Automatizacija procesa sterilizacije korištenjem robotskih manipulatora

Car, Antonio

SVEUČILIŠTE U RIJECI

TEHNIČKI FAKULTET

Diplomski sveučilišni studij elektrotehnike

Diplomski rad

AUTOMATIZACIJA PROCESA STERILIZACIJE KORIŠTENJEM ROBOTSKIH MANIPULATORA

Rijeka, srpanj 2024.

Antonio Car
0069084247

SVEUČILIŠTE U RIJECI

TEHNIČKI FAKULTET

Diplomski sveučilišni studij elektrotehnike

Diplomski rad

AUTOMATIZACIJA PROCESA STERILIZACIJE KORIŠTENJEM ROBOTSKIH MANIPULATORA

Mentor: prof. dr. sc. Zlatan Car

Komentor: Sandi Baressi Šegota, mag. ing. comp.

Rijeka, srpanj 2024.

Antonio Car
0069084247

SVEUČILIŠTE U RIJECI
TEHNIČKI FAKULTET
POVJERENSTVO ZA DIPLOMSKE ISPITE

Rijeka, 14.03.2024.

Zavod: Zavod za automatiku i elektroniku

Predmet: Osnove robotike

ZADATAK ZA DIPLOMSKI RAD

Pristupnik: **Antonio Car (0069084247)**

Studij: Sveučilišni diplomski studij elektrotehnike (1300)

Modul: Automatika (1331)

Zadatak: **Automatizacija procesa sterilizacije korištenjem robotskih manipulatora / Automation of sterilization process using robotic manipulators**

Opis zadatka:

Izvršiti pregled literature iz područja sterilizacije i klasifikacije objekata na slici. Prikupiti set podataka koji se sastoji od slika alata koji se steriliziraju u domeni javnog zdravstva. Trenirati model temeljen na algoritmu strojnog učenja i komentirati rezultate. Izvršiti izbor primjerenog industrijskog robotskog manipulatora, te isprogramirati robotski manipulator za vršenje kompletnog procesa sterilizacije, te isti proces simulirati. Diplomski rad mora biti napisan prema uputama za pisanje diplomskih radova koje su objavljene na mrežnim stranicama studija.

Rad mora biti napisan prema Uputama za pisanja diplomskih / završnih radova koje su objavljene na mrežnim stranicama studija.

Zadatak uručen pristupniku: 20.03.2024.

Mentor:
prof. dr. sc. Zlatan Car

Komentor:
Sandi Baressi Šegota

Predsjednik povjerenstva za diplomski ispit:
prof. dr. sc. Dubravko Franković

IZJAVA

Sukladno članku 9. Pravilnika o diplomskom radu, diplomskom ispitu i završetku diplomskih sveučilišnih studija Tehničkog fakulteta Sveučilišta u Rijeci izjavljujem da sam izradio ovaj diplomski rad samostalno, koristeći vlastito znanje i navedenu literaturu, u razdoblju od datuma zadavanja zadatka do datuma predaje.

Rijeka, 02. srpnja 2024.

Antonio Car

Želim izraziti duboku zahvalnost svojoj obitelji koja je od prvog dana školovanja bila moja najveća podrška. U trenucima kada ništa nije funkcioniralo ili nije išlo po planu, oni su uvijek bili uz mene. Također, zahvaljujem prof. dr. sc. Zlatanu Caru na prihvaćanju mentorstva za izradu rada i asist. Sandiju Baressi Šegoti, mag. ing. comp. na uputama i pomoći tijekom izrade.

Sadržaj

1. Uvod

U ovom radu opisan je postupak automatizacije procesa sterilizacije korištenjem Robot Studio programskog paketa i programskog jezika Python. Sterilizacija predstavlja ključan postupak u medicinskim okruženjima, koji uključuje uništavanje ili uklanjanje svih živih vrsta mikroorganizama poput bakterija, virusa, gljivica i spora.

S medicinskog stajališta, proces sterilizacije zahtijeva određeno vrijeme i angažman osoblja. Obuhvaćen je niz koraka, uključujući slaganje medicinskog pribora u setove koji se koriste tijekom operacija, postavljanje setova u posude, prijenos posuda do sterilizatora te čekanje na izvršenje procesa sterilizacije. Nakon završetka, postupak se ponavlja kako bi se kompletni pribor pripremio za novo korištenje.

U prvom dijelu rada opisan je povijesni razvoj procesa sterilizacije, izvođenje procesa sterilizacije te različite primjene. Nadalje, provedena je usporedba s procesom dezinfekcije, naglašavajući razlike između ta dva procesa te njihove specifične primjene.

Drugi dio rada posvećen je radu unutar Robot Studio programskog paketa i modeliranju objekata korištenih za simulaciju. Objekti su modelirani u softverima FreeCAD i SolidWorks, a potom su objekti korišteni prilikom izrade simulacije unutar Robot Studio programskog paketa u kojem je korišten robot IRB 1200. Detaljno je opisana interakcija pametnih komponenti s ostalim elementima, naglašavajući jednostavnost rada s pametnim komponentama. Nadalje, obrađeno je stvaranje putanja robota, kreiranje točaka izvršenja (targeta) te sinkronizacija putanja u RAPID kod, što omogućava pokretanje simulacije. Na samom kraju poglavlja, opisan je i način snimanja simulacije tijekom njezina izvršenja.

Uz to, istražene su mogućnosti primjene dubokog učenja za prepoznavanje predmeta unutar sterilizacijskog okruženja. Izrađeni su i obučeni modeli dubokog učenja za prepoznavanje različitih predmeta korištenih u procesu sterilizacije, a zatim su integrirani unutar Robot Studio programskog paketa. Automatizirano prepoznavanje predmeta omogućava prilagodbu rada robota prema predmetima, a time se definitivno povećava efikasnost samog procesa sterilizacije.

2. Sterilizacija

Sterilizacija je postupak kojim se uništavaju svi oblici živih mikroorganizama. Ovaj proces radi se kako bi se spriječio prijenos uzročnika bolesti s predmeta koji je kontaminiran. Medicinski djelatnici imaju dužnost svakog pacijenta zbrinuti prema određenim standardima, a kako bi dostigli razinu tog standarda, medicinski pribor, kojim pristupaju prema pacijentima, mora biti sterilan.

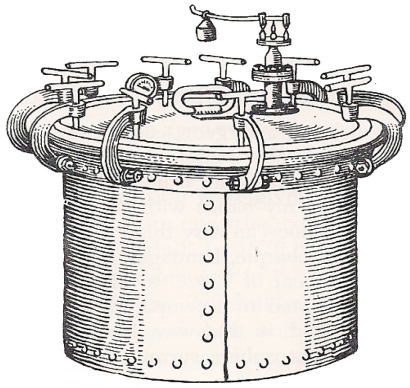
2.1. Povijesni razvoj sterilizacije

Sterilizacija se spominje još u razdoblju prije Krista u vrijeme stare Grčke, no prvi značajniji korak prema procesu sterilizacije kakav danas poznajemo napravio je francuski kemičar i mikrobiolog Louis Pasteur. 1862. godine objavio je svoja otkrića o tome kako mikroorganizmi mogu uzrokovati bolesti. Zahvaljujući tim saznanjima, Pasteur se smatra jednim od najvažnijih znanstvenika u području mikrobiologije i imunologije.

Nekoliko godina kasnije, točnije 1867. godine, britanski kirurg i znanstvenik Joseph Lister značajno je smanjio stopu smrtnosti svojih pacijenata uvodeći korištenje karbolne otopine na ranama, priboru koji dolazi u dodir s ranom, te dezinfekciju kirurških instrumenata i ruku. Lister je poznat po svojim revolucionarnim doprinosima u području kirurgije.

2.1.1. Prvi sterilizator

Slika 2.1 prikazuje prvi sterilizator koji je radio na principu pare, a osmislio ga je, prema [1], 1876. godine Charles Chamberland, poznati francuski mikrobiolog i bakteriolog, inače učenik i suradnik Louisa Pasteura.



Slika 2.1. Prvi sterilizator, izumitelj - Charles Chamberland, Izvor: [2]

2.2. Proces sterilizacije i njegova primjena

Pribor koji se koristi u operacijama treba prije same sterilizacije proći kroz proces pranja, dezinfekcije, ispiranja, te na kraju sušenja. U bolnicama, ovisno o radnom postupku, pribor se nakon ovih koraka može razvrstati u setove ili prema vrsti. Važno je prije stavljanja pribora u posudu za sterilizaciju postaviti indikatorsku traku koja će jasno signalizirati je li proces sterilizacije uspješno proveden.

Nakon postavljanja trake, pribor se stavlja u hermetički zatvorenu kutiju i prenosi u sterilizator. Kutija se postavlja na odgovarajuću poziciju, a sterilizator se uključuje i mora postići temperaturu od 180 °C. Proces sterilizacije medicinskog pribora traje oko jednog sata, nakon čega se pribor ostavlja u istoj kutiji oko sat vremena kako bi se ohladio. Po isteku odgovarajućeg vremena, kutija se izvadi iz sterilizatora, te se zatim kontroliraju indikatorske trake.

Postoje različite metode sterilizacije koje se koriste ovisno o materijalu od kojeg je izrađen predmet koji se sterilizira.

2.2.1. Primjena

Veliki napredak u proizvodnji poluvodiča postavio je nove zahtjeve za optimalnu proizvodnju. Parametri poput kvalitete zraka, temperature i vlage moraju se pažljivo kontrolirati kako bi se osigurala visoka razina čistoće. Važno je osigurati minimalan kontakt komponenti s prašinom i mikrobima koji se mogu prenositi zrakom, što bi moglo ugroziti proizvodnju poluvodiča. Slika 2.2 prikazuje dio tvornice u kojoj se proizvode poluvodiči.

Slične principe primjene možemo vidjeti i u prehrambenoj industriji. Prema [4], vrijeme i temperatura potrebni za sterilizaciju ovise o različitim faktorima, pri čemu je jedan od ključnih faktora vrsta mikroorganizama prisutnih u hrani. Procesi sterilizacije su dizajnirani kako bi uništili bakterije prisutne u hrani.

Slika 2.2. Prostor za proizvodnju poluvodiča, Izvor: [5]

2.3. Postupak izvođenja procesa sterilizacije

Nastavno na prethodno poglavlje u kojem je načelno opisan proces sterilizacije, u ovom poglavlju se opisuje svaki korak izvršenja procesa sterilizacije:

1. priprema pribora - prvi korak procesa sterilizacije medicinskog pribora je priprema koja je prikazana na slici 2.3. Sam pojam pripreme pribora sastoji se od mehaničkog uklanjanja vidljivih nečistoća kao što su krv, tkivo ili neki ostaci pomoću sapuna, vode i četke, a zatim potapanja u antiseptičko sredstvo i sušenja. Treba istaknuti da se danas za takve procese već polako uvode i automatski sustavi čišćenja.

Slika 2.3. Postupak pripreme pribora za sterilizaciju, Izvor: [8]

2. pakiranje pribora - nakon što su mehanički uklonjene sve vidljive nečistoće s pribora, sljedeći korak je pakiranje pribora u vrećice koje omogućavaju da pribor ostane sterilan nakon procesa sterilizacije kako je prikazano na slici 2.4.

Slika 2.4. Spakirani pribor spreman za postupak sterilizacije, Izvor: [9]

3. podešavanje sterilizatora i sam proces sterilizacije - podešavanje sterilizatora važan je korak budući da svaki proizvođač ima predviđene temperature pod koje je moguće postaviti pribor kao i vrijeme koliko pribor može biti izložen određenoj temperaturi. Pribor se, kako je

prikazano na slici 2.4, u ladicu postavlja tako da svaki komad bude maksimalno izložen tijekom postupka sterilizacije.

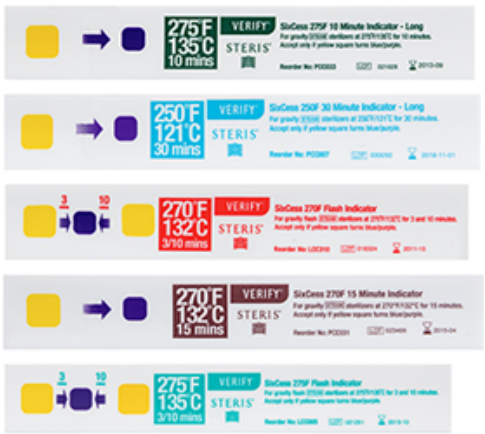
4. hlađenje i sušenje pribora nakon izvršenja procesa sterilizacije - sljedeći korak je hlađenje i sušenje pribora. Iznimno je važno pribor koji je izašao iz sterilizator ohladiti i osušiti jer vlažan pribor savršena je lokacije za nakupljanje mikroorganizama. Vlaga na priboru pojavljuje se zbog iznimno visoke temperature unutar procesa sterilizacije. Posljednji, ali ne manje važni faktor je i sigurnost djelatnika. Vrući pribor koji je izvučen iz sterilizatora mogao bi potencijalno izazvati opekline kod djelatnika koji rade u tom području budući da se kod procesa sterilizacije dostižu temperature do 180 °C.

5. testiranje sterilnosti - prije samog odlaganja steriliziranog pribora na za to predviđeno mjesto, potrebno je provjeriti uspješnost procesa sterilizacije. Uspješnost procesa sterilizacije moguće je testirati na sljedeće načine:

   a) biološkim indikatorima koji se smatraju najpouzdanijim indikatorima. U ovom slučaju se priboru dodaje indikatorska trakica sa sporama. Po završetku sterilizacije i laboratorijske analize indikatora može se utvrditi uspješnost procesa sterilizacije.

   b) kemijski indikatori, prikazani na slici 2.5, koriste se na način da se prati promjena boje indikatorske trakice. U slučaju dostizanja predviđene temperature, dolazi do promjene boje indikatorske trakice. Treba napomenuti da kemijski indikatori ne daju konkretno informaciju o sterilnosti pribora.

   c) fizikalni testovi mogu služiti za ispitivanje sterilnosti pomoću provjere sustava za filtriranje vode ili zraka. Ako se filtar kontaminira, možemo zaključiti da proces sterilizacije nije bio uspješno proveden.



Slika 2.5. Kemijski indikatori uspješnosti procesa sterilizacije, Izvor: [10]

Prema [3], proces dezinfekcije pomaže u smanjenju broja mikroorganizama u nekoj sredini. Suprotno procesu sterilizacije, koji uništava sve žive mikroorganizme, važno je napomenuti da se

procesom dezinfekcije ne ubijaju svi prisutni mikroorganizmi. Dezinfekcija smanjuje broj mikroorganizama do razine koja više nije opasna po zdravlje.

Primjenu procesa dezinfekcije moguće je pronaći u svakodnevnom životu. Jedan od primjera je bio aktualan tijekom pandemije COVID-19, kada su se u svakoj prostoriji nalazili dezinficijensi za dezinfekciju ruku. Također, dezinfekcija se primjenjivala na površinama koje se često dodiruju, kao što su kvake, stolovi, i javni prijevozni sustavi, kako bi se smanjila mogućnost prijenosa virusa. Dodatni primjeri uključuju redovito čišćenje i dezinfekciju privatnih i radnih prostora, kao i dezinfekciju medicinske opreme u zdravstvenim ustanovama.

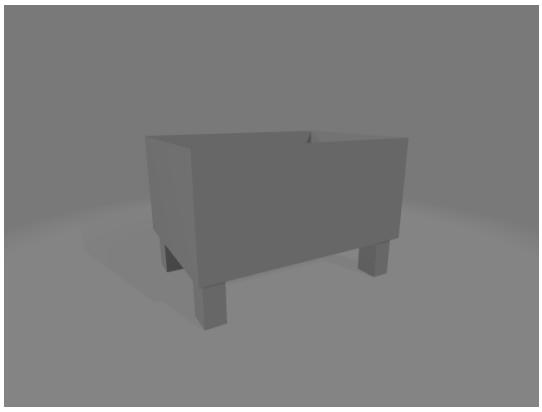
3. Automatizacija procesa sterilizacije u Robot Studio programskom paketu

Sam proces sterilizacije moguće je s pomoću robota izvesti na više različitih načina. U ovom radu predviđeno je prenošenje medicinskog pribora koji je spreman za proces sterilizacije u posudu koja se potom hermetički zatvara. Posuda se potom prenosi se u sterilizator koji je prethodno otvoren. Po postavljanju posude unutar sterilizatora, robot zatvara sterilizator te čeka nekoliko trenutaka do završetka samog procesa sterilizacije.

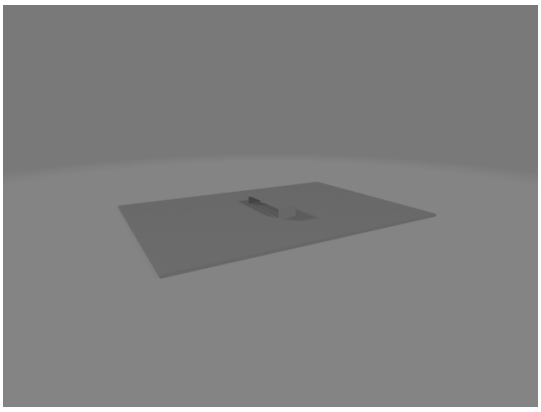
3.1. Izrada komponenti potrebnih za proces sterilizacije

Za automatizaciju procesa sterilizacije medicinskog pribora, potrebno je modelirati nekoliko komponenti koje će poslužiti za izvođenje samog procesa. Slike 3.1 i 3.2 prikazuju sterilizator bez poklopca i pripadajući poklopac. Medicinski pribor koji prolazi kroz proces sterilizacije prikazan je slikama 3.3, 3.4 i 3.5. Na samom kraju, na slici 3.7 prikazana je posuda bez poklopca u koju će biti smještan medicinski pribor. Slika 3.6 prikazuje poklopac koji služi za hermetičko zatvaranje posude unutar koje se nalazi pribor sa slike 3.3, 3.4 i 3.5. Izgled modeliranih objekata je sljedeći:

1. sterilizator s poklopcem koji je montiran na vodilice.

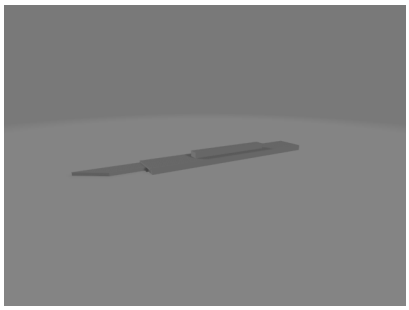


Slika 3.1. Sterilizator bez poklopca

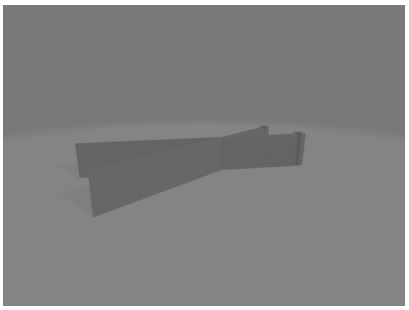


Slika 3.2. Poklopac sterilizatora

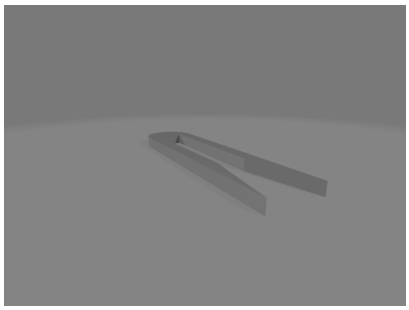
2. medicinski pribor - modelirane su tri različite komponente medicinskog pribora koje predstavljaju: medicinski skalpel, škare i medicinsku pincetu. Sve tri komponente nalaze se na prvom stolu.



Slika 3.3. Medicinski skalpel

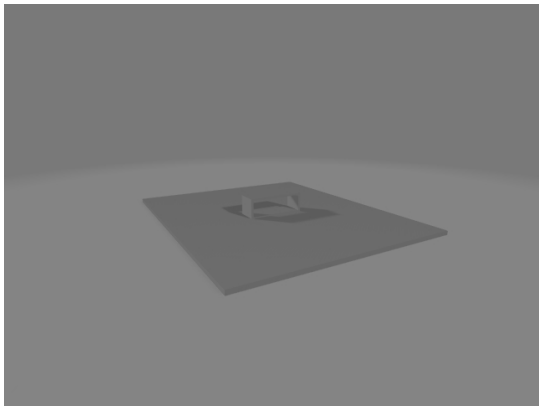


Slika 3.4. Medicinske škare

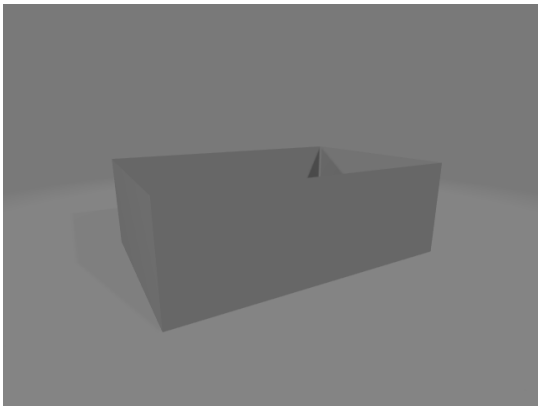


Slika 3.5. Medicinska pinceta

3. kutija s poklopcem - predviđena je za postavljanje pribora u nju, a potom za prijenos u sterilizator te na kraju povratak na početnu poziciju.



Slika 3.6. Poklopac posude



Slika 3.7. Posuda bez poklopca

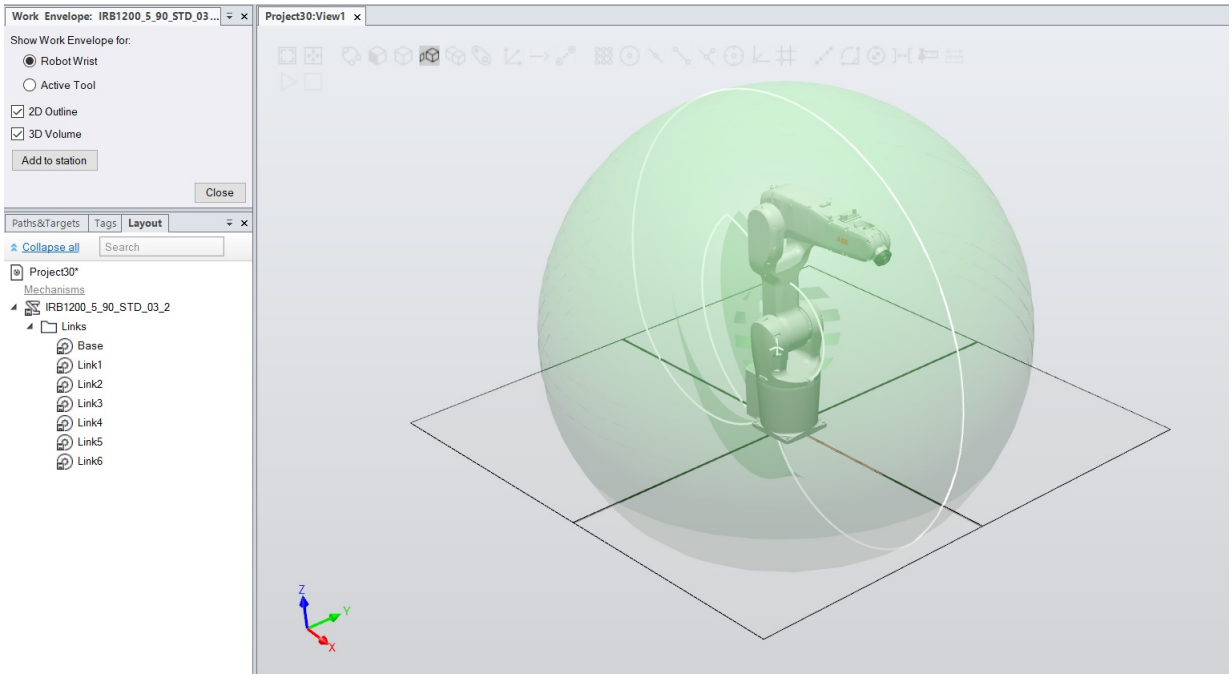
Svaka od navedenih komponenti modelirana je korištenjem softvera FreeCAD, koji je besplatan alat za računalom potpomognuto crtanje (CAD). Međutim, uvoz modela iz FreeCAD-a u Robot Studio programski paket zahtijeva konverziju u odgovarajući format. Nažalost, FreeCAD ne podržava izravan izvoz u Standard ACIS Text (.SAT) format, koji je potreban za uvoz u Robot Studio.

Kako bi se prevladala ova prepreka, koristi se dodatni softver za računalom potpomognuto crtanje, poput SolidWorks-a. SolidWorks je popularan i moćan 3D CAD softver koji omogućuje stvaranje, simuliranje i analizu 3D modela. Nakon što je komponenta modelirana u FreeCAD-u, izvozi se u .step format datoteke. Zatim se koristi SolidWorks za pretvorbu te .step datoteke u .SAT format datoteke.

Kada se postigne potrebna transformacija formata datoteke, sve predviđene komponente mogu se uvesti u radni prostor unutar Robot Studio programskog paketa. Ovaj proces omogućuje integraciju modeliranih komponenata u simulaciju automatizacije procesa sterilizacije unutar Robot Studio-a.

3.2. Robot i njegova konfiguracija

Za automatizaciju procesa sterilizacije medicinskog pribora, odabran je robot švicarsko - švedske tvrtka ABB (Asea Brown Boveri) koji se bave raznim inovacijama u području elektrotehnike, automatizacije, energetike i robotike, IRB 1200. Navedeni robot rješava dosta problema u industriji vezano za održavanje strojeva. Robot je iznimno fleksibilan, jednostavan za upotrebu, kompaktan te ima veliki radni prostor koji je prikazan na slici 3.8.



Slika 3.8. Prikaz radnog prostora robota IRB 1200 unutar Robot Studio programskog paketa.

3.2.1. Ključne značajke IRB 1200

Prema tablici 3.1, robot IRB 1200 moguće je nabaviti u tri različite varijante. Varijante robota ovise o masi i dometu samog robota. Za potrebe ovog rada korišten je robot 5 kg - 0.9 m.

Varijanta robota	Masa robota	Domet robota
IRB 1200 7 Kg / 0.9 m	7 kg	0.9 m
IRB 1200 7 Kg / 0.7 m	7 kg	0.7 m
IRB 1200 5 Kg / 0.9 m	5 kg	0.9 m

Tablica 3.1. Omjer mase i dometa robota kod IRB 1200., Izvor: [6]

Sve varijante robota iz tablice 3.1 mogu se montirati pod bilo kojim kutom, a treba istaknuti da svi dolaze s IP 40 zaštitom kao standardom.

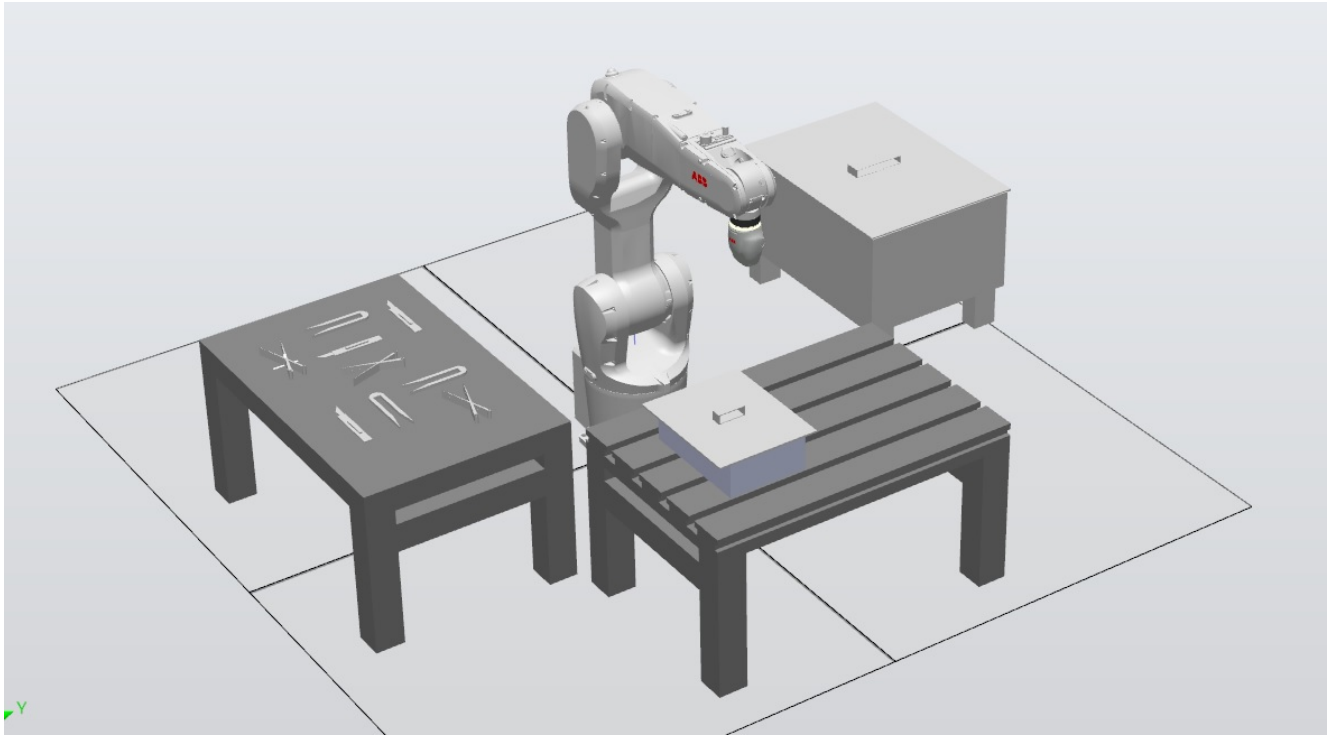
3.2.2. Kompaktnost IRB 1200

Stupanj slobode (DOF) je broj nezavisnih varijabli s pomoću kojih se definira moguća pozicija robota ili njegova kretnja unutar radnog prostora. Odabrani IRB 1200 ima šest stupnjeva slobode. U drugoj osi, IRB 1200 nema pomaka, a to nam osigurava duži korak nego kod manjih robota.

Duži korak nam omogućava pozicioniranje radnog objekta bliže samom robotu, a da se pritom ne izgubi funkcionalnost robota. Zbog ne postojanja pomaka u drugoj osi robota, omogućeno je kompaktnije postavljanje kada je robot učvršćen na strop unutar nekog radnog prostora. Primjer kada je robot učvršćen na strop pronalazimo u nekim vrstama obrade ili poliranja.

3.3. Radni prostor procesa sterilizacije

Navedene su sve komponente koje će se koristiti u samom procesu sterilizacije, objašnjen je način kretanja samog robota te izvršenje zadatka. Kako bi se lakše vizualiziralo prethodno napisano, na slici 3.9 prikazan je radni prostor unutar Robot Studio programskog paketa koji služi za automatizaciju procesa sterilizacije.



Slika 3.9. Prikaz radnog prostora u Robot Studio programskom paketu

Na lijevom stolu nalazi se razbacani pribor koji će biti razvrstan te će se prebacivati u posudu s poklopcem koja se nalazi na stolu ispred samog robota. Ovisno o zadanom zadatku, robot s pomoću umjetne inteligencije prepoznaje određeni komad medicinskog pribora, uzima ga te prebacuje na predviđeno mjesto (u posudu).

Posuda za pribor koja se nalazi na stolu ispred robota ima svoj poklopac koji se podiže na određenu visinu te se pomiče u lijevo u odnosu na početnu poziciju, a potom se odlaže na praznom dijelu stola. Nakon prebacivanja pribora, isti poklopac se vraća na posudu.

Posljednja stavka je sterilizator koji se nalazi s lijeve strane robota. Njegov način otvaranja je dolazak robota okomito na poklopac, minimalno podizanje, a potom pomicanje u desno na principu vodilica. Poklopac se otpušta minimalnim spuštanjem kako bi se vratio u početnu poziciju po visini u odnosu na početno stanje.

3.4. Što su pametne komponente i zašto se koriste

Prema [8], pametne komponente (eng. SmartComponents) Robot Studio programskog paketa, u simuliranom okružju stvaraju interakciju s ostalim elementima šaljući i primajući različite vrste signala. Pametne komponente sadrže svojstva koja pomoću signala stvaraju inteligentno ponašanje (podizanje i spuštanje objekta). Korisnici Robot Studio programskog paketa imaju mogućnost stvoriti direktnu interakciju između signala i svojstva budući da Robot Studio programski paket automatski stvara grafičko sučelje.

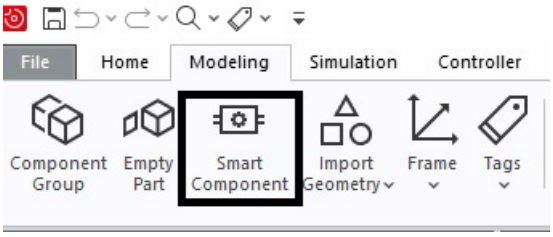
Signali su ulazi i izlazi koje pametne komponente koriste kako bi komunicirale s ostalim elementima. Svaki signal je definiran, a mogući signali su:

1. digitalni ulaz
2. digitalni izlaz
3. analogni ulaz
4. analogni izlaz
5. digitalna grupa ulaza
6. digitalna grupa izlaza

U nastavku će biti dan primjer upotrebe pametnih komponenti, njihov način korištenja te uvjeti koji moraju biti ispunjeni kako bi se pametne komponente mogle aktivirati.

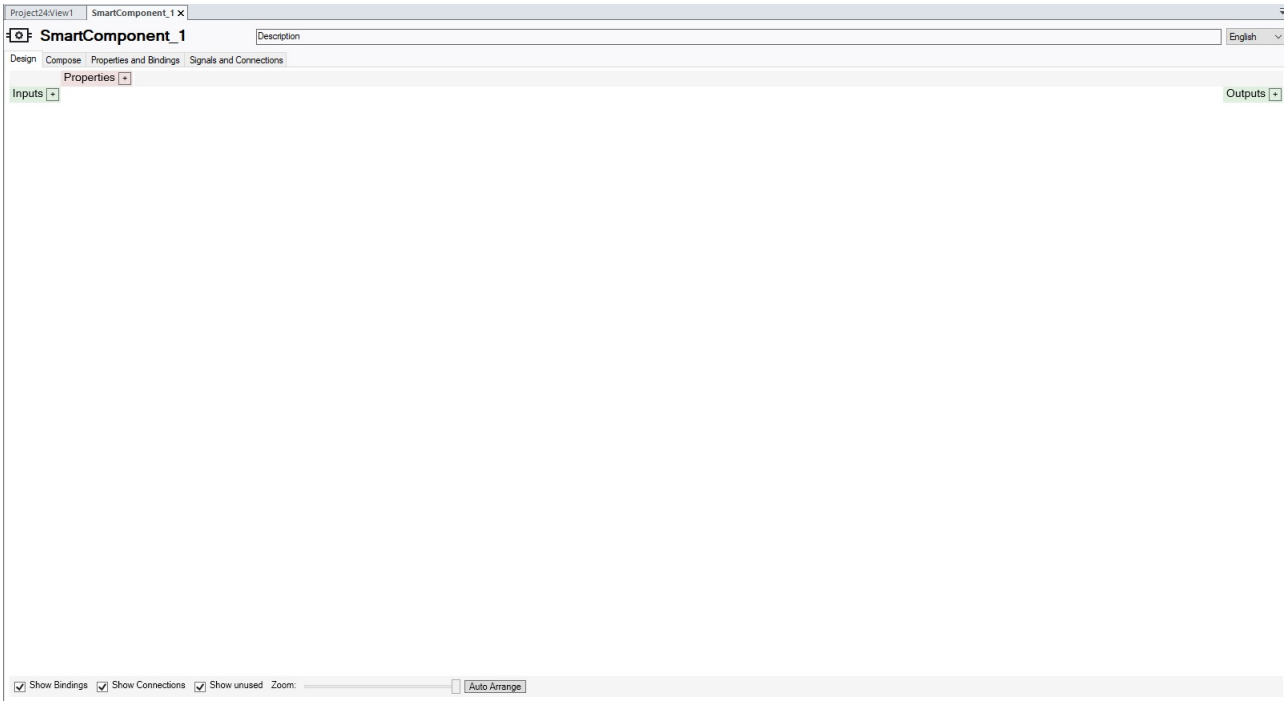
3.4.1. Primjena pametnih komponenata unutar automatizacije procesa sterilizacije

Prethodno je opisan osnovni način funkcioniranja pametnih komponenti te njihova zadaća. U nastavku će biti dan primjer dodavanja pametnih komponenti, koje se unutar Robot Studio programskog paketa odabiru prema slici 3.10. Dodatno, prikazat će se primjena pametnih komponenata na podizanje i premještanje određenog objekta (eng. pick and place).



Slika 3.10. Odabir pametne komponente.

Unutar projekta potrebno je odabrati karticu "Modeling", a potom pritiskom na "Smart Component" otvoriti pametnu komponentu. Slika 3.11 prikazuje prozor koji se dobije po otvaranju pametne komponente.



Slika 3.11. Prozor po otvaranju pametne komponente.

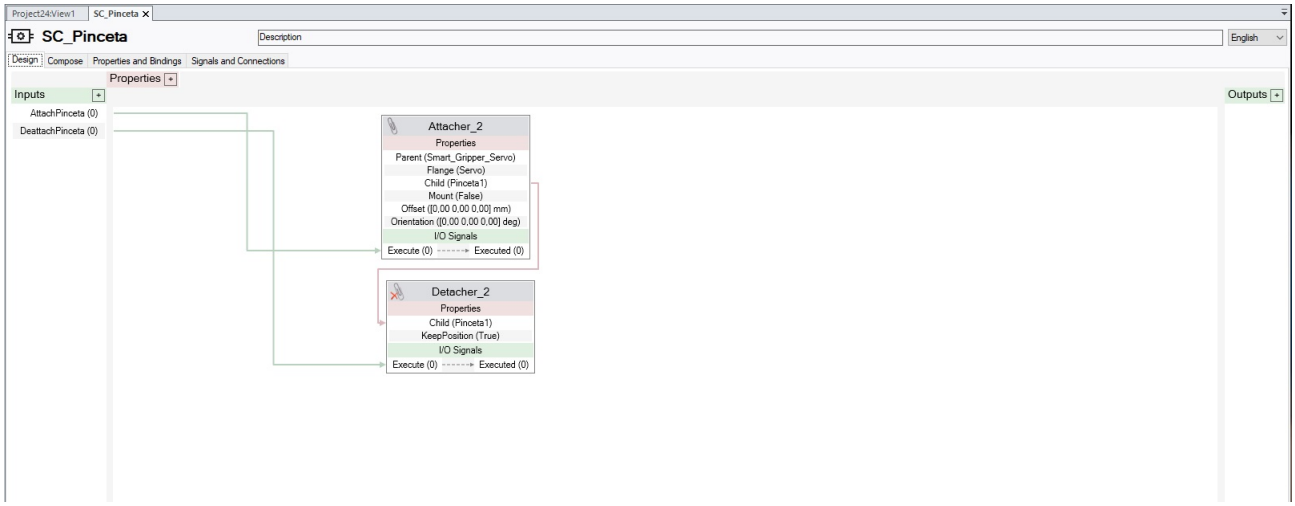
Također, na slici su prikazane četiri različite kartice pomoću kojih se postavljaju signali i postavke kako bi se izvršila određena radnja. Kartica "Design" prikazuje trenutačni prozor unutar kojeg je moguće povezivati određene komponente.

Unutar kartice "Compose" moguće je dodati različite komponente koje izvršavaju određene radnje. Npr. jedna od komponenata je "Attacher" koji u određenom trenutku zakači neki objekt

za robotski manipulator. Kartica "Properties and Bindings" prikazuje poveznicu između dvije komponente.

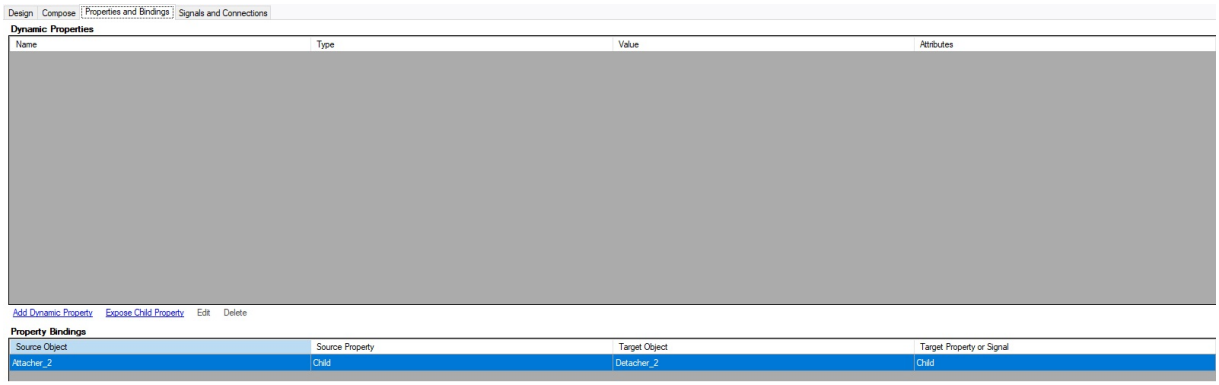
"Signals and Connections" kartica prikazuje signale koji su dodani pritiskom na "Inputs" i "Outputs" na kartici "Design". Sve navedene kartice prikazane su na slici 3.12.

Za rad s pametnim komponentama potrebno je znati koji će se signali koristiti kao ulazni, koji kao izlazni te komponente koje koristimo za interakciju između robota odnosno robotskog manipulatora i objekta.

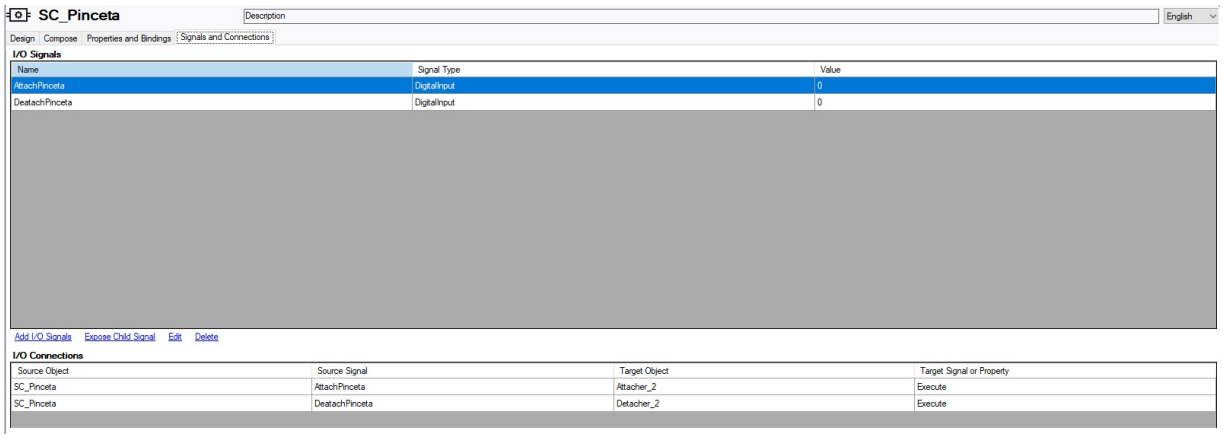


Slika 3.12. Primjer aktivne pametne komponente.

Slika 3.12 prikazuje dvije komponente: "Attacher" i "Deatacher" koje su povezane na dva ulazna signala "AttachPinceta (0)" i "DeatachPinceta (0)". Također, vidimo poveznicu između komponente "Attacher" i "Deatacher" tako što se "Attacher" iz dijela gdje piše "Child (Pinceta1)" povezuje sa "Deatacherom" na poziciju "Child (Pinceta1)". Povezivanjem unutar kartice "Design", automatski se sve poveznice prenose unutar kartica "Properites and Bindings" i "Signals and Connections" koje su prikazane na slikama 3.13 i 3.14.



Slika 3.13. Primjer kartice Properties and Bindings.



Slika 3.14. Primjer kartice Signals and Connections.

Pametne komponente mogu uvelike olakšati izradu simulacije. Jednostavnim povezivanjem komponenata s ulaznim signalima, a potom i između njih samih, stvara se mali sustav koji pomaže u izvršenju određenih zadataka.

Za svako tijelo unutar simulacije za koje postoji određena promjena pozicije aktivira se zasebna pametna komponenta kojoj se mijenjaju ulazni signal i "Child" koji se kontrolira odnosno čija se pozicija mijenja.

Unutar Robot Studio programskog paketa stvoreno je nekoliko pametnih komponenata za izvršavanje zadataka. Tri pametne komponente služe za promjenu pozicije pincete, tri za promjenu pozicije medicinskog skalpela, tri za promjenu pozicije škara, jedna za promjenu pozicije poklopca koji zatvara posudu gdje se odlaže pribor te jedna pametna komponenta za kontroliranje otvaranja i zatvaranja sterilizatora.

4. Izrada simulacije

Kako bi se s pomoću samog robota mogle izvršavati neke radnje, potrebno je prije svega osmisliti redoslijed kretanja robota. Potom se kreiraju točke u koje će robot doći te izvršiti određenu radnju. Na samom kraju, sve se sinkronizira u RAPID kod koji služi kako bi se simulacija pokrenula te bi robot izvršio predviđene zadatke.

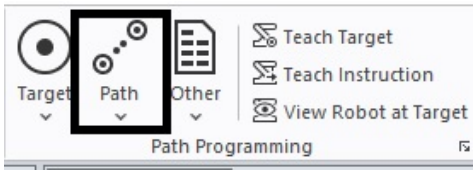
4.1. Putanja robota

S obzirom na količinu medicinskog pribora koji se nalazi na stolu, bit će potrebno kreirati putanju za svaki komad medicinskog pribora zasebno. Broj putanja po komadu pribora je sljedeći:

1. medicinska pinceta - 3 putanje.
2. medicinski skalpel - 3 putanje.
3. medicinske škare - 3 putanje.
4. otvaranje i zatvaranje poklopca - 2 putanje.
5. otvaranje i zatvaranje sterilizatora - 2 putanje.
6. prijenos hermetički zatvorene posude - 2 putanje.

4.1.1. Stvaranje putanje robota

Slika 4.1 prikazuje stvaranje prazne putanje robota. Na kartici "Home" u dijelu "Path Programming" može se pronaći kartica "Path". Pritiskom na "Path" stvara se prazna putanja unutar koje je moguće postaviti određene točke u koje se robot mora pozicionirati prije samog izvršenja zadatka.



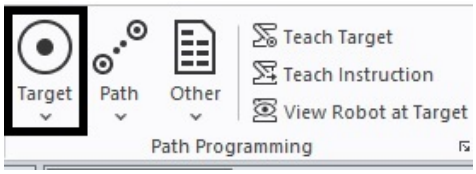
Slika 4.1. Primjer kreiranja putanje robota.

Nakon stvaranja, putanji robota moguće je postaviti ime kako bi se naknadno lakše razlikovalo o kojem je dijelu kretanja robota riječ.

4.2. Točke izvršenja

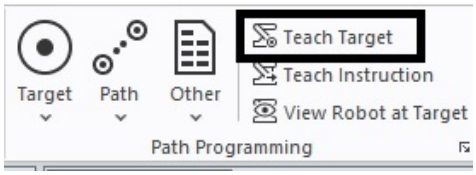
Nakon što je kreirana prazna putanja robota te joj je dodijeljeno neko ime, potrebno je u radnom prostoru naznačiti točke u koje se robot postavlja prije ili nakon izvršenja predviđenog zadatka.

Slika 4.2 prikazuje zadavanje točke izvršenja. Istu je moguće zadati ako se unutar softvera prvotno pritisne na karticu "Home", a potom se u dijelu "Path Programming" pronađe "Target". Unutar radnog prostora robota potrebno je tada odabrati poziciju gdje je potrebno pozicioniranje robota.



Slika 4.2. Primjer stvaranja Targeta robota.

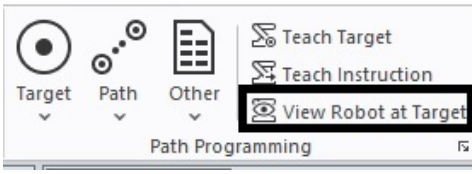
Postoji i opcija stvaranja "Targeta" odnosno točke izvršenja ako nam se robot nalazi u nekoj poziciji koja nije vezana za izvršenje neke radnje već je riječ o početnoj ili krajnjoj poziciji. Slika 4.3 prikazuje kako se pomoću naredbe "Teach Target" može stvoriti "Target" prema kojem se robot nakon što je izvršio prethodne naredbe vrati u određenu poziciju bez potrebe za izvršenjem nekog zadatka (pomicanje bloka).



Slika 4.3. Stvaranje Targeta pomoću opcije Teach Target.

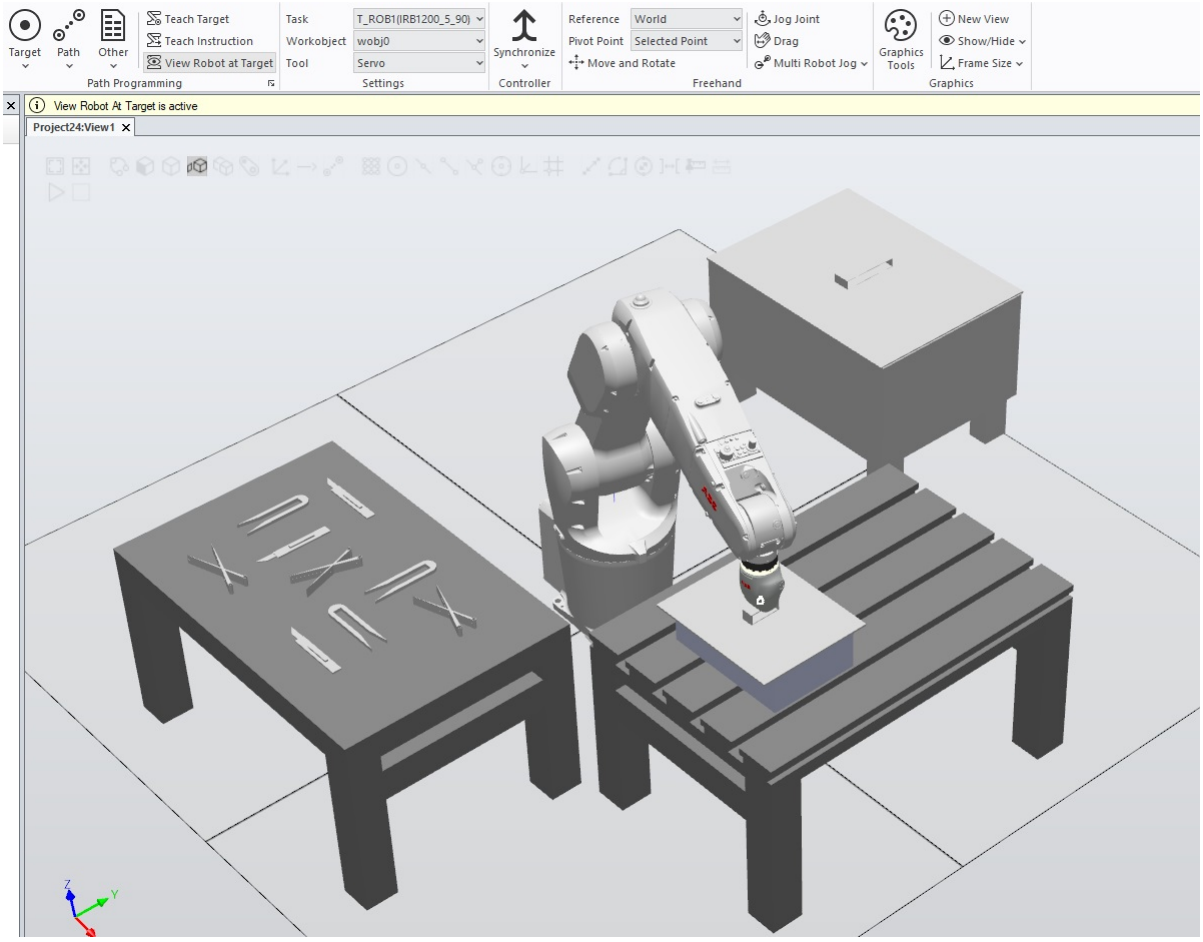
Prilikom programiranja kretanja robota, potrebno je voditi računa o mogućnostima koje robot ima. Ponekad je moguće da robot ne može izvršiti naredbu prema predviđenoj točki izvršenja zbog svoje konfiguracije, orijentacije "Targeta" ili je jednostavno točka izvršenja postavljena previše daleko.

Unutar Robot Studio programskog paketa postoji opcija koja omogućava korisnicima praćenje robota na točkama izvršenja prilikom pritiska na određenu točku izvršenja. Ako se aktivira opcija "View Robot at Target", kako je prikazano na slici 4.4, tada se može vidjeti točna pozicija robotskog manipulatora kada dođe do točke izvršenja. Ova opcija omogućava korisnicima lakše podešavanje pozicije prije sinkronizacije putanje u RAPID kod.



Slika 4.4. Aktivacije naredbe View Robot at Target.

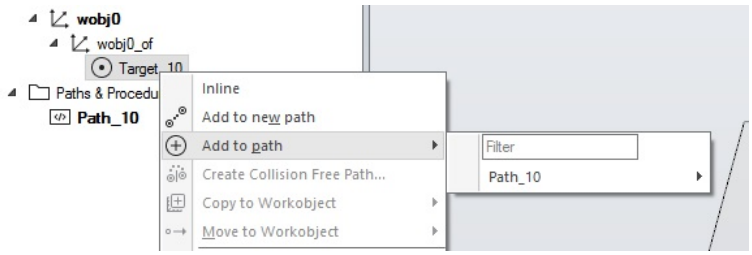
U nastavku, na slici 4.5 bit će prikazan primjer pozicioniranja robota na točku izvršenja uz aktivnu opciju "View Robot at Target".



Slika 4.5. Primjer aktivirane naredbe View Robot at Target.

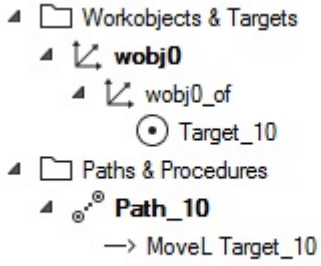
4.2.1. Popunjavanje putanje robota točkama izvršenja

Nakon što su postavljene sve točke izvršenja, tada je, kako bi se uopće mogao pokrenuti robot, potrebno sve točke izvršenja postaviti u predviđenu putanju. Slika 4.6 prikazuje kako se desnim klikom na "Target_10" i karticom "Add to path" dolazi do prikazanog na slici te se može točka izvršenja unutar putanje postaviti kao prva ili zadnja. Naknadno je moguće mijenjati redoslijed točaka izvršenja unutar putanje.



Slika 4.6. Primjer popunjavanja putanje robota točkama izvršenja.

Po izvršenju prethodno objašnjenog na slici 4.6, slika 4.7 prikazat će dodanu točku izvršenja unutar putanje robota.



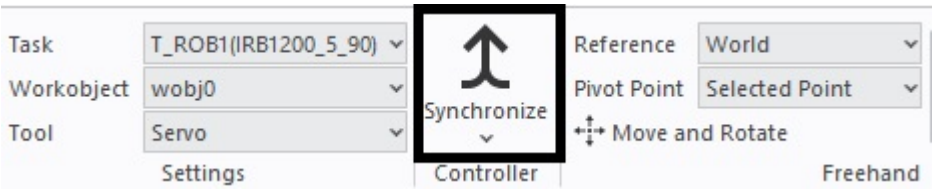
Slika 4.7. Primjer popunjavanja putanje robota točkama izvršenja.

Po odrađivanju prethodno opisanih akcija, putanja kretnje robota spremna je za sinkronizaciju u RAPID kod.

4.3. RAPID kod

Računalo samo po sebi naredbe koje mu čovjek postavi izvršava čitajući programski kod. Čovjek nije u stanju čitati jedinice i nule koje računalo čita te je zato postojala potreba za razvijanjem programskih jezika koji će biti razumljivi ljudima.

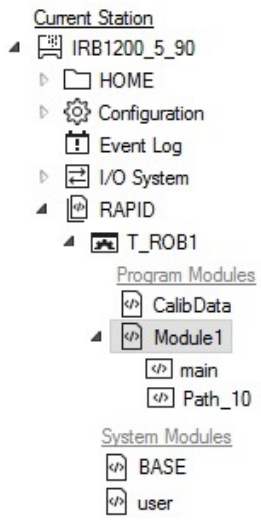
Upravo je RAPID programski jezik razvijen kako bi ga čovjek mogao koristiti i razumjeti. Kako bi robot izvršio predviđene putanje koje su prethodno postavljene, potrebno je unutar kartice "Home" pronaći opciju "Synchronize" kako je prikazano na slici 4.8.



Slika 4.8. Sinkronizacija predviđene putanje u RAPID kod.

Pritiskom na "Synchronize" pojavit će se prozor u kojem je moguće odabrati koje putanje korisnik želi sinkronizirati u RAPID kod pomoću "RAPID Sync". Upravo navedeni "RAPID Sync" služi za prijenos informacija između Robot Studio programskog paketa i virtualnog kontrolera koji je definiran na samom početku.

Po završetku sinkronizacije odabrane putanje robota, unutar kartice "RAPID" moguće je, kako je prikazano slikom 4.9 pronaći sljedeći izbornik, a unutar njega i glavni dio RAPID koda. Pritiskom na "main" otvara nam se glavni dio koda unutar kojeg se nalaze sve sinkronizirane putanje. Važno je istaknuti kako korisnik mora unutar "main" definirati koju putanju želi izvršiti.



Slika 4.9. Sinkronizacija predviđene putanje u RAPID kod.

U nastavku je prikazan RAPID kod koji prikazuje sve korake koje će robot napraviti tijekom izvođenja "Patha_10()". Svaki korak opisuje način kretanja robota, naziv točke izvršenja, brzinu izvođenja/kretanja, zonu i alat.

Pozivanje na izvršenje određene putanje izvršava se unutar "PROC main ()" gdje je potrebno definirati koju putanju robot mora izvršiti. Na kodu ispod, vidljivo je da je unutar "PROC main ()" definiran "Path_10;" te zapravo robot izvršava putanju istog naziva koja je također definirana.

```
1    PROC main()
2      Path_10;
3  ENDPROC
4      PROC Path_10()
5      MoveL Target_10, v150, fine, Servo\WObj:=wobj0;
6      MoveL Target_20, v150, fine, Servo\WObj:=wobj0;
7      MoveL Target_20_2, v150, fine, Servo\WObj:=wobj0;
8      MoveL Target_20_2_2, v150, fine, Servo\WObj:=wobj0;
9      MoveL Target_20_2_2_2, v150, fine, Servo\WObj:=wobj0;
10     MoveL Target_20_3, v150, fine, Servo\WObj:=wobj0;
11     MoveL Target_20_3_2, v150, fine, Servo\WObj:=wobj0;
12 ENDPROC
```

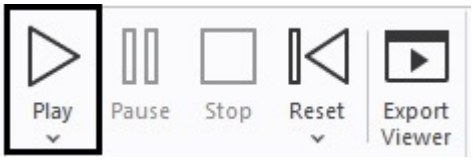
4.4. Simuliranje RAPID koda

Nakon definiranja putanje, točaka izvršenja te sinkronizacije putanje u RAPID kod, preostaje objasniti na koji način je moguće pokrenuti kretanje robota te izvršavanje predviđenih zadataka. Potrebno je objasniti mogućnosti koje Robot Studio programski paket pruža unutar kartice "Simulation" budući da se sve radnje koje će biti objašnjene nalaze upravo unutar navedene kartice.

4.4.1. Pokretanje simulacije

Pokretanje simulacije je posljednji korak prilikom izrađivanja kompletne simulacije. Na slici 4.10 su unutar bloka "Simulation Control" naznačene kontrole "Play", "Pause", "Stop" i "Reset".

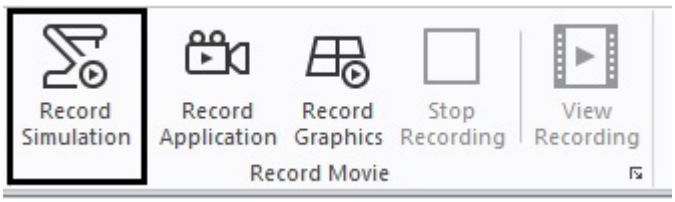
Kontrola "Play" pokreće RAPID kod unutar kojeg se nalazi definirana putanja koja je također definirana u glavnom bloku koji se izvršava. Kontrola "Pause" omogućava zaustavljanje simulacije tijekom izvođenja. S pomoću kontrole "Stop" u potpunosti zaustavljamo kretanje robota te se robot vraća u prethodnu točku. Kako i sam naziv kaže, kontrola "Reset" omogućava vraćanje simulacije na početak.



Slika 4.10. Kontrole za upravljanje radom simulacije.

4.4.2. Snimanje simulacije

Osim pokretanja, pauziranja, zaustavljanja i ponovnog pokretanja simulacije, unutar Robot Studio programskog paketa postoji i mogućnost snimanja simulacije. Slika 4.11 prikazuje unutar kartice "Simulation" i bloka "Record Movie" mogućnost kontrole "Record Simulation" s pomoću koje je moguće snimiti cijelu simulaciju koja se kasnije pojavi u obliku videa.



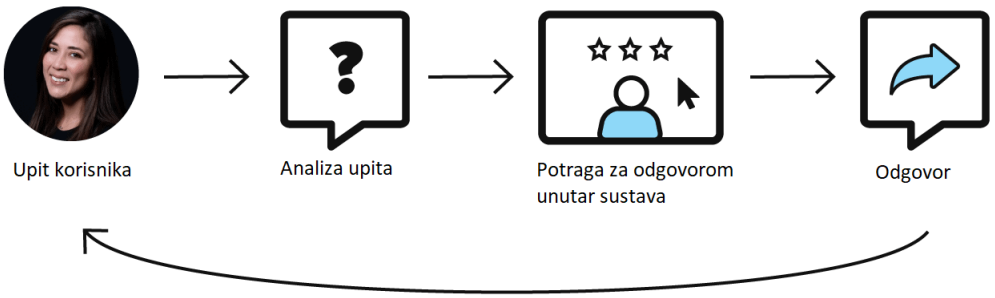
Slika 4.11. Dostupne kontrole za snimanje simulacije unutar Robot Studio programskog paketa

5. Primjena umjetne inteligencije na proces sterilizacije

Umjetna inteligencija (eng. Artificial Intelligence - AI) je, prema [11], tehnologija koja omogućava računalima i raznim strojevima simuliranje ljudske inteligencije te sposobnost rješavanja različitih problema.

Sama po sebi ili čak u primjeni s nekim drugim tehnologijama koje se danas koriste, umjetna inteligencija može izvršavati zadatke koji zahtijevaju razinu ljudske inteligencije ili reakcije odnosno prisutnost ljudskog faktora za rješavanje problema. Primjeri gdje se danas umjetna inteligencija pojavljuje, a bio bi potreban ljudski faktor:

1. digitalni asistent - različite tvrtke koje pružaju mogućnost razgovora s korisničkom podrškom bi u bliskoj budućnosti vrlo lako mogle zamijeniti agente sa umjetnom inteligencijom. Slika 5.1 prikazuje proces potraživanja odgovora od korisničke podrške. Umjetna inteligencija analizira pitanje koje je korisnik postavio, potom traži odgovor u bazi, a na kraju dostavlja odgovor korisniku.



Slika 5.1. Primjer primjene umjetne inteligencije u korisničkoj podršci, Izvor: [12]

2. GPS navođenje - ulaskom umjetne inteligencije u GPS sustave za navođenje, poboljšavaju se mogućnosti samih GPS sustava za navođenje s obzirom da umjetna inteligencija analizira dostupne podatke kao što su:

   a) stanja cesta - s obzirom na svakodnevne promjene na cestama, umjetna inteligencija prema dojavama na vrijeme prepoznaje potrebu za nuđenjem alternative s obzirom na stanje.

   b) korisnikove sklonosti - vozi li se korisnik češće cestom bez ili sa naplatom cestarine.

   c) prethodne vožnje - način i brzina iste.

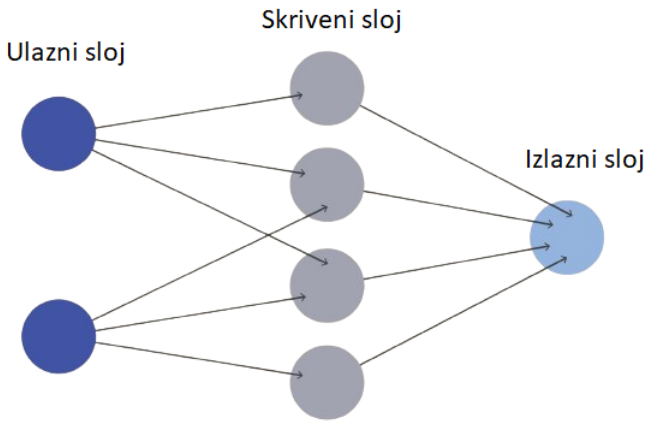
   Svi navedeni podaci koriste se kako bi se pomoću GPS navođenja moglo korisnicima ponuditi sigurnu i bezbrižnu vožnju gdje ne moraju provjeravati stanje na cestama ili provjeravati fizičku kartu kako se ne bi izgubili.

3. alati poput Chat GPT - a koji je dio ljudske svakodnevice s obzirom da ga se koristi za različite provjere informacija pa sve do pronalaska programskih kodova.

U ovom radu primjena umjetne inteligencije na proces sterilizacije bit će vidljiva prilikom odabira komada medicinskog pribora koji će robotski manipulator premještati s radnog stola u posudu za pribor. Zadatak umjetne inteligencije je prepoznati određeni komad medicinskog pribora koji se nalazi na stolu kao što je vidljivo na slici 3.9. Za izvršenje tog zadatka odabrano je korištenje konvolucijske neuronske mreže (eng. Convolutional Neural Network - CNN).

5.1. Neuronske mreže

Prije samog definiranja konvolucijske neuronske mreže koja će biti korištena u ovom radu, potrebno je definirati što su zapravo neuronske mreže. Sam pojam neuronske mreže moguće je definirati kao skup međusobno povezanih elemenata koji se nazivaju umjetni neuroni. Međusobna povezanost umjetnih neurona služi za razne postupke koji bi nadomjestili ljudski mozak. Slika 5.2 prikazuje jednostavnu neuronsku mrežu s njezinim slojevima.



Slika 5.2. Slika jednostavne neuronske mreže

Općenito unutar neuronskih mreža postoje tri različita sloja koji definiraju samu neuronsku mrežu:

1. ulazni sloj - najjednostavniji sloj koji prima podatke te ih prosljeđuje prema ostatku neuronske mreže.

2. skriveni sloj - unutar ovog sloja može se nalaziti samo jedna točka ili više njih budući da nije striktno definirano koliko se točaka unutar skrivenog sloja može ili smije nalaziti. Skriveni slojevi su zapravo odgovorni za izvrsne performanse i složenost neuronskih mreža. Istovremeno obavljaju višestruke funkcije poput transformacije podataka ili automatskog stvaranja značajki.

3. izlazni sloj - iako posljednji, ovaj sloj ne mora nužno sadržavati neke podatke već može sadržavati i određene informacije o problemu koji se pojavio.

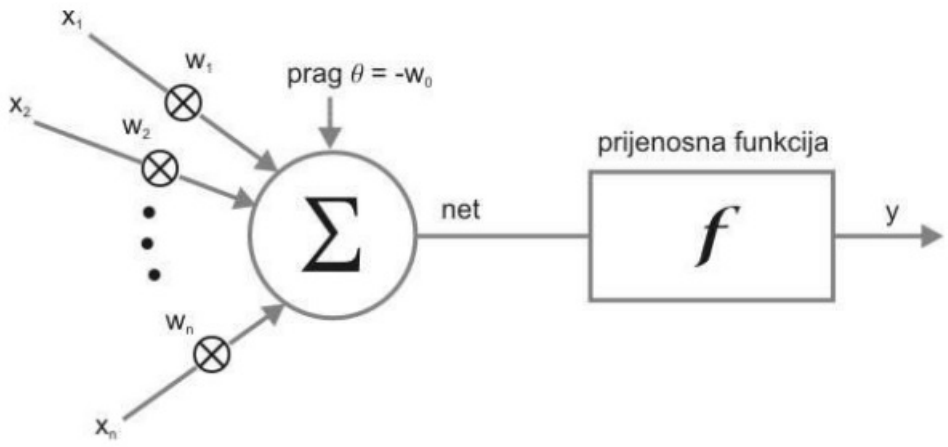
Prema [13], neuronske mreže svoju primjenu pronalaze u situacijama kada sama računala jednostavno nisu u mogućnosti pružiti dovoljno dobre rezultate. Postoje različite vrste neuronskih mreža koje se primjenjuju ovisno o situaciji ili zadatku koji je potrebno obaviti:

1. perceptron - najstariji i najjednostavniji model neurona. To je najmanja jedinica neuronske mreže koja provodi određene izračune kako bi otkrila značajke ili poslovnu inteligenciju u ulaznim podacima.
2. neuronska mreža s prosljeđivanjem ulaza (eng. Feed Forward Neural Network) - najjednostavniji oblik neuronske mreže gdje podaci putuju u samo jednom smjeru. Prolaze kroz umjetne neuronske čvorove te izlaze kroz postojeće izlazne čvorove. Skriveni slojevi u ovom slučaju ne moraju postojati dok su ulazni i izlazni slojevi prisutni.
3. višeslojni perceptron (eng. Multilayer Perceptron) - ova vrsta neuronske mreže predstavlja korak ka složenijim neuronskim mrežama. Koristi se za prepoznavanje govora, strojno prevođenje i složene klasifikacije.
4. rekurzivne neuronske mreže (RNN) - same su predviđene tako da spremaju izlaz sloja te se vraćaju na ulaz kako bi pomogle u predviđanju ishoda sloja. Primjena u provjeri gramatike, obradi teksta u vidu automatskog prijedloga riječi, pretvorba teksta u govor, prijevod općenito.
5. LSTM – dugoročna kratkoročna memorija (eng. Long Short-Term Memory) - specijalna vrsta rekurzivne neuronske mreže koja se pojavljuje kao dodatak standardnoj jedinici. Ova vrsta neuronske mreže ima posebnu memorijsku ćeliju koja može zadržavati informacije duži period.
6. modularna neuronska mreža - vrsta neuronske mreže koja ima više različitih mreža koje funkcioniraju zasebno te služe za izvođenje podzadataka. Tijekom procesa, mreže međusobno ne surađuju jedna s drugom već rade nezavisno kako bi se postigao željeni rezultat.
7. konvolucijske neuronske mreže - koristi trodimenzionalne podatke za klasifikaciju slika i zadatke prepoznavanje objekata. Primjena u obradi slika, prepoznavanju govora i strojnom prevođenju.

Posljednje navedena mreža, konvolucijska neuronska mreža, bit će, kako je prethodno već navedeno korištena u ovom radu. Odabrana je s obzirom na potrebu za prepoznavanjem objekata u prostoru. Svaka mreža ima svoju određenu vrstu primjene te ih je moguće koristiti za željeno dobivanje rezultata.

5.1.1. Umjetni neuron

Prethodno je navedeno da se neuronska mreža može definirati kao skup međusobno povezanih elemenata koji se nazivaju umjetni neuroni. U ljudskom tijelu nalaze se biološki neuroni koji, kao i umjetni, čiji je prikaz vidlji na slici 5.3, imaju određenu arhitekturu te zapravo svaki čvor ima mogućnost procesuiranja ulazne vrijednosti te slanje određenog signala prema drugom čvoru.



Slika 5.3. Princip rada umjetnog neurona, Izvor: [16]

Princip rada umjetnog neurona moguće je objasniti pomoću slike 5.3. Ulaze ($x_1, x_2..x_n$) definiramo kao ulazne vrijednosti umjetnog neurona. Slika 5.3 pokazuje ulaze koji su različito definirani ($x_1, x_2, .., x_n$), a potrebno ih je definirati na takav način da se informacije mogu interpretirati od strane računala. Kako bi se imitirala stvarna svojstva bioloških neurona, u priču oko umjetnog uvedene su težine koje su na slici 5.3 predstavljene kao $w_1, w_2, .., w_n$. Ulazne vrijednosti (x_n) množe se s težinama (w_n) te potom ulaze u zbrajalo gdje se svi ulazi pomnoženi sa svojim težinama zbrajaju. Izlaz iz zbrajala naziva se mrežni ulaz (eng. network input - net), a taj se iznos prenosi u prijenosnu funkciju kao ulaz. Prema [17], prijenosna funkcija služi za donošenje odluke hoće li se umjetni neuron aktivirati i prenijeti izlaz prema idućem neuronu koji potom obavlja isti postupak. Prijenosna funkcija ulaznu vrijednost koja dolazi iz zbrajala uspoređuje s pragom, a potom se prema tome donosi odluka hoće li se umjetni neuron aktivirati ili ne.

Primjeri prijenosnih funkcija unutar neuronskih mreža:

1. linearna funkcija - ova prijenosna funkcija je najjednostavnija funkcija gdje se ulaz množi s konstantnom te se to preslikava na izlaz ($f(x) = c * x$). Nema mogućnost učenja mreže.
2. tangens hiperbolna funkcija - koristi se za klasifikaciju između dvije klase. Daje vrijednosti između - 1 i 1 te je slična Sigmoidalnoj funkciji. Njezin izraz: $f(x) = tanh(x) = \frac{2}{1+e^{-2x}} - 1$.
3. ReLu - vrijednost funkcije je uvijek između 0 i 1 te se zbog nelinearnosti u praksi praktično koristi.

Primjer rada umjetnog neurona

Prema [17], primjer rada umjetnog neurona moguće je analizirati kroz različite boje. Prema tablici 5.1 žutoj boji dodijelimo vrijednost 1, zelenoj boji 2, a plavoj boji 3. Time smo informacije postavili tako da ih računalo može interpretirati, a ovi signali ulazi su u neuron.

Boja	Postavljena vrijednost
Žuta	1
Zelena	2
Plava	3

Tablica 5.1. Postavljene vrijednosti boja u primjeru rada umjetnog neurona

Objašnjeno je da se težine uvode kako bi se imitirala svojstva bioloških neurona, a u ovom slučaju se ulaz kojem je postavljena vrijednost 1 množi sa većom težinom tipa 10 u odnosu na zelenu i plavu boju koje se množe sa težinom 1. Jasno je da veća težina prilikom množenja ulazne vrijednosti donosi veći značaj u ukupnoj vrijednosti koja proizlazi iz zbrajala te služi kao ulaz u prijenosnu funkciju.

U ovom primjeru se, postavljanjem žute boje na prvo mjesto, zbog svojstava samog umjetnog neurona pretpostavlja da će biti najviše uzbuđenja oko žute boje. Ovisno o postavljenim vrijednostima za zelenu i plavu, može se provjeriti koliko će umjetni neuron biti uzbuđen radi ostalih boja.

Zaključno, umjetni neuron pokušava imitirati ponašanje bioloških neurona koje svatko ima u vlastitom tijelu. Prethodno je detaljno opisan postupak rada umjetnog neurona kao i kada će on poslati informacije prema sljedećem neuronu. Važna je stavka odabir kvalitetne prijenosne funkcije koje su spomenute prilikom objašnjavanja principa rada umjetnog neurona budući da zapravo prijenosna funkcija i njezin prag odlučuju hoće li se umjetni neuron aktivirati i poslati informaciju ka sljedećem neuronu koji će potom ponoviti postupak.

5.1.2. Primjena neuronskih mreža u medicini

Kroz proces zdravstvene skrbi, mnogi pacijenti razviju probleme sa umom i tijelom te ti problemi mogu dovesti do neugodnih osjećaja, skupih tretmana, invaliditeta i slično. Predviđajući ovakav potencijalni razvoj događaja unaprijed, zdravstveni djelatnici imaju priliku provesti preventivne mjere koje bi pacijentima mogle povećati razinu sigurnosti i kvalitetu brige, a paralelno umanjiti trošak. Kraće rečeno, predviđanje korištenjem mreža velikih podataka za procjenu specifičnih ljudi i određenih čimbenika rizika kod određenih bolesti moglo bi spasiti živote i izbjeći medicinske komplikacije.

Danas se svijet polako okreće ka tome da se metode predviđanja bolesti koje su postojale dugi niz godina polako zamjene sa umjetnim neuronskim mrežama pomoću kojih bi se iz baze podataka moglo puno lakše i brže zaključiti o čemu je riječ te kako dalje postupati sa pacijentima. Umjetna neuronska mreža je samo jedna od mnogih modela koji su predstavljeni u području zdravstva od strane umjetne inteligencije i obrade velikih baza podataka. Njihova želja je pretvoriti hrpu podataka u bazu iz koje bi se mogla izvlačiti rješenja koja će biti od velike koristi oko pružanja tretmana pacijentima i brige o njima.

Prema [19], neuronske mreže moguće je pronaći na mjestima gdje je umjetna inteligencija napravila prvi korak i zakoračila u prostore. U nastavku će biti navedeno nekoliko primjera primjene umjetne neuronske mreže u medicini, a potom i nekoliko primjera koji su slični po funkcionalnosti u odnosu na samu temu rada iz svakodnevnog života.

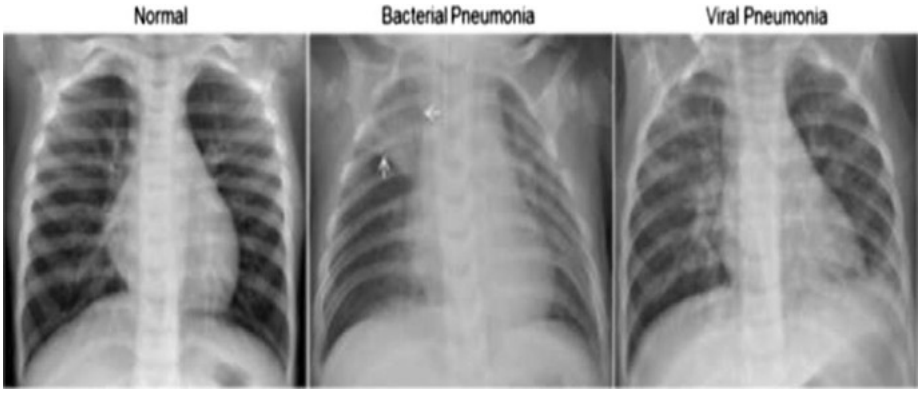
Jedan od primjera su dijagnostički sustavi. Umjetne neuronske mreže moguće je koristiti za detekciju problema sa srcem kao i za detekciju karcinoma. Također, moguće je otkriti i neke druge bolesti o kojima postoje podaci. Za analizu urina ili uzorka krvi moguće je koristiti biokemijsku analizu. Također, primjenu biokemijske analize moguće je pronaći kod praćenja razine glukoze kod osoba oboljelih od dijabetesa. Uz pomoć ove analize moguće je odrediti razinu iona u tekućini i detektirati različita patološka stanja. Umjetne neuronske mreže često su korištene za analizu medicinskih snimki u raznim granama medicine. Kao primjer možemo uzeti detekciju karcinoma gdje pomoću umjetne neuronske mreže možemo iščitavati klasifikaciju X zraka kao i magnetsku rezonancu. Jedna od stavki gdje pronalazimo primjenu umjetne neuronske mreže je razvoj lijekova. Tu se radi koristeći ogroman broj podataka kako bi se došlo do zaključka kako se može pomoći u slučaju određenog tretmana.

Može se reći da je potencijal primjene umjetnih neuronskih mreža u medicini ogroman, ali treba istaknuti da se problemi pojave kada se naiđe na podatke koje je mreži teško interpretirati. To se ističe kao razlog zašto se uglavnom mreže koriste u situacijama kada ima mnogo podataka koji su jasno definirani. Prije nego ih medicinski sustavi počnu koristiti, umjetne neuronske mreže trebat će proći kroz još dosta prilagodbe. Ideja je da umjetne neuronske mreže u potpunosti promjene medicinu i njezinu industriju te se vjeruje da će uloga biti velika, ali za sada još uvijek nije došlo do te točke.

5.1.3. Dodatni primjeri primjene

Jedan od primjera umjetne neuronske mreže je u svijetu drogerije gdje tvrtka koja se bavi analizom podataka kao i tvrtka koja radi u području ekonomije potpomažu tvrtku "Atomwise" koja koristi snagu strojnog učenja i neuronskih mreža kako bi pomogla medicinskim stručnjacima da brzo otkriju sigurnije i učinkovitije lijekove. Postoje još neke tvrtke koje su u procesu razvoja pojedinih sustava koji bi mogli pomoći u budućnosti. Primjer je tvrtka Butterfly Networks koja je još davne 2014. godine započela sa sakupljanjem sredstava za implementaciju umjetnih neuronskih mreža unutar sustava za dijagnostiku.

Prema [19], iCarbonX tvrtka razvija platformu na bazi umjetne inteligencije koja će olakšati istraživanja vezana za tretmane različitih bolesti i skrbi općenito. Upravo iCarbonX kompanija vjeruje da će uskoro biti u mogućnosti omogućiti princip rada u medicini gdje svaki pacijent ima poseban pristup i posebnu skrb. Jedan od primjera primjene umjetnih neuronskih mreža u medicini vidljiv je na slici 5.4



Slika 5.4. Prikaz snimka pluća koji se obrađuje primjenom umjetne neuronske mreže, Izvor: [20]

Osim primjene u medicini, slično na temu rada moguće je pronaći primjere robotskih sterilizatora. Prema [21], zadatak takvih robotskih sterilizatora je uklanjati sve oblike živih mikroorganizama te time spriječiti prijenos uzročnika bolesti. Najveća primjena ovakvih robota nalazi se u bolnicama, ali kroz rad je navedeno da se procesi sterilizacije pojavljuju i u tvornicama koje se bave proizvodnjom poluvodičkih komponenti. Osim navedenih na slikama 5.5 i 5.6, ovakve robotske sterilizatore moguće je vidjeti po trgovačkim centrima, kongresnim dvoranama i slično.

Slika 5.5. Robotski sterilizator u operacijskoj sali

Slika 5.6. Robotski sterilizator u trgovačkom centru

5.1.4. Konvolucijska neuronska mreža

Konvolucijske neuronske mreže (eng. Convolutional Neural Network - CNN) su poznate kao ConvNet mreže. Ova vrsta neuronske mreže specijalizirana je za prepoznavanje objekata uključujući klasifikaciju slika, prepoznavanje i segmentaciju. Ove mreže moguće je pronaći u različitim praktičnim situacijama kao što su sigurnosni sustavi, kamere, autonomna vozila.

Prije pojave konvolucijskih neuronskih mreža kakve danas poznajemo, postupci ekstrakcije značajki za identifikaciju objekta ili osobe su oduzimali previše vremena. Danas, konvolucijske neuronske mreže imaju mogućnost lakše prilagodbe na klasifikaciju slika, matrično množenje ili identificiranje uzoraka unutar slika. Može se zaključiti da su, s obzirom na zahtjeve koje mreže ispunjavaju, konvolucijske neuronske mreže računski i grafički zahtjevne.

Prethodno je ukratko navedena primjena konvolucijskih neuronskih mreža, a u nastavku se navode dodatni razlozi i ideje za korištenje konvolucijske neuronske mreže:

1. konvolucijske neuronske mreže razlikuju se u odnosu na klasične algoritme strojnog učenja zbog svoje sposobnosti autonomnog izdvajanja značajki u velikom opsegu. Time se izbjegava potreba za inženjeringom,a paralelno se povećava učinkovitost.
2. bez obzira na varijacije u položaju, orijentaciji, skali ili translaciji, konvolucijski slojevi konvolucijske neuronske mreže omogućavaju identificiranje te izdvajanje uzoraka i značajki iz podataka.
3. različite prethodno obučene arhitekture CNN-ova, uključujući VGG-16, ResNet50, Inceptionv3 i EfficientNet, pokazale su izuzetne performanse. Ovi modeli mogu se prilagoditi novim zadacima s relativno malo podataka putem postupka poznatog kao fino podešavanje.
4. osim klasifikacije slika, konvolucijske neuronske mreže mogu se primijeniti i u drugim područjima kao što je obrada jezika, prepoznavanje govora i analize različitih vremenskih nizova.

Kao i unutar svake neuronske mreže, tako i unutar konvolucijske neuronske mreže postoje određeni slojevi koji se posebno definiraju. Sama konvolucijska neuronska mreža sastoji se od četiri glavna dijela koji služe kao pomoć konvolucijskoj neuronskoj mreži kako bi mogla prepoznati te izdvojiti uzorke i značajke sa slike.

Slika 5.7 prikazuje slojeve unutar konvolucijske neuronske mreže. U nastavku će bit opisani slojevi sa slike.

1. konvolucijski sloj (eng. Convolutional layer) - ovaj sloj je osnovni gradivni blok konvolucijske neuronske mreže. Unutar konvolucijskog sloja se odvija većina računanja u sklopu konvolucijske neuronske mreže. Za izračune je potrebno nekoliko komponenata kao što su ulazni podaci, filtar i mapa značajki.

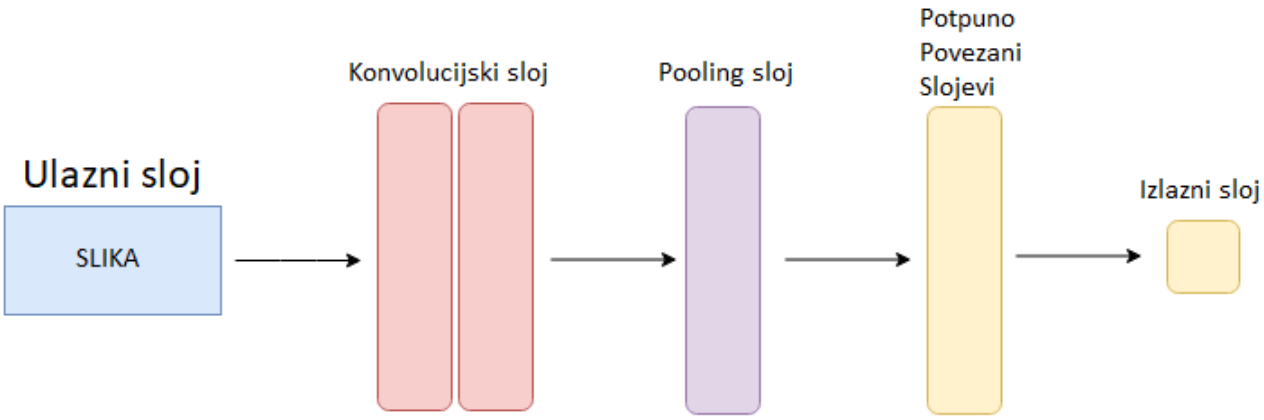
Uzmimo kao primjer da će ulazni sloj kao što je naznačeno na slici 5.7 biti slika u boji. Ta slika sastavljena je od matrice piksela u 3D - u. Jasno je da će ulazni sloj (slika) biti trodimenzionalna te ima određenu visinu, širinu i dubinu što odgovara RGB - u (Red, Green, Blue - aditivni model boja) slike. Također, u priči postoji detektor značajki koji je još poznatiji kao jezgra ili filtar koji se kreće kroz receptivna polja slike te provjerava jesu li značajke prisutne - proces konvolucije.

Prema [18], detektor značajki je dvodimenzionalno (2D) polje težina koje predstavlja dijelove slike. Iako se mogu razlikovati u veličini, veličina filtra je tipično matrica $3x3$ te je to ujedno i veličina receptivna polja. Filtar se primjenjuje na pojedino područje slike te se izračunava skalarni produkt ulaznih piksela i filtra. Taj umnožak koji je jednak skalarnom produktu se potom prebacuje u izlazno polje. Nakon izračuna, filtar se pomiče za polje te ponavlja proces dok se ne obiđe cijela slika. Konačni izlaz iz niza skalarnih produkata poznat je kao mapa značajki.

2. pooling sloj - ovaj sloj služi za smanjenje broja parametara ulaza. Slično kao i kod konvolucijskog sloja, proces uzorkovanja prelazi filtrom preko cijelog ulaza, ali je važna razlika da taj filtar nema nikakvu težinu. Umjesto toga, jezgra primjenjuje agregacijsku funkciju na vrijednost unutar receptivna polja popunjavajući izlazno polje.

3. potpuno povezani slojevi - naziv sloja jasno opisuje sam sebe. Vrijednosti piksela sa ulazne slike nisu direktno povezane na vanjski sloj u djelomično povezanim slojevima. Kada se u priči pojave potpuno povezani slojevi, svaka točka vanjskog sloja je povezana direktno s točkama iz prethodnih slojeva.

   Ovaj sloj ima zadatak klasifikacije na bazi značajki koje su izvučene iz prethodnih slojeva i njihovih različitih filtara. Dok konvolucijski i pooling sloj koriste ReLu kao prijenosnu funkciju, potpuno povezani slojevi obično koriste softmax aktivacijsku funkciju kako bi odgovarajuće klasificirali ulaze, proizvodeći vjerojatnost od 0 do 1.



Slika 5.7. Prikaz arhitekture konvolucijske neuronske mreže

5.2. Analiza Python koda

U ovom poglavlju bit će analiziran Python kod koji se koristio u svrhu prepoznavanja predmeta unutar programskog paketa Robot Studio. Rad se svodio na podizanje predmeta sa stola, prebacivanje u kutiju, a potom prijenosa u sterilizator. Kako bi robot prepoznao komad medicinskog pribora koji se nalazi na stolu, potrebno je preko umjetne inteligencije i učenja modela istrenirati sustav koji prepoznaje određeni komad medicinskog pribora te ga prenosi za na to predviđeno mjesto.

```
from tensorflow.keras.models import Sequential, Model
from tensorflow.keras.layers import Dense, Dropout, Flatten, Input,
 BatchNormalization
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint,
 ReduceLROnPlateau
from tensorflow.keras.applications import VGG16, ResNet101, NASNetMobile,
 MobileNet, InceptionV3
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.optimizers import Adam
from tensorflow.keras import regularizers
from tensorflow.keras import backend as K
import numpy as np
import tensorflow as tf
import uuid
import pandas as pd
```

U ovom primjeru koriste se različite komponente iz biblioteke tensorflow.keras, koje su ključne za rad s dubokim učenjem. Modul Sequential i Model omogućuju definiranje jednostavnih sekvencijalnih modela, kao i složenijih modela pomoću funkcionalnog API-ja. Sloj Dense služi za potpuno povezane slojeve u neuronskoj mreži, dok Dropout pomaže u regularizaciji modela smanjujući pretreniranost isključivanjem slučajnih neurona tijekom treninga. Sloj Flatten se koristi za pretvaranje višedimenzionalnih podataka u jednodimenzionalni niz, što je često potrebno prije ulaska u potpuno povezane slojeve. Input definira ulazni sloj mreže, a BatchNormalization normalizira izlaze iz prethodnog sloja, što može ubrzati treniranje i stabilizirati proces učenja.

Callback funkcije poput EarlyStopping, ModelCheckpoint i ReduceLROnPlateau su ključne za kontrolu treninga modela. EarlyStopping prekida treniranje kada se performanse na skupu za provjeru ne poboljšavaju, ModelCheckpoint automatski sprema najbolji model tijekom treniranja, dok ReduceLROnPlateau smanjuje stopu učenja kada se performanse modela počnu smanjivati.

Pretrenirani modeli kao što su VGG16, ResNet101, NASNetMobile, MobileNet i InceptionV3 pružaju arhitekture dubokih mreža koje su već istrenirane na velikim skupovima podataka, što omogućava transfer učenje za specifične zadatke.

Biblioteka ImageDataGenerator omogućava augmentaciju slike tijekom treniranja, čime se umjetno povećava veličina skupa podataka kroz transformacije slika. Optimizator *Adam*je po-

pularan izbor za treniranje neuronskih mreža zbog svoje efikasnosti i prilagodljivosti.

Osim tensorflow.keras komponenti, koristi se regularizers za dodavanje tehnika koje pomažu u sprječavanju pretreniranosti modela, dok backend modul pruža osnovne funkcionalnosti potrebne za rad modela. Dodatne biblioteke uključuju numpy za efikasnu numeričku obradu podataka, tensorflow kao osnovnu biblioteku za duboko učenje, uuid za generiranje jedinstvenih identifikatora, te pandas za manipulaciju i analizu podataka.

U nastavku se unutar koda se definiraju dimenzije na koje će se postaviti slike kako bi treniranje bilo optimalno.

```
# Postavljanje veličine slike
velicina_slike_x = 224
velicina_slike_y = 224
```

Nakon definiranja dimenzija slika, bit će opisan dio koda u kojem se radi augmentacija slika. Pojam augmentacije veže se na tehniku pomoću koje se umjetno može povećati veličina i raznolikost skupa podataka generirajući nove slike iz postojećih putem različitih transformacija.

```
# Generatori za augmentaciju podataka
train_datagen = ImageDataGenerator(
rescale=1./255,
rotation_range=40,
width_shift_range=0.2,
height_shift_range=0.2,
shear_range=0.2,
zoom_range=0.2,
horizontal_flip=True,
fill_mode='nearest'
)
```

Upravu se u ovom primjeru za potrebe proširenja podataka kako je prethodno opisano i navedeno koristi ImageDataGenerator iz biblioteke tensorflow.keras.preprocessing.image.

Prvo, skaliranje vrijednosti piksela (rescale = 1./255) pretvara svaki piksel slike u raspon od 0 do 1, što je standardna praksa za predobradu slika prije ulaska u neuronsku mrežu. Slike se zatim nasumično rotiraju do 40 stupnjeva (rotation_range = 40), što pomaže modelu da prepozna objekte bez obzira na njihovu orijentaciju.

Pomicanje slika vodoravno i okomito do 20 % njihove širine i visine (width_shift_range = 0.2, height_shift_range = 0.2) omogućava modelu da bude robusniji na pomake objekata unutar slike. Transformacija šišanja (shear_range = 0.2) primjenjuje geometrijske promjene koje mogu pomoći modelu da prepozna iskrivljene verzije objekata. Nasumično uvećavanje i smanjivanje slika do 20% (zoom_range = 0.2) omogućava modelu da prepozna objekte na različitim udaljenostima.

Horizontalno okretanje slika (horizontal_flip = True) povećava raznolikost skupa podataka, posebno kada objekti na slikama mogu biti simetrični. Konačno, postavka fill_mode = 'nearest'

osigurava da se prazni dijelovi slike, koji nastaju kao rezultat ovih transformacija, popune najbližim pikselima, čime se izbjegavaju artefakti.

```
val_datagen = ImageDataGenerator(rescale=1./255)
```

Ova linija koda ima svoju primjenu u pripremi podataka za provjeru. Prethodni dio bio je vezan za podatke za treniranje dok je ova linija kod bazirana na provjeri. Ovdje se vrijednosti piksela slika skaliraju tako da budu između 0 i 1, dijeljenjem sa 255. Ovo je standardni postupak za pripremu slika kako bi bile kompatibilne s neuronskim mrežama, baš kao i kod trening skupa podataka. Međutim, za slike koje se koriste za provjeru ne primjenjuju se dodatne transformacije poput rotacije, pomicanja ili približavanja. Cilj je osigurati da model bude evaluiran na originalnim, neizmijenjenim slikama, kako bi se dobila točna procjena njegovih performansi.

Dakle, dok se train_datagen koristi za augmentaciju i povećanje raznolikosti trening skupa podataka, val_datagen osigurava dosljednost i točnost pri provjeri modela. Ovaj pristup omogućava nam da iskoristimo prednosti augmentacije tijekom treniranja, a istovremeno zadržimo točnost pri procjeni modela na skupu za provjeru.

U nastavku će bit objašnjeno i prikazano učitavanje slika za treniranje i provjeru modela. Slike su obrađene pomoću prethodnih linija koda, a ovdje se, uz samo učitavanje slika prikazuje i način skaliranja slika na dimenzije 224 x 224 kako je i postavljeno u dijelu koda koji se nalazi nakon učitavanja biblioteka. Postavljena je veličina batch-a na 32 (batch_size = 32), što znači da će se slike učitavati u skupovima od 32 tijekom treniranja. Class_mode = categorical ukazuje na to da su podaci kategorizirani u više klasa to jest slike pripadaju različitim klasama.

```
train_ds = train_datagen.flow_from_directory(
'/content/drive/MyDrive/Colab_Notebooks/Slike_Za_Treniranje',
target_size=(224, 224),
batch_size=32,
class_mode='categorical'
)
```

Kod vezan za učitavanje slika koje služe za provjeru je više manje identičan kodu za učitavanje slika za treniranje. Jedna razlika vezana je zapravo uz put do direktorija u kojem se nalaze slike za provjeru. Može se primijetiti da se skaliranje odrađuje na iste dimenzije kao i kod slika za treniranje modela. Batch_size se postavlja na vrijednost 32 dok se class_mode postavlja kao categorical.

```
val_ds = val_datagen.flow_from_directory(
'/content/drive/MyDrive/Colab_Notebooks/Slike_Za_Provjeru',
target_size=(224, 224),
batch_size=32,
class_mode='categorical'
)
```

U nastavku se definira funkcija model_() koja vraća složeni model dubokog učenja koristeći unaprijed istrenirani model VGG16 kao osnovu.

```
def model_():
K.clear_session()
base_model = VGG16(weights='imagenet', include_top=False, input_shape=(224,
 224, 3))
base_model.trainable = True

model = Sequential([
base_model,
Flatten(),
Dense(256, activation='relu', kernel_regularizer=regularizers.l2(0.01)),
BatchNormalization(),
Dropout(0.6),
Dense(3, activation='softmax', kernel_regularizer=regularizers.l2(0.01))
])

return model
```

Funkcija model započinje naredbom K.clear_session(), a pomoću te naredbe se zapravo brišu svi prethodni grafovi kao i varijable koje su možda bile definirane radi nekih drugih slučajeva. Korisno je jer oslobađa memoriju koja bi mogla ostati zauzeta nakon prethodnih operacija, što može dovesti do problema s preopterećenjem memorije, posebno kada se radi o iterativnom razvoju modela.

Nakon toga, slaže se osnovni model VGG16, učitavajući prethodno istrenirane težine iz imagenet skupa podataka, pri čemu se definira da su svi njegovi slojevi trenirajući postavljanjem trainable = True. To znači da će svi slojevi biti prilagodljivi tijekom postupka treniranja. Ako bismo postavili da je trainable = False, tada bi dio slojeva bio zamrznut te se oni ne bi koristili u treniranju.

Zatim se definira novi model koristeći Sequential(). Prvi sloj u ovom modelu je osnovni model VGG16, a nakon toga slijede dodatni slojevi koji su dodani na vrh VGG16 kako bi se model prilagodio problemu koji je postavljen pred njega.

Dodani slojevi uključuju Flatten() sloj, koji pretvara izlaz iz osnovnog modela (koji je višedimenzionalni tenzor) u jednodimenzionalni vektor kako bi ga se moglo proslijediti potpuno povezanim slojevima. Zatim slijedi potpuno povezani sloj s 256 neurona i ReLU aktivacijskom funkcijom, uz L2 regularizaciju radi sprječavanja pretreniranosti. Nakon toga dolazi BatchNormalization() sloj koji normalizira podatke iz prethodnog sloja, što stabilizira i ubrzava treniranje.

Slijedi Dropout(0.6) sloj koji isključuje 60% neurona tijekom treniranja kako bi se spriječila pretreniranost. Konačno, izlazni sloj s 3 neurona i softmax aktivacijskom funkcijom za vjerojatnosnu distribuciju izlaza, uz L2 regularizaciju, zatvara model. Softmax aktivacijska funkcija je vrsta funkcije koja se koristi u zadnjem sloju neuronske mreže kako bi se riješili problemi klasifikacije.

U nastavku će biti postavljeni parametri prema kojima će se trenirati model.

```
brzina_ucenja = 0.0001
np.random.seed(94)

solveri = ['adam']
lista_broja_epoha = [30]
lista_batch_size = [32]

results = pd.DataFrame(columns=['slucaj', 'solver', 'broj epoha', 'batch
 size', 'accuracy_score', 'roc_auc_score_ovr', 'roc_auc_score_ovo', '
 precision_score_macro', 'precision_score_micro', 'recall_score_macro', '
 recall_score_micro', 'uuid'])

print("Početak petlje hiperparametara")

brojac = 1
```

Započinje se postavljanjem brzine učenja koja se postavlja na vrijednost 0.0001. Potom se sjeme generatora postavlja na 94 kako bi se osigurala reproducibilnost. Pojam reproducibilnosti veže se na sposobnost ponovnog stvaranja ili reprodukcije istih rezultata ili rezultata koji su vrlo slični nakon ponovnog izvođenja istog eksperimenta ili koda.

Nakon toga se definira lista solvera koju sačinjavaju optimizatori. Jedini koji se koristi u ovom programskom kodu je Adam, a on je uveden na samom početku iz biblioteke tensorflow.keras.optimizers. Potom se specificira broj epoha koji će biti korišten kao i batch_size. Oboje je moguće mijenjati tijekom treniranja modela. Broj epoha predstavlja koliko će puta model biti provučen kroz isti skup podataka, a batch_size određuje koliko se primjera koristi u svakoj iteraciji.

Osma linija koda donosi stvaranje prazne tablice koja će sadržavati informacije o kombinacijama hiperparametara i rezultatima evaluacije modela za svaku kombinaciju. Nakon postavljanja parametara, započinje se petlja hiperaparametara. Brojač se postavlja na vrijednost 1.

```
for i in solveri:
  for j in lista_broja_epoha:
    for z in lista_batch_size:
      mreza = model_()
      mreza.compile(optimizer=Adam(learning_rate=brzina_ucenja)
      loss='categorical_crossentropy',
      metrics=['AUC'])
```

U ovom dijelu bit će opisano postavljanje eksperimenta za treniranje modela strojnog učenja, pri čemu se ispituju brojevi epoha i veličine batch-eva (skupova podataka). Svrha ovog postupka je pronaći najbolju kombinaciju ovih parametara kako bi se postigla najbolja moguća točnost modela.

Kroz tri ugrađene petlje, kod prolazi preko različitih kombinacija parametara. U prvoj petlji, varijabla *i* prolazi kroz popis optimizatora, koji uključuje jedino optimizator *adam*. U drugoj petlji,

varijabla *j* prolazi kroz popis brojeva epoha, dok treća petlja s varijablom *z* prolazi kroz različite veličine batch-eva.

Za svaku kombinaciju ovih parametara, stvara se novi model pozivom funkcije model_(). Nakon toga, model se izvršava s odgovarajućim optimizatorom, gubitkom (loss) i metrikama. Ako je optimizator *adam*, koristi se Adam optimizer s odgovarajućom brzinom učenja.

U sljedećem dijelu koda bit će zapravo pojašnjen ključni proces treniranja neuronske mreže.

```
history = mreza.fit(
train_ds,
epochs=j,
validation_data=val_ds,
callbacks=[early_stopping, reduce_lr, checkpoint]
)
```

Model koji je prethodno definiran podvrgava se procesu učenja pomoću podataka o slikama koje su korištene u ovom radu. U ovom dijelu cilj je naučiti model prepoznavanje različitih kategorija slika na temelju primjera iz skupa za treniranje.

Pozivanjem metode fit zapravo započinje proces treniranja budući da je metoda fit samo srce treniranja modela. U metodu se prosljeđuju podaci za treniranje kao i podaci za provjeru. Prethodno je već definirano da se unutar trening podataka nalaze slike koju su unaprijed pripremljene i augmentirane.

Parametar epochs postavljen je na *j*, što označava broj puta koliko će model proći kroz cijeli skup trening podataka. Broj epoha određuje koliko će puta model vidjeti svaki primjer iz trening skupa, što je ključno za učinkovito učenje.

Podaci za provjeru nalaze se unutar val_ds te se koriste za procjenu performansi modela tijekom treniranja. Ovi podaci nisu augmentirani, već se koriste u izvornom obliku kako bi se osigurala što točnija procjena sposobnosti modela.

Jedna od ključnih stavki unutar ovog dijela koda bazirana je u petoj liniji koda gdje je riječ o callbacks funkciji gdje se kombiniraju early_stopping, reduce_lr i checkpoint. Prva od njih, EarlyStopping, je funkcija koja pomno prati gubitke provjere, što je mjera koliko dobro model radi na skupu podataka koji nije korišten za treniranje. Ako se taj gubitak ne smanji nakon određenog broja epoha, funkcija prekida daljnje treniranje. Ovo je izuzetno važno jer sprječava pretreniranost modela.

Druga funkcija, ReduceLROnPlateau, smanjuje brzinu učenja ako se gubitak vezan za provjeru stabilizira i ne pokazuje znakove poboljšanja. Brzina učenja određuje koliki korak model napravi u smjeru smanjenja gubitka tijekom svakog ažuriranja težina. Ako je brzina učenja prevelika, model može preskočiti optimalna rješenja, ako je premala, treniranje može biti vrlo sporo.

Konačno, ModelCheckpoint funkcija je poput sigurnosne mreže za treniranje modela. Ona kontinuirano prati performanse modela i sprema najbolju verziju koju model postigne tijekom

treniranja. Ako se kasnije epohe pokažu manje uspješnima i performanse modela se pogoršaju, ModelCheckpoint osigurava da najbolja verzija modela ostane sačuvana.

Kombinacija ove tri callback funkcije omogućava modelu da inteligentno i brzo reagira tijekom procesa treniranja. Svaka od ovih funkcija ima specifičnu ulogu, a zajedno čine moćan alat za optimizaciju performansi modela.

```
    uuid_ = uuid.uuid4()
    new_row = pd.DataFrame({
      'slucaj': [brojac],
      'solver': [i],
      'broj epoha': [j],
      'batch size': [z],
      'accuracy_score':  ['N/A'],  # Placeholder
      'roc_auc_score_ovr': [max(history.history['val_auc'])],
      'roc_auc_score_ovo': ['N/A'],  # Placeholder
      'precision_score_macro': ['N/A'],  # Placeholder
      'precision_score_micro': ['N/A'],  # Placeholder
      'recall_score_macro': ['N/A'],  # Placeholder
      'recall_score_micro': ['N/A'],  # Placeholder
      'uuid': [str(uuid_)]
    })
    results = pd.concat([results, new_row], ignore_index=True)
    brojac += 1

    results.to_excel('resultsVGG16.xlsx', sheet_name='Sheet1', index=False)

    print("Trening završen")
```

Nakon što se završi proces treniranja modela, potrebno je pohraniti rezultate, a to se radi pomoću pd.concat funkcije. Ovaj dio koda započinje generiranjem jedinstvenog identifikatora (UUID) za svaki eksperiment. UUID je jedinstveni identifikator koji osigurava da svaki proces treniranja bude prepoznatljiv i jedinstven, što je korisno za praćenje i analizu rezultata. Nakon toga, kreira se novi redak (new_row) u obliku DataFrame objekta, koji sadrži različite atribute eksperimenta koji su navedeni u kodu.

Podaci se potom dodaju u varijablu *results* pomoću, kako je već navedeno, pd.concat funkcije. Funkcija ignore_index = True osigurava da se indeksi redaka ponovno postave, čime se izbjegavaju duplikati ili neispravni indeksi.

Nakon što su svi eksperimenti završeni i rezultati su pohranjeni u results DataFrame, rezultati se spremaju u Excel datoteku nazvanu 'resultsVGG16.xlsx' pomoću to_excel metode. Ova datoteka sadrži sve važne informacije o svakom eksperimentu, što omogućava kasniju analizu i pregled performansi modela.

Konačno, na samom kraju, ispisuje se poruka kako bi se signaliziralo da je cijeli proces treniranja i pohrane rezultata uspješno dovršen.

5.2.1. Modeli

Na početku su navedeni modeli koji će biti korišteni u ovom radu: VGG16, ResNet101, InceptionV3, NASNetMobile i NASNetLarge. Svaki od njih bit će ukratko opisan zajedno s njihovom primjenom.

VGG16

Prema [23], VGG16 je vrsta konvolucijske neuronske mreže koja se koristi u ovom radu. Dizajniran je s 16 slojeva, uključujući konvolucijske i potpuno povezane slojeve. VGG16 se smatra jednim od najboljih modela za računalni vid zbog svoje jednostavnosti i učinkovitosti. Model koristi male 3x3 filtre u svim slojevima, što omogućava postizanje velike dubine uz relativno mali broj parametara.

Primjena VGG16 modela uključuje detekciju objekata i klasifikaciju slika. Može klasificirati 1000 različitih slika iz 1000 različitih kategorija s točnošću od 92.7%. Ovaj model je postao popularan zbog svoje jednostavnosti za korištenje i učinkovitosti u raznim zadacima računalnog vida. Zbog ovih svojstava, VGG16 se često koristi kao osnova za mnoge druge složenije modele i aplikacije u dubokom učenju.

Primjer korištenja VGG16 modela:

```
keras.applications.VGG16(
include_top=True,
weights="imagenet",
input_tensor=None,
input_shape=None,
pooling=None,
classes=1000,
classifier_activation="softmax",)
```

ResNet101

ResNet101 je varijacija ResNet modela s 101 slojem, što ga čini dubljim od ResNet50 modela. Slično kao i kod ResNet50, ResNet101 se temelji na ideji dubokih rezidualnih mreža, koristeći blokove s vezama kako bi olakšao treniranje dubokih modela.

Ova arhitektura zadržava ključne karakteristike dubokih rezidualnih mreža, ali dodatnih 51 sloj omogućava ResNet101 modelu da nauči složenije značajke i stvori još detaljnije predstave o podacima.

Primjena ResNet101 modela u računalnom vidu uključuje klasifikaciju slika, detekciju objekata i segmentaciju slika. Zahvaljujući većem broju slojeva, ovaj model može postići višu kvalitetu u svojim predviđanjima, pružajući veću preciznost i detaljnost u svojim rezultatima.

Arhitektura ResNet101 je popularna u području računalnog vida zbog svoje sposobnosti postizanja visokih performansi u raznim zadacima. Njezina dubina i kompleksnost omogućavaju joj da nauči složene obrasce u podacima i pruži precizna predviđanja u različitim scenarijima.

```
keras.applications.ResNet101V2(
include_top=True,
weights="imagenet",
input_tensor=None,
input_shape=None,
pooling=None,
classes=1000,
classifier_activation="softmax",)
```

InceptionV3

InceptionV3 je model koji pripada seriji Inception modela, poznatih po složenoj arhitekturi koja uključuje paralelne konvolucijske blokove. Ova struktura omogućava efikasno izvlačenje značajki iz slika uz kontroliranje broja parametara.

Dizajn InceptionV3 modela osmišljen je kako bi omogućio dublje mreže, ali istovremeno spriječio prevelik rast broja parametara. Kombinacija različitih konvolucijskih blokova omogućava modelu da uhvati različite prostorne značajke u slikama, što doprinosi njegovoj efikasnosti i performansama.

InceptionV3 se često koristi u raznim zadacima računalnog vida, uključujući klasifikaciju slika, detekciju objekata, segmentaciju slika i još mnogo toga. Model je postao popularan zbog svoje sposobnosti prilagodbe različitim zadacima i postizanja visokih performansi.

Primjer korištenja modela InceptionV3:

```
keras.applications.InceptionV3(
include_top=True,
weights="imagenet",
input_tensor=None,
input_shape=None,
pooling=None,
classes=1000,
classifier_activation="softmax",)
```

MobileNet

MobileNetV2 je arhitektura konvolucijske neuronske mreže koja je posebno optimizirana za izvođenje s visokim performansama na mobilnim uređajima. Ova arhitektura temelji se na invertiranoj rezidualnoj strukturi, koja omogućava efikasno učenje složenih značajki uz minimalne resurse.

Kao cjelina, arhitektura MobileNetV2 sadrži početni potpuno konvolucijski sloj s 32 filtra, nakon čega slijedi 19 rezidualnih slojeva. Ovaj dizajn omogućava modelu da nauči duboke reprezentacije slika uz minimalnu računalnu složenost, što ga čini idealnim za mobilne uređaje s ograničenim resursima.

Prema [24], MobileNetV2 se često koristi u raznim zadacima računalnog vida na mobilnim uređajima, uključujući klasifikaciju slika, detekciju objekata, prepoznavanje lica i druge zadatke. Njegova sposobnost efikasnog izvođenja na mobilnim uređajima čini ga popularnim izborom za aplikacije koje zahtijevaju obradu slika u stvarnom vremenu na mobilnim platformama.

Primjer korištenja MobileNetV2 modela:

```
keras.applications.MobileNetV2(
include_top=True,
weights="imagenet",
input_tensor=None,
input_shape=None,
pooling=None,
classes=1000,
classifier_activation="softmax",)
```

NASNetMobile

NASNetMobile je arhitektura konvolucijske neuronske mreže koja je dizajnirana s ciljem maksimiziranja performansi u određenom zadatku, uzimajući u obzir različite arhitektonske komponente kao što su veličine kernela, broj slojeva i veličina prostora za učenje.

Ovaj model je posebno koristan za primjene koje zahtijevaju visoku preciznost i efikasnost na mobilnim uređajima s ograničenim resursima. NASNetMobile je optimiziran za rad na mobilnim uređajima te omogućava brzu i pouzdanu obradu slika uz minimalnu potrošnju resursa.

NASNetMobile se ističe po svojoj skalabilnosti i prilagodljivosti različitim zadacima računalnog vida. Njegova arhitektura omogućava brzo učenje i prilagodbu različitim skupovima podataka, čineći ga popularnim izborom za razne aplikacije, uključujući klasifikaciju slika, detekciju objekata i druge zadatke računalnog vida.

Primjer korištenja NASNetMobile modela:

```
keras.applications.NASNetMobile(
include_top=True,
weights="imagenet",
input_tensor=None,
input_shape=None,
pooling=None,
classes=1000,
classifier_activation="softmax",)
```

5.2.2. Usporedba korištenja različitih modela

Nakon završetka treniranja modela, kako je i objašnjeno prilikom opisivanja koda, svaki model donio je određene rezultate koji će u nastavku biti prikazani tablicom. Broj epoha postavljen je na 30 budući da je bilo potrebno postaviti broj epoha tako da nam model napravi što detaljniju analizu te time ostvari što bolje rezultate.

Analizirajući rezultate dobivene za različite modele (VGG16, ResNet101, NasNetMobile, MobileNet i InceptionV3), moguće je uočiti značajne razlike u njihovoj učinkovitosti u predviđanju rezultata. Ovi rezultati su izraženi putem AUC mjere, koja se često koristi u evaluaciji performansi modela strojnog učenja, posebno u zadacima klasifikacije.

Prvo što se primjećuje je raznolikost u brzini učenja i stabilnosti modela tijekom epoha. Na primjer, modeli poput VGG16 i ResNet101 pokazuju postupno poboljšanje točnosti kako prolazi vrijeme u obliku epoha, dok drugi modeli poput NasNetMobile i InceptionV3 pokazuju manje stabilne trendove ili imaju sporiji rast točnosti.

VGG16 i ResNet101, postižu visoku točnost na kraju treninga (0.96 i 0.98), što ukazuje na njihovu sposobnost da dobro nauče značajke podataka i daju pouzdane rezultate. Ti modeli su duboki i složeni, što im omogućava da uhvate suptilne obrasce u podacima.

S druge strane, NasNetMobile, MobileNet i InceptionV3 imaju nižu konačnu točnost (0.88, 0.9 i 0.94). Iako su ti rezultati još uvijek pristojni, oni su niži u usporedbi s VGG16 i ResNet101. Ovi modeli možda nisu uspjeli uhvatiti složenije obrasce u podacima ili nisu imali dovoljno parametara za postizanje više točnosti.

Važno je napomenuti da izbor modela ovisi o konkretnim zahtjevima projekta. Ako je potrebna visoka preciznost, VGG16 i ResNet101 su optimalni izbori zbog njihove visoke točnosti. Međutim, ako je brzina ili veličina modela važnija, NasNetMobile, MobileNet i InceptionV3 mogu biti prikladniji izbori, iako s nešto nižom točnošću.

Ukratko, ovi rezultati pružaju uvid u relativnu učinkovitost različitih modela u zadatku predviđanja rezultata, što omogućava odabir optimalnog modela prema specifičnim potrebama i ograničenjima projekta.

U nastavku će svaki model biti zasebno predstavljen rezultatima, kratkom analizom postignutog te grafičkim prikazom dobivenih rezultata, a na samom kraju bit će prikazan i graf na kojem će biti prikazane sve krivulje dobivenih rezultata.

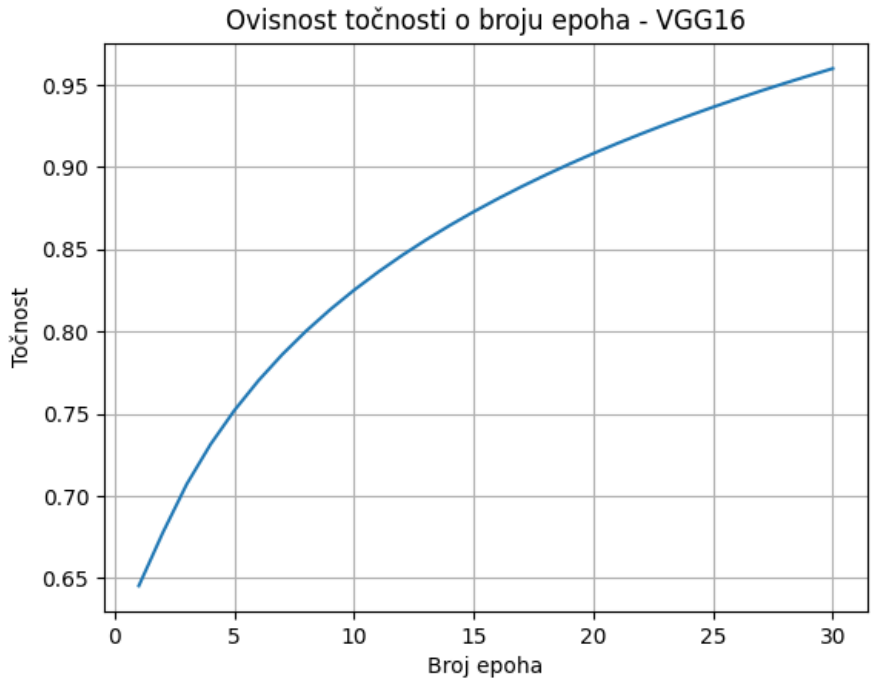
VGG16

Prvi model za treniranje bio je VGG16, a rezultati su prikazani tablicom 5.2.

Epoha br.	Rezultat (AUC)	Epoha br.	Rezultat (AUC)	Epoha br.	Rezultat (AUC)
1	0.64531331	11	0.836207668	21	0.914479452
2	0.677563392	12	0.846339831	22	0.920338719
3	0.707103955	13	0.855783222	23	0.925960915
4	0.731452809	14	0.864625555	24	0.931364482
5	0.752164915	15	0.872938784	25	0.936565788
6	0.770186791	16	0.880782672	26	0.94157943
7	0.78613763	17	0.888207394	27	0.94641848
8	0.800444745	18	0.895255485	28	0.95109469
9	0.81341548	19	0.901963318	29	0.955618663
10	0.8252783	20	0.908362239	30	0.96

Tablica 5.2. Prikaz rezultata za model VGG16.

Vidljivo je da je kod prve epohe vrijednost AUC - a bila 0.64531331, a na samom kraju kod tridesete epohe vrijednost se popela do 0.96 te je time prikazan dobar odabir modela za treniranje. Osim brojčano, slika 5.8 grafički prikazuje dobivene rezultate u odnosu na broj epoha.



Slika 5.8. Ovisnost točnosti VGG16 modela o broju epoha.

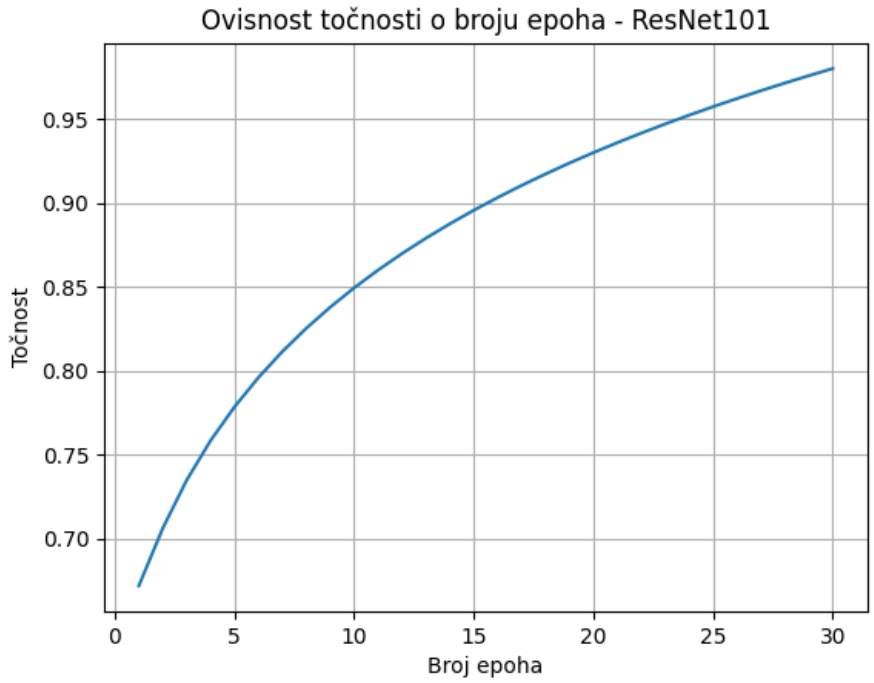
ResNet101

Drugi model za treniranje bio je ResNet101, a tablicom 5.3 prikazani su sljedeći rezultati:

Epoha br.	Rezultat (AUC)	Epoha br.	Rezultat (AUC)	Epoha br.	Rezultat (AUC)
1	0.672111321	11	0.860076179	21	0.935901969
2	0.706389536	12	0.869891711	22	0.941578134
3	0.735006957	13	0.879039996	23	0.947024636
4	0.758594909	14	0.887606006	24	0.952259342
5	0.778659761	15	0.895659447	25	0.957298107
6	0.796118454	16	0.903258214	26	0.962155073
7	0.811570829	17	0.910450913	27	0.966842903
8	0.825430847	18	0.917278751	28	0.971372981
9	0.837996247	19	0.923776964	29	0.97575558
10	0.849488353	20	0.929975919	30	0.98

Tablica 5.3. Prikaz rezultata za model ResNet101.

Rezultati pokazuju da je AUC vrijednost bila 0.672111321 u prvoj epohi, dok je na kraju, nakon tridesete epohe, dosegnula visokih 0.98. Ova značajna promjena ukazuje na to da je odabir modela za treniranje bio uspješan. Podaci su prikazani i grafički preko slike 5.9.



Slika 5.9. Ovisnost točnosti ResNet101 modela o broju epoha.

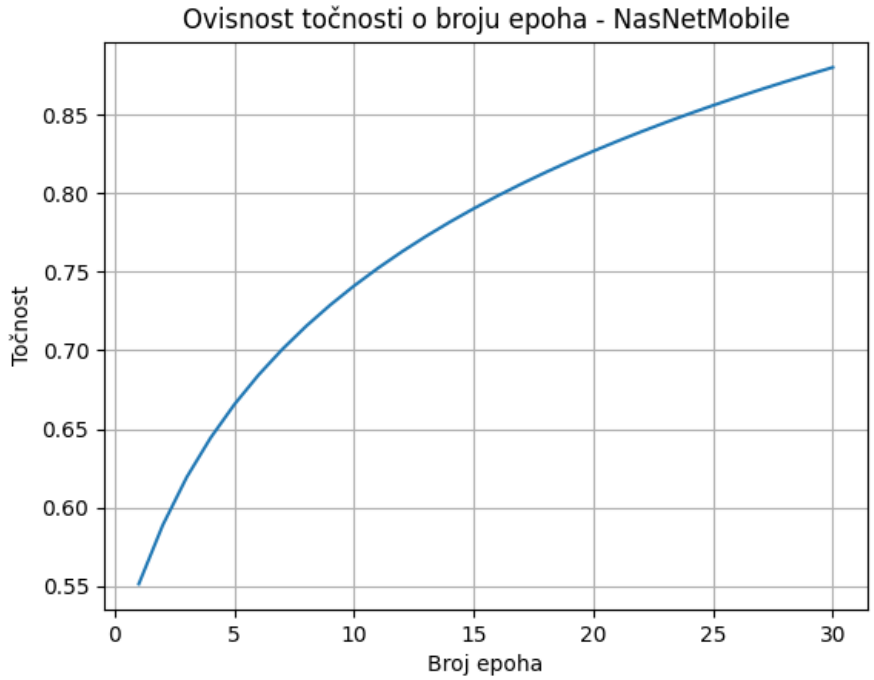
NasNetMobile

Treći model za treniranje bio je NasNetMobile, a dobiveni rezultati prikazani su tablicom 5.4.

Epoha br.	Rezultat (AUC)	Epoha br.	Rezultat (AUC)	Epoha br.	Rezultat (AUC)
1	0.551313312	11	0.752339158	21	0.833056935
2	0.588737248	12	0.762787951	22	0.839099304
3	0.619200954	13	0.772526448	23	0.844897194
4	0.64431071	14	0.781645103	24	0.850469622
5	0.665670068	15	0.790218121	25	0.855833469
6	0.684255128	16	0.798307131	26	0.861003787
7	0.700704431	17	0.805963875	27	0.865994058
8	0.715458644	18	0.813232219	28	0.870816399
9	0.728834714	19	0.820149672	29	0.875481746
10	0.741068247	20	0.826748559	30	0.88

Tablica 5.4. Prikaz rezultata za model NasNetMobile.

Analiza rezultata pokazuje da je vrijednost AUC-a bila 0.551313312 u prvoj epohi za model NasNetMobile, dok je na kraju, nakon tridesete epohe, dosegnula visokih 0.88. Slična i značajna promjena vrijednosti AUC-a primijećena je i kod drugih modela. Model je uspješan, a grafički prikaz slikom 5.10 to potvrđuje.



Slika 5.10. Ovisnost točnosti NasNetMobile modela o broju epoha.

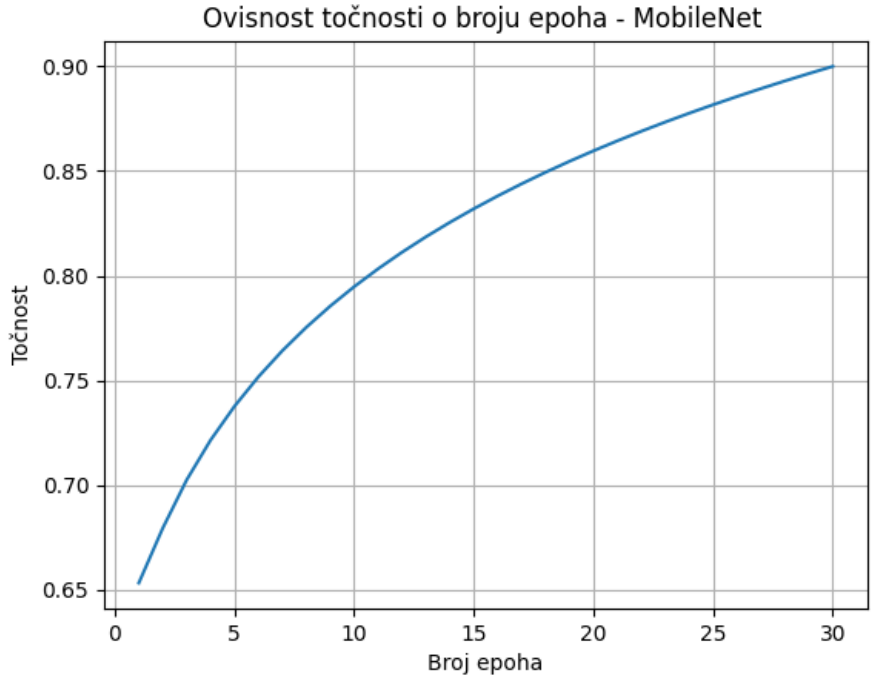
MobileNet

Četvrti model za treniranje bio je MobileNet te su tablicom 5.5 prikazani dobiveni rezultati.

Epoha br.	Rezultat (AUC)	Epoha br.	Rezultat (AUC)	Epoha br.	Rezultat (AUC)
1	0.653165454	11	0.803287241	21	0.864437072
2	0.6793464	12	0.811202993	22	0.869014624
3	0.702424965	13	0.818580642	23	0.873406965
4	0.721447507	14	0.825488714	24	0.877628501
5	0.73762884	15	0.831983425	25	0.881692022
6	0.75170843	16	0.838111463	26	0.88560893
7	0.764170024	17	0.843912026	27	0.889389438
8	0.775347457	18	0.849418348	28	0.893042727
9	0.785480844	19	0.854658842	29	0.896577081
10	0.794748672	20	0.859657999	30	0.9

Tablica 5.5. Prikaz rezultata za model MobileNet.

Rezultati pokazuju da je AUC vrijednost za model MobileNet bila 0.653165454 u prvoj epohi, dok je na kraju, nakon tridesete epohe, dosegnula visokih 0.9. Ova promjena ukazuje na to da je odabir modela za treniranje bio izuzetno uspješan. Osim što su ovi podaci izraženi numerički, prikazani su i grafički slikom 5.11.



Slika 5.11. Ovisnost točnosti NasNetMobile modela o broju epoha.

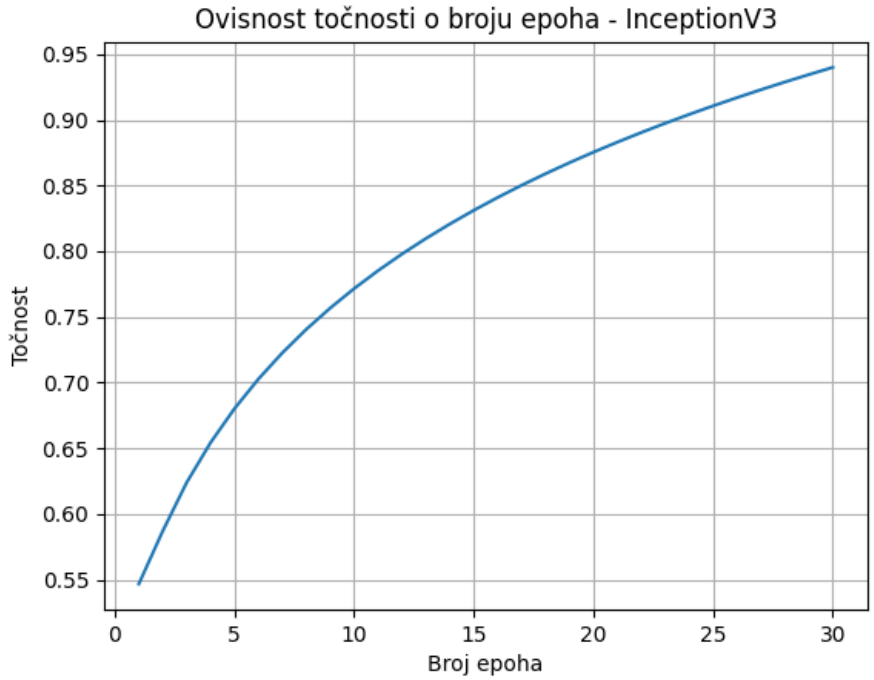
InceptionV3

Posljednji peti model za treniranje bio je InceptionV3, a tablica 5.6 prikazuje sljedeće rezultate:

Epoha br.	Rezultat (AUC)	Epoha br.	Rezultat (AUC)	Epoha br.	Rezultat (AUC)
1	0.546615135	11	0.785259585	21	0.883099315
2	0.586954239	12	0.797924789	22	0.890423398
3	0.623879944	13	0.809729028	23	0.897451144
4	0.654316012	14	0.820781943	24	0.904205602
5	0.680206143	15	0.83117348	25	0.910707235
6	0.702733489	16	0.84097834	26	0.916974287
7	0.722672038	17	0.850259242	27	0.9230231
8	0.740555932	18	0.859069356	28	0.928868362
9	0.75676935	19	0.867454148	29	0.934523329
10	0.771597875	20	0.875452799	30	0.94

Tablica 5.6. Prikaz rezultata za model InceptionV3.

Rezultati pokazuju da je AUC vrijednost za model InceptionV3 bila 0.546615135 u prvoj epohi, dok je na kraju, nakon tridesete epohe, dosegnula visokih 0.94. Ova postupna i značajna promjena ukazuje na to da je odabir modela za treniranje bio izuzetno uspješan. Osim što su ovi podaci izraženi numerički, prikazani su i grafički slikom 5.12.



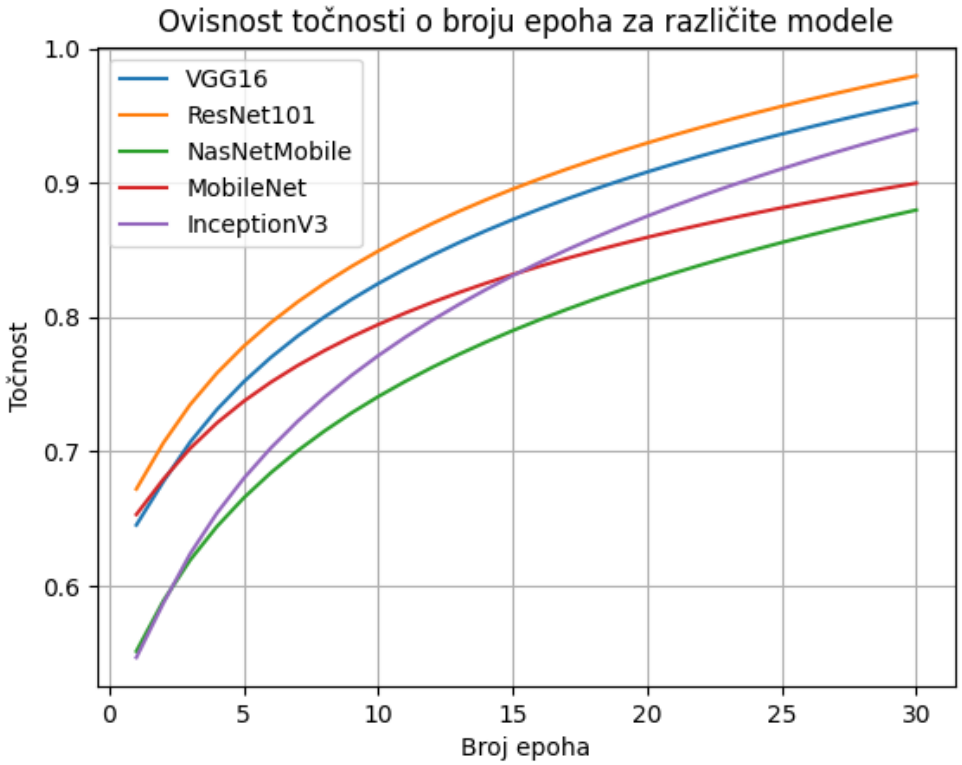
Slika 5.12. Ovisnost točnosti NasNetMobile modela o broju epoha.

Odabir modela

Modeli VGG16 i ResNet101 pokazali su se kao najuspješniji u postizanju visoke točnosti klasifikacije našeg skupa podataka. Ovi modeli postižu vrlo visoke vrijednosti površine ispod krivulje (AUC) već nakon nekoliko epoha, stabilizirajući se na vrijednostima oko 0.98. Ovo sugerira da su ovi modeli vrlo učinkoviti u prepoznavanju uzoraka u podacima i daju pouzdane rezultate.

Konačni izbor najboljeg modela ovisi o specifičnim zahtjevima problema, resursima na raspolaganju i željenim performansama. U ovom kontekstu, VGG16 i ResNet101 modeli nude najvišu točnost, dok su NasNetMobile, MobileNet i InceptionV3 također solidni izbori, posebno ako su resursi ograničeni ili ako je potrebno postići dobru točnost s manje složenim modelom.

Zaključno, s obzirom na sve navedeno, u ovom radu je ResNet101 model pokazao najbolje rezultate te je prema tome jasno da se taj model može iskoristiti za implementaciju umjetne inteligencije i programskog paketa Robot Studio na proces sterilizacije medicinskog pribora. Osim numerički, rezultate je moguće potvrditi i grafički putem slike 5.13.



Slika 5.13. Ovisnost točnosti svih modela o broju epoha.

6. Zaključak

Sterilizacija je ključni proces koji osigurava potpunu eliminaciju svih oblika živih mikroorganizama, a u usporedbi s dezinfekcijom, koja samo smanjuje broj mikroorganizama do određene razine, sterilizacija je kritična za osiguranje sigurnosti u mnogim područjima, uključujući elektrotehniku. S obzirom na to da medicinski uređaji moraju biti potpuno čisti i ne smiju sadržavati nikakve nečistoće, prašinu ili mikrobe, sterilizacija postaje sve važnija u proizvodnji elektroničkih komponenata.

Proces modeliranja komponenti unutar Robot Studio programskog paketa započeo je korištenjem softvera FreeCAD, no nedostatak opcija za izvoz određenih vrsta datoteka ograničio je njegovu upotrebljivost. Premda FreeCAD nudi jednostavan rad i intuitivno sučelje, nedostatak kompatibilnosti s potrebnim formatima datoteka potaknuo je prelazak na SolidWorks, kompleksniji CAD softver koji nudi mogućnosti izvoza datoteka potrebnih za Robot Studio.

Rad u Robot Studio programskom paketu sastojao se od složenih aktivnosti poput slaganja radnog prostora, kreiranja putanja kretanja i točaka izvršavanja radnji te sinkronizacije tih putanja u RAPID kodu. Važno je osigurati da svi objekti koji se moraju kretati unutar radnog prostora robota budu pravilno postavljeni kako bi se izbjegli incidenti ili oštećenja.

Umjetna inteligencija (AI) igra ključnu ulogu u ovom procesu, omogućavajući prepoznavanje predmeta na stolu unutar Robot Studio programskog paketa. Konvolucijska neuronska mreža, primijenjena u ovom radu pomoću programskog jezika Python, pokazala se kao moćan alat za prepoznavanje objekata na temelju slika. Iako su se pojavili izazovi, kao što su prilagodba dimenzija slika i odabir optimalnog modela, rezultati su kroz testiranje i iterativni proces odabira modela pokazali zadovoljavajuće performanse.

Kombinacija CAD softvera, Robot Studio programskog paketa i umjetne inteligencije omogućava automatizaciju procesa sterilizacije na visokoj razini preciznosti i pouzdanosti. Integracija ovih tehnologija otvara vrata za naprednije i učinkovitije procese sterilizacije u medicinskoj i elektrotehničkoj industriji.

Bibliografija

[1] - Skellie, B.: "A brief history of sterilization", s Interneta, https://brnskll.com/shares/a-brief-history-of-sterilization/, 1. travnja 2024.

[2] - Andrews, N.: "Charles Chamberland autoclave graphic", s Interneta, https://statimusa.com/2022/09/the-history-of-the-autoclave/charles-chamberland-autoclave-graphic/, 1. travnja 2024,

[3] - Prlić, N.: "Opća načela zdravlja i njege", Školska knjiga, 2014.

[4] - Desrosier, N.W.; Singh, R.P.: s Interneta, https://www.britannica.com/topic/food-preservation/Sterilization, 5. travnja 2024.

[5] - s Interneta, https://www.stockwell.com/semiconductor-manufacturing-equipment/, 5. travnja 2024.

[6] - "IRB 1200", s Interneta, https://new.abb.com/products/robotics/robots/articulated-robots/irb-1200, 7. Travnja 2024.

[7] - "Properties and signals", s Interneta, https://developercenter.robotstudio.com/api/robotstudio/articles/Concepts/SmartComponentTopics/SmartComponents.html, 15. travnja 2024.

[8] - "Odgovarajuća sterilizacija i dezinfekcija", s Interneta, "https://www.art-kozmetika.hr/objave/novice/odgovarajuca-sterilizacija-i-dezinfekcija", 15. travnja 2024.

[9] - "Autoklav medicinski sterilizator detaljan vodič", s Interneta, https://recolo.hr/autoklav-medicinski-sterilizator-detaljan-vodic/, 20. travnja 2024.

[10] - "Chemical Indicators for Sterilization", s Interneta, https://www.steris.com/healthcare/products/sterility-assurance-and-monitoring/chemical-indicators, 25. travnja 2024.

[11] - "What is artificial intelligence (AI)?", s Interneta, https://www.ibm.com/topics/artificial-intelligence, 10. svibnja 2024.

[12] - "An introduction to AI chatbots" s Interneta, https://www.drift.com/learn/chatbot/ai-chatbots/, 10. svibnja 2024.

[13] - "Neuronska mreža", s Interneta, https://www.enciklopedija.hr/clanak/neuronska-mreza, 10. svibnja 2024.

[14] - Chen, J.: "What Is a Neural Network?", s Interneta, https://www.investopedia.com/terms/n/neuralnetwork.asp, 10. svibnja 2024.

[15] - Robinson, Scott.: "Artificial neuron", s Interneta https://www.techtarget.com/searchcio/definition/artificial-neuron, 10. svibnja 2024.

[16] - Car, Z.: "Umjetne neuronske mreže", interna skripta, Sveučilište u Rijeci, Tehnički fakultet., 20. svibnja 2024.

[17] - Buettgenbach, M.H.: "Explain like I'm five: Artificial neurons", s Interneta, https://towardsdatascience.com/explain-like-im-five-artificial-neurons-b7c475b56189, 20. svibnja 2024.

[18] - "Što su konvolucijske neuronske mreže", s Interneta, https://www.ibm.com/topics/convolutional-neural-networks, 20. svibnja 2024.

[19] - Ferguson, J.: "Neural Networks in Healthcare", s Interneta, https://royaljay.com/healthcare/neural-networks-in-healthcare/, 20. svibnja 2024.

[20] - Yadav, S.S.; Jadhav, S.M.: "Deep convolutional neural network based medical image classification for disease diagnosis", s Interneta, https://journalofbigdata.springeropen.com/articles/10.1186/s40537-019-0276-2, 25. svibnja 2024.

[21] - "Intelligent Sterilization Robot (ISR)", s Interneta https://www.time-medical.com/intelligent-sterilization-robot, 25. svibnja 2024.

[22] - "Keras Applications", s Interneta, https://keras.io/api/applications/, 25. svibnja 2024.

[23] - Rohini, G.: "Everything you need to know about VGG16", s Interneta https://medium.com/@mygreatlearning/everything-you-need-to-know-about-vgg16-7315defb5918, 25. svibnja 2024.

[24] - "MobileNetV2", s Interneta https://paperswithcode.com/method/mobilenetv2, 25. svibnja 2024.

Sažetak i ključne riječi

Proces sterilizacije opreme je čest i dugotrajan proces u modernom zdravstvu. Ovaj diplomski rad cilja ka automatizaciji ovog procesa korištenjem artikulirane robotske ruke i umjetne inteligencije. U radu je korištenjem Robot Studio programskog paketa prikazano idejno rješenje automatizacije postupka sterilizacije. Uz to, izrađen je i model temeljen na konvolucijskim neuronskim mrežama za razlikovanje različitih objekata za sterilizaciju. Istraživanje dano u radu pokazuje da je moguće realizirati ovakvo rješenje za potrebe sterilizacije, bez velikih modifikacija postojeće opreme.

Ključne riječi: artikulirani robotski manipulator, konvolucijske neuronske mreže, sterilizacija, primjena automatizacije u zdravstvu

Summary and key words

The process of sterilizing equipment is a common and time-consuming procedure in modern healthcare. This thesis aims to automate this process using an articulated robotic arm and artificial intelligence. The conceptual solution for automating the sterilization procedure is presented using the Robot Studio software package. Additionally, a model based on convolutional neural networks has been developed to distinguish between different objects for sterilization. The research presented in this thesis demonstrates that such a solution can be implemented for sterilization purposes without significant modifications to existing equipment.

Keywords: articulated robotic manipulator, convolutional neural networks, sterilization, application of automation in healthcare

A RAPID kod

```
MODULE Module1
PROC main()
Podizanje_Poklopca;
Pinceta1;
Skalpel;
Pinceta3;
Vracanje_Poklopca;
Otvaranje_Sterilizatora;
Sterilizacija1;
Zatvaranje_Sterilizatora;
ENDPROC
PROC Podizanje_Poklopca()
MoveL Target_10,v150,fine,Servo\WObj:=wobj0;
MoveL Target_20,v150,fine,Servo\WObj:=wobj0;
PulseDO AttachPoklopac;
MoveL Target_20_2,v150,fine,Servo\WObj:=wobj0;
MoveL Target_20_2_2,v150,fine,Servo\WObj:=wobj0;
MoveL Target_20_2_2_2,v150,fine,Servo\WObj:=wobj0;
PulseDO DeattachPoklopac;
MoveL Target_20_2_2_2,v150,fine,Servo\WObj:=wobj0;
MoveL Target_10,v150,fine,Servo\WObj:=wobj0;
ENDPROC
PROC Vracanje_Poklopca()
MoveL Target_10,v150,fine,Servo\WObj:=wobj0;
MoveL Target_20_2_2,v150,fine,Servo\WObj:=wobj0;
MoveL Target_20_2_2_2,v150,fine,Servo\WObj:=wobj0;
PulseDO AttachPoklopac;
MoveL Target_20_2_2,v150,fine,Servo\WObj:=wobj0;
MoveL Target_20_2,v150,fine,Servo\WObj:=wobj0;
MoveL Target_20,v150,fine,Servo\WObj:=wobj0;
PulseDO DeattachPoklopac;
MoveL Target_10,v150,fine,Servo\WObj:=wobj0;
ENDPROC
PROC Pinceta1()
MoveL Target_10,v150,fine,Servo\WObj:=wobj0;
MoveL Target_30_3,v150,fine,Servo\WObj:=wobj0;
MoveL Target_30,v150,fine,Servo\WObj:=wobj0;
PulseDO AttachPinceta1;
MoveL Target_30_3,v150,fine,Servo\WObj:=wobj0;
MoveL Target_10,v150,fine,Servo\WObj:=wobj0;
MoveL Target_20,v150,fine,Servo\WObj:=wobj0;
MoveL Target_20_3,v150,fine,Servo\WObj:=wobj0;
MoveL Target_20_3_2,v150,fine,Servo\WObj:=wobj0;
```

```
PulseDO DeattachPinceta1;
MoveL Target_20_3,v150,fine,Servo\WObj:=wobj0;
MoveL Target_20,v150,fine,Servo\WObj:=wobj0;
MoveL Target_10,v150,fine,Servo\WObj:=wobj0;
ENDPROC
PROC Pinceta2()
MoveL Target_10,v150,fine,Servo\WObj:=wobj0;
MoveL Target_40_2,v150,fine,Servo\WObj:=wobj0;
MoveL Target_40,v150,fine,Servo\WObj:=wobj0;
PulseDO AttachPinceta2;
MoveL Target_40_2,v150,fine,Servo\WObj:=wobj0;
MoveL Target_20,v150,fine,Servo\WObj:=wobj0;
MoveL Target_20_4,v150,fine,Servo\WObj:=wobj0;
MoveL Target_20_4_2,v150,fine,Servo\WObj:=wobj0;
PulseDO DeattachPinceta2;
MoveL Target_20_4,v150,fine,Servo\WObj:=wobj0;
MoveL Target_20,v150,fine,Servo\WObj:=wobj0;
MoveL Target_10,v150,fine,Servo\WObj:=wobj0;
ENDPROC
PROC Pinceta3()
MoveL Target_10,v150,fine,Servo\WObj:=wobj0;
MoveL Target_50_2,v150,fine,Servo\WObj:=wobj0;
MoveL Target_50,v150,fine,Servo\WObj:=wobj0;
PulseDO AttachPinceta3;
MoveL Target_50_2,v150,fine,Servo\WObj:=wobj0;
MoveL Target_20,v150,fine,Servo\WObj:=wobj0;
MoveL Target_20_5,v150,fine,Servo\WObj:=wobj0;
MoveL Target_20_5_2,v150,fine,Servo\WObj:=wobj0;
PulseDO DeattachPinceta3;
MoveL Target_20_5,v150,fine,Servo\WObj:=wobj0;
MoveL Target_20,v150,fine,Servo\WObj:=wobj0;
MoveL Target_10,v150,fine,Servo\WObj:=wobj0;
ENDPROC
PROC Otvaranje_Sterilizatora()
MoveL Target_90,v150,fine,Servo\WObj:=wobj0;
MoveL Target_100_2,v150,fine,Servo\WObj:=wobj0;
MoveL Target_100,v150,fine,Servo\WObj:=wobj0;
PulseDO AttachPoklopacSterilizatora;
MoveL Target_100_2,v150,fine,Servo\WObj:=wobj0;
MoveL Target_100_2_2_2,v150,fine,Servo\WObj:=wobj0;
MoveL Target_100_2_2,v150,fine,Servo\WObj:=wobj0;
PulseDO DeattachPoklopacSterilizatora;
MoveL Target_100_2_2_2,v150,fine,Servo\WObj:=wobj0;
MoveL Target_90,v150,fine,Servo\WObj:=wobj0;
ENDPROC
PROC Zatvaranje_Sterilizatora()
MoveL Target_100_2_2_2,v150,fine,Servo\WObj:=wobj0;
```

```
MoveL Target_100_2_2,v150,fine,Servo\WObj:=wobj0;
PulseDO AttachPoklopacSterilizatora;
MoveL Target_100_2_2_2,v150,fine,Servo\WObj:=wobj0;
MoveL Target_100_2,v150,fine,Servo\WObj:=wobj0;
MoveL Target_100,v150,fine,Servo\WObj:=wobj0;
PulseDO DeattachPoklopacSterilizatora;
MoveL Target_10,v150,fine,Servo\WObj:=wobj0;
ENDPROC
PROC Sterilizacija()
MoveL Target_10,v150,fine,Servo\WObj:=wobj0;
MoveL Target_20,v150,fine,Servo\WObj:=wobj0;
PulseDO SpajanjeObjekata;
MoveL Target_20_2,v150,fine,Servo\WObj:=wobj0;
MoveL Target_70_4,v150,fine,Servo\WObj:=wobj0;
MoveL Target_70,v150,fine,Servo\WObj:=wobj0;
PulseDO OtpajanjeObjekata;
MoveL Target_70_4,v150,fine,Servo\WObj:=wobj0;
WaitTime 5;
MoveL Target_70,v150,fine,Servo\WObj:=wobj0;
PulseDO SpajanjeObjekata;
MoveL Target_70_4,v150,fine,Servo\WObj:=wobj0;
MoveL Target_20_6,v150,fine,Servo\WObj:=wobj0;
WaitTime 2;
MoveL Target_20,v150,fine,Servo\WObj:=wobj0;
PulseDO OtpajanjeObjekata;
MoveL Target_10,v150,fine,Servo\WObj:=wobj0;
ENDPROC
PROC Sterilizacija1()
MoveL Target_10,v150,fine,Servo\WObj:=wobj0;
MoveL Target_20,v150,fine,Servo\WObj:=wobj0;
PulseDO SpajanjeObjekata;
MoveL Target_20_2,v150,fine,Servo\WObj:=wobj0;
MoveL Target_70_4,v150,fine,Servo\WObj:=wobj0;
MoveL Target_70,v150,fine,Servo\WObj:=wobj0;
PulseDO OtpajanjeObjekata;
MoveL Target_70_4,v150,fine,Servo\WObj:=wobj0;
MoveL Target_100_2_2_2,v150,fine,Servo\WObj:=wobj0;
MoveL Target_100_2_2,v150,fine,Servo\WObj:=wobj0;
PulseDO AttachPoklopacSterilizatora;
MoveL Target_100_2_2_2,v150,fine,Servo\WObj:=wobj0;
MoveL Target_100_2,v150,fine,Servo\WObj:=wobj0;
MoveL Target_100,v150,fine,Servo\WObj:=wobj0;
PulseDO DeattachPoklopacSterilizatora;
MoveL Target_70_4,v150,fine,Servo\WObj:=wobj0;
WaitTime 5;
MoveL Target_100,v150,fine,Servo\WObj:=wobj0;
PulseDO AttachPoklopacSterilizatora;
```

```
MoveL Target_100_2,v150,fine,Servo\WObj:=wobj0;
MoveL Target_100_2_2_2,v150,fine,Servo\WObj:=wobj0;
MoveL Target_100_2_2,v150,fine,Servo\WObj:=wobj0;
PulseDO DeattachPoklopacSterilizatora;
MoveL Target_100_2_2_2,v150,fine,Servo\WObj:=wobj0;
MoveL Target_100_2,v150,fine,Servo\WObj:=wobj0;
MoveL Target_70,v150,fine,Servo\WObj:=wobj0;
PulseDO SpajanjeObjekata;
MoveL Target_70_4,v150,fine,Servo\WObj:=wobj0;
MoveL Target_20_6,v150,fine,Servo\WObj:=wobj0;
WaitTime 2;
MoveL Target_20,v150,fine,Servo\WObj:=wobj0;
PulseDO OtpajanjeObjekata;
MoveL Target_10,v150,fine,Servo\WObj:=wobj0;
ENDPROC
PROC Skalpel()
MoveL Target_10,v150,fine,Servo\WObj:=wobj0;
MoveL Target_40_2,v150,fine,Servo\WObj:=wobj0;
MoveL Target_40,v150,fine,Servo\WObj:=wobj0;
PulseDO AttachPinceta2;
MoveL Target_40_2,v150,fine,Servo\WObj:=wobj0;
MoveL Target_20,v150,fine,Servo\WObj:=wobj0;
MoveL Target_20_4,v150,fine,Servo\WObj:=wobj0;
MoveL Target_20_4_2,v150,fine,Servo\WObj:=wobj0;
PulseDO DeattachPinceta2;
MoveL Target_20_4,v150,fine,Servo\WObj:=wobj0;
MoveL Target_20,v150,fine,Servo\WObj:=wobj0;
MoveL Target_10,v150,fine,Servo\WObj:=wobj0;
ENDPROC
PROC Path_10()
MoveL Target_10,v150,fine,Servo\WObj:=wobj0;
MoveL Target_20,v150,fine,Servo\WObj:=wobj0;
MoveL Target_20_2,v150,fine,Servo\WObj:=wobj0;
MoveL Target_20_2_2,v150,fine,Servo\WObj:=wobj0;
MoveL Target_20_2_2_2,v150,fine,Servo\WObj:=wobj0;
MoveL Target_20_3,v150,fine,Servo\WObj:=wobj0;
MoveL Target_20_3_2,v150,fine,Servo\WObj:=wobj0;
ENDPROC
ENDMODULE
```

B Python kod

```
from tensorflow.keras.models import Sequential, Model
from tensorflow.keras.layers import Dense, Dropout, Flatten, Input,
 BatchNormalization
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint,
 ReduceLROnPlateau
from tensorflow.keras.applications import VGG16, VGG19, ResNet101V2,
 NASNetMobile, MobileNet, InceptionV3
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.optimizers import Adam
from tensorflow.keras import regularizers
from tensorflow.keras import backend as K
import numpy as np
import tensorflow as tf
import uuid
import pandas as pd

# Postavljanje veličine slike
velicina_slike_x = 224
velicina_slike_y = 224

# Generatori za augmentaciju podataka
train_datagen = ImageDataGenerator(
rescale=1./255,
rotation_range=40,
width_shift_range=0.2,
height_shift_range=0.2,
shear_range=0.2,
zoom_range=0.2,
horizontal_flip=True,
fill_mode='nearest'
)

val_datagen = ImageDataGenerator(rescale=1./255)

train_ds = train_datagen.flow_from_directory(
'/content/drive/MyDrive/Colab_Notebooks/Slike_Za_Treniranje',
target_size=(224, 224),
batch_size=32,
class_mode='categorical'
)

val_ds = val_datagen.flow_from_directory(
'/content/drive/MyDrive/Colab_Notebooks/Slike_Za_Provjeru',
```

```
target_size=(224, 224),
batch_size=32,
class_mode='categorical'
)

def model_():
K.clear_session()
base_model = VGG16(weights='imagenet', include_top=False, input_shape=(224, 224, 3))
base_model.trainable = True

model = Sequential([
base_model,
Flatten(),
Dense(256, activation='relu', kernel_regularizer=regularizers.l2(0.01)),
BatchNormalization(),
Dropout(0.6),
Dense(3, activation='softmax', kernel_regularizer=regularizers.l2(0.01))
])

return model

brzina_ucenja = 0.000001
np.random.seed(94)

solveri = ['adam']
lista_broja_epoha = [30]  # Povećan broj epoha da vidimo trend
lista_batch_size = [32]

results = pd.DataFrame(columns=['slucaj', 'solver', 'broj epoha', 'batch size', 'accuracy_score', 'roc_auc_score_ovr', 'roc_auc_score_ovo', 'precision_score_macro', 'precision_score_micro', 'recall_score_macro', 'recall_score_micro', 'uuid'])

print("Početak petlje hiperparametara")

brojac = 1

for i in solveri:
for j in lista_broja_epoha:
for z in lista_batch_size:
mreza = model_()
mreza.compile(optimizer=Adam(learning_rate=brzina_ucenja) if i == 'adam' else RMSprop(learning_rate=brzina_ucenja) if i == 'rmsprop' else SGD(learning_rate=brzina_ucenja, momentum=0.9),
loss='categorical_crossentropy',
metrics=['AUC'])
```

```
82
83  early_stopping = EarlyStopping(monitor='val_loss', patience=5,
     restore_best_weights=True)
84  reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.2, patience=5,
     min_lr=0.00001)
85  checkpoint = ModelCheckpoint('best_model.h5', monitor='val_loss',
     save_best_only=True)
86
87  history = mreza.fit(
88  train_ds,
89  epochs=j,
90  validation_data=val_ds,
91  callbacks=[early_stopping, reduce_lr, checkpoint]
92  )
93
94  uuid_ = uuid.uuid4()
95  new_row = pd.DataFrame({
96    'slucaj': [brojac],
97    'solver': [i],
98    'broj epoha': [j],
99    'batch size': [z],
100   'accuracy_score':  ['N/A'],  # Placeholder
101   'roc_auc_score_ovr': [max(history.history['val_auc'])],
102   'roc_auc_score_ovo': ['N/A'],  # Placeholder
103   'precision_score_macro': ['N/A'],  # Placeholder
104   'precision_score_micro': ['N/A'],  # Placeholder
105   'recall_score_macro': ['N/A'],  # Placeholder
106   'recall_score_micro': ['N/A'],  # Placeholder
107   'uuid': [str(uuid_)]
108 })
109 results = pd.concat([results, new_row], ignore_index=True)
110 brojac += 1
111
112 results.to_excel('resultsVGG16.xlsx', sheet_name='Sheet1', index=False)
113
114 print("Trening završen")
```