (Lee i Grewal, 2004). Nove tehnologije sežu u gotovo sve djelatnosti kako bi se unaprijedili radni procesi, u vidu izrade novih inovativnih rješenja te optimizirali radni procesi. Razvojem novih tehnologija javlja se potreba za razvojem umjetne inteligencije koja bi unaprijedila i olakšala poslove i zadatke s kojima se ljudi svakodnevno susreću.

Umjetna inteligencija općeniti je pojam koji podrazumijeva upotrebu računala za modeliranje inteligentnog načina ponašanja uz minimalnu ljudsku intervenciju. U literaturi se umjetna inteligencija opisuje se kao znanost i inženjerstvo izrade inteligentnih strojeva. Pojam umjetne inteligencije primjenjiv je na široki spektar u medicini poput robotike, medicinske dijagnoze te medicinske statistike (Hamet i Tremblay, 2017). Umjetna inteligencija primjenjuje se u medicinske svrhe od svog početka, a neki od najranijih radova u uspješnoj primjeni tehnologije umjetne inteligencije dogodili su se upravo u medicinskom kontekstu (Altman, 1999). Umjetna inteligencija koristi se za razne zdravstvene i istraživačke svrhe. Do sada se najviše koristila u svrhu otkrivanja bolesti, upravljanje kroničnim stanjima, kod pružanja zdravstvenih usluga i u otkrivanju lijekova (Datta, 2019).

Jedna od primjena umjetne inteligencije je u području interpretacije slika u kojoj softver može prepoznati te karakterizirati objekte (Rhoads, 2020). Čovjek lako prepozna objekte i okolinu oko sebe, dok je u računalnom vidu ovaj proces mnogo zahtjevniji. Zajedno s razvojem računalnih tehnologija, radi se na usavršavanju modela strojnog učenja kako bi se replicirala ljudska inteligencija. U ovome radu promatrati će se umjetna inteligencija u prepoznavanju i detekciji kirurških instrumenata. Glavni cilj ovoga diplomskog rada bio je klasificirati kirurške instrumente. U sklopu ovoga rada izrađen je model strojnog učenja koji ukomponiran u aplikaciju kojoj korisnik može pristupiti preko preglednika ili mobilnog uređaja.

U ovome radu objedinjen je pojam umjetne inteligencije, navedene su prednosti umjetne inteligencije u medicini, objašnjen je pojam računalnog vida, te su nabrojane i objašnjeni algoritmi strojnog učenja koji se koriste u računalnom vidu. Objašnjen je pojam neuronskih mreža, umjetnih neuronskih mreža, konvolucijske neuronske mreže.

U izradi aplikacije korišteni su sljedeći razvojni alati: Cloud Annotations, Colaboratory i Node-RED. Cloud Annotations je alat pomoću kojega su označeni kirurški instrumenti. Colaboratory je razvojno okruženje korišteno za izradu modela, za detekciju kirurških instrumenata. Grafičko sučelje aplikacije izrađeno je pomoću razvojnog okruženja Node-RED. Aplikacija korisniku omogućuje izravno fotografiranje slike za prepoznavanje te učitavanje slike s računala. Model je treniran na pet različitih kirurških instrumenata: zakrivljene kirurške škare, ravne kirurške škare, skalpel, kirurška pinceta te hvataljke.

2 Umjetna inteligencija

Polje umjetne inteligencije (*eng. artificial intelligence*), pokušava razumjeti inteligentne entitete. Za razliku od filozofije i psihologije, koje se također bave inteligencijom, umjetna inteligencija nastoji izgraditi i razumjeti inteligentne entitete (Russell i Norvig, 2010). Često se postavljaju pitanja: „Što je inteligencija“, „Kako se može mjeriti inteligencija“ ili „Kako mozak radi“. Ovo su glavna pitanja pri pokušavanju shvaćanja umjetne inteligencije. Međutim, središnje pitanje za inženjere, posebno informatičare, pitanje je inteligentnih strojeva koji se ponašaju kao osoba, pritom pokazujući inteligentno ponašanje (Ertel, 2011). John McCarthy 1955. godine prvi je definirao pojam umjetne inteligencije: „Cilj umjetne inteligencije je razviti strojeve koji se ponašaju kao da su inteligentni“.

U Enciklopediji Britannica (1991). ponuđena je definicija umjetne inteligencije koja glasi: „Umjetna inteligencija je sposobnost digitalnih računala da rješavaju probleme koji su obično povezani s višim sposobnostima intelektualne obrade ljudi“. Ova definicija ima svojih slabosti. Računalo s velikom memorijom može spremiti dugi tekst i dohvatiti ga na zahtjev inteligentne sposobnosti, pamćenje dugih tekstova zasigurno se može smatrati većom sposobnosti od intelektualne sposobnosti ljudi. Prema ovoj definiciji, svako računalo bi se moglo smatrati sustavom umjetne inteligencije (Eritel, 2011).

Elaine Rich (1983). je ponudio definiciju: „Umjetna inteligencija je grana koja proučava kako natjerati računala da rade stvari u kojima, su ljudi bolji“. Rich jezgrovito karakterizira sve ono što su istraživači umjetne inteligencije radili posljednjih 50 godina. Zadaci kao što je izvršavanje mnogih izračuna u veoma kratkom vremenu su jake strane digitalnih računala, u ovom pogledu računalna mnogostruko nadmašuju ljude. Međutim, u mnogim područjima ljudi su daleko superiorniji od strojeva. Na primjer, osoba koja ulazi u nepoznatu sobu prepoznati će okolinu u djeliću sekunde, jednako tako ako je potrebno donositi odluke i planirati radnje. Do danas ovaj je zadatak veoma zahtjevan za autonomne robote (Eritel, 2011). Inteligentni sustavi, ne mogu se graditi bez dubokog nerazumijevanja ljudskog rasuđivanja i inteligentnog djelovanja uopće, zbog čega neuroznanost igra veliku ulogu u razumijevanju umjetne inteligencije. Posebna snaga ljudske inteligencije je prilagodljivost. Ljudi su sposobni prilagoditi se na razne uvjete okoline i u skladu s time mijenjati svoje ponašanje kroz učenje. Prema Richovoj definiciji, strojno učenje je središnje polje umjetne inteligencije, iz razloga što je ljudska sposobnost učenja toliko superiornija od računala (Eritel, 2011).

U Tablici 1. prikazane su četiri definicije umjetne inteligencije, raspoređene u dvije dimenzije. U prvom redu prikazane su definicije s misaonim procesima i zaključivanjem. U

drugom redu, prikazane su definicije čiji je fokus na ponašanju. Na lijevoj strani tablice prikazane su definicije koje mjere uspjeh u smislu vjernosti ljudskom učinku. Na desnoj strani, prikazane su definicije koje promatraju racionalnost. Sustav se smatra racionalnim ako čini „pravu stvar“, s obzirom na ono što zna. Pristup usmjeren na čovjeka, dijelom je empirijska znanost, koja uključuje promatranja i hipoteze o ljudskom ponašanju. racionalistički pristup uključuje kombinaciju matematike i tehnike (Russell i Norvig, 2010).

Tablica 1. Definicije umjetne inteligencije, organizirane u četiri kategorije.

Sustavi koji razmišljaju poput ljudi	**Sustavi koji racionalno razmišljaju**
„Aktivnosti koje povezujemo s ljudskim razmišljanjem, aktivnosti poput donošenja odluka, rješavanja problema, učenja ...“ (Bellman, 1978).	*„Proučavanje mentalnih sposobnosti kroz upotrebu računskih modela "* *(Charniak i McDermott, 1985).*
Sustavi koji se ponašaju poput ljudi	**Sustavi koji djeluju racionalno.**
"Umjetnost stvaranja strojeva koji izvode funkcije koje zahtijevaju inteligenciju, kao kada je izvode ljudi" (Kurzweil, 1990).	*"Područje proučavanja koje nastoji objasniti i oponašati inteligentno ponašanje u terminima računskih procesa" (Schalkoff, 1990).*

Izvor: Prilagođeno prema: Russell i Norvig (2010)

2.1 Strojno učenje

Strojno učenje (*eng. Machine learning*) je podskup umjetne inteligencije koji gradi matematički model temeljen na uzorcima, poznatim kao "podaci o obuci", kako bi donosio predviđanja ili odluke bez eksplicitnog programiranja za izvršavanje zadatka (Samuel, 1959). Alpaydin (2014). je definirao strojno učenje kao „Programiranje računala za optimiziranje kriterija izvedbe, na temelju primjera podataka ili prošlog iskustva“. Strojno učenje obično se odnosi na promjene u sustavima koji izvode zadatke povezane s umjetnom inteligencijom. Ti zadaci uključuju planiranje, upravljanje, prepoznavanje, dijagnosticiranje,

predviđanje itd. „Promjene“ označavaju poboljšanja već postojećeg sustava ili sintezu novih (Nilsson, 1998).

Podaci o obuci se odnose na prošle informacije dostupne učeniku, koje su obično u obliku elektroničkih podataka prikupljenih i dostupnih za analizu. Podaci bi trebali biti u obliku digitaliziranih skupova s oznakom informacija dobivenih interakcijom s okolinom. Kvaliteta i veličina skupa informacija ključna je za uspjeh predviđanja učenika (Mohri, Rostamizadeh, Talwalkar, 2018). Učenje, poput inteligencije, pokriva širok raspon procesa te ga je teško precizno definirati (Nilsson, 1998). Procesi učenja uključuju stjecanje novih deklarativnih znanja, razvoj motoričkih i kognitivnih vještina, poukom ili praksom, organiziranje novih znanja u općenite, učinkovite prikaze te otkrivanje novih činjenica i teorija promatranjem i eksperimentiranjem (G.Carbonell, S.Michalski i M.Mitchell, 1983).

Strojno učenje sastoji se od dizajniranja učinkovitih i točnih algoritama predviđanja. Kao i u drugim područjima računalnih znanosti, neke su kritične mjere kvalitete tih algoritama su njihova vremenska i prostorna složenost. Budući da uspjeh algoritma učenja ovisi o korištenim podacima, strojno učenje je usko povezano s analizom podataka i statistikom. Općenito, tehnike učenja su metode vođene podacima koje kombiniraju temeljne pojmove u računalnoj znanosti sa idejama iz statistike, vjerojatnosti i optimizacije (Mohri, Rostamizadeh, Talwalkar, 2018).

Tehnike temeljene na strojnom učenju uspješno su primijenjene u različitim područjima, od prepoznavanja uzoraka, računalnog vida, inženjeringa svemirskih letjelica, financija, zabave i računalne biologije do biomedicinskih i medicinskih primjena (El Naqa i J. Murphy, 2015).

S obzirom dali je u postupku strojnog učenja poznat izlaz ili nije moguće je proces strojnog učenja podijeliti u dvije kategorije: nadzirano učenje i nenadzirano učenje. U nastavku su objašnjene ove dvije kategorije.

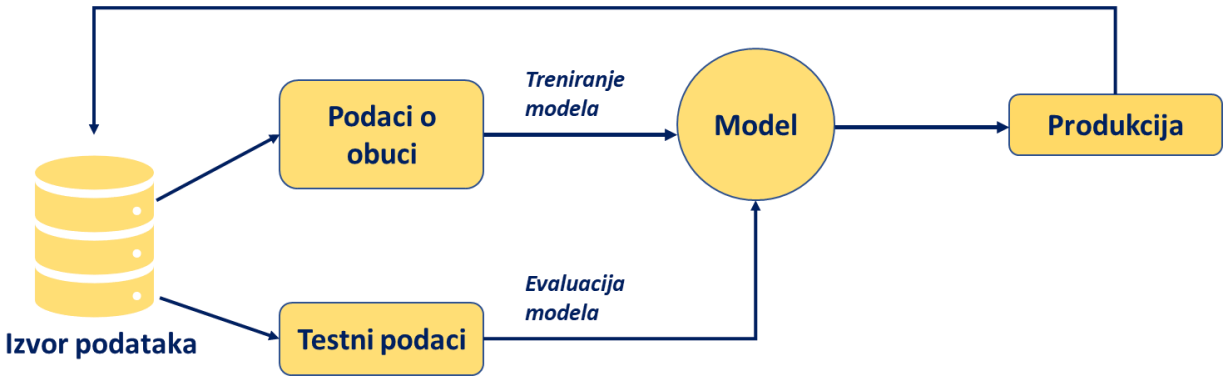
2.1.1 Nadzirano učenje

Nadzirano učenje (*eng. Supervised learning*) zadatak je strojnog učenja funkcije koja preslikava ulaz na izlaz na temelju primjera parova ulaz-izlaz. On zaključuje funkciju iz označenih podataka o obuci koji se sastoje od niza primjera obuke. Algoritmi nadziranog strojnog učenja su oni algoritmi kojima je potrebna vanjska pomoć (Mahesh, 2018).

Glavna karakteristika nadziranog učenja je dostupnost označenih podataka o obuci. Naziv se poziva na ideju "učitelja" koji upućuje sustav učenja na oznakama da se poveže s primjerima obuke. Obično su ove oznake klasa u problemima klasifikacije. Nadgledani

algoritmi učenja induciraju modele iz ovih podataka o obuci, a ti se modeli mogu koristiti za klasifikaciju drugih podataka koji ne sadržavaju oznake (Cunningham, Cord i Delany, 2008). Na Slici 1. prikazan je tijek nadziranog strojnog učenja.

Slika 1. Tijek nadziranog učenja



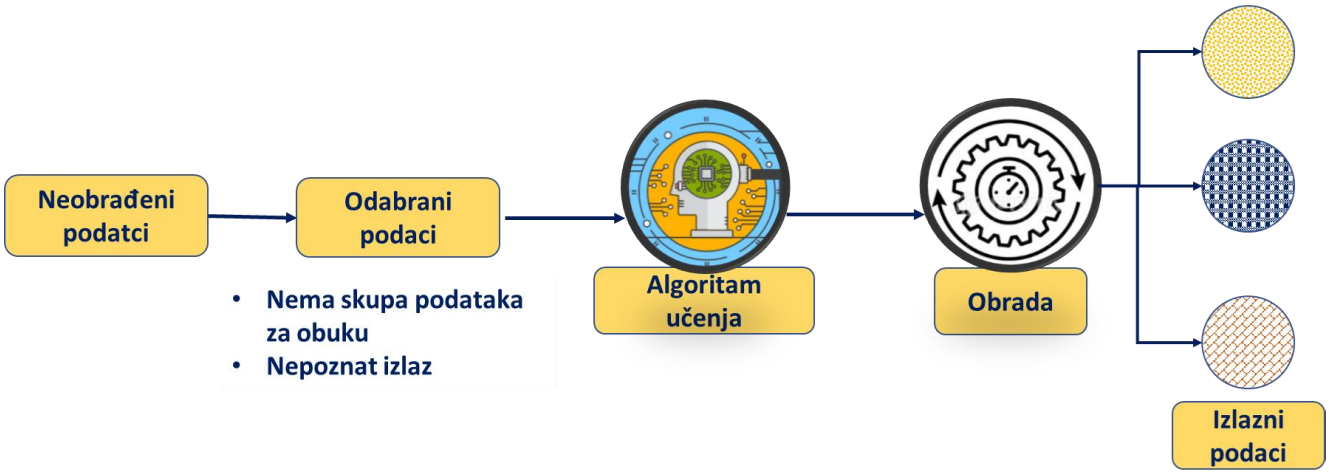
Izvor: izrada autora prema: Mahesh (2018).

2.1.2 Nenadzirano učenje

Nenadzirano učenje (*eng. Unsupervised learning*) ne koristi nikakve podatke o obuci ili testiranju. Umjesto toga, “pokušava” klasificirati nepoznate podatke odvajanjem podataka u različite klase (klastere). Ova metoda se još naziva "učenjem bez učitelja". Metoda pokušava izravno izgraditi modele ne temeljeći se ni na jednom unaprijed izgrađenom modelu ili znanju. Uči iz neoznačenih podataka, zadatak ove metode je otkriti klase sličnih primjera iz neoznačenih podataka i organizirati podatke u grupe sličnosti, što je poznato kao grupiranje, ili procjenom distribucije podataka unutar ulaznog prostora koji se naziva procjena gustoće. Grupiranje je proces organiziranja neoznačenih podataka u klastere, gdje su podaci u istom klasteru međusobno slični, a podaci u različitim klasterima različiti (Bajrami, Derawi i Bours, 2011). Algoritmi nenadziranog učenja prepušteni su sami sebi, njihov je zadatak otkriti i predstaviti zanimljivu strukturu podataka. Kad se uvedu novi podaci, koriste se prethodno naučene značajke za prepoznavanje klase podataka (Mehesh, 2018).

Na Slici 2. prikazan je tijek nenadziranog učenja.

Slika 2. Tijek nenadziranog učenja



Izvor: izrada autora prema: Mahesh (2018).

2.2 Duboko učenje

Duboko učenje *(eng. Deep learning)* predstavlja najnoviju, modernu tehniku za obradu slika i analizu podataka, s obećavajućim rezultatima i velikim potencijalom (Kamilaris i Prenafeta-Boldú, 2018). Duboko učenje je podskup strojnog učenja gdje umjetne neuronske mreže i algoritmi temeljeni na strukturi i funkcioniranju ljudskog mozga, uče iz velikih količina podataka kako bi stvorili obrasce za donošenje odluka. Neuronske mreže s različitim (dubokim) slojevima omogućuju učenje kroz ponavljanje izvršavanja zadataka i njihovo malo prilagođavanje kako bi se poboljšao ishod (LeCun, Bengio i Hinton, 2015).

Duboko učenje omogućuje računalnim modelima koji se sastoje od više slojeva obrade da nauče prikaze podataka s više razina apstrakcije. Ove su metode dramatično poboljšale stanje u području prepoznavanja govora, prepoznavanja vizualnih objekata, otkrivanja objekata i mnogih drugih domena, poput otkrivanja lijekova i genomike (LeCun, Bengio i Hinton, 2015). Duboko učenje podrazumijeva automatsko učenje više razina prikaza temeljne distribucije podataka koje treba modelirati. Na primjer, u kontekstu računalnog vida, to implicira da će algoritam dubokog učenja naučiti vlastite prikaze niske razine iz neobrađene slike, zatim izgraditi reprezentacije koje ovise o tim reprezentacijama niske razine i uzastopno ponavljati isti postupak do više razine (Lauzon, 2012). Učenje s automatskim predstavljanjem, ključna je točka interesa ove vrste pristupa jer je eliminirana potreba za dugotrajnim ručno izrađenim dizajnom značajki (Lauzon, 2012).

U aplikacijama je duboko učenje postiglo veliki napredak u govoru i slici, čime se promiče napredak umjetne inteligencije i interakcija čovjeka i računala (Hao, Zhang i Ma, 2016).

3 Umjetna inteligencija u medicini

Tehnika umjetne inteligencije najučinkovitija je tehnologija korištena u suvremenom zdravstvenom području. Pristup medicinskim podacima koji postaju sve dostupniji, kao i napredak dijagnostičkih tehnika velikih podataka, doprinijeli su uspješnoj upotrebi umjetne inteligencije u zdravstvenom sustavu (Datta, 2019). Potencijalne tehnike umjetne inteligencije mogu obraditi velike količine podataka, što može pomoći u donošenju zdravstvenih odluka. Umjetna inteligencija u medicini može se podijeliti na dvije glavne grane: virtualna i fizička. Virtualnu komponentu može se predstaviti strojnim učenjem. Postoje tri vrste algoritama strojnog učenja: (1) bez nadzora (sposobnost pronalaženja obrazaca) (2) s nadzorom (temelji se na algoritmima klasifikacije i predviđanja na prethodnim primjerima) (3) učvršćivanje učenja (upotrebom nagrada i kazni oblikuje se strategija za rad u određenom prostoru). Umjetna inteligencija pruža brojne inovacije na području medicine. Učinkovito analizira informacije, medicinsku dokumentaciju i sustave te poboljšava digitalnu automatizaciju za brže i dosljednije rezultate. Pomaže liječnicima u postizanju boljih rezultata (Haleem, Javaid i Haleem Khan, 2019). Na Slici 3. prikazane su neke od prednosti korištenja umjetne inteligencije u području medicine, kirurgije, radiologije, bolničke uprave i medicinske dijagnostike te kardiologije.

Slika 3. Prednosti umjetne inteligencije u medicini

Izvor: izrada autora prema: Haleem, Javaid i Haleem Khan (2019).

Strojno učenje primjenjuje se na širokom rasponu računalnih zadataka, poput optičkog prepoznavanja znakova, filtriranja e -pošte, računalnog vida i mnogih drugih (Park, Took i Seong, 2018).

Strojno učenje u biomedicinskom inženjeringu pokušava obuhvatiti niz različitih aplikacija strojnog učenja u području biomedicinskog inženjeringa, s posebnim naglaskom na najreprezentativnije tehnike strojnog učenja ili pristupom dubokom učenju. Zadaci strojnog učenja obično se razvrstavaju u dvije široke kategorije, ovisno o tome postoji li sustav ocjenjivanja učenja ili povratne informacije: nadzirano učenje i učenje bez nadzora.

4 Računalni vid

Baumgart (1975). okarakterizirao je računalni vid kao inverzna računalna grafika. U računalnoj grafici, svijet je predstavljen dovoljno detaljno tako da slika proces formatiranja može se numerički simulirati za generiranje sintetičke digitalne slike; obrnuto, percipirane digitalne slike se analiziraju da bi se mogli izračunati detaljniji geometrijski modeli.

Perceptivni psiholozi desetljećima su pokušavali razumjeti kako vizualni sustav funkcionira te kako osmisliti optičke iluzije. Istraživači računalnog vida paralelno su istraživali matematičke tehnike za prikaz trodimenzionalnog oblika i izgleda objekata na slikama. Danas računalni vid nudi mnoge mogućnosti, neke od njih su:

1. Struktura iz algoritma kretanja može rekonstruirati 3D model točki velike složene scene iz stotina fotografija koje se djelomično preklapaju. Snavely, Seitz i Szeliski (2007). koristili su velike zbirke fotografija s interneta i drugih izvora te ih koristili u dobivanju 3D modela i informacija o sceni na slici, koje olakšavaju brojne aplikacije u vizualizaciji, lokalizaciji te pregledavanju slika. Izazovi s kojima se susretali je usklađivanje i rekonstrukcija 3D informacija iz stotinu slika koje pokazuju velike varijacije u gledištu, osvjetljenju, vremenskim uvjetima, razlučivosti itd. U rješavanju ovoga problema koriste se usklađivanje značajki i struktura iz kretanja. Koristeći položaje 3D kamera i rijetku geometriju točaka, dobili su rekonstrukciju na desetke poznatih mjesta. Na Slici 4. prikazana je rekonstruirana scena Koloeuma u Rimu.

Slika 4. Primjer rekonstruirane scene: Koloseum

Izvor: Snavely, Seitz i Szeliski (2007).

2. Algoritmi stereo usklađivanja mogu izraditi detaljan 3D model od stotina fotografija, koje snimaju objekt iz različitih kutova. Goesele, Snavely, Curless, Hoppe i Seitz (2007). izračunali su stereo prikaz s više podataka (*eng. MVS - multi-view stereo*) rekonstrukcije nekoliko baza podataka. Slike u bazi variraju u pogledu veličine, broja fotografija. Slika 5. prikazuje prikaz 3D modela Kipa Slobode dobivenog metodom stereo usklađivanja. Rezultati dokazuju da MVS može rekonstruirati detaljne i visoko kvalitetne karte dubine za vrlo različite ulazne podatke.

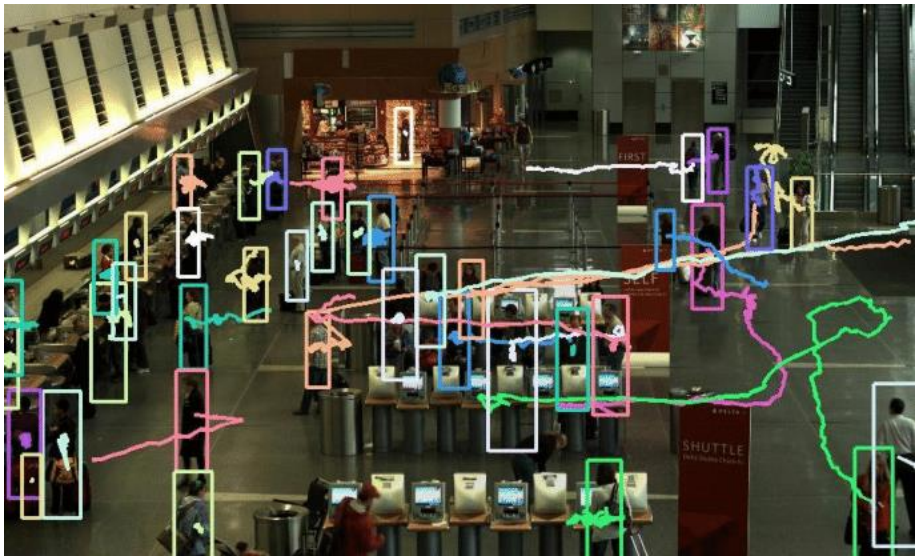
Slika 5. Stereo algoritam Kipa Slobode

Izvor: Goesele, Snavely, Curless, Hoppe i Seitz (2007).

3. Algoritmi praćenja osoba mogu pratiti osobu koja hoda ispred pretrpane pozadine. Shu, Dehghan, Oreifej i Hand (2012) razvili su sustav za praćenje osoba koji se temelji na jednoj kameri. Njihov pristup uči djelomično specifičnu osobu metodom potpornih vektora (*eng. SMV, Support vector machines*) koji bilježe artikulacije ljudskog i tijela u dinamički promjenjivom izgledu i pozadini. S djelomičnim modelom ovaj pristup je sposoban za rješavanje djelomičnih začepljenja i u otkrivanja faza praćenja. U fazi otkrivanja odabire se podskup dijelova koji povećava vjerojatnost otkrivanja, što značajno povećava

vjerojatnost otkrivanja u prepunoj sceni. U fazi praćenja dinamički su riješene okluzije raspodjeljujući ocjenu klasifikatora osobe među odgovarajuće dijelove, što omogućuje otkrivanje i predviđanje djelomičnog začepljenja te sprječavanje performansi klastifikatora od degradacije. Na Slici 6. prikazan je rad sustava praćenja osoba.

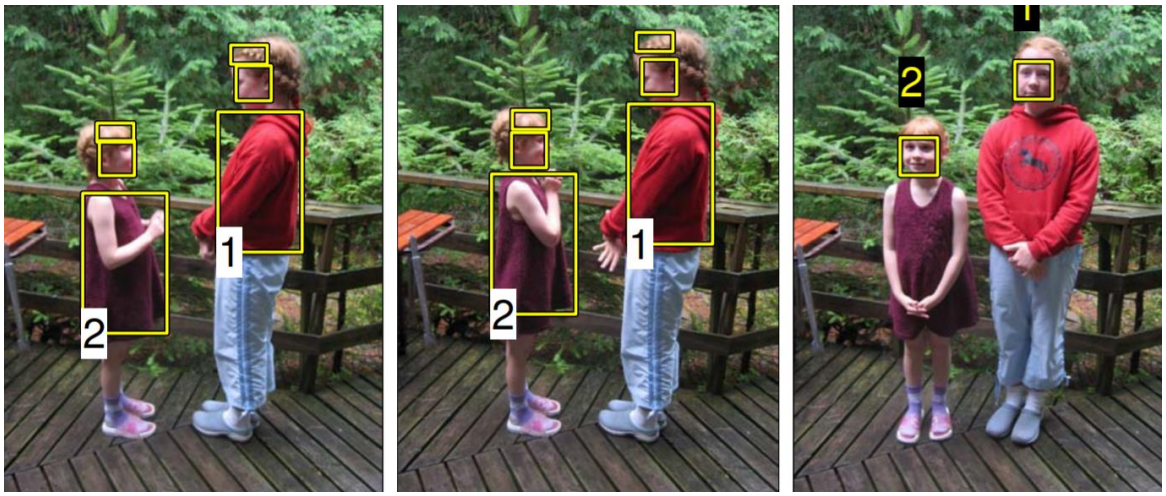
Slika 6. Algoritam za praćenje osoba



Izvor: Shu, Dehghan, Oreifej i Hand (2012).

4. Algoritmi za otkrivanje lica, zajedno s odjećom na bazi boje i algoritma za prepoznavanje kose mogu locirati i prepoznati pojedince na slici. Sivic, Zitnick i Szeliski (2006). razvili su model za detekciju određene osobe u nizu fotografija snimljenih u kratkom vremenskom razdoblju. Radi lakše identifikacije, pretpostavljaju da kosa i odjeća svakog pojedinca ostaje ista. Njihov zadatak bio je izazovan, jer se ljudi mogu kretati, mijenjati pozu te djelomično zakretati. Za otkrivanje osoba na fotografiji Sivic i sur. (2006) koriste dvostupanjsku metodu. Prvo, pojedinci se identificiraju grupiranjem frontalnih detekcija lica pomoću podataka o boji odjeće. Drugo, model slikovne strukture zasnovan na boji, koristi se za pronalaženje pojavljivanja svake osobe na slikama na kojima je propušteno otkrivanje frontalnog lica. Na Slici 7. prikazana je detekcija ljudi na slici.

Slika 7. Otkrivanje ljudi na fotografijama



Izvor: Sivic, Zitnick i Szeliski (2006).

4.1 Raspoznavanje objekta

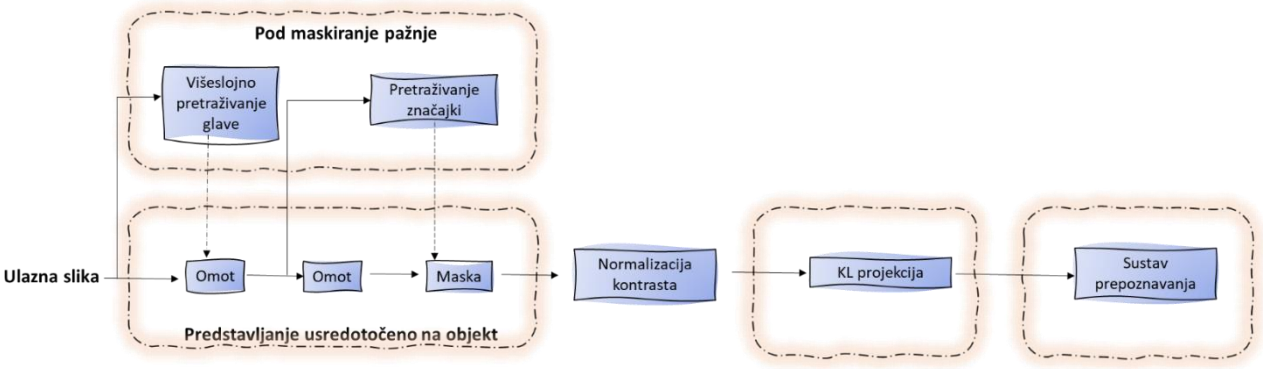
Od svih zadaća s kojima se računalni vid danas susreće, zadatak analize scene i prepoznavanje sastavnih objekta scene i dalje predstavlja velik izazov u području računalnog vida. Glavni izazov u raspoznavanju objekata predstavlja činjenica da je stvarni svijet sačinjen od zbirke predmeta koji zaklanjaju jedni druge te se pojavljuju u različitim pozama (Szeliski, 2010). Najizazovnija verzija prepoznavanja je raspoznavanje opće kategorije (ili klase), što može uključivati prepoznavanje primjera izrazito različitih klasa kao što su životinje ili namještaj. U mnogim slučajevima prepoznavanje ovisi uvelike o kontekstu okolnih objekata i elemenata scene (Szeliski, 2010). Učinkovitost otkrivanja i prepoznavanja objekata uvelike ovisi o kvaliteti izdvojenih značajki i robusnosti klasifikatora, jer na izgled slika mogu utjecati mnogi čimbenici poput uvjeta osvjetljenja, položaja, refleksije objekata i unutarnjih karakteristika kamera. Da bi se postiglo robusno otkrivanje i prepoznavanje, izdvojene značajke koje se koriste za provjeru moraju biti invarijantne na osvjetljenje, pozu i druge transformacije (Jiang, Hadid, Pang, Granger i Feng, 2019).

4.1.1 Prepoznavanje lica

Tijekom godina razvijen je veliki broj algoritama za brzo i učinkovito prepoznavanje lica. Moghaddam i Pentland (1997). predstavili su tehniku bez nadzora za vizualno učenje, koja se temelji na procjeni gustoće u visoko-dimenzionalnom prostoru pomoću dekompozicije vlastitog prostora. Za modeliranje podataka o obuci izvedene su dvije vrste procjena gustoće: multivarijatni Gaussov model i mješoviti-Gaussov model. Ove vjerojatnosti se zatim koriste za vizualno pretraživanje i automatsko prepoznavanje objekta. Njihova tehnika učenja temelji se na vjerojatnosnom vizualnom modeliranju, otkrivanju,

prepoznavanju i kodiranju ljudskih crta lica. Na Slici 8. prikazan je dijagram koji opisuje sustav za obradu lica koji su 1997. osmislili Moghaddam i Pentland.

Slika 8. Sustav za obradu lica



Izvor: Izrada autora prema: Moghaddam i Pentland (1997).

5 Algoritmi strojnog učenja

Algoritmi učenja prisutni su u mnogim aplikacijama koje svakodnevno koristimo. Svaki put kad se web tražilica poput Googlea koristi za pretraživanje Interneta, jedan od razloga koji tako dobro funkcionira je taj što je algoritam za učenje koji je naučio rangirati web stranice. Ovi se algoritmi koriste u razne svrhe, poput rudarstva podataka, obrade slika, prediktivne analitike i mnogim drugim (Mahesh, 2020). U nastavku su opisane neke od najčešće korištenih algoritama strojnog učenja. Ovi algoritmi mogu se primijeniti na gotovo svaki problem s podacima.

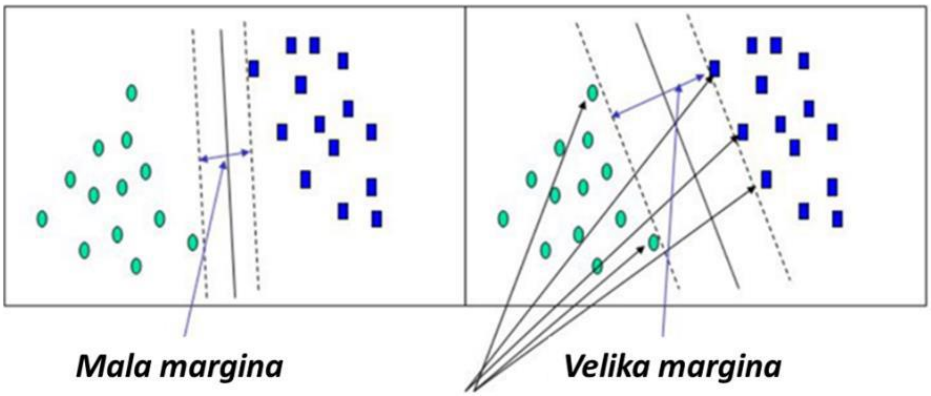
5.1 Metoda potpornih vektora

Metoda potpornih vektora (*eng. Suport-vector machine – SVM*) je računalni algoritam koji na zadanom primjeru uči dodjeljivanju oznaka objektima. Cilj ovog algoritma je pronaći hiperravinu u N-dimenzionalnom prostoru (N- broj značajki) koja klasificira podatkovne točke. Za odvajanje dvije klase podatkovnih točaka postoji mnogo hiperravnina koje se mogu odabrati, no cilj je pronaći ravninu koja ima najveću marginu, odnosno najveću udaljenost između podatkovnih točaka obije klase. Povećanje udaljenosti marže pruža određeno pojačanje tako da se buduće točke podataka mogu pouzdanje klasificirati (Gandhi, 2018).

Metoda potpornih vektora koristi jednostavni matematički model $y=wx+y$, naime se manipulira kako bi se omogućila podjela domene. Stroj potpornih vektora može se podijeliti na linearan i nelinearan model. Koraci u linearnom vektorskom stroju su: preslikavanje

domene podataka u skup odgovora i podjela domene podataka. Koraci u nelinearnom vektorskom stroju su: preslikavanje domene podataka u prostor obilježja pomoću funkcije jezgre, preslikavanje domene prostora obilježja u skup odgovora, a zatim podjela domene podataka (Suthaharan, 2016).

Slika 9. Metoda potpornih vektora



Izvor: Gandhi (2018).

Neka od područja gdje se primjenjuje ovaj algoritam su: prepoznavanje lažne aktivnosti na kreditnim karticama, algoritam metode potpornih vektora može naučiti prepoznati rukom pisane znamenke ispitivanjem velike baze skeniranih slika ručno napisanih nula, jedinica, itd. Također, metoda potpornih vektora se uspješno primjenjuje u sve široj raznolikosti bioloških promjena. Uobičajena biomedicinska primjena potpornih vektora je automatska klasifikacija profila ekspresije gena u mikro redovima (Nobe, 2006). SVM se često primjenjuje u aplikacijama s više domena u okruženju velikih podataka. Međutim, vektorski stoj je matematički složen i računski skup (Suthaharan, 2016).

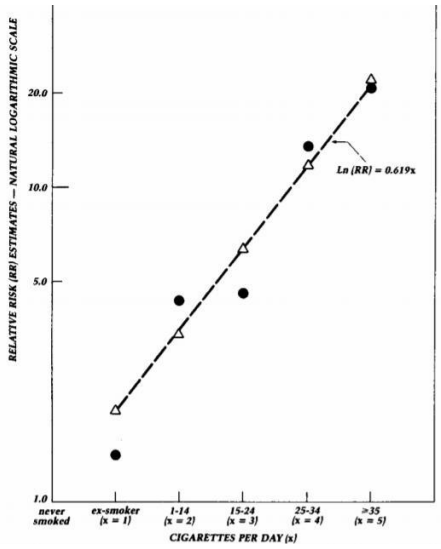
5.2 Linearna Regresija

Linearna regresija (*eng. Linear regression*) je jedna od najčešćih i najopsežnijih algoritama statističkog i strojnog učenja. Linearna regresija se koristi za pronalaženje linearnog odnosa između jednog ili više prediktora. Linearna regresija se dijeli na: jednostavnu regresiju i višestruku regresiju (Maulud i Abdulazeez, 2020).

Strojno učenje se obično koristi u raznim područjima za rješavanje teških problema koji se ne mogu lako riješiti računalnim pristupima. Linearna regresija je jedan od matematičkih pristupa koji se koriste za izvođenje produktivne analize u strojnom učenju, naročito za računalni vid (Meer, Mintz i Rosenfeld, 1991). Ono dopušta kontinuirano/ stvarno ili matematičko projekcije varijabli. Koncept linearne regresije prvi je predložio Sir Francis Galton 1894., te se od tada koristi kao pouzdani matematički test za vrednovanje i kvantificiranje odnosa između promatranih varijabli (Meer, Mintz i Rosenfeld, 1991).

Linearna regresija se obično koristi u matematici istraživačke metode, gdje je moguće mjeriti predviđene učinke i modelirati ih prema više ulaznih varijabli. To je metoda vrednovanja i modeliranja podataka koja uspostavlja linearan odnos između varijabli koje se zavisne i nezavisne (Maulud i Abdulazeez, 2020). Na Slici 10. prikazan je primjer jednostavne linearne regresije. U ovome primjeru zavisna varijabla (x) je broj cigareta koji bi osoba dnevno konzumirala, nezavisna varijabla (y) je relativni rizik od bolesti miokarda (Godfrey, 1985).

Slika 10. Primjer grafa jednostavne linearne regresije



Izvor: Godfrey (1985).

Iako je linearna regresija, prvenstveno statistička metoda, ona se koristi i u strojnom učenju. Strojno učenje je područje koje se bavi prediktivnim modeliranjem, prvenstveno se bavi smanjenjem pogreške modela ili davanjem što točnijih predviđanja (Brownlee, 2020).

5.3 Logistička regresija

Logistička regresija (*eng. Logistic regression*) jedan je od najpopularnijih algoritama strojnog učenja koji potpada pod tehniku nadziranog učenja. Koristi se za predviđanje kategorijalne ovisne varijable pomoću zadanog skupa neovisnih varijabli. Logistička regresija predviđa izlaz kategorijalno ovisne varijable. Stoga ishod mora biti kategorička ili diskretna vrijednost. Može biti ili Da ili Ne, 0 ili 1, točno ili netočno itd., Ali umjesto da daje točnu vrijednost kao 0 i 1, daje vjerojatne vrijednosti koje se nalaze između 0 i 1 (Javatpoint, 2021).

5.4 Stabla odlučivanja

Stablo odlučivanja (*eng. Decision Tree*) predstavlja postupak odlučivanja za određivanje klase date instance (Utgoff, 1989). Stabla odlučivanja se smatraju jednim od

najpopularnijih pristupa za predstavljanje klasifikatora. Stablo odlučivanja je klasifikator izražen kao rekurzivna particija prostora instance. Stablo odlučivanja sastoji se od čvorova koji tvore ukorijenjeno stablo, što znači da je usmjereno stablo s čvorom zvanim "korijen" koji nema dolazne rubove. Svi ostali čvorovi imaju točno jedan dolazni rub. Čvor s izlaznim rubovima naziva se unutarnji ili testni čvor (Rokach i Maimon, 2005). Svi ostali čvorovi nazivaju se listovi (također poznati kao čvorovi odluke). Svaki list je dodijeljen jednoj klasi koja predstavlja najprikladniju ciljnu vrijednost. Alternativno, list može sadržavati vektor vjerojatnosti koji ukazuje na vjerojatnost da ciljni atribut ima određenu vrijednost (Rokach i Maimon, 2005). Stabla odlučivanja konstruiraju se analizom niza primjera obuke za koje su poznate oznake klasa. Zatim se primjenjuju za razvrstavanje prethodno neviđenih primjera.

Ako se obučavaju na visokokvalitetnim podacima, stabla odluka mogu donijeti vrlo točna predviđanja (Kingsford i Salzberg, 2008). Stabla odlučivanja ponekad su lakša za tumačenje od drugih klasifikatora, poput neuronskih mreža i podržavaju vektorske strojeve, jer kombiniraju jednostavna pitanja o podacima na razumljiv način (Kingsford i Salzberg, 2008).

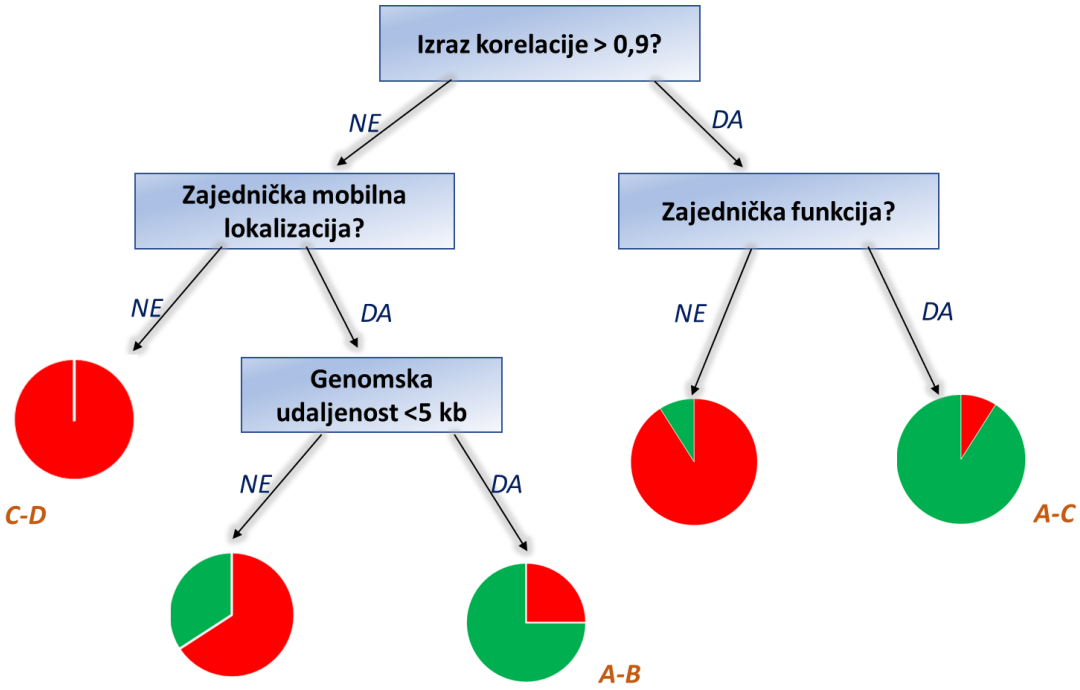
Na Slici 11. je prikazano hipotetičko stablo odlučivanja u kojem svaki čvor sadrži pitanje da/ ne koje postavlja pitanje o jednoj značajci stavki podataka, izrađeno prema Tablici 2. Primjer dolazi do lista prema odgovorima na pitanja. Kružni grafikoni pokazuju postotak interaktora (zelenih) i neinteraktora (crvenih) iz primjera obuke koji dopiru do svakog lista (Kingsford i Salzberg, 2008).

Tablica 2. Stavke podataka klasifikatora stabla odlučivanja.

Par gena	Interakcija?	Korelacija izraza	Zajednička lokalizacija?	Zajednička funkcija?	Genomska udaljenost
A-B	DA	0,77	DA	NE	1 kb
A-C	DA	0,91	DA	DA	10 kb
C-D	NE	0,1	NE	NE	1 Mb

Izvor: Izrada autora prema: Kingsford i Salzberg (2008)

Slika 11. Primjer stabla odluke



Izvor: Izrada autora prema: Kingsford i Salzberg (2008)

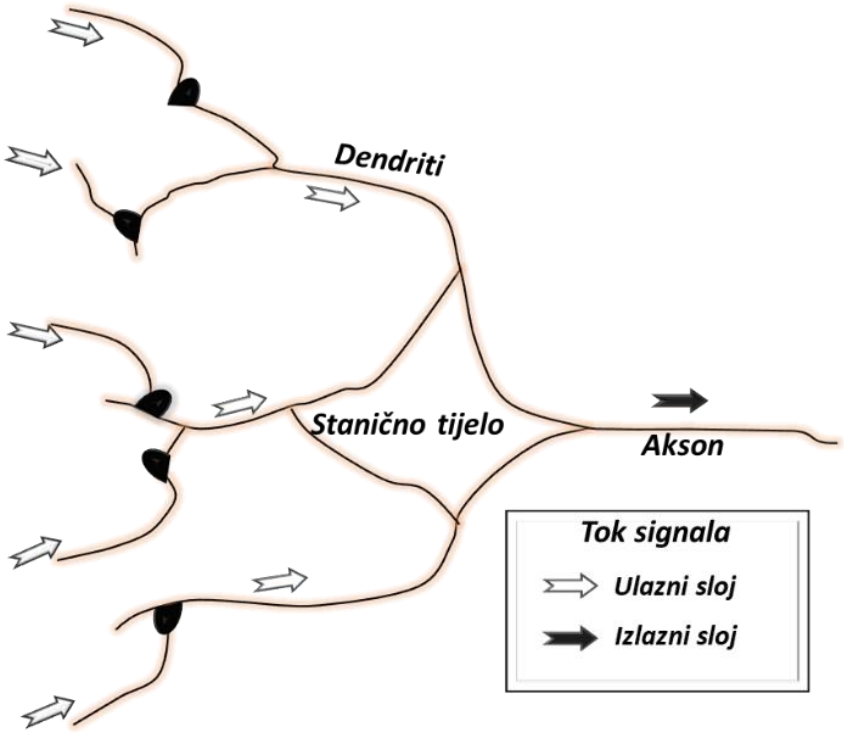
6 Neuronske mreže

Neuronska mreža (*eng. Neural Network*) je međusobno povezan sklop jednostavnih procesnih elemenata, jedinica ili čvorova, čija se funkcionalnost temelji na životinjskom neuronu. Stabilnost obrade mreže pohranjena je u jedinici jačine veze ili težine, dobivene prilagodbe skupu ili učenjem na skupu obrazaca treninga (Gurney, 1997). Standardna neuronska mreža sastoji se od mnogih jednostavnih, povezanih procesora koji se nazivaju neuroni, a svaki proizvodi niz aktivacija u stvarnoj vrijednosti. Ulazni neuroni aktiviraju se pomoću senzora koji percipiraju okoliš, drugi neuroni se aktiviraju putem ponderiranih veza iz prethodno aktivnih neurona. Neki neuroni mogu utjecati na okoliš pokrećući radnje. Učenje ili bodovanje odnosi se na pronalaženje utega zbog kojih neuronska mreža pokazuje željeno ponašanje, poput vožnje automobila. Ovisno o problemu i načinu na koji su neuroni povezani, takvo ponašanje može zahtijevati dugačke uzročne lance računskih faza, gdje svaka faza transformira agregatnu aktivaciju mreže. Duboko učenje se bavi točnom dodjeljivanju kredita u mnogim takvim fazama (Schmidhuber, 2015).

Ljudski se mozak sastoji od 100 milijardi živčanih stanica ili neurona, čiji je stiliziran primjer prikazan na Slici 12. Neuroni komuniciraju putem električnih signala koji su

kratkotrajni impulsi u naponu stanične stjenke ili membrane. Interne neuronske veze posreduju elektrokemijskim spojevima zvanim sinapse, koje se nalaze na granama stanice koje se nazivaju dendriti. Svaki neuron prima više tisuća veza s drugih neurona i stoga stalno prima mnoštvo dolaznih signala, koji na kraju dopiru do staničnog tijela. Oni su integrirani, grubo rečeno, ako rezultirajući signal prelazi neki prag, neuron će „aktivirati“ ili generirati naponski impuls kao odgovor. To se zatim prenosi na druge neurone putem razgranatog vlakna poznatog kao akson.

Slika 12. Bitne komponente neurona prikazane u stiliziranom obliku



Izvor: izrada autora prema: Gurney (1997).

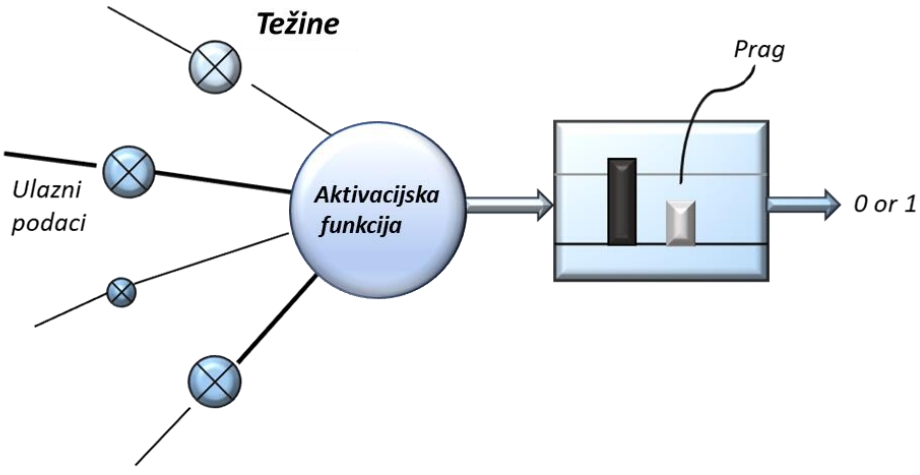
6.1 Umjetne neuronske mreže

Umjetne neuronske mreže (*eng. Artificial Neural Network – ANN*) inspirirane su ljudskim mozgom i mrežom neurona prisutnih u mozgu. Podaci se obrađuju i prenose s jednog neurona na drugi putem neuro sinaptičkih spojeva. Slično, u umjetnim neuronskim mrežama postoje različiti slojevi stanica raspoređeni i međusobno povezani. Izlazne informacije iz unutarnjih slojeva neuronske mreže prenose se na sljedeće slojeve i na kraju u sloj koji daje izlaz (Sharma, Sharma i Athaiya, 2017).

Umjetni ekvivalenti bioloških neurona čvorovi su ili jedinice u preliminarnoj definiciji, njihov prototipni primjer prikazan je na Slici 13. sinapse se modeliraju jednim brojem ili težinom tako da svaki ulaz pomnoži s težinom prije nego se pošalje u ekvivalent tijela stanice. Ovdje se ponderirani signali zbrajaju jednostavnim aritmetičkim zbrojem kako bi se

omogućila aktivacija čvora. U tipu čvora prikazanog na Slici 13. prikazana je logička jedinca praga *(eng. Treshold logic unit- TLU*)- aktivacija se uspoređuje s pragom; ako aktivacija prelazi prag; jedinica proizvodi visoko vrijedni izlaz konvencionalno „1“, u suprotnom vraća „0“. Na Slici 13. veličina signala predstavljena je širinom odgovarajućih strijelaca, a utezi su predstavljeni simbolom množenja u krugovima; njihove vrijednosti su proporcionalne veličini simbola. Logička jedinca praga- TLU je najjednostavniji i povijesno najraniji model umjetnog neurona (Gurney, 1997). Složenost pravih neurona vrlo je apstraktna pri modeliranju umjetnih neurona. Oni se u osnovi sastoje od ulaza (poput sinapsi), koji se množe s težinama (jakost odgovarajućih signala), a zatim se izračunavaju matematičkom funkcijom koja određuje aktivaciju neurona. Druga funkcija (koja može biti identitet) izračunava izlaz umjetnog neurona (ponekad ovisno o određenom pragu). Što je veća težina umjetnog neurona, jači će biti ulaz koji se množi s njim. Težine također mogu biti negativne, pa možemo reći da je signal inhibiran negativnom težinom. Ovisno o težini, proračun neurona bit će različit. Podešavanjem težina umjetnog neurona možemo dobiti željeni izlaz za određene ulaze. No, kad imamo umjetnu neuronsku mrežu od stotina ili tisuća neurona, bilo bi prilično komplicirano pronaći ručno sve potrebne utege. Ali možemo pronaći algoritme koji mogu prilagoditi težine umjetnih neuronskih mreža kako bi dobili željeni izlaz iz mreže. Taj proces prilagođavanja utega naziva se učenje ili trening (Gershenson, 2003).

Slika 13. Jednostavni umjetni neuron



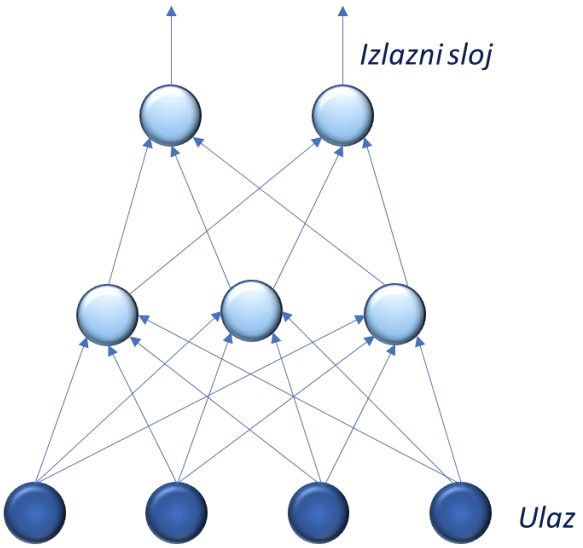
Izvor: izrada autora prema: Gurney (1997).

Na Slici 14. prikazan je primjer jednostavne neuronske mreže. Čvorovi su raspoređeni u slojevitu strukturu u kojoj svaki signal izlazi iz ulaza i prolazi kroz dva čvora prije nego što

dosegne izlaz izvan kojega se više ne transformira. Ova napredna struktura samo je jedna od nekoliko dostupnih i obično se koristi za postavljanje ulaznog uzorka u jednu od nekoliko klasa prema rezultirajućem uzorku izlaza. U stvarnim neuronima sinaptičke snage mogu se pod određenim okolnostima, primijeniti tako da se ponašanje svakog neurona može primijeniti ili prilagoditi njegovom posebnom unosu simulansa. U umjetnim neuronima ekvivalent tome je promjena vrijednosti utega. Što se tiče obrade informacija, ne postoje računalni programi –„znanje“ koje bi trebalo biti pohranjeno u njezinim težnjama, koje se razvijaju procesom prilagodbe na poticaj iz niza promjera uzorka (Gurney, 1997).

U strojnom učenju, neuronskim mrežama, vektorskim strojevima za podršku i evolucijskim proračunima obično nam se daje set za obuku i testni skup. Pod skupom obuke to će značiti sjedinjenje označenog skupa i neoznačenog skupa primjera koji su dostupni učenicima strojeva (Mohri, Rostamizadeh, Talwalkar, 2018).

Slika 14. Jednostavni primjer neuronske mreže



Izvor: izrada autora prema: Gurney (1997).

6.1.1 Aktivacijske funkcije u neuronskim mrežama

Funkcije aktivacije se koriste u umjetnim neuronskim mrežama za pretvaranje ulaznog signala u izlazni signal koji se zatim dovodi kao ulaz u sljedeći sloj u hrpi. U umjetnoj neuronskoj mreži izračunava se zbroj proizvoda inputa i njihove odgovarajuće težine i konačno primjenjivanje funkcije aktivacije da bi se dobio izlaz tog određenog sloja i isporučio se kao izlaz sljedećem sloju (Sharma, Sharma i Athaiya, 2017).

Točnost predviđanja neuronske mreže ovisi o broju slojeva koji se koriste i što je još važnije o vrsti upotrijebljene aktivacijske funkcije. Kada imamo aktivacijsku funkciju, najvažnije je uzeti u obzir klasifikator temeljen na pragu, što znači da bez obzira mora li vrijednost linearne transformacije aktivirati neuron ili ne, ili možemo reći da se neuron aktivira ako je ulaz u aktivacijsku funkciju veći od vrijednosti praga ili će se u protivnom deaktivirati. U tom slučaju izlaz se ne dovodi kao ulaz na sljedeći sloj (Sharma, Sharma i Athaiya, 2017).

6.1.1.1 Funkcija binarnog koraka

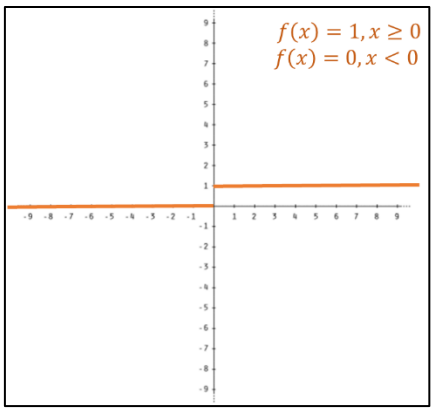
Funkcija binarnog koraka (*eng. Binary Step Function*) je najjednostavnija aktivacijska funkcija koja postoji, može se implementirati jednostavnim if-else izrazima u Pythonu. Prilikom stvaranja binarnog klasifikatora općenito se koriste binarne aktivacijske funkcije. Funkcija binarnog koraka ne može se koristiti u slučaju klasifikacije više klasa. Također, gradijent binarne funkcije koraka je nula što može uzrokovati smetnje u koraku privatnog širenja, tj. ako izračunamo derivaciju *f(x)* u odnosu na *x,* jednaka je nuli. Matematički funkcija binarnog koraka može se definirati kao (Sharma, Sharma i Athaiya, 2017):

$$\boldsymbol{f(x) = 1, x \geq 0}$$

$$\boldsymbol{f(x) = 0, x < 0}$$

Funkcija binarnog koraka se obično koristi u primitivnim neuronskim mrežama bez skrivenog sloja, koje se još nazivaju jednoslojni perceptroni (Serengil, 2020). Graf funkcije binarnog skoka prikazan je na Slici 15.

Slika 15. Funkcija binarnog koraka



Izvor: izrada autora prema: Sharma, Sharma i Athaiya (2017).

6.1.1.2 Linearna aktivacijska funkcija

Funkcija linearnog aktiviranja (*eng. Linear activation function*) izravno je proporcionalna ulazu. Glavni nedostatak binarne funkcije koraka je što ima nulti gradijent jer nema komponente *x* u binarnoj funkciji koraka. Kako bi se ovaj problem uklonio, koristi se linearna funkcija. Ona se definira sljedećom formulom:

$$\boldsymbol{f(x) = ax}$$

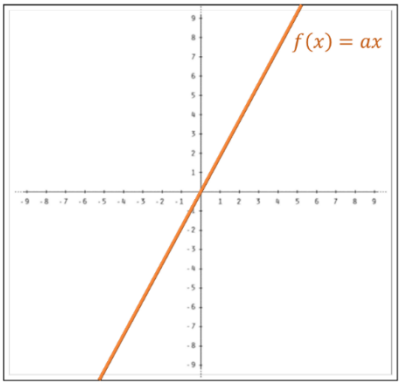
Derivacija funkcije *f(x)* nije nula, već je jednaka vrijednosti korištene konstante. Gradijent nije nula, već je konstantna vrijednost koja je neovisna o ulaznoj vrijednosti *x*, što znači da će se ponderi i pristranosti ažurirati tijekom koraka unatrag širenja iako će faktor ažuriranja ostati isti (Sharma, Sharma i Athaiya, 2017).

Vrijednost varijable a može biti bilo koja konstanta vrijednost po izboru korisnika. No linearna aktivacijska funkcija ima dva glavna nedostatka (Little, 2020):

1) Nije moguće koristiti prostiranje unatrag (gradijalni silazak) za uvježbavanje modela- derivacija funkcije je konstanta i nema veze s ulazom, x. Nije moguće vratiti se i promijeniti težine u ulaznim neuronima, što može dati bolje predviđanje.
2) Svi slojevi neuronske mreže urušavaju se u jedan sloj – s linearnim aktivacijskim funkcijama, bez obzira koliko slojeva u neuronskoj mreži ima, posljednji sloj bit će linearna funkcija prvog sloja. Dakle, funkcija linearne aktivacije pretvara neuronsku mrežu, koja sadrži samo jedan sloj. Neuronska mreža s linearnom aktivacijom jednostavno je linearni regresijski model.

Linearne aktivacijske funkcije idealne su tamo gdje je potrebna interpretacija za jednostavnije zadatke (Sharma, Sharma i Athaiya, 2017). Na Slici 16. prikazan je primjer linearne aktivacijske funkcije.

Slika 16. Linearna aktivacijska funkcija



Izvor: izrada autora prema: Sharma, Sharma i Athaiya (2017).

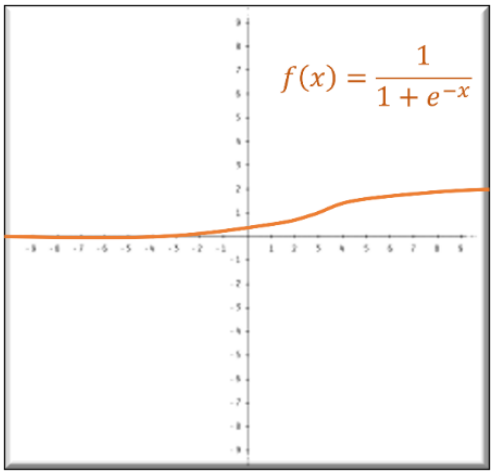
6.1.1.3 Sigmoidna aktivacijska funkcija

Sigmoidna aktivacijska funkcija (*eng. Sigmoid activation function*) još se naziva i logistička funkcija, tradicionalno je vrlo poznata funkcija aktivacije za neuronske mreže. Riječ je o najčešće korištenoj funkciji aktivacije, glavni razlog njezine popularnosti je činjenica da je riječ o nelinearnoj funkciji. Sigmoidna funkcija transformira vrijednosti u rasponu od 0 do 1. Definira se sljedećom formulom:

$$\boldsymbol{f(x) = \frac{1}{e^{-x}}}$$

Na Slici 17. prikazan je graf Sigmoidne aktivacijske funkcije.

Slika 17. Sigmoidna aktivacijska funkcija



Izvor: izrada autora prema: Sharma, Sharma i Athaiya (2017).

6.1.1.4 ReLu

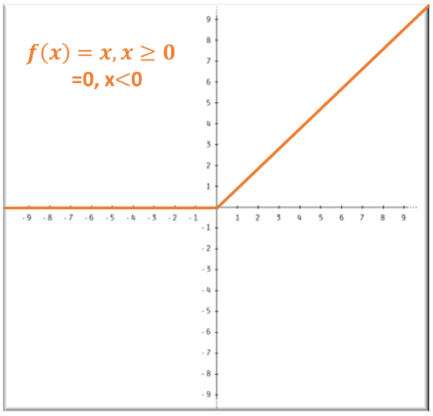
ReLU označava ispravljenu linearnu jedinicu, ona je nelinearna je aktivacijska funkcija koja se široko koristi u neuronskim mrežama.

Prednost korištenja ReLU funkcije je to da se svi neuroni se ne aktiviraju u isto vrijeme. Iz čega slijedi da će neuron biti deaktiviran tek kada je izlaz linearne transformacije nula. Može se prikazati matematičkom formulom:

$$\boldsymbol{f(x) = max(0, x)}$$

Na Slici 18. prikazan je graf ReLu aktivacijske funkcije.

Slika 18. ReLu aktivacijska funkcija



Izvor: izrada autora prema: Sharma, Sharma i Athaiya (2017).

6.1.1.5 Tanh aktivacijska funkcija

Tanh funkcija je samo još jedna moguća funkcija koja se može koristiti kao nelinearna aktivacijska funkcija između slojeva neuronske mreže. Najveća prednost tanh funkcije je ta što proizvodi nultocentrirani izlaz, čime se podržava proces povratnog širenja (Goyal, 2020).

Tanh funkcija je kontinuirana, vrijednosti se nalaze u rasponu od -1 do 1. U usporedbi sa Sigmoidnom funkcijom, gradijent Tanh funkcije je strmiji. Tanh je preferiran u odnosu na Sigmoidnu funkciju jer ima gradijente koji nisu ograničeni na promjenu u određenom smjeru, a također je nulti centar. Tanh funkcija uglavnom se koristi u ponavljajućim neuronskim mrežama za obradu prirodnog jezika i zadatke prepoznavanja govora.

Tanh aktivacijska funkcija se može prikazati sljedećom formulom (Nwankpa, Ijomah, Gachagan i Marshall, 2018):

$$\boldsymbol{f(x) = 2\,sigmoid(2x) - 1}$$

6.1.1.6 Softmax

Softmax funkcija kombinacija je više Sigmoidnih funkcija. Kako znamo da Sigmoidna funkcija vraća vrijednosti u rasponu od 0 do 1, to se može tretirati kao vjerojatnost podatkovnih točaka određene klase. Funkcija Softmax koristi se u modelima s više klasa gdje vraća vjerojatnosti svake klase, pri čemu ciljna klasa ima najveću vjerojatnost. Glavna razlika između Sigmoida i Softmax aktivacijske funkcije je ta što se Sigmoid koristi u binarnoj klasifikaciji, dok se Softmax se koristi za multivarijantne klasifikacijske zadatke (Nwankpa, Ijomah, Gachagan i Marshall, 2018).

Može se izraziti sljedećom formulom (Nwankpa, Ijomah, Gachagan i Marshall, 2018):

$$\theta(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \text{, za } j = 1, \ldots, K$$

7 Konvolucijske neuronske mreže

Konvolucijska neuronska mreža (*eng. Convolutional Neural Network - CNN*) imala je revolucionarne rezultate u proteklom desetljeću u raznim područjima vezanim za prepoznavanje uzoraka; od obrade slike do prepoznavanja glasa (Albawi, Mohammed i Al-Zawi, 2017). Konvolucijska neuronska mreža je vrsta umjetne neuronske mreže koja potječe iz neuroznanosti, koja datira od prijedloga prvog umjetnog neurona 1943. godine (Liang i Hu, 2017).

Koncepcijski, konvolucijska neuronska mreža nalikuje višeslojnom perceptronu (*eng. Multilayer perceptron- MLP).* Svaki pojedini neuron u MLP -u ima aktivacijsku funkciju koja preslikava ponderirane ulaze na izlaz. Višeslojni perceptron postaje duboki višeslojni perceptron kada se mreži doda više skrivenih slojeva (Zhao, Lu, Chen, Liu i Wu, 2017). Konvolucijske neuronske mreže su prvi doista uspješan pristup dubokog učenja gdje su slojevi hijerarhije uspješno obučeni na robustan način. One su izbor arhitekture koja koristi prostorne i vremenske odnose kako bi smanjilo broj parametara koji se moraju naučiti i na taj način poboljšava opću obuku unaprijeđenog širenja unaprijed (Mishra i Gupta, 2017).

Ova tehnika ilustrira koliko je poboljšanje dubokih slojeva značajno za obradu informacija. Konvolucijske neuronske mreže su stara tehnika, koja je razvijena 1980 -ih i 1990 -ih (Kim, 2017). Konvolucijske neuronske mreže prvi su predložili (LeCun, Boser, Denker, Henderson, Howard, Hubbard i Jackel (1990). u radu „Handwritten Digit Recognition with a Back-Propagation Network“ gdje su predstavili su primjenu mreža za širenje unatrag za prepoznavanje rukom pisanih znamenki. U osnovi se konvolucijska neuronska mreža se razlikuje od Neokognitirona ugrađivanjem algoritma za širenje unatrag za učenje receptivnih polja jednostavnih jedinica (Liang i Hu, 2015).

Konvolucijska neuronska mreža nije samo duboka neuronska mreža koja ima mnogo skrivenih slojeva. To je duboka mreža koja oponaša način na koji vizualni korteks mozga obrađuje i prepoznaje slike (Kim, 2017). Kao i drugi hijerarhijski modeli, uključujući Neocognitron i HMAX , usko su povezani s nalazima Hubela i Wiesela o jednostavnim stanicama i složenim stanicama u primarnom vidnom korteksu (Liang i Hu, 2015). Konvolucijske neuronske mreže od svog začetka karakteriziraju lokalne veze, dijeljenje

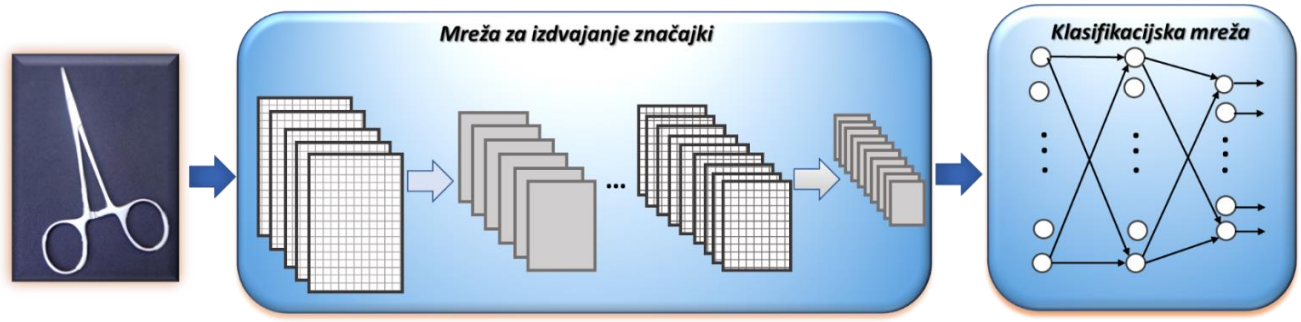
težine i lokalno udruživanje. Prva dva svojstva omogućuju modelu otkrivanje lokalnih informativnih vizualnih uzoraka s manje prilagodljivih parametara od višeslojnog perceptrona. Treće svojstvo oprema mrežu s nekom prevoditeljskom invarijantnošću (Liang i Hu, 2015).

Konvolucijske neuronske mreže su nedavno potaknula dramatičan napredak u prepoznavanju slika zbog njihove sposobnosti da prilagodljivo nauče značajke klasifikacije, umjesto da se oslanjaju na značajke koje je odabrao čovjek. Ove se značajke izdvajaju iz slike putem skupa konvolucijskih filtera čiji se koeficijenti uče tehnikom poznatom kao prostiranje unatrag, a zatim se agregiraju pomoću operacije poznate kao udruživanje (Bayar i Stamm, 2016).

Konvolucijske neuronske mreže imaju izvrsne performanse u problemima strojnog učenja. Posebno u aplikacijama koje se bave slikovnim podacima, poput najvećeg skupa podataka o klasifikaciji slika (Image Net) i računalnog vida i obrade prirodnog jezika (*eng. Natural language processing- NLP)* (Albawi, Mohammed i Al-Zawi, 2017). U osnovi, CNN - ovi se sastoje od nekoliko različitih vrsta slojeva koji su međusobno povezani (Xie, Du, Li, Liang, Tang, Ong i Gosh, 2018).

Konvolucijska mreža se sastoji od neuronske mreže koja izdvaja značajke ulazne slike i druge neuronske mreže koja klasificira tu značajku slika. Slika 19. prikazuje tipičnu arhitekturu Konvolucijske neuronske mreže. Ulazna slika ulazi u mrežu za izdvajanje značajki. Izvučeni signali značajki ulaze u klasifikacijsku neuronsku mrežu. Klasifikacijska neuronska mreža tada djeluje na temelju značajki slike i generira izlaz. Neuronska mreža za izdvajanje značajki sastoji se od hrpa slojevitog sloja i parova slojeva za sažimanje. Sloj savijanja, kako mu i naziv govori, pretvara sliku pomoću operacije savijanja. Može se smatrati zbirkom digitalnih filtera. Skupni sloj kombinira susjedne piksele u jedan piksel. Stoga skupni sloj smanjuje dimenziju slike (Kim, 2017).

Slika 19. Tipična arhitektura Konvolucijske mreže

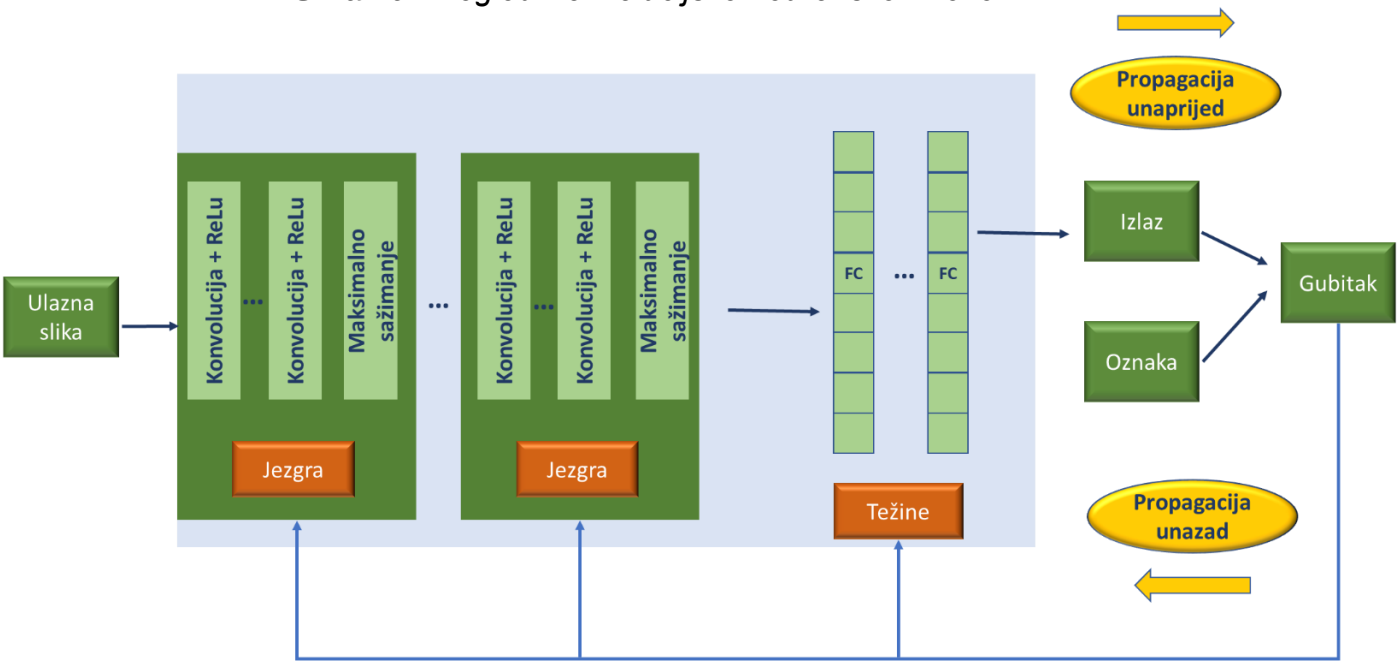


Izvor: izrada autora prema: Kim (2017)

7.1 Konvolucijski elementi neuronske mreže

Postoje četiri vrste slojeva za konvolucijsku neuronsku mrežu: konvolucijski sloj, sloj sažimanja, korekcijski sloj i potpuno povezani sloj. Tipična arhitektura sastoji se od ponavljanja hrpe od nekoliko slojeva konvolucije i sloja za sažimanje, nakon čega slijedi jedan ili više potpuno povezanih slojeva. Korak u kojem se ulazni podaci pretvaraju u izlazne kroz navedene slojeve naziva se promicanje prema naprijed (Yamashita, Nishio, Gian Do i Togashi, 2018). Na Slici 20. prikazan je pregled konvolucijske neuronske mreže, Učinkovitost modela pod određenim jezgrama i težinama izračunava se pomoću funkcije gubitka putem širenja unaprijed, u skupu podataka za obuku, a parametri za učenje, tj. jezgre i ponderi, ažuriraju se u skladu s vrijednošću gubitka putem širenja unatrag uz algoritam optimizacije gradijentnog spuštanja.

Slika 20. Pregled Konvolucijske neuronske mreže



Izvor: izrada autora prema: Yamashita, Nishio, Gian Do i Togashi (2018).

7.1.1 Konvolucijski sloj

Konvolucijski sloj je temeljna komponenta arhitekture konvolucijske neuronske mreže, koja izvodi ekstrakciju značajki, koja se tipično sastoji od kombinacije linearnih i nelinearnih operacija, tj. operacije konvolucije i aktiviranja (Yamashita, Nishio, Gian Do i Togashi, 2018).

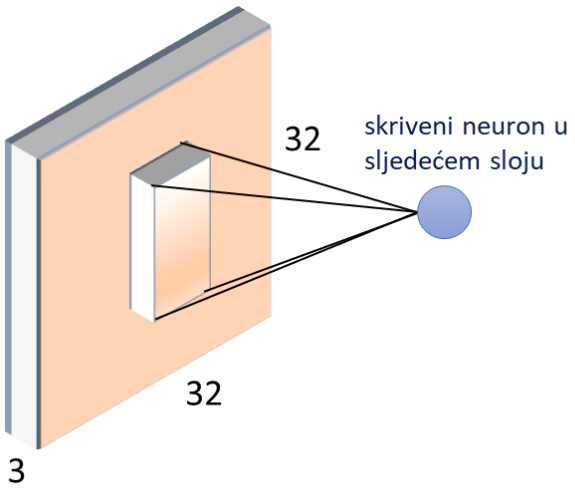
Sloj konvolucije generira nove slike koje se nazivaju mape značajki. Karta značajki naglašava jedinstvene značajke izvorne slike. Konvolucijski sloj djeluje na vrlo različit način

u usporedbi s drugim slojevima neuronske mreže. Ovaj sloj ne koristi težine povezivanja i ponderirani zbroj. Umjesto toga, sadrži filtre koji pretvaraju slike (Kim, 2017).

Konvolucija je specijalizirana vrsta linearne operacije koja se koristi za izdvajanje značajki, gdje se mali niz brojeva, nazvan jezgra, primjenjuje na ulaz, što je niz brojeva, koji se naziva tenzor. Elementni umnožak između svakog elementa jezgre i ulaznog tenzora izračunava se na svakom mjestu tenzora i zbraja kako bi se dobila izlazna vrijednost u odgovarajućoj poziciji izlaznog tenzora, nazvana karta značajki (Yamashita, Nishio, Gian Do i Togashi, 2018).

Slika 21. prikazuje regionalnu vezu za sljedeći sloj. Skriveni neuroni u sljedećem sloju dobivaju samo ulaze iz odgovarajućeg dijela prethodnog sloja.

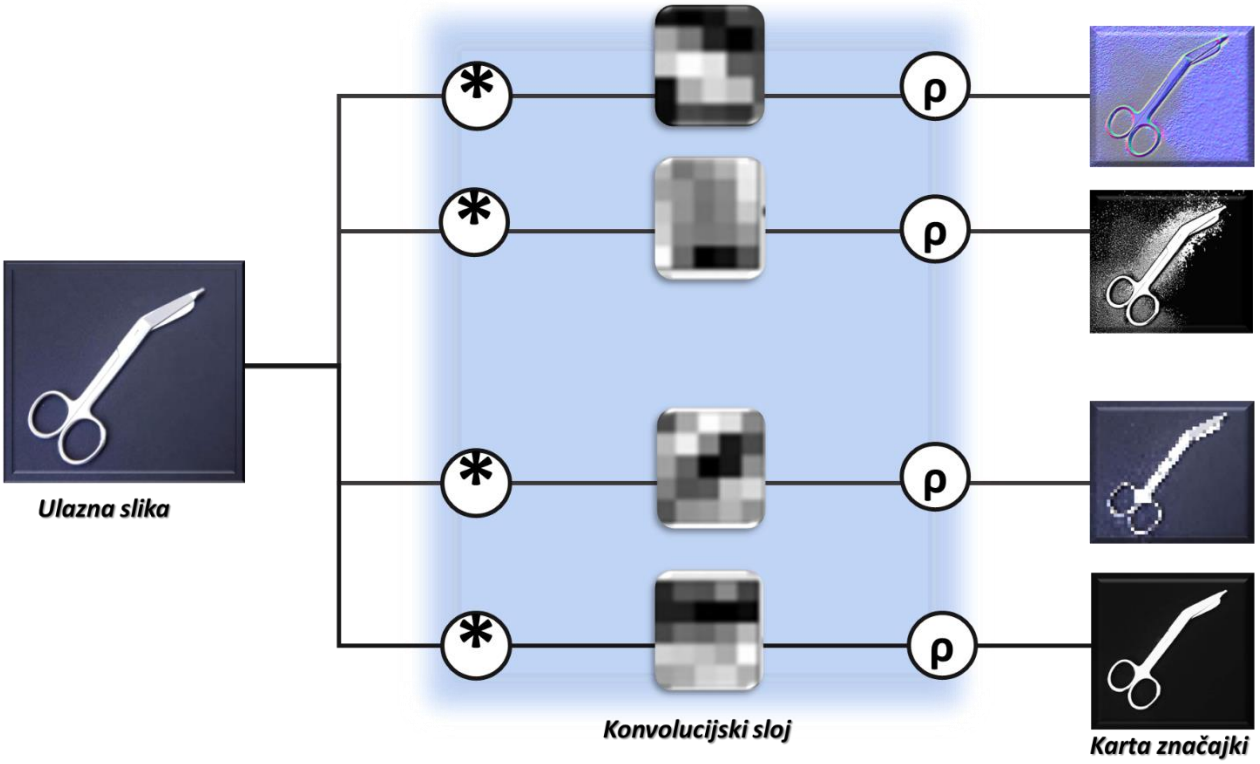
Slika 21. Konvolucija kao alternativa za potpuno povezanu mrežu.



Izvor: izrada autora prema: Albawi, Mohammed i Al-Zawi, (2017).

Slika 22. prikazuje postupak sloja savijanja, gdje zaokružena oznaka * označava radnju savijanja, a oznaka ρ je funkcija aktiviranja. Ikone u sivim tonovima između ovih operatora označavaju filtre za konvoluciju. Sloj konvolucije generira isti broj karata značajki kao i filteri konvolucije. Slika 22. prikazuje vrijednosti 5 x 5 filtera u pikselima sive boje. Vrijednosti matrice filtra određuju se kroz proces obuke. Stoga se te vrijednosti kontinuirano usavršavaju tijekom cijelog procesa obuke.

Slika 22. Postupak sloja konvolucije



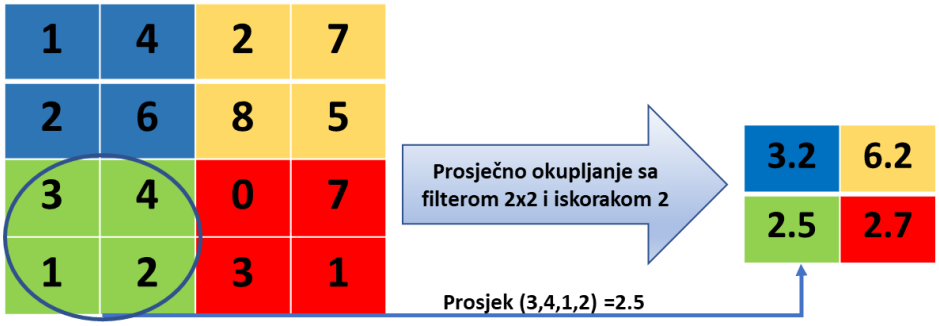
Izvor: izrada autora prema: Kim (2017).

7.1.2 Sloj sažimanja

Sloj sažimanja nudi tipičnu operaciju smanjenja uzorkovanja koja smanjuje dimenzionalnost u ravnini karata značajki kako bi se uvodila invazivna translacija na male pomake i izobličenja te smanjio broj sljedećih parametara koji se mogu naučiti. Potrebno je napomenuti da ne postoji parametar koji se može naučiti ni u jednom sloju spremišta, dok su veličina filtra, korak i padding hiperparametri u operacijama spremanja, slični operacijama konvolucije (Yamashita, Nishio, Gian Do i Togashi, 2018). Svrha sažimanja je pretvorba zajedničkog prikaza značajki u korisniji skup koji čuva važne informacije, a odbacuje nebitne detalje. Upotreba sloja sažimanja u konvolucijskim neuronskim mrežama ima za cilj postići nepromjenjivost na promjene položaja ili uvjete osvjetljenja, robusnost i kompaktnost prikaza. Općenito, skupni sloj sažima izlaze susjednih skupina neurona na istoj karti jezgre (Yu, Wang, Chen i Wei, 2014). Ovaj sloj postiže bolju generalizaciju, bržu konvergenciju, robustan na translaciju i izobličenja i obično se postavlja između konvolucijskih slojeva (Coskun, Ucar, Yildirim i Demir, 2017). Postoje više vrsta sažimanja: prosječno sažimanje, maksimalno sažimanje, mješovito sažimanje, Lp sažimanje, stohastičko, sažimanje prostornih piramida i okupljanje regija interesa (Gholamalinezhad i Khosravi, 2020).

Na Slici 23. prikazan je primjer operacije prosječnog sažimanja, to je prva duboka neuronska mreža temeljena na konvoluciji. Kao što se vidi sa Slike 23., prosječni sloj sažimanja, izvodi smanjenje uzrokovanja dijeljenjem ulaza u pravokutna područja okupljanja izračunavanjem prosječnih vrijednosti svakog područja (Gholamalinezhad i Khosravi, 2020).

Slika 23. Primjer operacije prosječnog okupljanja



Izvor: izrada autora prema: Gholamalinezhad i Khosravi (2020).

7.1.3 Potpuno povezani sloj

Izraz potpuno povezan sloj (*eng. Fully connected layer - FCL*) odnosi se na to da je svaki filtar u prethodnom sloju povezan sa svakim filterom u sljedećem sloju.

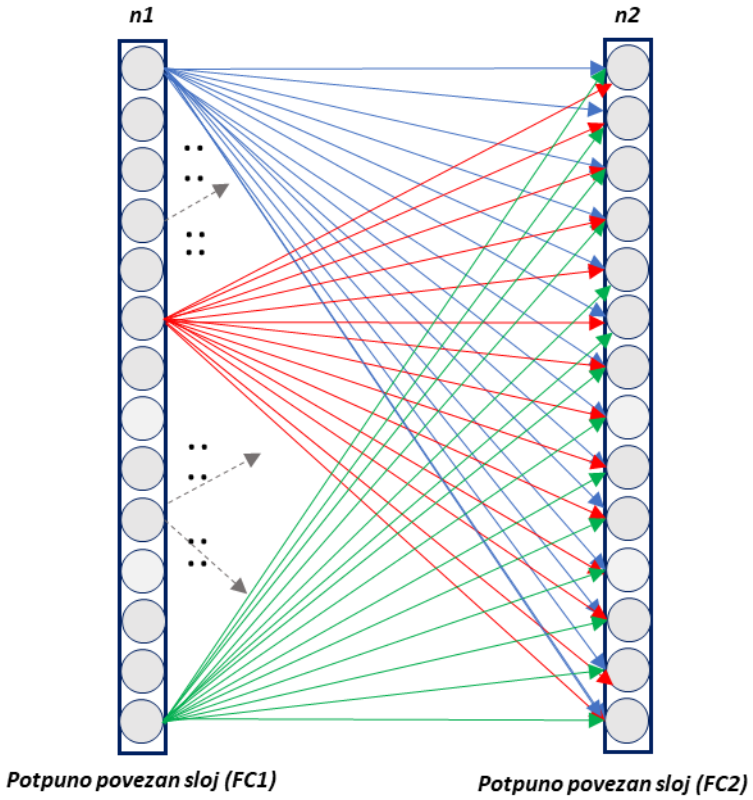
Cilj korištenja potpuno povezanog sloja je upotrijebiti značajke za klasifikaciju ulazne slike u različite klase na temelju skupa podataka za obuku (Coskun, Ucar, Yildirim i Demir, 2017). Na Slici 24. prikazana je mreža dva potpuno povezana sloja s n_1 i n_2 neurona u svakom sloju. Dva sloja označena su kao FC_1 i FC_2 gdje je $x \in R^{n_1 x\, 1}$ (Ma i Lu, 2017).

Potpuno povezani slojevi su vrsta napredne umjetne neuronske mreže i slijede načelo tradicionalne višeslojne perceptronske neuronske mreže (MLP). Potpuno povezani slojevi uzimaju ulaz iz konačnog konvolucijskog ili skupnog sloja, koji je u obliku skupa metrika (mapa značajki), a ti se metrički poravnavaju kako bi se stvorio vektor, a taj se vektor zatim dovodi u potpuno povezani sloj kako bi se generirao konačni izlaz konvolucijske neuronske mreže (Ghosh, Sufian, Sultana i Chakrabarti, 2019).

Mape izlaznih značajki konačne konvolucije obično su spljoštene, tj. transformirane u jednodimenzionalni (1D) niz brojeva i povezane s jednim ili više potpuno povezanih slojeva, također poznatim kao gusti slojevi, u koji je svaki ulaz povezan uočljiva težina. Nakon što se stvore značajke izdvojene slojevima konvolucije i smanjene uzorke slojeva za združivanje, one se preslikavaju podskupom potpuno povezanih slojeva u konačne izlaze mreže, poput

vjerojatnosti za svaku klasu u zadacima klasifikacije (Yamashita, Nishio, Gian Do i Togashi, 2018).

Slika 24. Primjer potpuno povezanog sloja



Izvor: *izrada autora prema:* Ma, Lu (2017).

7.2 Funkcija gubitka

Funkcija koju želimo minimizirati ili maksimizirati naziva se ciljna funkcija ili kriterij. Kad ga minimiziramo, možemo ga nazvati i funkcijom troška, funkcijom gubitka ili funkcijom pogreške (Goodfellow, Bengio i Courville, 2016).

U izlaznom sloju izračunavamo pogrešku predviđanja koju generira model konvolucijske neuronske mreže na podacima za obuku pomoću neke funkcije gubitka. Ova pogreška predviđanja govori mreži kako je njihovo predviđanje od stvarnog izlaza, a zatim će se ta pogreška optimizirati tijekom procesa učenja modela konvolucijske neuronske mreže (Ghosh, Sufian, Sultana i Chakrabarti, 2019).

Funkcija gubitka koristi dva parametra za izračun pogreške, prvi parametar je procjena rezultata modela konvolucijske neuronske mreže (koji se naziva i predviđanje), a drugi je stvarni izlaz (poznat i kao oznaka) (Ghosh, Sufian, Sultana i Chakrabarti, 2019). U različitim

vrstama problema koriste se različite vrste funkcija gubitka. Neke od najčešće korištenih funkcija gubitka ukratko su opisane u nastavku. Uobičajeno korištena funkcija gubitka za klasifikaciju više klasa je unakrsna entropija, dok se srednja kvadratna pogreška tipično primjenjuje na regresiju na kontinuirane vrijednosti (Yamashita, Nishio, Gian Do i Togashi, 2018).

Unakrsna entropija za funkciju gubitka funkcije Softmax

Gubitak unakrsne entropije, koji se naziva i funkcija gubitka, naširoko se koristi za mjerenje performansi modela konvolucijske neuronske mreže, čiji je izlaz vjerojatnost $p \in \{0,1\}$. Široko se koristi kao alternativa kvadratne funkcije gubitka pogreške u klasifikacijskim problemima više klasa. Koristi softmax aktivacije u izlaznom sloju za generiranje izlaza unutar raspodjele vjerojatnosti $p, \mathrm{y} \in \mathrm{R^N}$ gdje je p vjerojatnost za svaku izlaznu kategoriju, a y označava željeni izlaz i vjerojatnost svake izlazne klase može se dobiti na sljedeći način (Ghosh, Sufian, Sultana i Chakrabarti, 2019):

$$p_i = \frac{e^{a_i}}{\sum_{k=1}^{N} e^{a}{}_{k}}$$

gdje je *N* broj neurona u izlaznom sloju i e a_i označava svaki normalizirani izlaz s prethodnog sloja u mreži.

Euklidska funkcija gubitka

Euklidov gubitak koji se naziva i pogreška srednjeg kvadrata, naširoko se koristi u regresijskim problemima. Srednja kvadratna pogreška nalazi se između predviđenog izlaza $p \in R^N$ i stvarnog izlaza $Y \in R^N$ svaki neuron izlaznog sloja konvolucijske neuronske mreže definiran je kao $H(p, y) = (p - y)^2$ako postoji N neurona u izlaznom sloju, tada se procjena euklidskog gubitka definira kao (Ghosh, Sufian, Sultana i Chakrabarti, 2019):

$$H(p, y) = \frac{1}{2N} \sum_{i=1}^{N} (p_i - y_i)^2$$

7.2.1 Gradijentno spuštanje

Gradijentno spuštanje jedan je od najpopularnijih algoritama za izvođenje optimizacije i daleko najčešći način optimizacije neuronskih mreža (Ruder, 2016). Gradijentno spuštanje obično se koristi kao optimizacijski algoritam koji iterativno ažurira parametre koji se mogu naučiti, tj. Jezgre i težine, mreže kako bi se smanjili gubici. Gradijent funkcije gubitka daje smjer u kojem funkcija ima najveću stopu povećanja, a svaki parametar

koji se može naučiti ažurira se u negativnom smjeru gradijenta s proizvoljnom veličinom koraka određenom na temelju hiperparametra koji se naziva brzina učenja.

Matematički, djelomična izvedenica gubitka u odnosu na svaki parametar koji se može naučiti, a jedno ažuriranje parametra formulirano je kako slijedi (Yamashita, Nishio, Gian Do i Togashi, 2018):

$$w: = w - \alpha * \frac{\vartheta L}{\vartheta w}$$

gdje *w* označava svaki parametar koji se može naučiti, α označava brzinu učenja, a L označava funkciju gubitka. Potrebno je napomenuti da je u praksi brzina učenja jedan od najvažnijih hiperparametara koji se postavlja prije početka treninga. U praksi se iz razloga kao što su ograničenja memorije izračunavaju gradijenti funkcije gubitka s obzirom na parametre. Pomoću podskupa skupa podataka za obuku koji se naziva mini-serija i primjenjuje se na ažuriranje parametara. Ova metoda naziva se mini-serija gradijentnog spuštanja, često se naziva i stohastičko gradijentno spuštanje, a veličina male serije je također hiperparametar (Yamashita, Nishio, Gian Do i Togashi, 2018).

7.3 Primjeri arhitekture Konvolucijskih neuronskih mreža

7.3.1 Klasifikacija slika

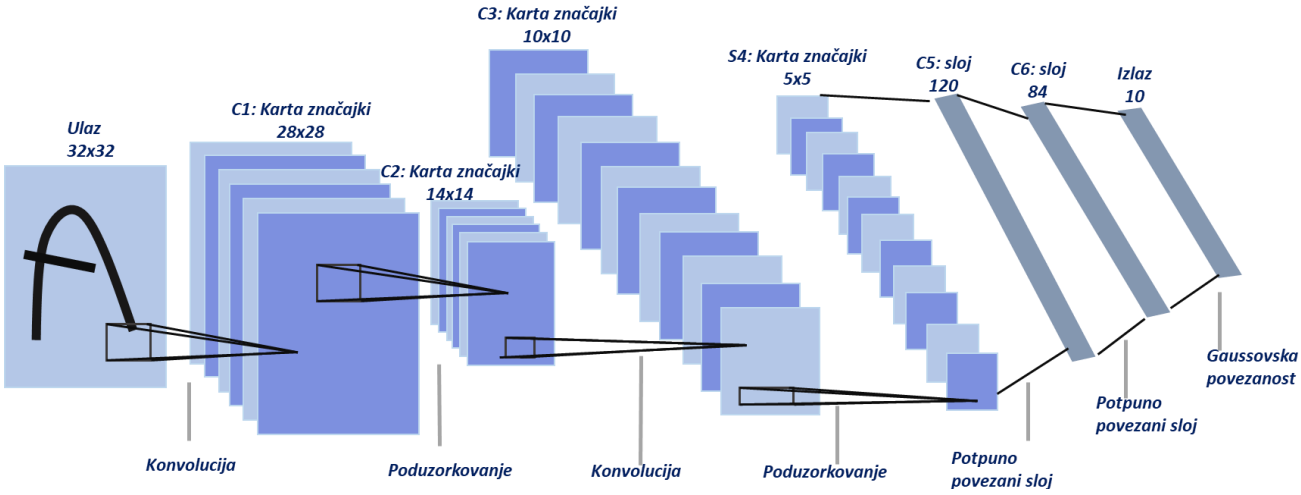
U klasifikaciji slika pretpostavljamo da ulazna slika sadrži jedan objekt, a zatim je sliku potrebno klasificirati u jednu od unaprijed odabranih ciljnih klasa pomoću modela konvolucijske neuronske mrež (Ghosh, Sufian, Sultana i Chakrabarti, 2019). U klasifikaciji slike svaka slika ima glavni objekt koji zauzima veliki dio slike. Slika se razvrstava u jednu od klasa na temelju identiteta njenog glavnog objekta (Wu, 2017).

7.3.1.1 LeNet-5

LeNET-5 su predstavili LeCun, Bottou, Bengio i Haffner (1998). u radu „*Učenje zasnovano na gradijentima namijenjeno za prepoznavanje dokumenata*“. LeNET-5 je bila jedna od prvih konvolucijskih primjena obrade slika. LeNet ima 5 ponderiranih slojeva, 3 sloja sažimanja i 2 potpuno povezana sloja (Ghosh, Sufian, Sultana i Chakrabarti, 2019). Cilj LenNet-5 bio je klasificirati slike znamenki u sivim tonovima iz skupa podataka MNIST (Modificirani nacionalni institut za standarde i tehnološku bazu podataka) (Agarwal, 2020). Prvi sloj je ulazni sloj - to se općenito ne smatra slojem mreže jer se u ovom sloju ništa ne

nauči. Ulazni sloj napravljen je tako da prima 32x32, a to su dimenzije slika koje se prenose u sljedeći sloj. Arhitektura LeNet-5 koristi dvije značajne vrste konstrukcija slojeva: konvolucijski slojevi i slojevi poduzorkovanja. Kako bi dimenzija MNIST slika zadovoljila zahtjeve ulaznog sloja, slike 28x28 su podstavljene (Alake, 2020). Na Slici 25. prikazana je arhitektura LeNet-5.

Slika 25. Arhitektura LeNet-5

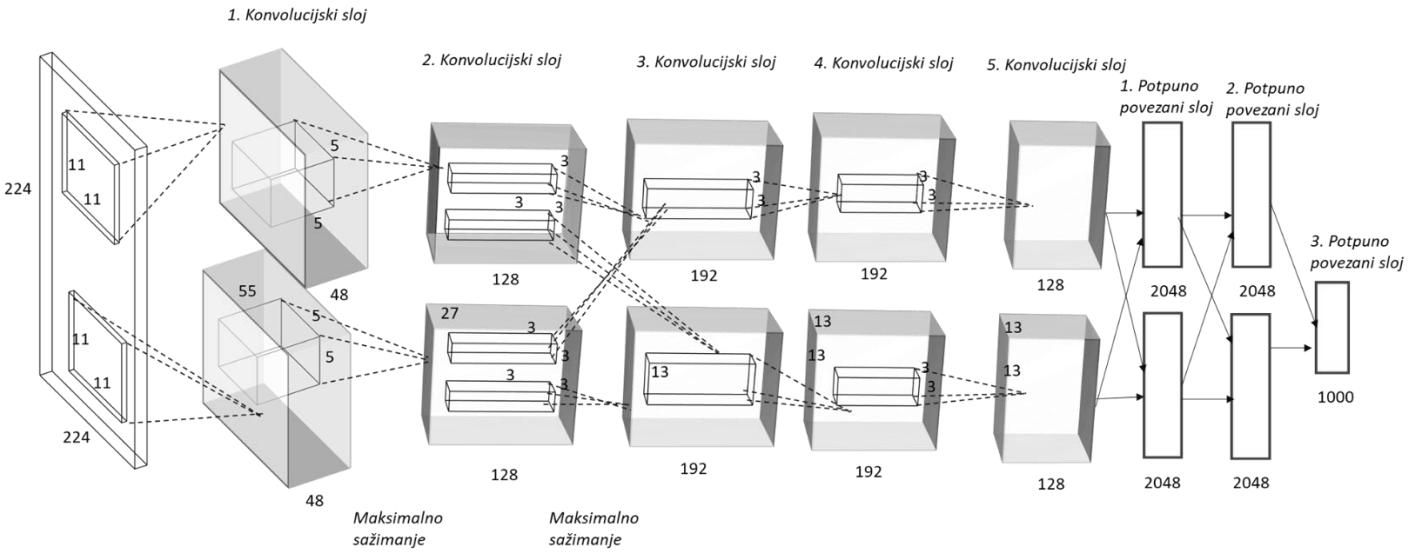


Izvor: izrada autora prema: LeCun, Bottou, Bengio i Haffner (1998).

7.3.1.2 AlexNet

Krizhevsky, Sutskever i E. Hinton (2012). dizajnirali su model konvolucijske neuronske mreže nazvan AlexNet, koji je osmišljen za klasifikaciju ImageNet podataka. AlexNet se sastoji od osam ponderiranih slojeva među kojima su prvih pet konvolucijski, a nakon toga tri potpuno povezana sloja. AlexNet maksimizira cilj multinomijalne logističke regresije, što je ekvivalentno maksimiziranju prosjeka za sve slučajeve treninga log-vjerojatnosti ispravne oznake prema distribuciji predviđanja. AlexNet koristi funkciju aktiviranja nelinearnosti s ispravljenom linearnom jedinicom (ReLU) nakon svakog konvolucijskog i potpuno povezanog sloja. Na Slici 26. prikazana je arhitektura AlexNeta.

Slika 26. Arhitektura AlexNet

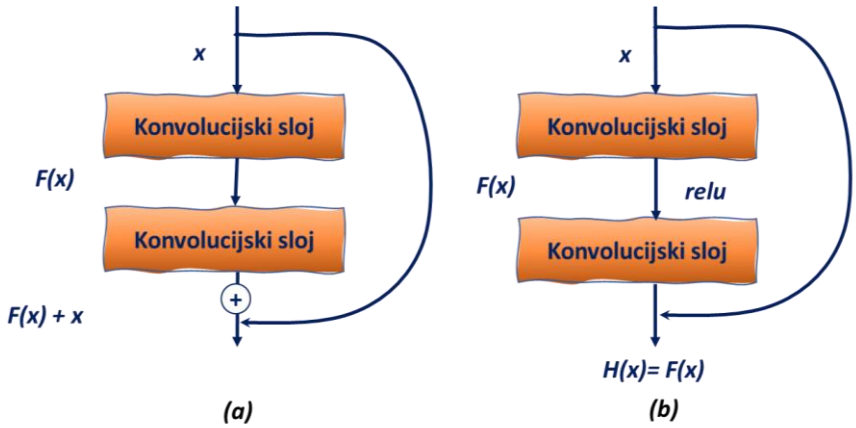


Izvor: izrada autora prema: Krizhevsky, Sutskever i E. Hinton (2012).

7.3.1.3 ResNet

Budući da duboki model konvolucijske neuronske mreže pati od problema nestajanja gradijenta, He, Zhang, Ren i Sun (2016). predložili su „veze za preskakanje identiteta“ za rješavanje nestajućeg gradijenta problem predlažući model ResNet, za olakšavanje obuke mreža koje su znatno dublje od onih koje su se ranije koristile. Arhitektura ResNeta koristi rezidualno preslikavanje, umjesto da uči izravno preslikavanje. Cjelovita ResNet arhitektura sastoji se od mnogih rezidualnih ploča sa 3 × 3 sloja zavoja. Slika 27. ilustrira razliku između izravnog preslikavanja i rezidualnog preslikavanja (Ghosh, Sufian, Sultana i Chakrabarti, 2019).

Slika 27. (a) Mapiranje unutar Rezidualnog bloka, (b) Jednostavna izravna preslikavanja.



Izvor: *izrada autora prema:* Ghosh, Sufian, Sultana i Chakrabarti (2019).

7.3.1.4 GoogLeNet

GoogLeNet arhitektura se razlikuje od svih prethodno razmatranih konvencionalnih modela konvolucijskih neuronskih mreža. Koristi mrežne grane umjesto korištenja jednoredne sekvencijalne arhitekture. Glavni znak ove arhitekture je poboljšana upotreba računalnih resursa unutar mreže (Szegedy i sur., 2015). To je postignuto pomno izrađenim dizajnom koji omogućuje povećanje dubine i širine mreže uz održavanje proračuna konstantnim. Kako bi se optimizirala kvaliteta, arhitektonske odluke temeljile su se na hebbijskom principu i intuiciji obrade u više razmjera (Szegedy i sur., 2015). GoogleNet ima 22 ponderirana sloja za učenje, koji je koristio "Inception Module" kao osnovni građevinski blok mreže (Ghosh, Sufian, Sultana i Chakrabarti , 2019).

7.3.2 Detekcija objekta

Za razliku od klasifikacije slika, otkrivanje zahtijeva lokalizaciju (mnogo) objekata unutar slike (Girshick, Donahue, Darrell i Malik, 2013). Detekcija objekta pokušava detektirati objekte unutar ulazne slike odgovarajućom identifikacijom svakog objekta zajedno s njihovim ispravnim položajem unutar slike pomoću modela konvolucijske neuronske mreže (Ghosh, Sufian, Sultana i Chakrabarti, 2019). Dakle, otkrivanje objekata u osnovi je instancijalni zadatak računalnog vida. Prije popularnosti dubokog učenja u računalnom vidu, otkrivanje objekata vršeno je ručno izrađenim značajkama strojnog učenja, poput promjene invarijantne značajke pomaka (Sultana, Sufian i Dutta, 2019).

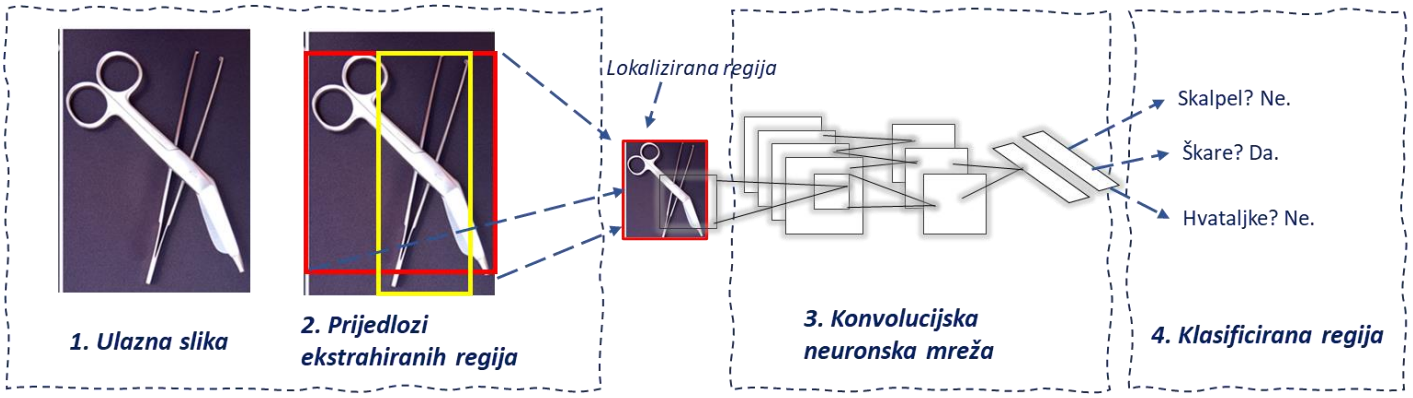
U nastavku su objašnjeni neke od poznatijih arhitektura konvolucijskih neuronskih mreža za detekciju objekta.

7.3.2.1 R-CCC

Prva konvolucijska neuronska mreža dizajnirana za otkrivanje objekata je R-CNN (*Region-based CNN*), koja koristi pristupe temeljene na kliznim prozorima za uspješno otkrivanje objekata (Girshick, Donahue, Darrell i Malik, 2013). R-CNN -ov sustav otkrivanja objekata sastoji se od tri modula. Prvi generira prijedloge regija neovisnih o kategorijama. Ovi prijedlozi definiraju skup otkrivanja objekata koji su dostupni detektoru. Drugi modul je velika konvolucijska neuronska mreža koja izdvaja vektor značajki fiksne duljine iz svake regije. Treći modul je skup linearnih potpornih vektora specifičnih za klasu (Girshick, Donahue, Darrell i Malik, 2013). Na Slici 28. prikazan je sustav za detekciju objekta temeljen na R-CNN arhitekturi. Sustav (1) uzima ulaznu sliku, (2) izdvaja prijedloge regija odozgo prema gore, (3) izračunava značajke za svaki prijedlog pomoću konvolucijske neuronske

mreže, a zatim (4) klasificira svaku regiju pomoću specifične linearne metode potpornih vektora (Girshick, Donahue, Darrell i Malik, 2013).

Slika 28. Arhitektura R-CNN



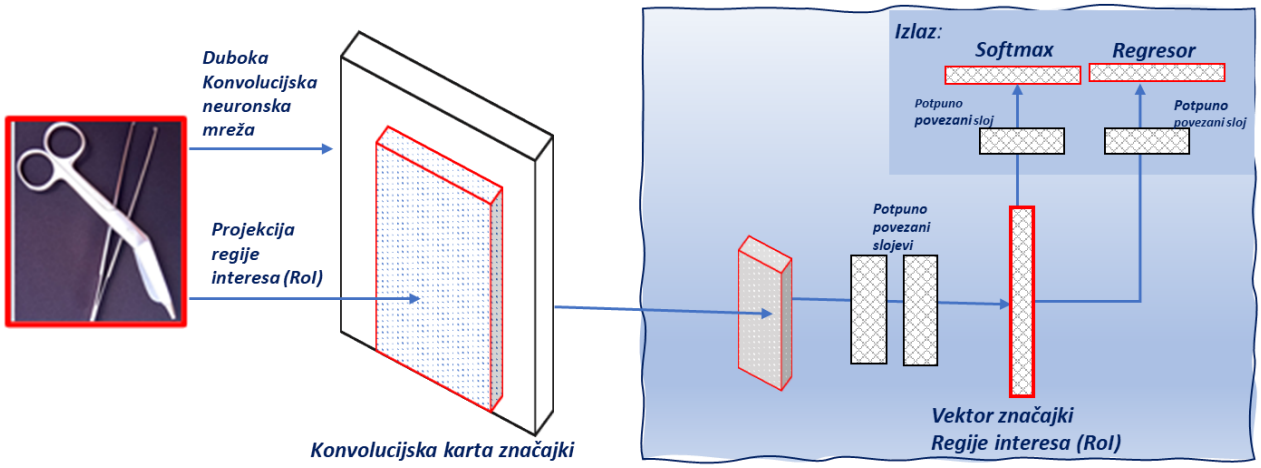
Izvor: izrada autora prema: Girshick, Donahue, Darrell i Malik (2013).

7.3.2.2 Fast R-CNN

Fast R-CNN nadograđuje se na R-CNN za učinkovitiju klasifikaciju objekata pomoću dubokih konvolucijskih neuronskih mreža. Fast R-CNN koristi nekoliko inovacija za poboljšanje brzine obuke i testiranja, a istovremeno povećava točnost otkrivanja (Girshick, 2015). Kako bi riješio problem brzine Fast R-CNN, koristi sloj konvolucije prije izdvajanja prijedloga regija. Ova metoda značajno smanjuje broj prijedloga regija, što je rezultiralo da je Fast R-CNN postao učinkovitiji od R-CNN-a. Fast R-CNN trenira vrlo duboku mrežu VGG16, 9 puta brže od R-CNN-a (Girshick, 2015). Na Slici 29. prikazana je Fast R-CNN arhitektura. Ulazna slika i više područja interesa (*RoI*) unose se u potpunu konvolucijsku mrežu. Svaki prijedlog regije (RoI) se udružuje u mapu značajki fiksne veličine, a zatim zajedno s potpuno povezanim slojevima (*FC)* preslikava u vektor obilježja. Mreža ima dva izlazna vektora po regiji interesa: softmax vjerojatnosti i regresijske pomake graničnog okvira po klasi (Girshick, 2015). Jedno od velikih poboljšanja Fast R-CNN-a od R-CNN-a je to što se propagacija konvolucijske neuronske mreže prema naprijed izvodi na cijeloj slici. U usporedbi s R-CNN-om, u Fast R-CNN-u ulaz konvolucijske neuronske mreže za izdvajanje značajki je cijela slika, a ne pojedinačni prijedlozi regija. Pretpostavimo da selektivno pretraživanje generira prijedloge n regija. Ovi prijedlozi regija (različitih oblika) označavaju područja interesa (različitih oblika) na izlazima konvolucijske neuronske mreže. Zatim ta područja interesa dodatno izdvajaju značajke istog oblika (navedena je visina h2 i širina w2) kako bi se lako spojile. Da bi se to postiglo, Fast R-CNN uvodi skupni sloj područja interesa : izlaz CNN-a i prijedlozi regija ulaze u ovaj sloj, ispisujući spojene značajke oblika

n × c × h2 × w2 koje se dalje izdvajaju za sve prijedloge regije (Zhang, C. Lipton, Li i J. Smola, 2021).

Slika 29. Fast R-CNN arhitektura

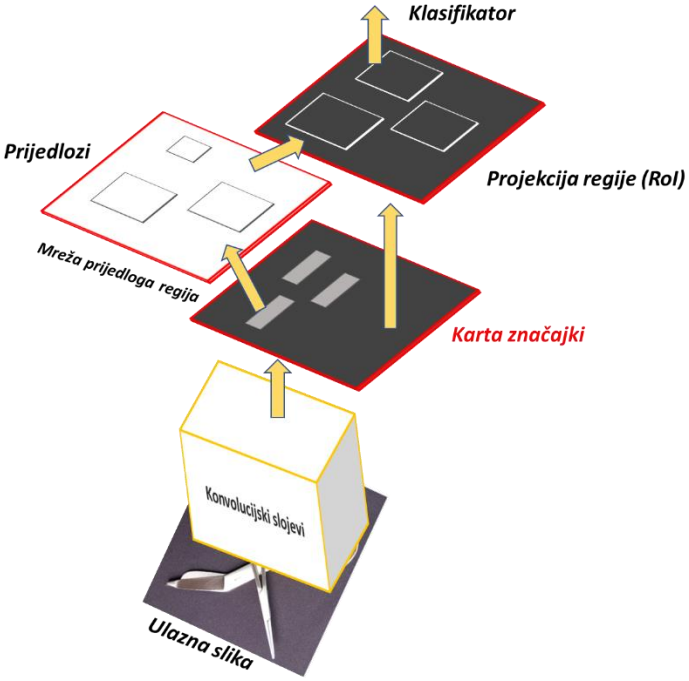


Izvor: izrada autora prema: Girshick (2015).

7.3.2.3 Faster R-CNN

Faster R-CNN je gotovo sličan prethodno opisanoj arhitekturi Fast R-CNN, ali u ovoj arhitekturi autori Ren, He, Girshick i Sun (2015). zamijenili prethodnu tradicionalnu tehniku selektivnog pretraživanja mrežom prijedloga regija (*eng. Region proposal network- RPN*). Mreža prijedloga regija je potpuno konvolucijska mreža koja se koristi za izradu prijedloga visokokvalitetnih regija. Faster R-CNN (kombinirajući RPN s brzim R-CNN-om) može se obučavati s kraja na kraj. Korištenjem RPN-a i Fast R-CNN-a postiže daljnje ubrzanje radi otkrivanja objekata unutar ulazne slike. Slično Fast R-CNN-u, cijela se slika daje kao ulaz u slojeve konvolucije Faster R-CNN-a za izradu konvolucijske karte značajki. Tada se umjesto korištenja algoritma selektivnog pretraživanja na karti značajki za identifikaciju prijedloga regija koristi RPN za predviđanje prijedloga regija. U RPN -u prijedlozi regija otkriveni po lokacijama kliznih prozora nazivaju se sidra. Relevantan okvir sidra odabire se primjenom granične vrijednosti na ocjenu "objektivnosti". Odabrani sidreni okviri i karte značajki izračunate po početnom modelu konvolucijske neuronske mreže zajedno se šalju u skupni sloj regije interesa za preoblikovanje, a izlaz skupnog sloja regije interesa, unosi se u potpuno povezane slojeve radi konačne klasifikacije i regresije graničnih okvira (Sultana, Sufian i Dutta, 2020). U usporedbi s brzim R-CNN-om, Faster R-CNN mijenja samo metodu prijedloga regije iz selektivnog pretraživanja u mrežu prijedloga regije. Ostatak modela ostaje nepromijenjen (Zhang, C. Lipton, Li i J. Smola, 2021). Na Slici 30. prikazana je Faster R-CNN arhitektura.

Slika 30. Faster R-CNN arhitektura

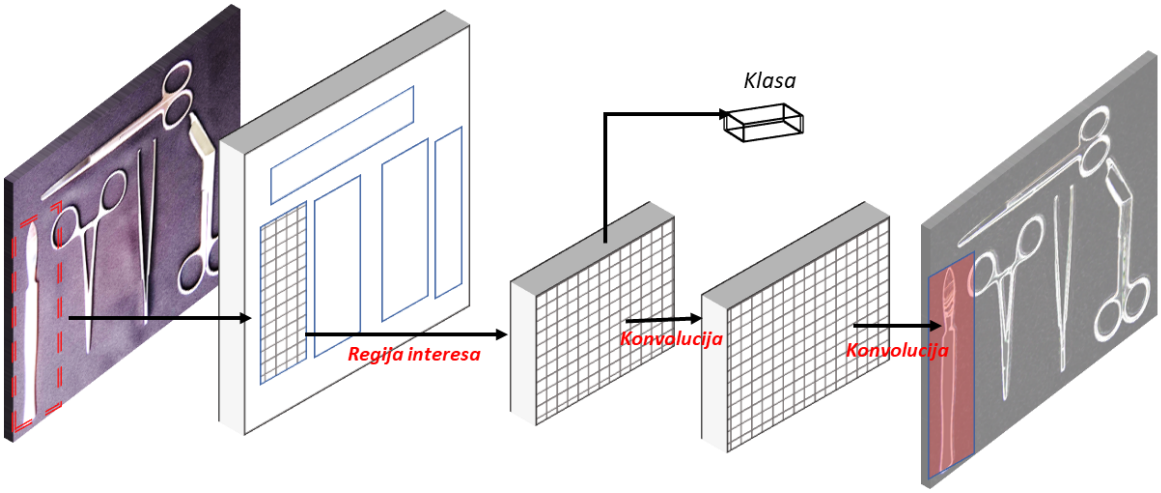


Izvor: izrada autora prema: Ren, He, Girshick i Sun (2015).

7.3.2.4 Mask R-CNN

Mask R-CNN, proširuje Fast R-CNN dodavanjem grane za predviđanje maske objekta paralelno s postojećom granom za prepoznavanje graničnih okvira. U skupu podataka za obuku, ako su položaji objekta na razini piksela također označeni na slikama, Mask R-CNN može učinkovito koristiti takve detaljne oznake za daljnje poboljšanje točnosti otkrivanja objekata. Mask R-CNN je jednostavna za implementaciju i obuku s obzirom na Fast R-CNN. Osim toga, grana maske dodaje samo male računske troškove, omogućujući brzi sustav i brzo eksperimentiranje. Mask R-CNN zamjenjuje područje okupljanja područja interesa slojem poravnanja područja interesa. Ovaj sloj poravnanja područja interesa koristi bilinearnu interpolaciju za očuvanje prostornih informacija na kartama značajki, što je prikladnije za predviđanje na razini piksela. Ispis ovog sloja sadrži karte značajki istog oblika za sva područja interesa. Koriste se za predviđanje ne samo klase i graničnog okvira za svako područje interesa, već i položaj piksela na razini objekta kroz dodatnu potpuno konvolucijsku mrežu (Zhang, C. Lipton, Li i J. Smola, 2021). Na Slici 31. prikazana je arhitektura Mask R-CNN-a za segmentaciju slike.

Slika 31. Arhitektura Mask R-CNN za segmentaciju slike

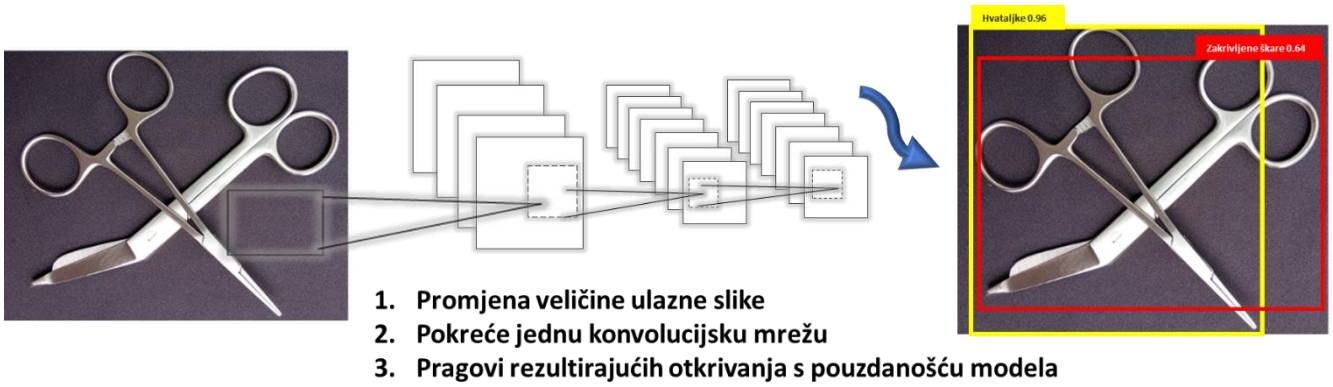


Izvor: izrada autora prema: He, Gkioxari, Dollar i Girshick, (2017).

7.3.2.5 YOLO

YOLO (*You Only Look Once*) pristup otkrivanju objekata predstavili su Redmon, Divvala, Girshick i Farhadi (2016). Oni su uveli su otkrivanje objekata kao problem regresije u prostorno odvojene granične okvire i vjerojatnosti pridružene klase. Jedna neuronska mreža predviđa granične okvire i vjerojatnosti klasa izravno iz punih slika u jednoj procjeni. YOLO čini više pogrešaka u lokalizaciji, ali je manje vjerojatno da će predvidjeti lažne pozitivne rezultate u pozadini. Za razliku od pristupa koje se temelje na klasifikatorima, YOLO se osposobljava za funkciju gubitka koja izravno odgovara performansama otkrivanja, a cijeli se model zajedno obučava. YOLO je jedna od jednostavnijih arhitektura za detekciju objekata. Jedna konvolucijska mreža istodobno predviđa više ograničavajućih okvira i vjerojatnost klasa za te okvire. YOLO trenira slike i izravno optimizira performanse otkrivanja. Ovaj jedinstveni model ima nekoliko prednosti u odnosu na tradicionalne metode otkrivanja objekata (Redmon, Divvala, Girshick i Farhadi, 2016). Na Slici 32. prikazana je arhitektura YOLO sustava za detekciju slika.

Slika 32. Sustav detekcije YOLO

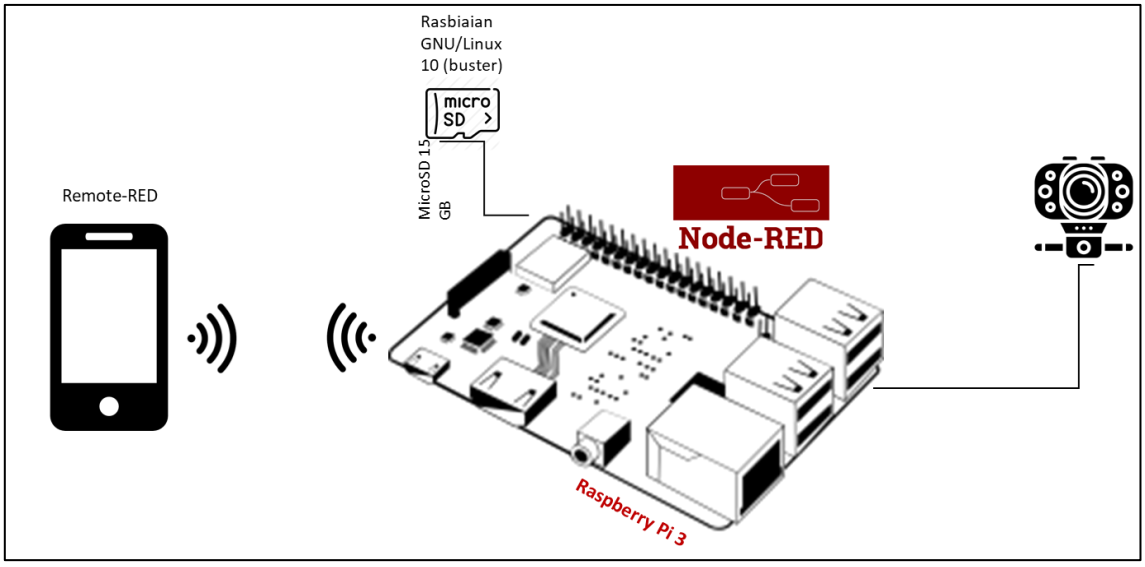


Izvor: izrada autora prema: Redmon, Divvala, Girshick i Farhadi, (2016)

8 Razvojno okruženje za rad

Na Slici 33. prikazano je razvojno okruženje za rad. Sklopovlje čine Raspberry Pi računalo, na koji je spojena kamera. Kao spremište korištena je microSD kartica ukupnog kapaciteta 15 GB sa instaliranim Raspbian GNU Linux 10 operativnim sistemom, na kojemu je instaliran Node-RED. Pokrenutoj aplikaciji na Node-RED-u, moguće je pristupiti preko pametnog telefona koristeći aplikaciju Remote-RED.

Slika 33. Shematski prikaz razvojnog okruženja



8.1 Sklopovlje

Sklopovlje čine Raspberry Pi računalo i uređaji koji su s njime povezani. Raspberry Pi, učinkovito i isplativo računalo veličine kreditne kartice . Raspberry Pi razvijen je od strane tvrtke United Kingdom Raspberry Pi. Raspberry Pi od svog je lansiranja redovito poboljšava kako u pogledu hardvera tako i softvera (Nayyar i Puri, 2015). Raspberry Pi omogućuje

spajanje ulazno-izlaznih uređaja putem USB priključaka kao što su tipkovnica, miš, kamera, itd., monitora putem HDMI priključka, spajanje na lokalnu mrežu bežično ili putem žice (Zhao, Jegatheesan i Loon, 2015).

Za ovaj projekt korišten je Raspberry Pi 3 Model B, sljedećih specifikacija:

- **SoC**: Broadcom BCM2837 chipset na 1,2 GHz
- **Spremište**: MicroSD
- **Povezivanje mreža**: 802.11 b/g/n bežični LAN, Bluetooth 4.1
- **GPIO**: 40-pin GPIO
- **RAM**: 1 GB LPDDR2 memorije
- **Povezivanje:** 4 x USB 2.0 priključak

8.2 Programsko okruženje

Za izradu aplikacije za prepoznavanje kirurških instrumenata, korišteni su programi Colaboratory, Node-RED, i Cloud Annotations.

Za izradu modela strojnog učenja korišten je Colaboratory. Colaboratory je integrirano razvojno za Python. Colaboratory omogućuje izradu modela strojnog učenja s pohranom u oblaku. Ono omogućuje, jednostavno djelanje te ne zahtjeva posebnu konfiguraciju, omogućuje besplatan pristup GPU-ima i TPU-ima, što znači da korisnik može iskoristiti snagu Googleovih hardvera, bez obzira na snagu računala kojeg se koristi. Colaboratory se koristi za strojno učenje s aplikacijama koje uključuju: rad s TensorFlowom, razvoj i trening neuronskih mreža, eksperimentiranje s TPU-ima.

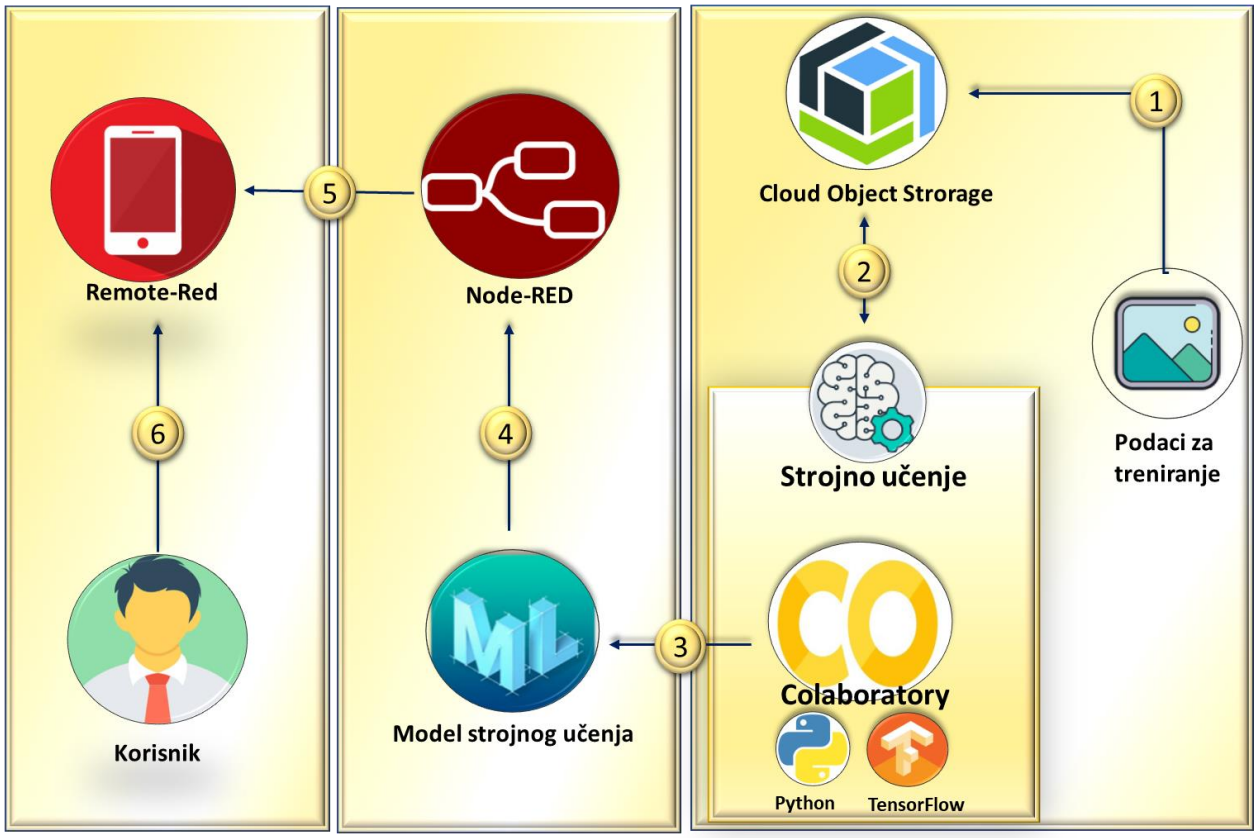
TensorFlow je programska biblioteka za umjetnu inteligenciju otvorenog koda koji se koristi u razvoju i obuci modela strojnog učenja. TensorFlow se primjenjuje u velikim razmjerima i u heterogenim okruženjima (Abadi i sur., 2016).

Za izradu ove aplikacije Colaboratory je poslužio za izradu modela, za detekciju kirurških instrumenata. Nakon što je model istreniran, on se sprema na računalo, te se uvozi u Node-RED razvojno okruženje, koje je poslužilo za izradu grafičkog korisničkog sučelja.

Na Slici 34. prikazan je dijagram koji opisuje programsko okruženje izrade aplikacije. Prvo, su prikupljeni podaci za treniranje, zatim su podaci su pohranjeni u Cloud Object Storage, potom, su prikupljene slike označene pomoću alata Cloud Annotations, zatim slijedi priprema i treniranje modela koristeći Colaboratory, nakon uspješno istreniranog modela, koristeći Node-RED razvojno okruženje implementirana je aplikacija koja će omogućiti jednostavno korištenje. Korisnik aplikaciji može pristupiti preko pametnog telefona, preko

aplikacije Remote-RED, skeniranjem QR koda, aplikacije. Pomoću Remote-RED aplikacije moguće je pristupiti sučelju aplikacije bez obzira na gdje se nalazili.

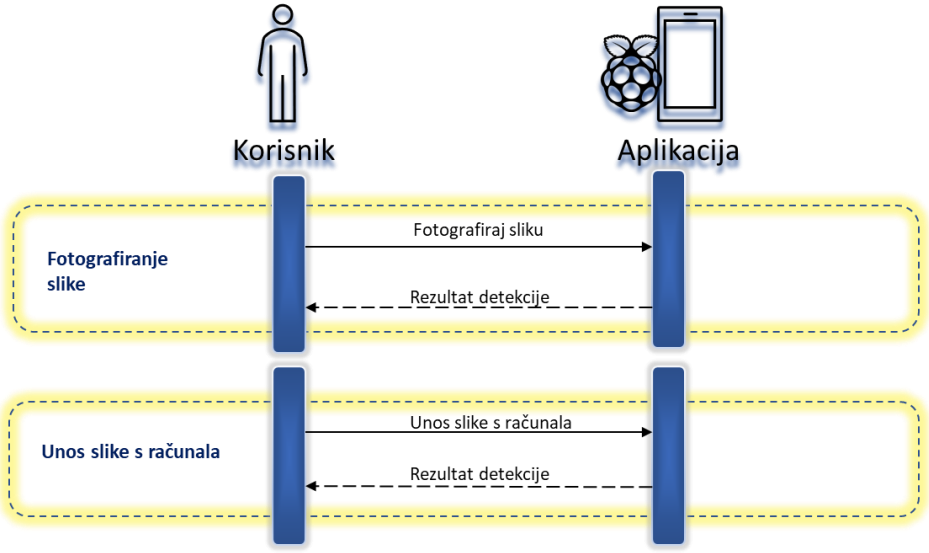
Slika 34. Dijagram programskog okruženja



8.2.1 Dijagram sekvenci

Na Slici 35. prikazan je dijagram sekvenci za aplikaciju detekcije objekte. Na dijagramu je prikazan odnos između korisnika i aplikacije. Korisnik aplikacije ima dvije mogućnosti fotografirati sliku i učitati sliku s računala. U oba slučaja aplikacija vraća rezultat prepoznavanja slike.

Slika 35. Dijagram sekvenci



8.3 Skup podataka

Skup podataka na kojima je model treniran sadrži 1500 slika, za pet različitih kategorija kirurških instrumenata, koji su prikazani na Slici 36.:

1. Škare za zavoje i gips
2. Kirurška pinceta
3. Skalpel
4. Kirurške škarice ravne
5. Hvataljka za arterije

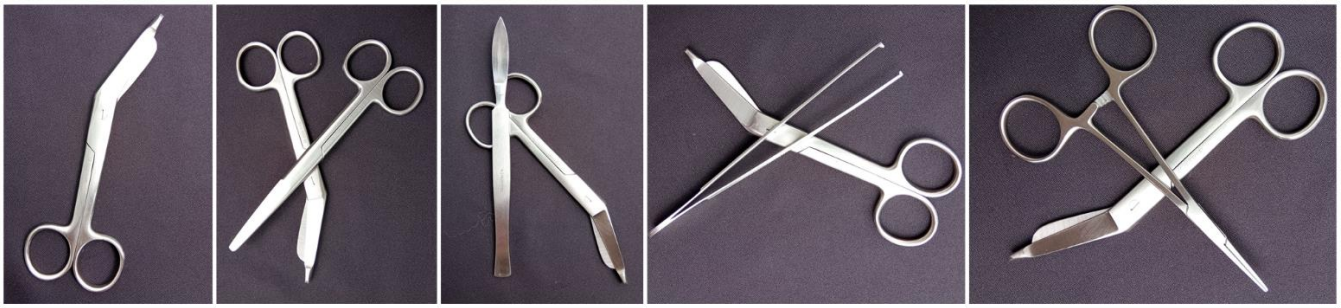
U nastavku su opisani pojedini kirurški instrumenti na kojima je model treniran.

Slika 36. Skup kirurških instrumenata

Škare za zavoje i gips

Škare za zavoje i gips prikladne su za rezanje i uklanjanje debljeg gipsanog zavoja i zavoja. Na Slici 37. prikazane su škare za zavoj i gips na kojima je model treniran. Na Slikama su dostupne slike gdje škare za zavoje i gips same na slici (a) , zajedno sa kirurškim škaricama ravnim (b), skalpelom (c), pincetom (d) i hvataljkama za arterije (e).

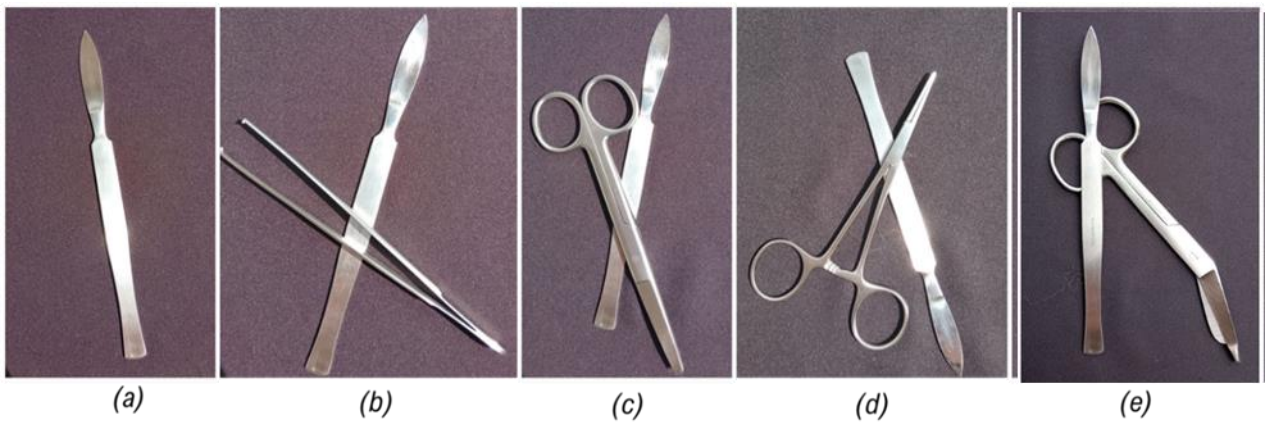
Slika 37. Primjer slika za zakrivljene škare



Skalpel

Skalpel je mali i iznimno oštar instrument s oštricom koji se koristi u kirurgiji. Na Slici 38. prikazani su primjeri slika skalpela na kojima je model treniran. Na slici (a) prikazan je primjer slike na kojoj je skalpel sam na slici. Na Slici 38. (b) prikazan je skalpel s pincetom, na Slici (c) prikazan je skalpel s ravnim kirurškim škaricama, na Slici (d) nalazi se skalpel s hvataljkama za arterije, (e) s škarama za zavoj i gips.

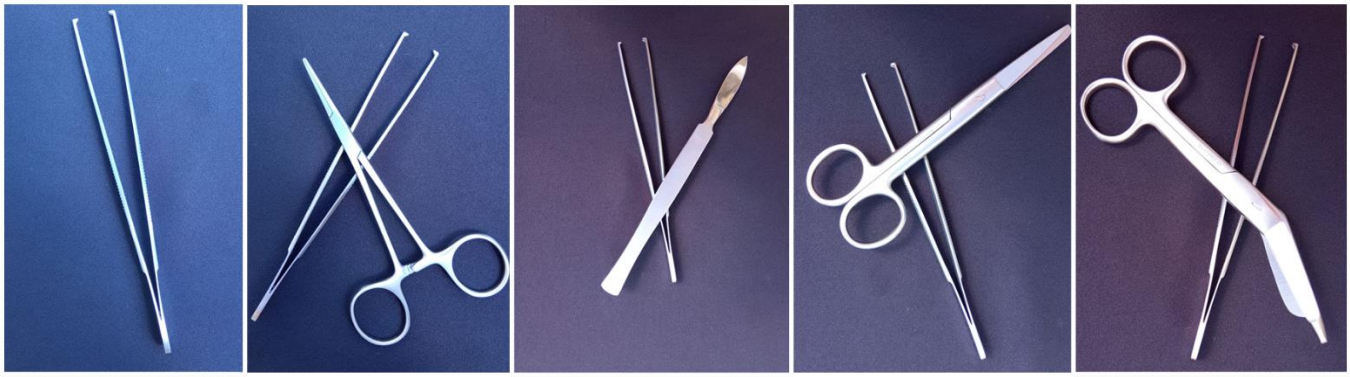
Slika 38. Primjer slika za skalpel



Kirurška pinceta

Kirurška pinceta je kirurški instrument hvatajućeg tipa koji se koristi tijekom operacija i drugih medicinskih zahvata. Na Slici 39. prikazani su primjeri slika pincete na kojoj je model treniran. Na Slici 39. (a) prikazan je primjer slike na kojemu se pinceta nalazi sama. Na Slici (b) s hvataljkama za arterije (c) skalpelom (d) ravnim kirurškim škaricama (e) škarama za zavoje i gips.

Slika 39. Primjer slika za pincetu

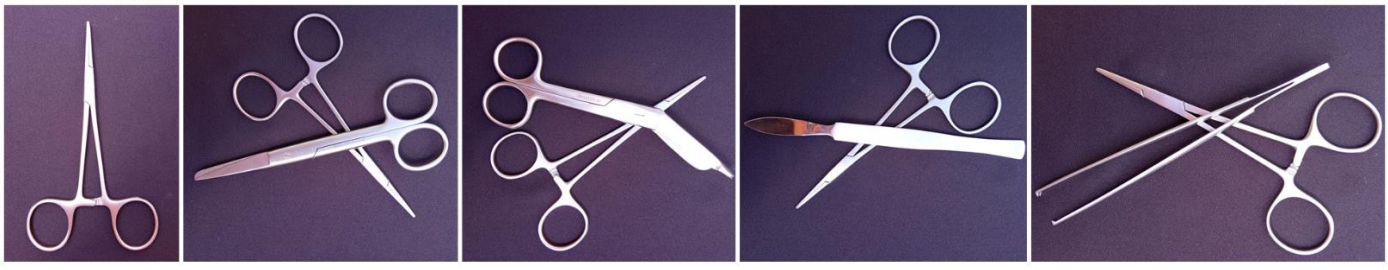


(a) (b) (c) (d) (e)

Hvataljka za arterije

Hvataljka za arterije služi za pridržavanje i hvatanje presječenih i ozlijeđenih krvnih žila sprječavajući krvarenje. Na Slici 40. prikazan je primjer slika za hvataljke za arterije (a) same hvataljke, (b) s ravnim kirurškim škaricama, (c) škarama za zavoj i gips (d) skalpelom (e) s pincetom.

Slika 40. Primjer slika hvataljke za arterije



(a) (b) (c) (d) (e)

Kirurške škarice ravne

Ravne kirurške škarice se koriste prvenstveno za rezanje tkiva, šavova te medicinskih potrošnih materijala poput zavoja i kompresa. Na Slici 41. prikazane su primjeri slika na kojima je model treniran za instrument ravne kirurške škare. Na Slici (a) prikazane su same kirurške škarice ravne (b) sa skalpelom (c) pincetom (d) škarama za zavoj i gips (e) hvataljkama za arterije.

Slika 41. Kirurške škarice ravne

9 Treniranje modela

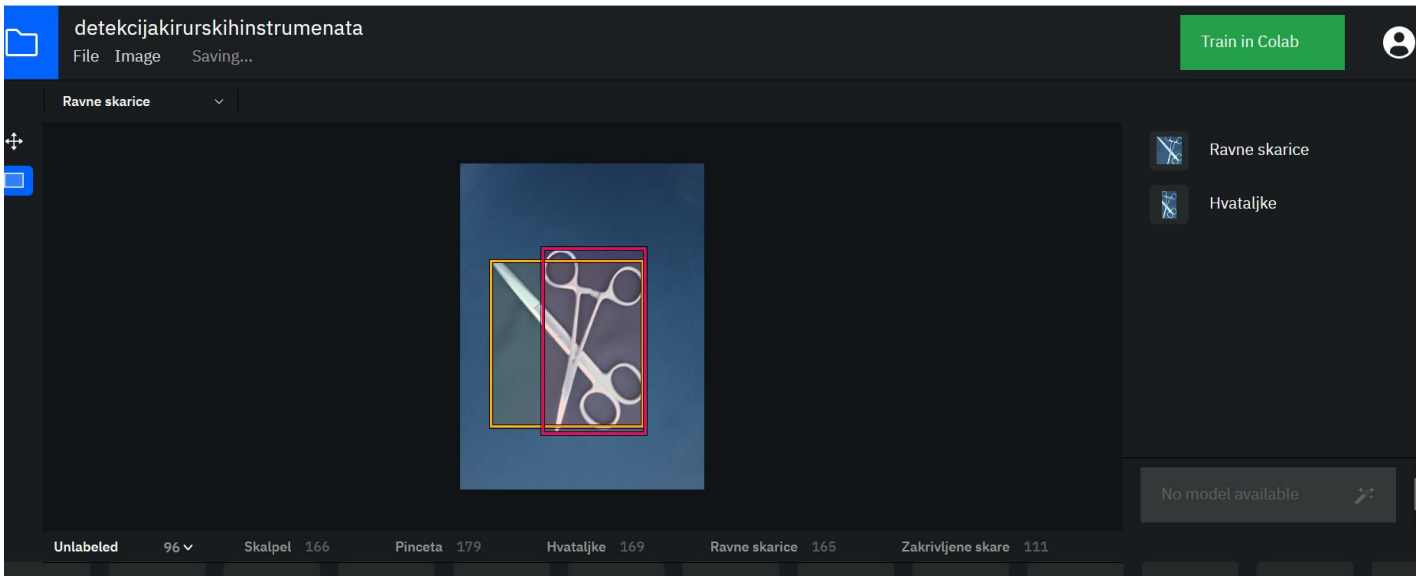
Za osposobljavanje modela za detekciju objekata potreban je skup slika i oznaka s ograničavajućim okvirima. Za dodavanje oznaka slikama na kojima će model učiti, korišten je alat IBM Cloud Annotations. Korištenje IBM Cloud spremišta podataka daje pouzdano mjesto za čuvanje podataka o treningu. Također, omogućuje potencijal za suradnju, dopuštajući timu da istovremeno bilježi skup podataka u stvarnom vremenu (Cloud Annotations, 2021). Na Slici 42. prikazan je dijagram koji opisuje postupak treniranja modela, slike su označene koristeći alat Cloud Annotations, slike i oznake su pohranjene unutar IBM Cloud Object Storage, nakon čega slijedi obuka modela, koristeći programsku biblioteku Tensorflow.

Slika 42. Postupak obuke podataka

9.1 Priprema podataka za obuku

Slikama se oznake dodane koristeći alat Cloud Annotations. Na Slici 43. prikazan je alat Cloud Annotations, te postupak dodavanja oznaka na primjeru hvataljka za arterije i ravne kirurške škarice. Nakon što su slike označene u gornjem desnom kutu, klikom na 'Train in Colab' započinje postupak treniranja podataka.

Slika 43. Dodavanje oznaka slikama



9.2 Treniranje modela

Kako bi se omogućio pristup podacima o obuci pohranjenih u IBM Cloud Storage, potrebne su vjerodajnice za segment za pohranu objekata. Vjerodajnice su kopirane iz prethodnog koraka u prvu ćeliju unutar Colaboratory-a.

Slika 44. Dodavanje vjerodostojnica u Google Colab Notebook

```
credentials = {
  "bucket": "oznakapodataka",
  "access_key_id": "6c2d286f79541c4ec98afb4aa1fadab9a0",
  "secret_access_key": "0632545369e640890d7cdd76c0f63ee018314de7a4adfcd97gc",
  "endpoint_url": "https://s3.us.cloud-object-storagge.appdomain.cloud"
}
```

9.2.1 Instalacija API-a za otkrivanje objekata TensorFlow

Većina potrebnih ovisnosti dolazi unaprijed učitana u Colaboratory. Jedini dodatni paket koji je potrebno instalirati je TensorFlow.js koji se koristi za pretvaranje obučenog modela u model koji je kompatibilan za web koji je dostupan u Tensorflow.js formatu. Da bismo koristili API za otkrivanje objekata, moramo ga dodati u PYTHON-PATH zajedno s *'slim'* paketom koji sadrži kod za trening i ocjenu nekoliko široko korištenih modela klasifikacije slika konvolucijskih neuronskih mreža.

Slika 45. Instalacija API-a za detekciju objekata TensorFlow

```
%tensorflow_version 1.x
import os
import pathlib

# Kloniranje spremišta modela tensorflow ako već ne postoji
if "models" in pathlib.Path.cwd().parts:
  while "models" in pathlib.Path.cwd().parts:
    os.chdir('..')
elif not pathlib.Path('models').exists():
  !git clone --depth 1 https://github.com/cloud-annotations/models

!pip install cloud-annotations==0.0.4
!pip install tf_slim
!pip install lvis
!pip install --no-deps tensorflowjs==1.4.0

%cd /content/models/research
!protoc object_detection/protos/*.proto --python_out=.

pwd = os.getcwd()
os.environ['PYTHONPATH'] += f':{pwd}:{pwd}/slim'
```

Treniranje započinje pozivom skripte `*model_main*`, prosljeđujući:

- Mjesto `*pipepline.config*` koji je stvoren u prethodnom koraku
- Tamo gdje će se spremiti model
- Koliko koraka želimo trenirati model (što je veći broj koraka, to je veći potencijal za učenje)
- Broj koraka ocjenjivanja

Na Slici 46. prikazan je kod kojim se poziva trening modela.

Slika 46. Trening modela

```
!rm -rf $OUTPUT_PATH
!python -m object_detection.model_main \
    --
pipeline_config_path=$DATA_PATH/pipeline.config \
    --model_dir=$OUTPUT_PATH \
    --num_train_steps=$NUM_TRAIN_STEPS \
    --num_eval_steps=100
```

9.2.2 Pretvaranje modela TensorFlow.js format

Kad je model uspješno istreniran, može se koristiti u Pythonu, kako bi se mogao koristiti u pregledniku potrebnog ga je pretvoriti u Tensorflow.js format. Na Slici 47. prikazan je kod koji pretvara model u Tensorflow.js format.

Slika 47. Pretvaranje modela TensorFlow.js format

```
!tensorflowjs_converter \
  --input_format=tf_frozen_model \
  --output_format=tfjs_graph_model \
  --output_node_names='Postprocessor/ExpandDims_1,Postprocessor/Slice' \
  --quantization_bytes=1 \
  --skip_op_check \
  $EXPORTED_PATH/frozen_inference_graph.pb \
  /content/model_web

import json

from object_detection.utils.label_map_util import get_label_map_dict

label_map = get_label_map_dict(LABEL_MAP_PATH)
label_array = [k for k in sorted(label_map, key=label_map.get)]

with open(os.path.join('/content/model_web', 'labels.json'), 'w') as f:
  json.dump(label_array, f)

!cd /content/model_web && zip -r /content/model_web.zip *
```

9.2.3 Preuzimanje modela

Potrebno je preuzeti model kako bi ga mogli implementirati u aplikaciju. Na Slici 48. prikazan je kod za preuzimanje modela.

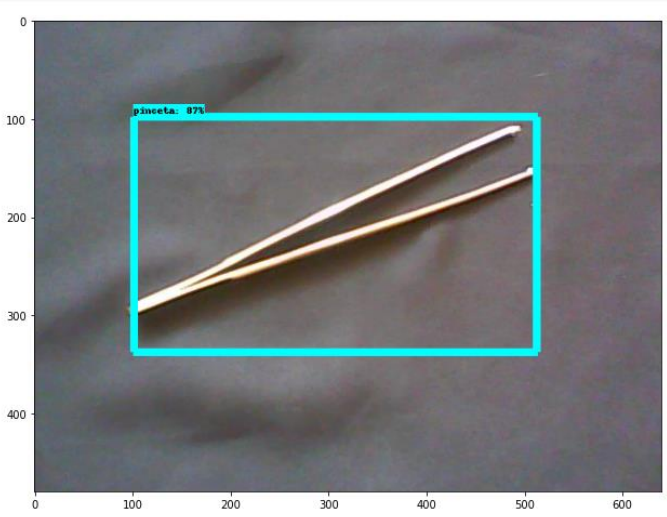
Slika 48.Naredba za preuzimanje modela

```
from google.colab import files
files.download('/content/model_web.zip')
```

9.2.4 Testiranje modela

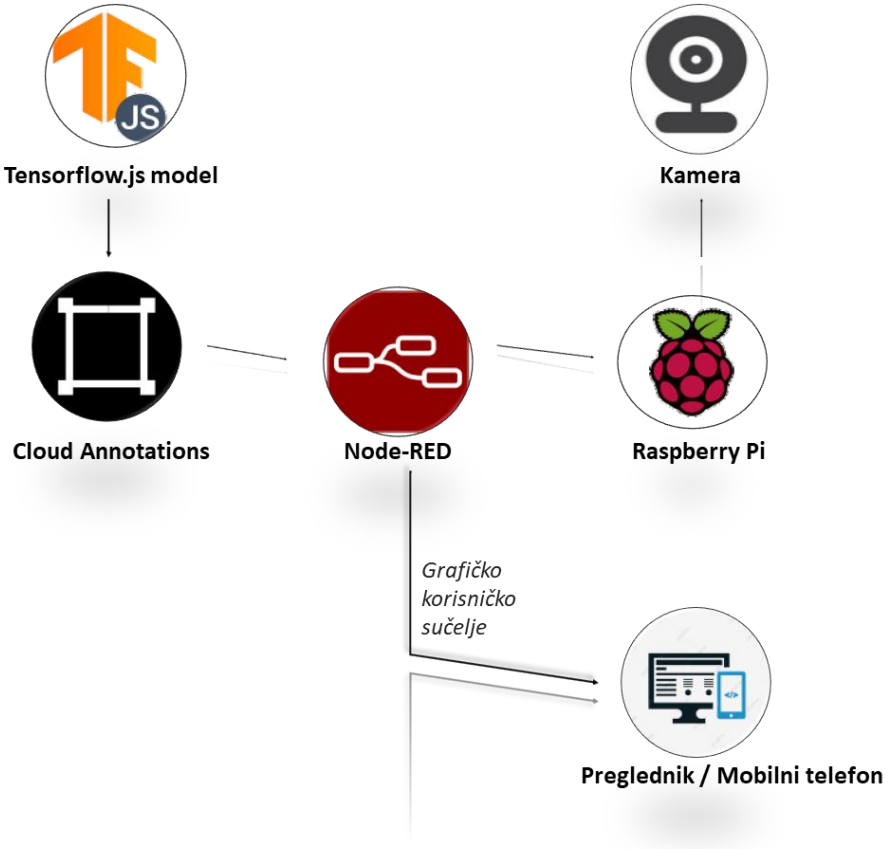
Na Slici 49. prikazano je testiranje modela za kiruršku pincetu. Model je procijenio vjerojatnost s 87%.

Slika 49. Testiranje modela



Nakon što je model uspješno istreniran te preuzet, izrađeno je sučelje aplikacije koje će olakšati korisniku korištenje modela. Za izradu sučelja aplikacije te spajanje modela s komponentama sučelja, korišteno je razvojno okruženje Node-RED. Na Slici 50. prikazan je shematski prikaz koji opisuje izradu aplikacije. U samom središtu je Node-RED koji spaja hardverske i softverske komponente sustava. Node-RED je pokrenut na Raspberry Pi-u, na kojega je spojena kamera. Prethodno izrađeni model u Tensorflow.js formatu dodaje su u Node-RED aplikaciju. Tensorflow.js pruža prednost što se modeli izvode izravno na uređaju bez interakcije s vanjskim poslužiteljem ili oblakom.

Slika 50. Shematski prikaz sustava



10.1 Node-RED

Node-Red je program otvorenog koda zasnovan za integraciju IoT hardverskih uređaja, API-ja (Application Programming Interfaces) i mrežnih usluga koje je razvio IBM. Node-RED je zasnovan na JavaScriptu, izražen je u platformi Node.js, koja pruža vizualni uređivač protoka zasnovan na pregledniku. Sustav sadrži čvorove koji su predstavljeni odgovarajućim

ikonama. Čvorovi se mogu koristiti na dva načina: povlačenjem/ ispuštanjem i spajanjem čvorova ili uvezom JavaScript koda (Lekić i Gardašević, 2018).

Vizualni prikaz temeljen na pregledniku pomaže dizajnerima i programerima u boljem razumijevanju interakcija u čitavoj mreži Internet stvari (*eng. Internet of things*). U stvari, mnogi entiteti mogu biti uključeni, i hardverski (npr. senzori) i softverski (npr. uslužni). Node-RED također omogućuje povezivanje hardverskih uređaja i sučelja za programiranje aplikacija (Sicari, Rizzardi i Coen-Porisini, 2018).

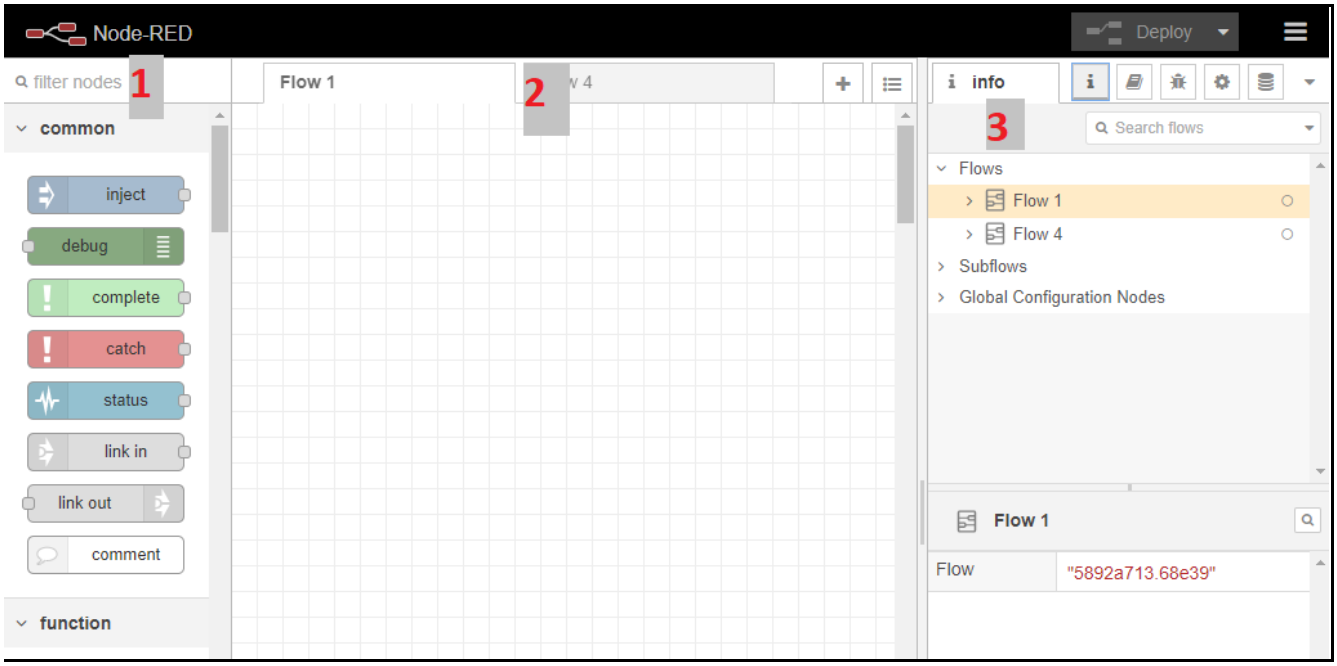
Stvoreni čvorovi se pohranjuju pomoću JSON-a (*JavaScript Object Notation*). Node-Red omogućuje programerima da povežu ulaz, izlaz i obradu čvorova kako bi se stvorili tokovi za obradu podataka, kontroliranje stvari ili stanje upozorenja. Djeluje na sljedeće načelo: omogućuje povezivanje web usluga ili prilagođava čvorova jedni drugima ili stvarima kako bi mogle obaviti funkcije poput stanja podataka senzora putem e-pošte ili na usluge poput Twittera za jednostavnije izvođenje složenih analiza (Lekić i Gardašević, 2018).

Node-Red ima tri osnovne komponente (Rajalakshmi i Shahnasser, 2017):

1. **Izbornik čvorova** (*eng. Node Panel*) – čvorovi su napisani u Node.js. Mogu se jednostavno preuzeti iz knjižnice Node-Red-a.
2. **Izbornik protoka** (*eng. Flow Panel*) – dijagrami protoka stvaraju se integriranjem različitih čvorova koji su konfigurirani i pohranjeni pomoću JSON-a.
3. **Info i ploča za otklanjanje pogrešaka** (*eng. Info and Debug Panel*) – prikazuje poruke proslijeđene čvorovima za otklanjanje pogrešaka u tijeku.

Na Slici 51. prikazano je sučelje aplikacije Node-RED, s označenim osnovnim komponentama.

Slika 51. Sučelje programa Node-Red



10.2 Korištene komponente

U Tablici 3. prikazane su komponente koje su korištene za izradu aplikacije kao i njihov opis.

Tablica 3. Opis komponenta korištenih za izradu aplikacije u Node-RED-u

Komponenta	Opis
Ui_button	Korisničkom sučelju dodaje gumb. Klikom na gumb generira se poruka s *msg.payload* postavljenim na polje *Payload*.
Function	JavaScript funkcija za pokretanje formuliranje poruka koje čvor prima. Poruke se šalju kao JavaScript objekt koji se naziva *msg*. Prema dogovoru imat će svojstvo *msg.payload* koje sadrži tijelo poruke.
Ui_text	Na korisničkom sučelju prikazat će se tekstno polje koje se ne može uređivati. Svaka primljena *msg.payload* ažurirat će tekst na temelju navedenog formata vrijednosti.
Fs-file-lister	Ovaj čvor pretražuje mapu (i po potrebi podmape) za datoteke. Po želji može filtrirati određenu datoteku ili obrazac mape ili vratiti samo ime mape. Mapa i datoteke dolaze s računala i operativnog sustava na kojem je pokrenut Node-Red.

Ui_dropdown	Korisničkom sučelju dodaje okvir za padajući izbornik. Po potrebi se može dodati više parova vrijednosti / oznaka. Ako oznaka nije navedena, vrijednost će se koristiti za oboje. Konfigurirana vrijednost odabrane stavke vratit će se kao *msg.payload.*
Change	Omogućuje postavljanje, promjenu, brisanje ili premještanje svojstava poruke, konteksta toka ili globalnog konteksta. Čvor može odrediti više pravila koja će se primijeniti onim redoslijedom kako su definirana.
Template	Postavlja svojstvo na temelju priloženog predloška.
File in	Čita sadržaj datoteke kao niz ili kao binarni međuspremnik.
Usbcamera	Ako se aktivira čvor USB kamere za Raspberry Pi kameru, snimit će fotografiju s danom rezolucijom i formatom.
Switch	Usmjerava poruke na temelju njihovih vrijednosti svojstva ili položaja sekvence.
Debug	Prikazuje odabrana svojstva poruke na kartici bočne trake za otklanjanje pogrešaka. Prema zadanim postavkama prikazuje *msg.payload*, ali se može konfigurirati za prikaz bilo kojeg svojstva, pune poruke ili rezultata izraza JSON.
Base64	Funkcija koja pretvara izabrano svojstvo (zadani *msg.payload*) base64 format. Ako je ulaz međuspremnik, pretvara ga u niz kodiran s Base64. Ako je ulaz Base64 niz, pretvara ga natrag u binarni međuspremnik.
Ui_table	Čvor koji prikazuje tablicu na sučelju aplikacije.
Ui_audio	Reproducira audio ili tekst u govor na nadzornoj ploči.
Cloud-Annotations	Čvor za predviđanje s modelima Cloud Annotations.

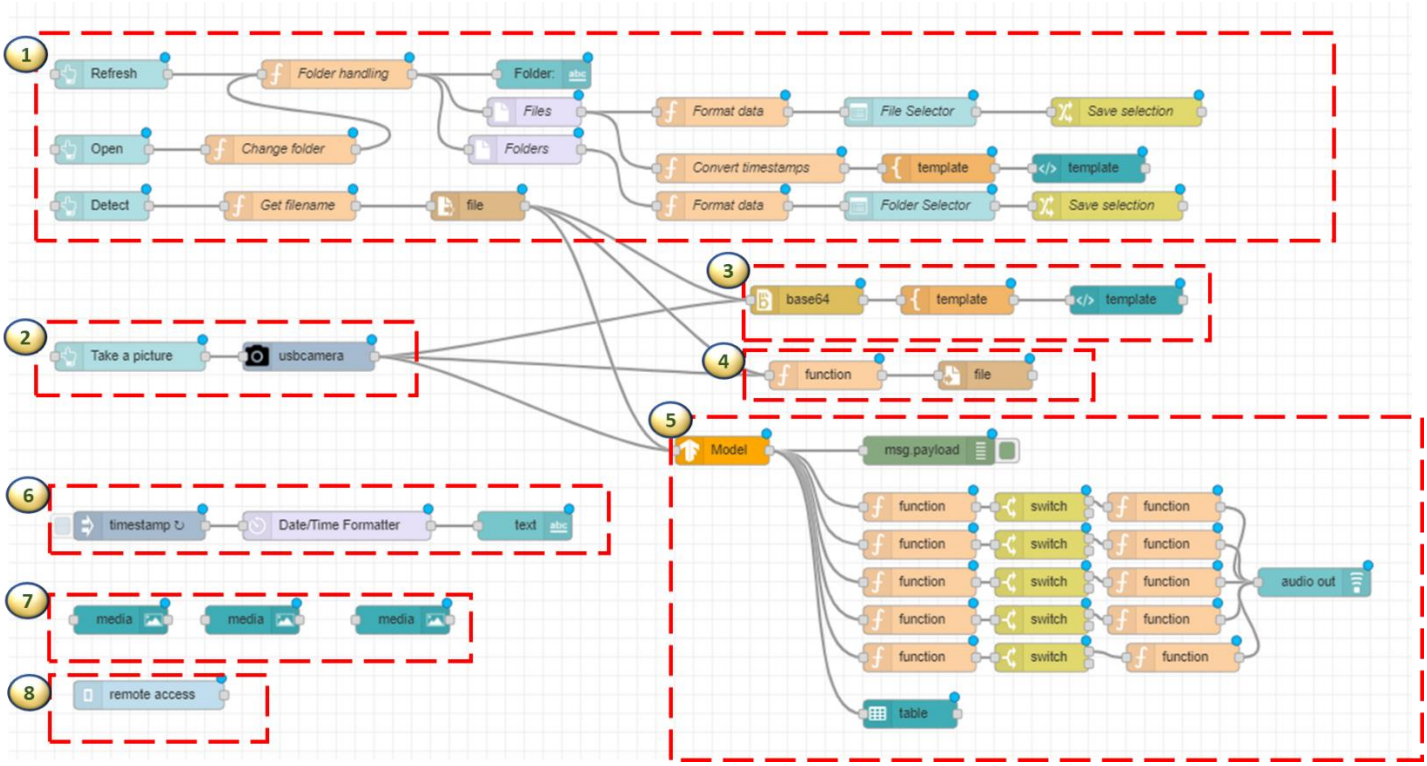
Ui_template	Ovaj čvor može se koristiti za stvaranje dinamičkog elementa korisničkog sučelja koji mijenja svoj izgled na temelju ulazne poruke i može vraćati poruke na Node-RED.
Inject	Ubrizgava poruku u tok ručno ili u redovitim intervalima. *Msg.payload* može biti raznih vrsta, uključujući nizove, JavaScript objekte ili trenutno vrijeme.
Moment	Pretvara datumski / vremenski objekt ili niz u lijepo oblikovan tekst ili datumski / vremenski objekt.
Ui_media	Prikazuje medijske datoteke i URL-ove na nadzornoj ploči
Remote-access	Pristupni čvor omogućuje pristup web mjestu lokalno ili udaljeno iz aplikacije.

10.3 Glavni Node-RED tijek

Na Slici 52. prikazan je glavni Node-RED tijek koje je poslužio za izradu grafičkog sučelja aplikacije te povezao prethodno izrađen model strojnog učenja, koji će poslužiti za klasifikaciju kirurških instrumenata. Glavni tijek je podijeljen na osam pod tijekova:

1. Tijek za učitavanje slike s računala
2. Izravno fotografiranje slike
3. Prikaz fotografije slike/ slike koja je učitana s računala na sučelju aplikacije
4. Spremanje fotografirane slike na računalo
5. Detekcija i prikaz rezultata detekcije
6. Prikaz vremena i datuma
7. Prikaz logotipa projekta na sučelju aplikacije
8. Daljinski pristup

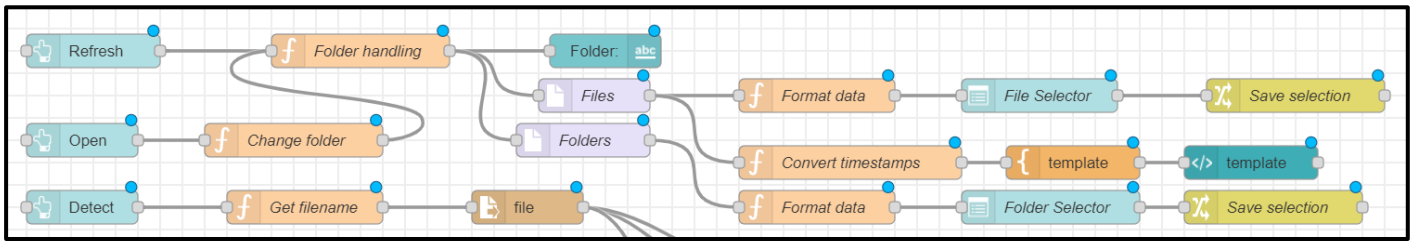
Slika 52. Node-RED tijek



10.3.1 Učitavanje slike s računala

Na Slici 53. prikazan je tijek koji omogućuje korisniku da učita sliku koju želi prepoznati s računala. Čvorovi koji su potrebni su: *ui_button*, *function*, *ui_text*, *fs-file-lister*, *ui_dropdown*, *change*, *file in, template, ui_template*.

Slika 53. Tijek za učitavanje slika s računala



Čvor pod nazivom „*Folder handling*“ je funkcijski čvor koji služi za rukovanje mapama. Na Slici 54. prikazan je kod ovog funkcijskog čvora.

Slika 54. Funkcija: Rukovanje mapama

```
let folder = context.get("folder");
if (folder===undefined) {
    folder="/";
    context.set("folder", folder);}
let hidden = context.get("hidden");
if (hidden===undefined) {
    hidden=false;
    context.set("hidden", hidden);}
if (msg.topic==="up") {
    var the_arr = folder.split('/');
    the_arr.pop();
    folder=the_arr.join('/');
    context.set("folder", folder);}
if (msg.topic==="change") {
    folder=msg.payload;
    context.set("folder", folder);}
if (msg.topic==="hidden") {
    hidden=msg.payload;
    context.set("hidden", hidden);}
msg.payload = {"start":folder, "hidden": hidden};
return msg;
```

Funkcijski čvor pod nazivom „*Format data*“ služi za formatiranje podataka za datoteke. Na Slici 55. prikazan je kod unutar ovoga funkcijskog čvora.

Slika 55. Funkcija: Formatiranje podataka za datoteke

```
// formatiranje podataka za padajući izbornik
msg.options = [];
for (var i=0; i<msg.payload.length; i++) {
    // This is a file
    obj = {};
    obj [msg.payload[i].name.replace(/^.*(\\|\/|\:)/, '')]=msg.payload[i].name;
    msg.options.push(obj);
}
msg.payload={};
return msg;
```

Sljedeći čvor pod nazivom „*Folder*:“ je *Ui_text*, čiji je format vrijednosti postavljen je na: *{{msg.payload.start}}.* Ovaj čvor ispisuje informaciju na sučelju aplikacije u kojemu se folderu korisnik trenutno nalazi.

Čvor pod nazivom „*Files*“ je *fs-file-lister.* Ovaj čvor pretražuje datoteke. Početak pretraživanja postavljen je u mapi *„/home/pi*“.

Funkcijski čvor „*Format data*“ oblikuje podatke za padajući izbornik tj. za prikaz datoteka na sučelju. Na Slici 56. prikazan je kod za formatiranje podataka.

Slika 56. Funkcija: Formatiranje podataka

```
// oblikovanje podataka za padajući izbornik
msg.options = [];
for (var i=0; i<msg.payload.length; i++) {
    // This is a file
    obj = {};
    obj [msg.payload[i].name.replace(/^.*(\\|\/|\:)/, '')]=msg.payload[i].name;
    msg.options.push(obj);
}
msg.payload={};
return msg;
```

Čvor pod nazivom „*File Selector* “ je čvor ui_dropdown on korisničkom sučelju dodaje okvir za padajući izbornik.

Čvor pod nazivom „Save selection“ je change čvor. Postavke čvora su prikazane na Slici 57.

Slika 57. Postavke čvora "Save selection"

Čvor pod nazivom : „*Change folder*“ je funkcijski čvor. Kod ovog funkcijskog čvora prikazan je na Slici 58. Ovaj čvor dohvaća naziv datoteke iz konteksta toka.

Slika 58. Funkcija: primjena mape

```
// Dohvati naziv datoteke iz konteksta toka
let folderselected = flow.get("folderselected");

// provjerite, ako je naziv datoteke nedefiniran, to znači da još ne postoji, još ništa nije odabrano
// return: ne ispisujte ništa
if (folderselected===undefined) {
    return;
}
msg.topic = "change";
msg.payload = folderselected;

return msg;
```

Čvor pod nazivom : *„Convert timestamps*“ je funkcijski čvor. Kod ovog funkcijskog čvora prikazan je na Slici 59.

Slika 59. Funkcija: Pretvaranje vremenskih oznaka

```
for (var i=0; i<msg.payload.length; i++) {
    msg.payload[i].stat.created = msg.payload[i].stat.created.toISOString().slice(0, 19).replace('T', ' ');
    msg.payload[i].stat.changed = msg.payload[i].stat.changed.toISOString().slice(0, 19).replace('T', ' ');
    msg.payload[i].stat.accessed = msg.payload[i].stat.accessed.toISOString().slice(0, 19).replace('T', ' ');
    msg.payload[i].stat.statusChanged = msg.payload[i].stat.statusChanged.toISOString().slice(0, 19).replace('T', ' ');
    msg.payload[i].fname = msg.payload[i].name.replace(/^.*(\\|\/|\:)/, '');
}
return msg;
```

Čvor pod nazivom „*Get filename*“ je funkcijski čvor koji služi za dohvaćanje naziva datoteke. Kod ovog funkcijskog čvora prikazan je na Slici 60.

Slika 60. Funkcija: Dohvaćanje naziva datoteke

```
// Dohvaćanje naziva datoteke iz konteksta toka
let filename = flow.get("fileselected");
// provjerite, ako je naziv datoteke nedefiniran, to znači da još ne postoji, još ništa nije odabrano
// return: ne ispisujte ništa
if (filename===undefined) {
    return;
}
// vraća naziv datoteke u čvor datoteke za brisanjemsg.filename = filename;
if (msg.filename.replace(/^.*(\\|\/|\:)/, '')[0]!==".") {
// Učinite to samo ako je ovo datoteka
return msg;
}
```

Čvor *Template* postavlja svojstvo na temelju priloženog predloška. Na Slici 61. prikazan je kod template čvora, koji priprema prikaz datoteka na sučelju odabrane mape.

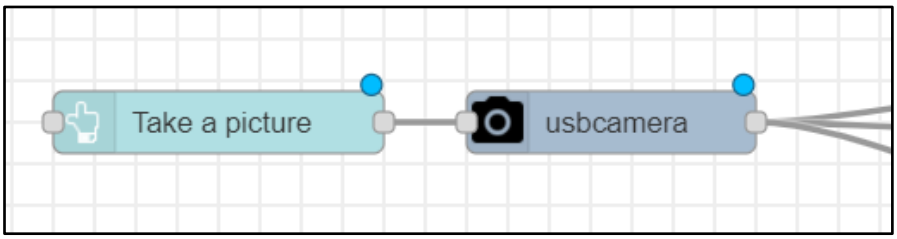
Slika 61. Template čvor

```
<table width="100%">
    <tr><th>File Name</th><th>Size</th><th>Created</th><th>Changed</th></tr>
    {{#payload}}
        <tr>
            <td><a href="/download?filename={{name}}" target="blank">{{fname}}</a></td>
            <td>{{stat.size}}</td>
            <td>{{stat.created}}</td>
            <td>{{stat.changed}}</td>
        </tr>
    {{/payload}}
</table>
```

10.3.2 Izravno fotografiranje slike

Na Slici 62. prikazan je tijek koji korisniku omogućuje izravno fotografiranje slike s USB kamere koja je spojena na Rasberry Pi. Kako bi se to omogućilo potrebna su dva čvora: ui_button, usbcamera. Ui_button korisničkom sučelju dodaje gumb. Klikom na gumb pod nazivom „Take a picture“, pokreće se usb kamera koja fotografira sliku te je šalje na čvor modela koji je zadužen za prepoznavanje te slike. Čvor usbcamera postavljen je na „Buffer“ vrijednost.

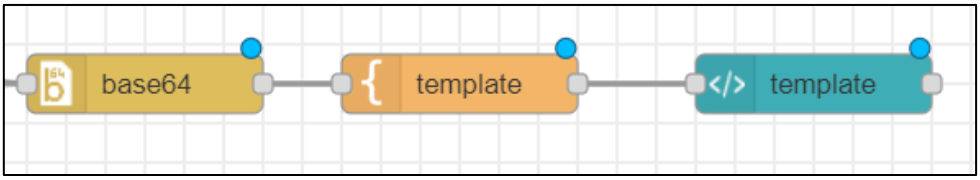
Slika 62. Izravno fotografiranje slike



10.3.3 Prikaz fotografije

Za prikaz fotografirane slike ili slike koja je učitana s računala potrebni su sljedeći čvorovi: *base64, template, ui_template*. Na Slici 63. prikazan je tijek koji omogućuje ovu funkcionalnost. Base64 pretvara odabranu sliku u format *base64*. Template čvor priprema sliku za prikaz, postavljene su određene dimenzije slike (Slika 64.)

Slika 63. Tijek za prikaz slike na sučelju aplikacije



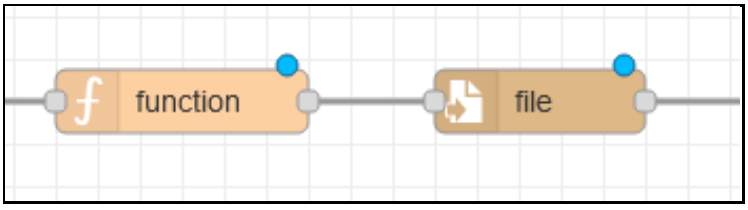
Slika 64. Template kod

```
1  <img width="352px"
2  height="288px" src="
3  {{{payload}}}">
```

10.3.4 Spremanje slika u odabrani folder

Za spremanje snimljenih slika pod određenim imenom koje će sadržavati: datum i vrijeme kad je slika fotografirana, potreban je tijek prikazan na Slici 65.

Slika 65. Tijek za spremanje slika pod određenim imenom.



Slika 66. prikazuje kod unutar čvora funkcije koji dohvaća datum i vrijeme. Snima sliku pod određenim imenom dodavanjem godine, mjeseca, dana, sati, minuta i sekundi nakon prefiksa.

Slika 66. Čvor funkcije

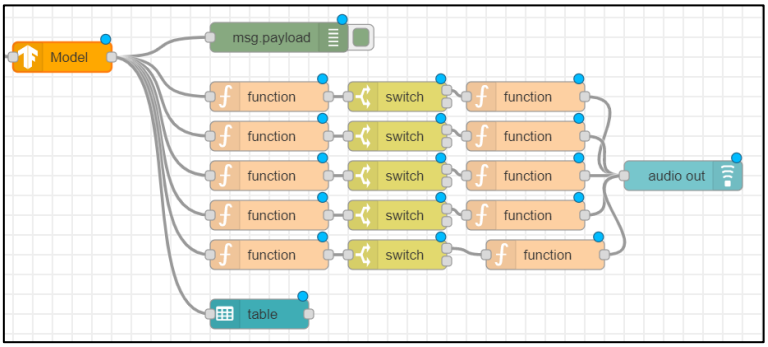
```
var now = new Date();
var yyyy = now.getFullYear();
var mm = now.getMonth() < 9 ? "0" + (now.getMonth() + 1) : (now.getMonth() + 1);
var dd  = now.getDate() < 10 ? "0" + now.getDate() : now.getDate();
var hh = now.getHours() < 10 ? "0" + now.getHours() : now.getHours();
var mmm  = now.getMinutes() < 10 ? "0" + now.getMinutes() : now.getMinutes();
var ss  = now.getSeconds() < 10 ? "0" + now.getSeconds() : now.getSeconds();


msg.imagename = "image_"+  yyyy + mm + dd + "-" + hh + mmm + ss + ".jpg";

msg.filename = "D:/pic/"+ msg.imagename;

node.status({fill:"blue",shape:"ring",text:msg.imagename});
return msg;

```

Čvor *File* zapisuje *msg.payload* u datoteku. Unutar svojstava ovog čvora nudi se mogućnost dodavanja datoteke na kraj ili zamjene postojećeg sadržaja. Također može izbrisati datoteku.

10.3.5 Tijek za prepoznavanje i prikaz prepoznate klase

Za detekciju i prikaz rezultata detekcije korišteni su sljedeći čvorovi: *cloud-annotations-gpu, ui_table, function, switch i ui_audio, debug.*

Slika 67. Prepoznavanje i prikaz prepoznate klase



Unutar čvora *cloud-Annotations* postavljena je putanja modela, koja upućuje na prethodno izrađen model.

Ui_table prikazuje rezultate klasifikacije u obliku tablice. Prvi stupac u tablici je klasa, u drugom stupcu prikazana je vjerojatnost. Na Slici 68. prikazane su postavke čvora *ui_table*.

Slika 68. Postavke čvora ui_table

Sljedeći korak je izgovor prepoznate klase. Da bi zvuk izgovorio funkciju, potrebni su čvorovi *function*, *switch* i *ui_audio*.

Funkcijski čvor koji je potreban za funkcionalnost izgovora prepoznate klase prikazan je na Slici 69. Ovaj funkcijski čvor provjerava dali je klasa objekta skalpel. Ovaj funkcijski čvor izrađen je za svaki kirurški instrument.

Slika 69. Čvor funkcije

```
let a= false;
for(let object of msg.payload){
   if (object.class === 'Skalpel'){
      a = true;
      break;
   }}
   msg.payload= a;
   return msg;
```

Switch je čvor koji usmjerava poruke na temelju njihovih vrijednosti svojstva ili položaja sekvence. Kada stigne poruka, čvor će procijeniti svako od definiranih pravila i proslijediti poruku na odgovarajuće izlaze svih odgovarajućih pravila. U ovom slučaju čvor *switch*, treba podijeliti na opcije: *'is true'* i *'otherwise'*. Ako poruka vrati vrijednost true, nastavite na sljedeću funkciju čvora, inače ne poduzmite ništa. Na Slici 70. prikazane su postavke *switch* čvora.

Slika 70. Postavke čvora switch

Ako je vrijednost istinita, nastavlja se na sljedeći funkcijski čvor koji prenosi poruku koju će reproducirati ui_audio. Ovaj je funkcijski čvor postavljen kako je prikazano na Slici 71.

Slika 71. Postavke čvora Function

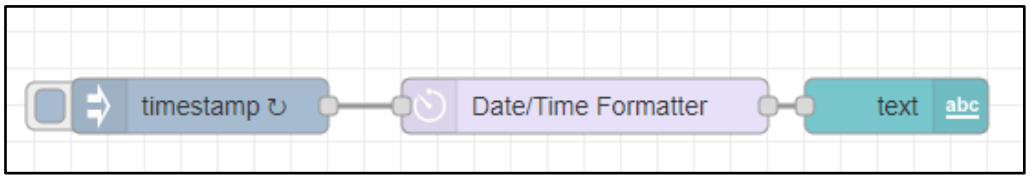
```
msg.topic = 'a';
msg.payload = 'Skalpel';
return msg;
```

Ui_audio je čvor koji reproducira audio ili tekst u govor na nadzornoj ploči. Čvor očekuje da *msg.payload* sadrži međuspremnik *wav* ili *mp3* datoteke. Unutar ovog čvora potrebno je smjestiti grupu i odabrati glas koji će izgovoriti poruku tj. prepoznatu klasu.

10.3.6 Postavljanje vremena i datuma

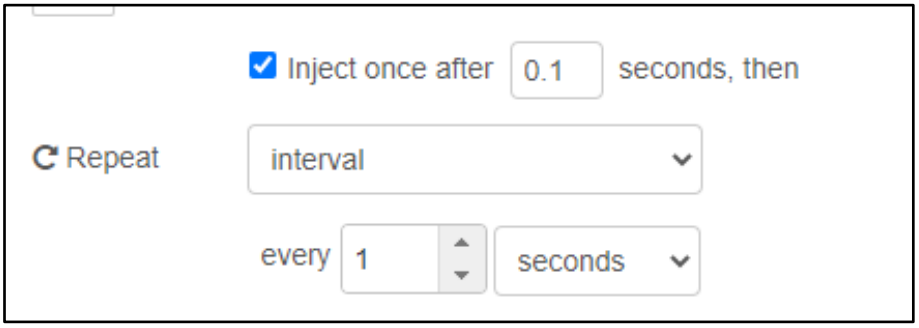
Tok prikazan na Slici 72. korišten je za prikaz datuma i vremena na nadzornoj ploči. Potrebni čvorovi su inject, date/ time formater i *ui_text.*

Slika 72. Tijek za postavljanje datuma



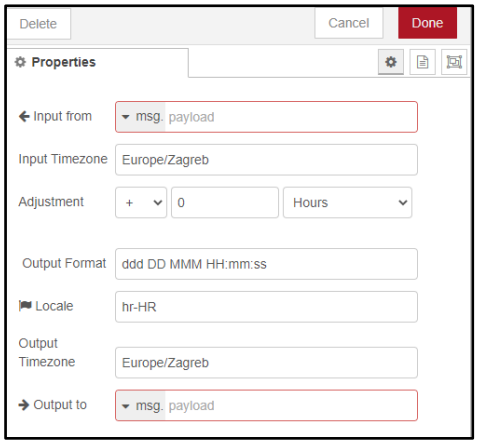
Čvor *Inject* ubrizgava poruku u tok bilo ručno ili u redovitim intervalima. Korisni teret poruke može biti raznih vrsta, uključujući nizove, JavaScript objekte ili trenutno vrijeme. U svojstvima je potrebno postaviti *msg.payload* kao vremensku oznaku. Također je potrebno prilagoditi postavke kao što je prikazano na Slici 73. kako bi se svake sekunde ažuriralo vrijeme.

Slika 73. Postavke čvora Inject



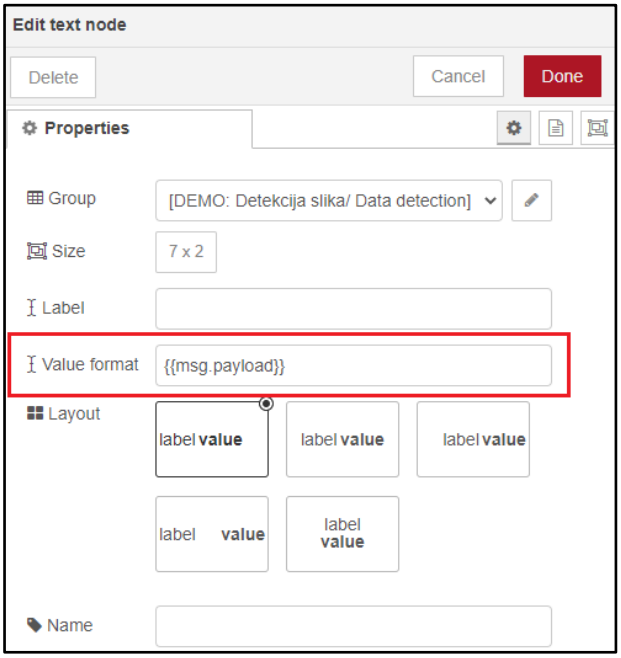
Date/ Time formatter čvor je koji objekt ili niz datuma i vremena pretvara u oblikovani tekst ili oblik datuma / vremena. Unutar ovog čvora potrebno je postaviti vremensku zonu i izlazni format. Vremenska zona postavljena na „Europa / Zagreb“. Izlazni format postavljen je na: “*ddd DD MMM HH: mm: ss*”. Postavke čvora prikazane su na Slici 74.

Slika 74. Date/ Time formatter postavke čvora



Ui_text je čvor koji na korisničkom sučelju prikazuje tekstualno polje koje se ne može uređivati. Svaka primljena msg.payload ažurirat će tekst na temelju navedenog formata vrijednosti. Unutar ovog čvora potrebno je odabrati grupu nadzorne ploče te postaviti format vrijednosti na {{*msg.payload*}} kao što je prikazano na Slici 75.

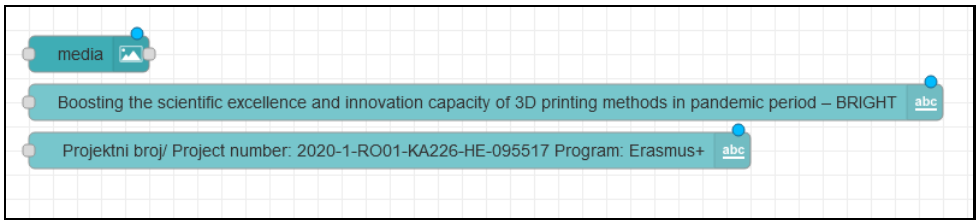
Slika 75. Postavke čvora Ui_text



10.3.7 Postavljanje logotipa i opisa projekta na sučelje

Na Slici 76. prikazan je tijek koji prikazuje logotip i opis projekta na sučelju aplikacije. Za prikaz slike na sučelju potreban je čvor: *Ui_media,* a za prikaz opisa korišten je ui_text čvor.

Slika 76. Tijek za dodavanje opisa projekta na grafičko sučelje



Čvor *Ui_media* koristi se za prikaz slike na nadzornoj ploči, ovaj čvor prikazuje medijske datoteke i URL-ove na sučelju aplikacije. Čvor Ui_text koristi se za prikaz opisa projekta na sučelju aplikacije.

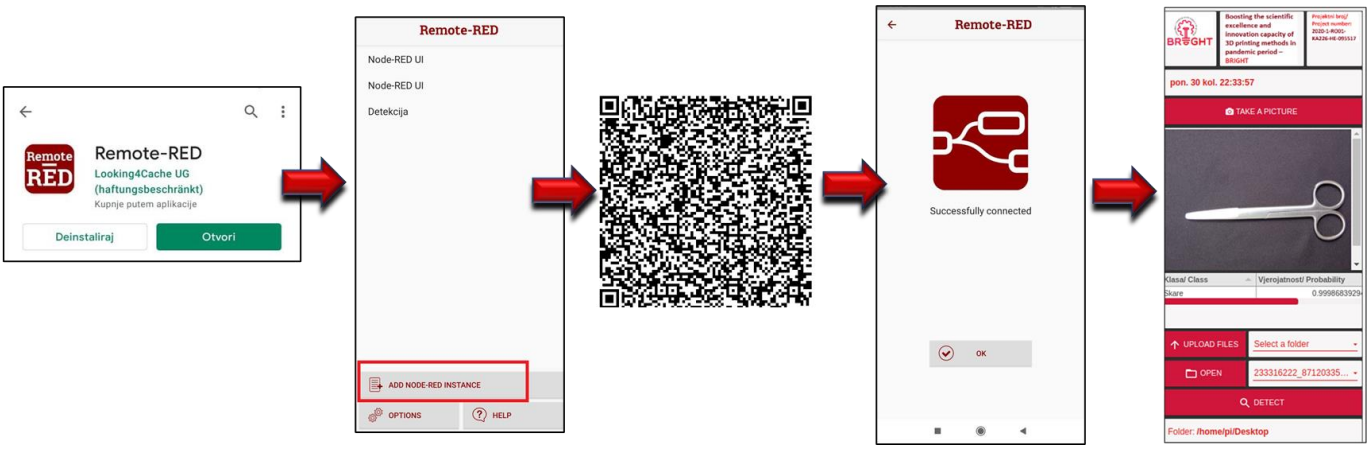
10.3.8 Daljinski pristup

Remote-access – Pristupni čvor omogućuje pristup web mjestu lokalno ili udaljeno iz mobilne aplikacije.

Preuzimanjem mobilne aplikacije Remote-RED, koja je dostupna na Google Play-u, moguće je pristupiti aplikaciji, iz udaljene lokacije. Nakon preuzimanja aplikacije odabere se opcija „Add Node-RED instance“, potom se skenira QR kod. Nakon skeniranja QR koda korisnik

dobiva potvrdu "*Successfully connected*“. Na Slici 77. prikazan je postupak daljinskom pristupu aplikaciji.

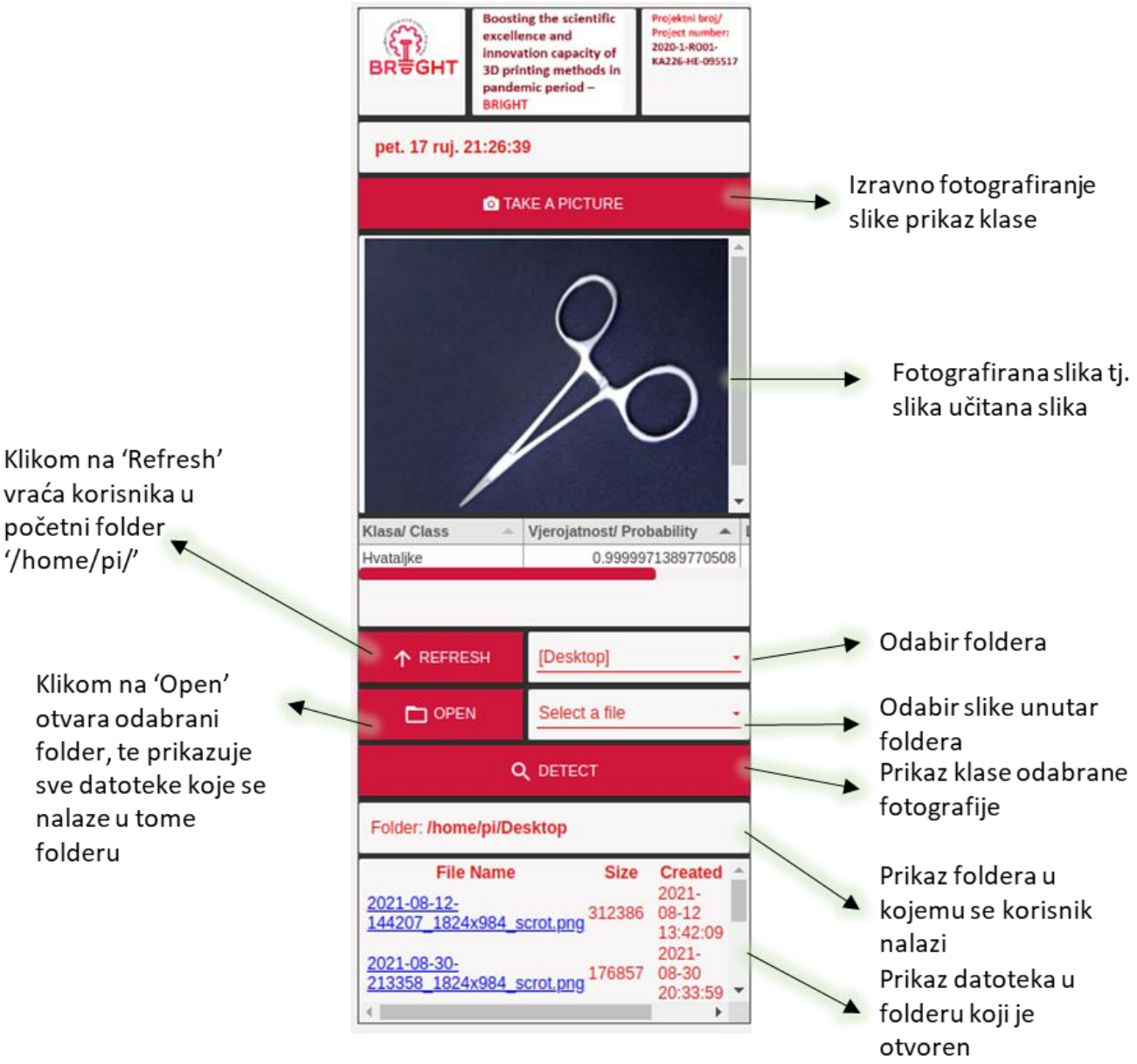
Slika 77. Daljinski pristup (Remote-RED)



10.4 Prikaz rezultata

Na Slici 78. prikazano je sučelje aplikacije. Preko kojega korisnik fotografira sliku ili je učitava s računala. U tablici se ispisuju rezultati prepoznate klase kao i vjerojatnost.

Slika 78. Prikaz aplikacije



11 Zaključak

Čovjek lako prepozna objekte i okolinu oko sebe, dok je računalnom vidu ovaj proces mnogo zahtjevniji. Zajedno s razvojem računalnih tehnologija, radi se na usavršavanju modela strojnog učenja kako bi se replicirala ljudska inteligencija. Pojam umjetne inteligencije primjenjuje široki spektar u medicini kao što je robotika, medicinska dijagnoza, medicinska statistika. Neki od najranijih radova u uspješnoj primjeni umjetne inteligencije dogodili su se u medicinskom kontekstu. Umjetna inteligencija nudi brojna rješenja u području medicine, osobito u ciljanim tretmanima te personaliziranoj terapiji. Umjetna inteligencija postaje jednostavna za korištenje i trebala bi postati ključni alat za cijeli niz biomedicinskih pitanja za unaprjeđenje razumijevanja sustava.

U ovome radu objašnjene su osnove umjetne inteligencije, strojnog i dubokog učenja. Dan je pregled umjetne inteligencije u medicini te prednosti korištenja umjetne inteligencije u raznim područjima medicine. Objašnjeno je područje računalnog vida s naglaskom na detekciju objekta te su opisana istraživanja u tom području. U radu su opisani algoritmi strojnog učenja, neuronske mreže, umjetne neuronske mreže, konvolucijske neuronske mreže te neke od poznatijih arhitektura konvolucijskih neuronskih mreža za klasifikaciju i detekciju objekta.

U praktičnom dijelu ovoga rada opisana je izrada aplikacije koja pomoću kamere prepoznaje kirurški instrument i javlja kojoj klasi taj objekt pripada. Model strojnog učenja treniran je koristeći programski jezik Python i programsku biblioteku Tensorflow, u razvojnom okruženju Colaboratory. Model je pretvoren u tensorflow.js format kako bi se mogao jednostavni koristiti u pregledniku. Za izradu grafičkog sučelja aplikacije korišteno je razvojno okruženje Node-RED. U aplikaciji je uspješno implementirana klasifikacija određenog kirurškog instrumenta, kao buduće poboljšale aplikacije nudi se mogućnost detekcije i lokalizacije koja bi omogućila prepoznavanje više kirurških instrumenata na jednoj fotografiji. Također, kao sugestija za budućnost moglo bi se navesti, veći broj kirurških instrumenata koje će aplikacija prepoznati.

12 Literatura

Članci:

1. Albawi, S., Mohammed, T. A., & Al-Zawi, S. (2017). Understanding of a convolutional neural network. 2017 International Conference on Engineering and Technology (ICET). Published.
2. Altman, R. (1999). AI in Medicine: The Spectrum of Challenges from Managed Care to Molecular Medicine. *AI Mag., 20*, 67-77.
3. Bajrami, G., Derawi, M. O., & Bours, P. (2011). Towards an automatic gait recognition system using activity recognition (wearable based). 2011 Third International Workshop on Security and Communication Networks (IWSCN). Published.
4. Baumgart, B. G. (1975). A polyhedron representation for computer vision. Proceedings of the May 19–22, 1975, National Computer Conference and Exposition on – AFIPS '75. Published.
5. Bayar, B., & Stamm, M. C. (2016). A Deep Learning Approach to Universal Image Manipulation Detection Using a New Convolutional Layer. Proceedings of the 4th ACM Workshop on Information Hiding and Multimedia Security. Published.
6. Carbonell, J. G., Michalski, R. S., & Mitchell, T. M. (1983). AN OVERVIEW OF MACHINE LEARNING. Machine Learning, 3–23.
7. Cascio, W. F., and Montealegre, R. (2016), How technology is changing work and organizations. Annu. Rev. Organ. Psychol. Organ. Behav. 3, 349–375.
8. Coskun, M., Ucar, A., Yildirim, O., & Demir, Y. (2017). Face recognition based on convolutional neural network. 2017 International Conference on Modern Electrical and Energy Systems (MEES). Published.
9. Datta, S., Barua, R., & Das, J. (2020). Application of Artificial Intelligence in Modern Healthcare System. Alginates – Recent Uses of This Natural Polymer. Published.
10. el Naqa, I., & Murphy, M. J. (2015). What Is Machine Learning? Machine Learning in Radiation Oncology, 3–11.
11. Gershenson, C. (2003). Artificial Neural Networks for Beginners. ArXiv, cs.NE/0308031.
12. Gholamalinezhad, H., & Khosravi, H. (2020). Pooling Methods in Deep Neural Networks, a Review. ArXiv, abs/2009.07485.
13. Ghosh, A., Sufian, A., Sultana, F., Chakrabarti, A., & De, D. (2019). Fundamental Concepts of Convolutional Neural Network. Intelligent Systems Reference Library, 519–567.

14. Girshick, R. (2015). Fast R-CNN. 2015 IEEE International Conference on Computer Vision (ICCV). Published.
15. Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. 2014 IEEE Conference on Computer Vision and Pattern Recognition.
16. Godfrey, K. (1985). Simple Linear Regression in Medical Research. New England Journal of Medicine, 313(26), 1629–1636.
17. Goesele, M., Curless, B., and Seitz, S. (2006). Multi-view stereo revisited. In IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'2006), New York City, NY.
18. Haleem, A., Javaid, M., & Khan, I. H. (2019). Current status and applications of Artificial Intelligence (AI) in medical field: An overview. Current Medicine Research and Practice, 9(6), 231–237.
19. Hamet, P., Tremblay, J., (2017). Artificial intelligence in medicine, Metabolism. 69.
20. Hao, X., Zhang, G., & Ma, S. (2016). Deep Learning. International Journal of Semantic Computing, 10(03), 417–439.
21. He, K., Gkioxari, G., Dollár, P., & Girshick, R.B. (2017). Mask R-CNN. 2017 IEEE International Conference on Computer Vision (ICCV), 2980-2988.
22. He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR).
23. Kamilaris, A., & Prenafeta-Boldú, F. X. (2018). Deep learning in agriculture: A survey. Computers and Electronics in Agriculture, 147, 70–90.
24. Kim, P. (2017). Convolutional Neural Network. MATLAB Deep Learning, 121–147.
25. Kingsford, C., & Salzberg, S. L. (2008). What are decision trees? Nature Biotechnology, 26(9), 1011–1013.
26. Kodali, R., Anjum, A., IoT Based HOME AUTOMATION Using Node-RED, Second International Conference on Green Computing and Internet of Things (ICGCIoT), 2018, pp. 386-390,
27. Krizhevsky, A., Sutskever, I. and Hinton, G.E. (2012) Imagenet Classification with Deep Convolutional Neural Networks. In: Pereira, F., Burges, C.J.C., Bottou, L. and Weinberger, K.Q., Eds., Advances in Neural Information Processing Systems, Vol. 25, Curran Associates, Inc., 1097-1105.

28. Lauzon, F. Q. (2012). An introduction to deep learning. 2012 11th International Conference on Information Science, Signal Processing and Their Applications (ISSPA). Published.
29. LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. Nature, 521(7553), 436–444.
30. LeCun, Y., Boser, B., Denker, J., Henderson, D., Howard, R., Hubbard, W., & Jackel, L. (1990). Handwritten Digit Recognition with a Back-Propagation Network. NIPS.
31. LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), 2278–2324.
32. Lee, R. P., & Grewal, R. (2004). Strategic Responses to New Technologies and Their Impact on Firm Performance. Journal of Marketing, 68(4), 157–171.
33. Lekic, M., & Gardasevic, G. (2018). IoT sensor integration to Node-RED platform. 2018 17th International Symposium INFOTEH-JAHORINA (INFOTEH). Published.
34. Liang, M., & Hu, X., (2015). Recurrent convolutional neural network for object recognition. 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR).
35. Ma, W., & Lu, J. (2017). An Equivalence of Fully Connected Layer and Convolutional Layer. ArXiv, abs/1712.01252.
36. Mahesh, B. (2020). Machine Learning Algorithms-A Review. International Journal of Science and Research (IJSR), 9, 381-386.
37. Maulud, D., & Abdulazeez, A. M. (2020). A Review on Linear Regression Comprehensive in Machine Learning. Journal of Applied Science and Technology Trends, 1(4), 140–147.
38. McCarthy, J. A. (1955). Search for Double Beta Decay inCa48. Physical Review, 97(5), 1234–1236.
39. Meer, P., Mintz, D., Rosenfeld, A., & Kim, D. Y. (1991). Robust regression methods for computer vision: A review. International Journal of Computer Vision, 6(1), 59–70.
40. Mishra, C., & Gupta, D. L. (2017). Deep Machine Learning and Neural Networks: An Overview. IAES International Journal of Artificial Intelligence (IJ-AI), 6(2), 66.
41. Moghaddam, B., & Pentland, A. (1997). Probabilistic visual learning for object representation. IEEE Transactions on Pattern Analysis and Machine Intelligence, 19(7), 696–710.
42. Nayyar, A., & Puri, V. (2015). Raspberry Pi-A Small, Powerful, Cost Effective and Efficient Form Factor Computer
43. Noble, W. S. (2006). What is a support vector machine? Nature Biotechnology, 24(12), 1565–1567.

44. Nwankpa, C., Ijomah, W., Gachagan, A., & Marshall, S. (2018). Activation functions: Comparison of trends in practice and research for deep learning. arXiv preprint arXiv:1811.03378.
45. Park, C., Took, C. C., & Seong, J. K. (2018). Machine learning in biomedical engineering. Biomedical Engineering Letters, 8(1), 1–3.
46. Rajalakshmi, A., & Shahnasser, H. (2017). Internet of Things using Node-Red and alexa. 2017 17th International Symposium on Communications and Information Technologies (ISCIT). Published.
47. Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Published.
48. Ren, S., He, K., Girshick, R., & Sun, J. (2017). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. IEEE Transactions on Pattern Analysis and Machine Intelligence, 39(6), 1137–1149.
49. Rhoads, D. D. (2020). Computer Vision and Artificial Intelligence Are Emerging Diagnostic Tools for the Clinical Microbiologist. Journal of Clinical Microbiology, 58(6).
50. Rieke, N., Tombari, F., & Navab, N. (2018). Computer Vision and Machine Learning for Surgical Instrument Tracking. Computer Vision for Assistive Healthcare, 105–126.
51. Rokach L., Maimon O. (2005) Decision Trees. In: Maimon O., Rokach L. (eds) Data Mining and Knowledge Discovery Handbook. Springer, Boston, MA.
52. Ruder, S., (2016). An overview of gradient descent optimization algorithms, Insight Centre for Data Analytics, NUI Galway Aylien Ltd., Dublin
53. Samuel, A. L. (1959). Some Studies in Machine Learning Using the Game of Checkers. IBM Journal of Research and Development, 3(3), 210–229.
54. Schmidhuber, J. (2015). Deep learning in neural networks: An overview. Neural Networks, 61, 85–117.
55. Sharma, S., Sharma, S., & Athaiya, A. (2020). ACTIVATION FUNCTIONS IN NEURAL NETWORKS. International Journal of Engineering Applied Sciences and Technology, 04(12), 310–316.
56. Shu, G., Dehghan, A., Oreifej, O., Hand, E., & Shah, M. (2012). Part-based multiple-person tracking with partial occlusion handling. 2012 IEEE Conference on Computer Vision and Pattern Recognition. Published.
57. Sicari, S., Rizzardi, A., & Coen-Porisini, A. (2019). Smart transport and logistics: A Node-RED implementation. Internet Technology Letters, 2(2), e88.

58. Sivic, J., Zitnick, C. L., & Szeliski, R. (2006). Finding people in repeated shots of the same scene. Procedings of the British Machine Vision Conference 2006. Published.
59. Snavely, N., Seitz, S. M., & Szeliski, R. (2007). Modeling the World from Internet Photo Collections. International Journal of Computer Vision, 80(2), 189–210.
60. Sultana, F., Sufian, A., & Dutta, P. (2020). A Review of Object Detection Models Based on Convolutional Neural Network. Advances in Intelligent Systems and Computing, 1–16.
61. Suthaharan, S. (2016). Support Vector Machine. Machine Learning Models and Algorithms for Big Internet Classification, 207–235.
62. Szegedy, C., Wei Liu, Yangqing Jia, Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., & Rabinovich, A. (2015). Going deeper with convolutions. 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Published.
63. Utgoff, P. E. (1989). Incremental Induction of Decision Trees. Machine Learning, 4(2), 161–186.
64. Wu, J., (2017). Introduction to Convolutional Neural Networks, National Key Lab for Novel Software Technology
65. Xie, X., Du, D., Li, Q., Liang, Y., Tang, W. T., Ong, Z. L., Lu, M., Huynh, H. P., & Goh, R. S. M. (2018). Exploiting Sparsity to Accelerate Fully Connected Layers of CNN-Based Applications on Mobile SoCs. ACM Transactions on Embedded Computing Systems, 17(2), 1–25.
66. Yamashita, R., Nishio, M., Do, R. K. G., & Togashi, K. (2018). Convolutional neural networks: an overview and application in radiology. Insights into Imaging, 9(4), 611–629.
67. Yu, D., Wang, H., Chen, P., & Wei, Z. (2014). Mixed Pooling for Convolutional Neural Networks. Rough Sets and Knowledge Technology, 364–375.
68. Zhang XD. (2020) Machine Learning. In: A Matrix Algebra Approach to Artificial Intelligence. Springer, Singapore.
69. Zhao, B., Lu, H., Chen, S., Liu, J., & Wu, D. (2017). Convolutional neural networks for time series classification. Journal of Systems Engineering and Electronics, 28(1), 162–169.
70. Zhao, C.W., Jegatheesan, J., & Loon, S.C. (2015). Exploring IOT Application Using Raspberry Pi.

Knjige:

1. Alpaydin, E. (2014). Introduction to Machine Learning. Amsterdam University Press.
2. Bellman, R. E. (1978). An Introduction to Artificial Intelligence: Can Computers Think? Boyd & Fraser Publishing Company, San Francisco.
3. Charniak, Eugene & Mcdermott, Drew. (1986). Introduction to Artificial Intelligence.
4. Encyclopedia Britannica (1991). Encyclopedia Britannica, London
5. Ertel, W. (2011). Introduction to Artificial Intelligence. Springer.
6. Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. Amsterdam University Press.
7. Gurney, K. (1997). An Introduction to Neural Networks (1st ed.). CRC Press.
8. Jiang, X., Hadid, A., Pang, Y., Granger, E., & Feng, X. (Eds.). (2019). Deep Learning in Object Detection and Recognition. doi:10.1007/978-981-10-5152-4
9. Kurzweil, R. (1990). The Age of Intelligent Machines. MIT Press, Cambridge, Massachusetts.
10. Mohri, M., Rostamizadeh, A., & Talwalkar, A. (2018). Foundations of machine learning. MIT press.
11. Nilsson, N., (1998). Introduction to Machine Learning, Department of Computer Science Stanford University, Stanford
12. Rich, E., (1983). Artificial Intelligence. McGraw–Hill, New York
13. Russell, S., Norvig, P. (2010). Artificial Intelligence: A Modern Approach – 3rd Edition. Pearson.
14. Schalkoff, R. I. (1990). Artificial Intelligence: An Engineering Approach. McGraw-Hill, New York.
15. Szeliski, R. (2010). *Computer Vision: Algorithms and Applications (Texts in Computer Science)* (2011th ed.). Springer.
16. Zhang, A., C. Lipton, Z., Li, M., J. Smola, A. (2021). Dive into Deep Learning

Internet izvori:

1. Agarwal, S. (2020). LeNet-5 CNN Architecture - Shaleen Agarwal. Medium. https://medium.com/@coolshaleen1/lenet-5-cnn-architecture-468ed7b895cd
2. Alake, R. (2020). Understanding and Implementing LeNet-5 CNN Architecture (Deep Learning). Medium. https://towardsdatascience.com/understanding-and-implementing-lenet-5-cnn-architecture-deep-learning-a2d531ebc342
3. Brownlee, J. (2020). Linear Regression for Machine Learning. Machine Learning Mastery. https://machinelearningmastery.com/linear-regression-for-machine-learning/
4. Gandhi, R. (2018). Support Vector Machine — Introduction to Machine Learning Algorithms. Medium. https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47
5. Javatpoint (2021). Logistic Regression in Machine Learning, Www.Javatpoint.Com. https://www.javatpoint.com/logistic-regression-in-machine-learning
6. Little, Z. (2020). Activation Functions (Linear/Non-linear) in Deep Learning —. Medium. https://xzz201920.medium.com/activation-functions-linear-non-linear-in-deep-learning-relu-sigmoid-softmax-swish-leaky-relu-a6333be712ea
7. Node-RED, node-red-contrib-usbcamera: https://flows.nodered.org/node/node-red-contrib-usbcamera
8. Serengil, S. (2020). Step Function as a Neural Network Activation Function. Sefik Ilkin Serengil. https://sefiks.com/2017/05/15/step-function-as-a-neural-network-activation-function/
9. Goyal, K. (2021). 6 Types of Activation Function in Neural Networks You Need to Know. UpGrad Blog. https://www.upgrad.com/blog/types-of-activation-function-in-neural-networks/
10. Abadi, M. i sur. (2016). TensorFlow: A System for Large-Scale Machine Learning | USENIX. Dostupno na: https://www.usenix.org/conference/osdi16/technical-sessions/presentation/abadi

Alati:

1. Cloud Annotations: https://cloud.annotations.ai/
2. Google Colaboratory: https://research.google.com/colaboratory/
3. Node-RED: https://nodered.org/
4. Tensorflow: https://www.tensorflow.org/

13 Popis slika

14 Popis tablica

Sažetak

Posljednjih nekoliko godina tehnologije zasnovane na umjetnoj inteligenciji brzo su napredovale, tijekom prethodna dva desetljeća svjedoci smo velikog napretka u umjetnoj inteligenciji i njenoj primjeni. Neki od najranijih radova u uspješnoj primjeni umjetne inteligencije dogodili su se upravo u medicinskom kontekstu.

U ovome radu objašnjene su osnove umjetne inteligencije, strojnog i dubokog učenja. Dat je pregled umjetne inteligencije u medicini te prednosti njenog korištenja u raznim područjima medicine. Objašnjeno je područje računalnog vida s naglaskom na detekciju objekta te su opisana neka istraživanja u tom području. U radu su opisani algoritmi strojnog učenja, neuronske mreže, umjetne neuronske mreže te konvolucijske neuronske mreže. U sklopu ovoga rada izrađena je aplikacija koja prepoznaje kirurške instrumente. U izradi aplikacije korišteni su sljedeći razvojni alati: Cloud Annotations, Colaboratory i Node-RED. Za trening modela poslužila je biblioteka strojnog učenja Tensorflowi programski jezik Python. Model strojnog učenja u Tensorflow.js formatu, ukomponiran je s grafičkim sučeljem, izrađenim koristeći razvojni alat Node-RED i programski jezik JavaScript.

Aplikacija korisniku omogućuje izravno fotografiranje slike te učitavanje slike s računala za njeno prepoznavanje. Aplikacija je prepoznaje pet različitih kirurških instrumenata: zakrivljene kirurške škare, ravne kirurške škare, skalpel, kirurška pinceta te hvataljke.

Ključne riječi: *Računalni vid, umjetna inteligencija, strojno učenje, Node-RED, Tensorflow.js*

Abstract

In the last few years, technologies based on artificial intelligence have advanced rapidly, and over the past two decades we have witnessed great progress in artificial intelligence and its application. Some of the earliest work in the successful application of artificial intelligence occurred precisely in the medical context.

This paper explains the basics of artificial intelligence, machine and deep learning. An overview of artificial intelligence in medicine field and the advantages of its use in various fields of medicine is given. The area of computer vision is explained with an emphasis on object detection and some research in this area is described. The paper describes machine learning algorithms, neural networks, artificial neural networks and convolutional neural networks. As part of this thesis, an application was developed that classifies surgical instruments. The following development tools were used in the development of the application: Cloud Annotations, Collaborative and Node-RED. The Tensorflow machine learning library was used for model training and the Python programming language. The machine learning model in Tensorflow.js format is integrated with a graphical interface, created using the development tool Node-RED and the JavaScript programming language.

The application allows the user to take a photo directly and download the image from a computer to recognize it. The app recognizes five different surgical instruments: curved surgical scissors, flat surgical scissors, scalpel, surgical tweezers, and forceps.

Keywords: Computer vision, artificial intelligence, machine learning, Node-RED, Tensorflow.js