Primjena računalnog vida kod prepoznavanja i interpretacije znakovnog pisma

Pačić, Ira

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

Ira Pačić

Zagreb, 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

Mentori:

Dr. sc. Tomislav Stipančić, dipl. ing.

Student:

Ira Pačić

Zagreb, 2024.

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

Velika zahvala mentoru izv. prof. dr. sc. Tomislavu Stipančiću na pristupačnosti i pruženoj pomoći tijekom izrade rada.

Zahvaljujem se kolegicama i kolegama koji su mi pružili pomoć i podršku tokom cijelog studija.

Zahvaljujem obitelji,prijateljima i dečku koji su uvijek spremni pomoći i podržati me,te su mi pružili sve kako bi mi olakšali cijeli studij.

Ira Pačić

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE
Središnje povjerenstvo za završne i diplomske ispite
Povjerenstvo za diplomske ispite studija strojarstva za smjerove:
Proizvodno inženjerstvo, inženjerstvo materijala, industrijsko inženjerstvo i menadžment, mehatronika i robotika, autonomni sustavi i računalna inteligencija

Sveučilište u Zagrebu Fakultet strojarstva i brodogradnje	
Datum	Prilog
Klasa: 602 - 04 / 24 - 06 / 1	
Ur.broj: 15 - 24 -	

DIPLOMSKI ZADATAK

Student: **Ira Pačić** JMBAG: 0035220496

Naslov rada na hrvatskom jeziku: **Primjena računalnog vida kod prepoznavanja i interpretacije znakovnog pisma**

Naslov rada na engleskom jeziku: **Application of computer vision in the recognition and interpretation of sign letters**

Opis zadatka:

Računalni vid se koristi u različite svrhe, uključujući rješavanje zadataka koji uključuju detekciju, klasifikaciju, prepoznavanje i slično. Pritom se često koriste različite metode umjetne inteligencije koje prilikom treninga modela koriste prikladne podatke. Tako modeli na temelju uspješnog treninga vrše akviziciju informacija, te uče ili identificiraju objekte, pojave ili ljude. Pritom je moguće koristiti Python kao programski jezik te različite programske biblioteke kao što su OpenCV ili MediaPipe.

Cilj ovog rada je razviti detektor znakova temeljen na pokretima ruku za američki znakovni jezik (engl. American Sign Language – ASL). U radu je potrebno:

- kreirati skup podataka za trening modela u vidu slikovnih datoteka tako da se snime pokreti ruke koji su karakteristični za svako slovo abecede
- trenirati model umjetne inteligencije za poslove klasifikacije na temelju ranije kreiranog skupa podataka
- evaluirati dobiveno rješenje u sklopu Laboratorija za projektiranje izradbenih i montažnih sustava te dati kritički osvrt.

U radu je potrebno navesti korištenu literaturu i eventualno dobivenu pomoć.

Zadatak zadan: 9. svibnja 2024.

Zadatak zadao: Izv. prof. dr. sc. Tomislav Stipančić

Datum predaje rada: 11. srpnja 2024.

Predviđeni datumi obrane: 15. – 19. srpnja 2024.

Predsjednik Povjerenstva: Prof. dr. sc. Ivica Garašić

SADRŽAJ

POPIS SLIKA

POPIS TABLICA

POPIS OZNAKA

Oznaka	Opis
CNN	konvolucijska neuronska mreža (convolutional neural network)
NumPy	Numerical Python
OpenCv	Open Source Computer Vision Library
ReLU	ispravljena linearna jedinica (rectified linear unit)
RGB	Crvena – Zelena - Plava (Red – Green - Blue)
2D	dvodimenzionalno
3D	trodimenzionalno

SAŽETAK

Računalni vid se svojim razvojem proširio na brojne grane, pa tako i na pomaganje ljudima sa svakodnevnim poteškoćama s kojima se susreću. U ovom radu pokazana je primjena računalnog vida kod prepoznavanja i interpretacije znakovnog jezika u programskom jeziku Python koristeći OpenCV i MediaPipe biblioteke. Prije izrade modela navedeno je teorijsko upoznavanje sa pojmovima i bibliotekama potrebnim za izradu modela,te je zatim objašnjen programski kôd koji raspoznaje slova znakovnog jezika. Na kraju se evaluira dobiveno rješenje i opisuju načini za poboljšanje modela.

Ključne riječi: Python, MediaPipe, računalni vid, znakovni jezik

SUMMARY

With its development, computer vision has spread to numerous branches, including helping people with everyday difficulties they face. In this paper, the application of computer vision in the recognition and interpretation of sign language in the Python programming language using the OpenCV and MediaPipe libraries is demonstrated. Before creating the model, we have a theoretical introduction to the concepts and libraries necessary for creating the model, and then the program code that recognizes the letters of the sign language is explained. At the end, the obtained solution is evaluated and ways to improve the model are described.

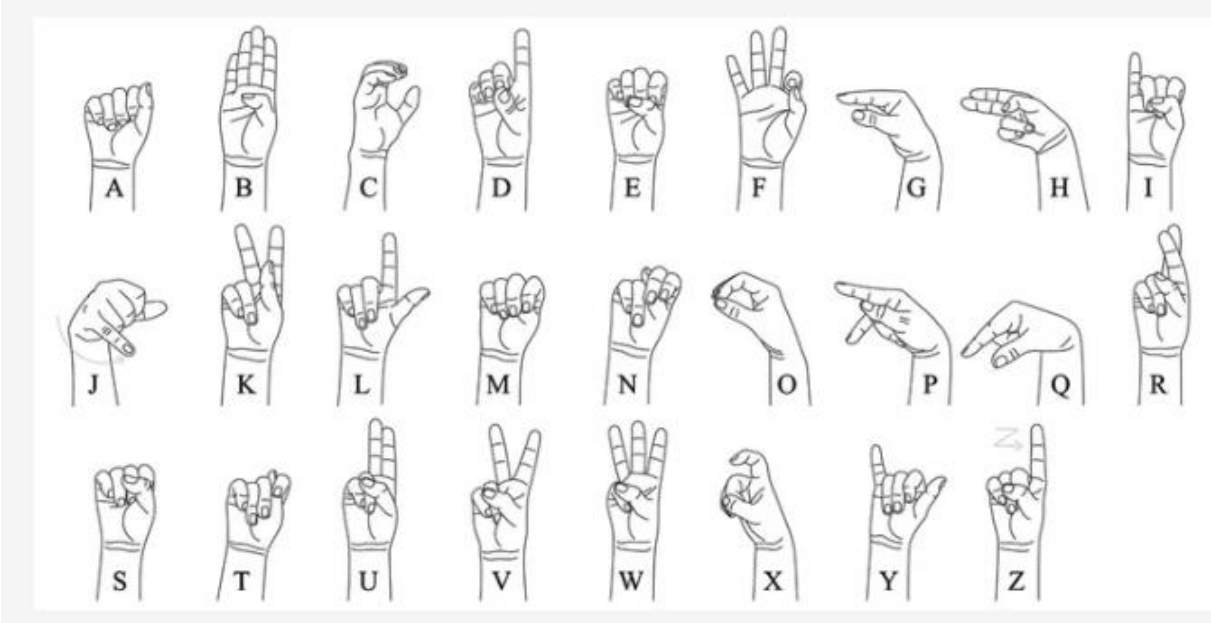
Key words: Python, MediaPipe, computer vision, sign language

1. UVOD

Povijest američkog znakovnog jezika (ASL) započela je 1814. godine kada je obrazovanje gluhih uvedeno u Sjedinjene Države. U to vrijeme nije postojao standardni znakovni jezik, ali su se u zajednicama gluhih razvili različiti sustavi znakova, danas poznati kao staroamerički znakovni jezik. Povijest ASL-a veže se uz Dr. Thomasa Hopkinsa Gallaudeta, svećenika iz Hartforda.Gallaudet je primijetio da je devetogodišnja gluha kći njegovog susjeda vrlo inteligentna, unatoč tome što nije mogla govoriti ni čuti, te je želio pronaći način da joj pomogne u komunikaciji. Međutim, nije imao uspjeha jer nije poznavao najučinkovitije metode obrazovanja gluhih. Gallaudet je dobio podršku zajednice i dovoljno novca da otputuje u Europu, gdje je obrazovanje gluhih bilo razvijenije. U Europi je upoznao Laurenta Clerca, jednog od najboljih učitelja na institutu za gluhe. Gallaudet je učio metode podučavanja i pohađao privatne sate kod Clerca. Zajedno su 1817. godine u Hartfordu osnovali prvu javnu besplatnu školu za gluhe u SAD-u, danas poznatu kao Američka škola za gluhe. Škola je brzo rasla, privlačeći gluhe učenike iz cijelih Sjedinjenih Država. Gallaudet je otišao u mirovinu 1830., dok je Clerc nastavio predavati do 1850-ih. Do 1863. godine, u SAD-u je bilo otvoreno dvadeset i dvije škole za gluhe, većinu kojih su osnovali Clercovi učenici koristeći njegove metode obrazovanja gluhih. [1]

Što se tiče Hrvatske, 2015. godine se donio Zakon o hrvatskom znakovnom jeziku i ostalim sustavima komunikacije gluhih i gluhoslijepih osoba u Republici Hrvatskoj, čime je hrvatski znakovni jezik postao priznat kao samostalan jezični sustav s vlastitim gramatičkim pravilima, potpuno neovisan o jeziku čujućih osoba. U današnje vrijeme, s razvojem tehnologije, umjetna inteligencija značajno olakšava život gluhim osobama. Umjetna inteligencija počinje imati veliki utjecaj na tumačenje znakovnog jezika, pružajući kreativna rješenja za prevladavanje izazova s kojima se zajednica gluhih suočava. Automatizirano tumačenje znakovnog jezika omogućuje prevođenje znakovnog jezika u govorni jezik i obrnuto, čime gluhim osobama više nije potreban prevoditelj za komunikaciju s bilo kim, na bilo kojem mjestu, u bilo koje vrijeme. Jedna od primjena ove tehnologije su rukavice za znakovni jezik koje koriste senzore i algoritme za detekciju i prijevod znakovnog jezika u govorni jezik u stvarnom vremenu, omogućujući gluhim osobama interakciju s bilo kim, bez obzira na poznavanje znakovnog jezika. Također, upotreba umjetne inteligencije za online usluge prevođenja znakovnog jezika omogućuje gluhim osobama komunikaciju s prevoditeljima putem mobitela ili računala,

otvarajući nove mogućnosti za pristup ključnim uslugama i informacijama, bez obzira na njihovu lokaciju. [2]



Slika 1.1 Jednoručna abeceda Američkog Znakovnog Jezika [1]

2. TEORIJSKA OSNOVA RADA

2.1. Umjetna inteligencija

Umjetna inteligencija je područje koje spaja više disciplina i koje se bavi razvojem i implementacijom računalnih sustava sposobnih za obavljanje zadataka koji zahtijevaju ljudske kognitivne vještine, kao što je učenje, razumijevanje, planiranje, prepoznavanje objekata i slično. Umjetna inteligencija se koristi u različitim područjima i uključuje stručne sustave, jezičnu obradu, identifikaciju govora i strojni vid. Sustavi umjetne inteligencije funkcioniraju kroz nekoliko ključnih koraka. Prvo se prikupljaju velike količine podataka koji su označeni ili klasificirani, a zatim se ti podaci koriste za obuku modela, analizirajući ih kako bi prepoznali obrasce. Na temelju tih prepoznatih obrazaca, sustavi mogu donositi odluke o novim, nepoznatim podacima. Prednosti AI uključuju učinkovitost u obavljanju detaljnih poslova, brzo izvršavanje zadataka koji uključuju velike količine podataka te dosljednost rezultata. Međutim, ovaj pristup zahtijeva duboku tehničku stručnost. AI može raditi samo s onim što je naučeno i nije u stanju generalizirati znanje s jednog zadatka na drugi. Tri vrste umjetne inteligencije podijeljene prema njijhovim karakteristikama navedene su u tablici. [6]

Uska umjetna inteligencija (UUI)	UUI je vrsta umjetne inteligencije koja je usmjerena na jedan zadatak ili područje, kao što je prepoznavanje govora ili lica, ili prevođenje jezika. UUI nije u stanju obavljati zadatke izvan svoje domene i nema svjesnost ili mogućnost razumijevanja šireg konteksta.
Opća umjetna inteligencija (OUI)	OUI je vrsta umjetne inteligencije koja teži obavljati širok spektar zadataka sličnih onima koje obavlja ljudska inteligencija, kao što je sposobnost učenja iz različitih izvora podataka, razumijevanja emocija i predlaganje novih ideja. OUI trenutno ne postoji, ali bi predstavljala vrhunac razvoja umjetne inteligencije, sposobnu prilagoditi se različitim situacijama, uz duboko razumijevanje okoline i ljudskih interakcija.

Superumjetna inteligencija (SUI)	SUI je vrsta umjetne inteligencije koja bi u teoriji mogla prestići ljudsku inteligenciju u svim spektrima, uključujući kreativnost, mudrost i društvene vještine. Kao takva bi imala spobnost kontrole nad drugim oblicima bilo prirodne ili umjetne inteligencije. Iako je SUI hipotetski koncept, ona već sada sa sobom povlači brojne rizike i etička pitanja.

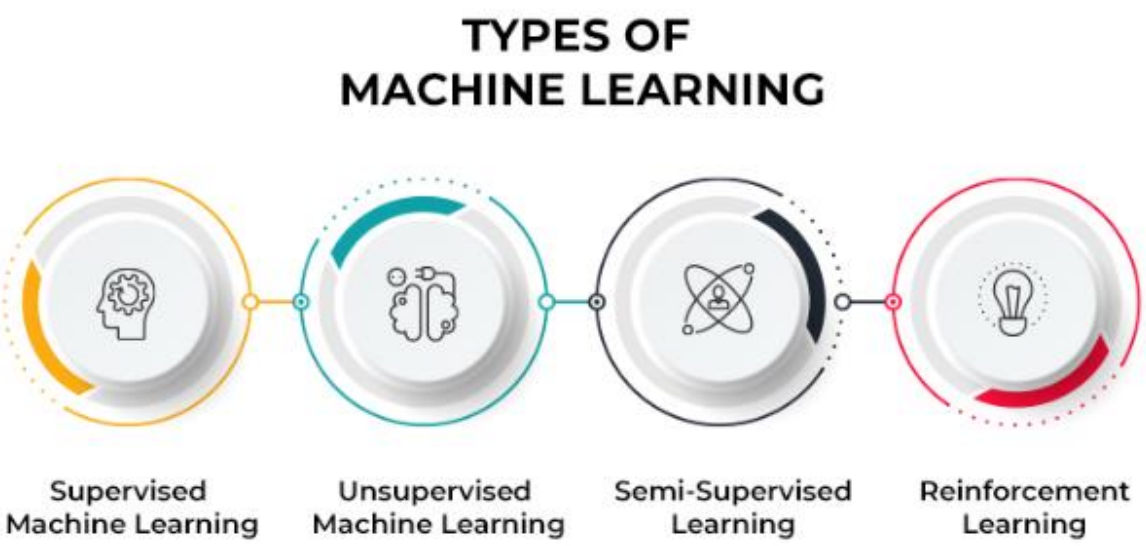
Tablica 1. Vrste umjetne inteligencije s obzirom na sposobnost

2.2. Strojno učenje

Strojno učenje spada u područje računalne znanosti i umjetne inteligencije, usredotočeno je na korištenje algoritama i podataka za imitaciju ljudskog učenja te na postepeno poboljšavanje točnosti algoritma. Navedeni algoritam strojnog učenja dijeli se na tri osonvna dijela, a to su:

1. Proces donošenja odluka: koristi algoritme strojnog učenja za klasifikaciju ili predviđanje. Na temelju podataka na ulazu, algoritam će generirati procjenu uzorka unutar tih podataka.
2. Funkcija pogreške: predstavlja funkciju koja procjenjuje uspješnost predikcije modela. Ukoliko su dostupni poznati primjeri, funkcija pogreške uspoređuje predikcije s tim primjerima kako bi se ocijenila preciznost modela.
3. Proces optimizacije modela: ukoliko model može bolje odgovarati podatkovnim točkama u skupu za obuku, u tom se slučaju prilagođavaju težine kako bi došlo do manje razlike između poznatog primjera i procjene modela. Iterativni proces optimizacije se ponavlja te se kontinuirano ažuriraju težine sve dok se ne dosegne zadovoljavajuća razina točnosti.

   Četiri osnovne kategorije strojnog učenja prikazane su na slici [Slika 2.1].



Slika 2.1 Tipovi strojnog učenja [3]

Nadzirano strojno učenje koristi označene skupove podataka za obuku algoritama kako bi se klasificirali podaci ili precizno predvidjeli ishodi. Dok ulazni podaci prolaze modelom, on prilagođava svoje težine kako bi postigao optimalno uklapanje. To se događa s ciljem da se izbjegne nedovoljno ili pak prekomjerno uklapanje. Nadzirano učenje se često koristi u različitim organizacijama za rješavanje raznih problema, poput filtriranja neželjene pošte iz pristigle pošte.

Nenadzirano učenje, s druge strane, koristi algoritme strojnog učenja za grupiranje i analizu podataka koji su neoznačeni (podskupovi koji se nazivaju klasteri). Ovi algoritmi mogu samostalno otkriti skrivene uzorke ili grupe unutar podataka bez ljudske pomoći. Ova metoda je posebno korisna za istraživačku analizu podataka i prepoznavanje slika i obrazaca zbog svoje sposobnosti da otkrije sličnosti i razlike među informacijama.

Polu-nadzirano učenje predstavlja kombinaciju nenadziranog i nadziranog učenja. Ovaj pristup koristi manji označeni skup podataka da bi se usmjerila klasifikacija i izdvojile značajke iz većeg skupa podataka koji je neoznačen. Polu-nadzirano učenje je korisno u situacijama kada nema dovoljno označenih podataka za potpuno nadzirano učenje ili kada je previše skupo označiti dovoljan broj podataka.

Pojačano učenje je model strojnog učenja sličan nadziranom učenju, ali se algoritam ne trenira korištenjem unaprijed označenih podataka. Umjesto toga, uči putem pokušaja i pogrešaka. Uspješni ishodi se ojačavaju kako bi se razvile optimalne preporuke za određeni problem.[3]

2.3. Duboko učenje

Osim strojnog učenja, važno je istaknuti i naglasiti i duboko učenje, također jednu od ključnih grana umjetne inteligencije. Iako se često spominju naizmjenično, između dubokog i strojnog učenja je velika razlika. Neuronske mreže, duboko i strojno učenje spadaju u potpodručja umjetne inteligencije. Neuronske mreže su dio strojnog učenja, dok je dubinsko učenje specifično područje unutar neuronskih mreža. Načini učenja algoritama različiti su kod dubokog i strojnog učenja. Iako nije nužno, duboko strojno učenje može koristiti označene skupove podataka kako bi informirao svoj algoritam. Za razliku od strojnog učenja, duboko učenje može direktno raditi s nestrukturiranim podacima poput teksta ili slika, automatski identificirajući značajke koje razlikuju različite kategorije podataka. To smanjuje potrebu za ljudskom intervencijom i omogućuje obradu većih količina podataka, dok strojno učenje često zahtijeva značajnu ljudsku pomoć za definiranje relevantnih značajki i obično ovisi o strukturiranim podacima. Specijalisti u strojnom učenju moraju odrediti skup značajki kako bi model mogao razumjeti razlike između unosa podataka, što često zahtijeva više strukturiranih podataka za proces učenja. [5] Duboko učenje, temeljeći se na višeslojnim neuronskim mrežama, omogućilo je brze napretke u mnogim područjima, npr. u području računalnog vida. Duboko učenje grana je strojnog učenja koja se sastoji od neuronske mreže s tri ili više slojeva: ulazni sloj kroz kojeg ulaze podaci, skriveni slojevi koji obrađuju i prenose podatke na druge slojeve te izlazni sloj u kojem se izrađuje konačni rezultat ili predviđanje. Rješenja za strojno učenje nude mnogo prilagodbi koje se nazivaju hiperparametrima koje se mogu podešavati u svrhu optimizacije učenja algoritma iz podataka. Rješenja za duboko učenje također koriste hiperparametre, ali koriste i više korisnički konfiguriranih slojeva (korisnik određuje broj i vrstu). Zapravo, ovisno o rezultirajućoj neuronskoj mreži, broj slojeva može biti prilično velik i tvoriti jedinstvene neuronske mreže sposobne za specijalizirano učenje: neki mogu naučiti prepoznavati slike, dok drugi mogu detektirati i analizirati glasovne naredbe.

Slika 2.2 Razlike između strojnog i dubokog učenja [5]

2.4. Umjetne neuronske mreže

Umjetna neuronska mreža predstavlja umjetnu imitaciju ljudskog mozga kojom se pokušava simulirati postupak učenja. Analogija s pravim ljudskim mozgom nije ni blizu slična jer postoje još mnoga područja živčanog sustava koja nisu replicirana uz pomoć umjetnih neuronskih mreža. Istovremeno postoje i karakteristike umjetnih neronskih mreža koje se ne poklapaju s biološkim sustavima. Neuroni ili živčane stanice su osnovne jedinice živčanog sustava i odgovorni su za prijenos i obradu informacija u tijelu. Sastoje se od staničnog tijela, dendrita koji primaju signale, aksona koji šalje signale dalje, i sinapsi koje su spojevi između neurona gdje dolazi do prijenosa informacija putem kemijskih ili električnih signala.[4] Neuroni su međusobno povezani i prenose informacije različitim elektrokemijskim putevima. Na slici [Slika 2.3] je prikazana građa biološkog neurona. Neuronske mreže imaju nekoliko prednosti u odnosu na tradicionalne metode obrade podataka:

• Mogu obrađivati nejasne ili nepotpune podatke što je tipično kada se radi o podacima iz različitih vrsta senzora poput kamera i mikrofona, te raspoznavanju uzorka među njima.

• Učinkovite su s velikim brojem varijabli ili parametara.

• Mogu se prilagoditi novim uvjetima ili podacima iz okoline. [19]

• Imaju mogućnost stjecanja znanja kroz učenje na temelju primjera.

Neuronske mreže vrlo uspješno rješavaju zadatke klasifikacije i predviđanja, gdje postoji veza između izlaznih i ulaznih varijabli, bez obzira na kompleksnost te veze (nelinearnost). Današnja primjena neuronskih mreža vezana je uz različita područja:

• prepoznavanje uzoraka,

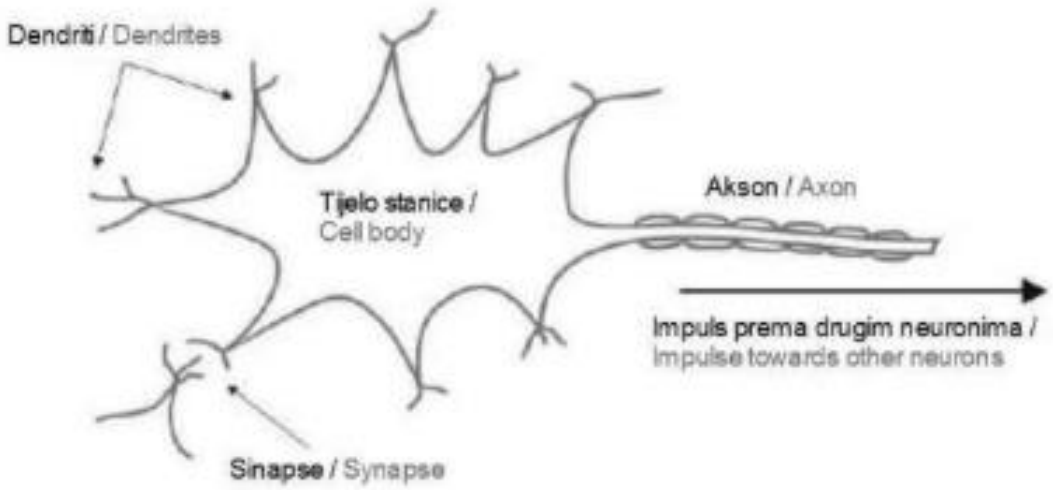
• obrada slike,

• obrada govora,

• problemi optimizacije,

• nelinearno upravljanje,

• simulacije i sl. [4]



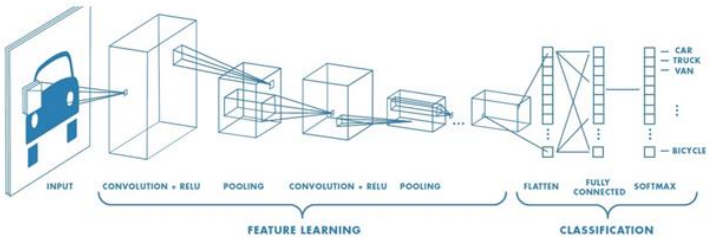
Slika 2.3 Građa biološkog neurona [19]

2.5. Konvolucijske neuronske mreže (CNN)

Konvolucijske neuronske mreže koriste trodimenzionalne podatke za klasifikaciju slika i prepoznavanje objekata. Neuronske mreže, koje su podskup strojnog učenja, čine srž algoritama dubokog učenja. One se sastoje od više slojeva čvorova: ulaznog sloja, jednog ili više skrivenih slojeva te izlaznog sloja. Za specifične slučajeve upotrebe i tipove podataka postoje različite vrste neuronskih mreža. Na primjer, rekurentne neuronske mreže često su u upotrebi za prepoznavanje govora i obradu prirodnog jezika, dok su konvolucijske neuronske mreže (ConvNets ili CNN) češće primjenjene za zadatke računalnog vida i također za zadatke klasifikacije. Konvolucijske neuronske mreže omogućuju pristup koji je skalabilniji za klasifikaciju slika i prepoznavanje objekata, te koristi principe linearne algebre, posebno matričnog množenja, za identifikaciju uzoraka unutar slike. Ipak, ove mreže mogu biti zahtjevne za računala te zahtijevaju upotrebu grafičkih procesorskih jedinica (GPU) za treniranje modela. [7] Konvolucijske neuronske mreže razlikuju se od ostalih neuronskih mreža svojim superiornim performansama s ulaznim signalima slike, govora ili zvuka. Postoje tri glavne vrste slojeva, a to su:

1. Konvolucijski sloj
2. Sloj sažimanja
3. Potpuno povezani (FC) sloj

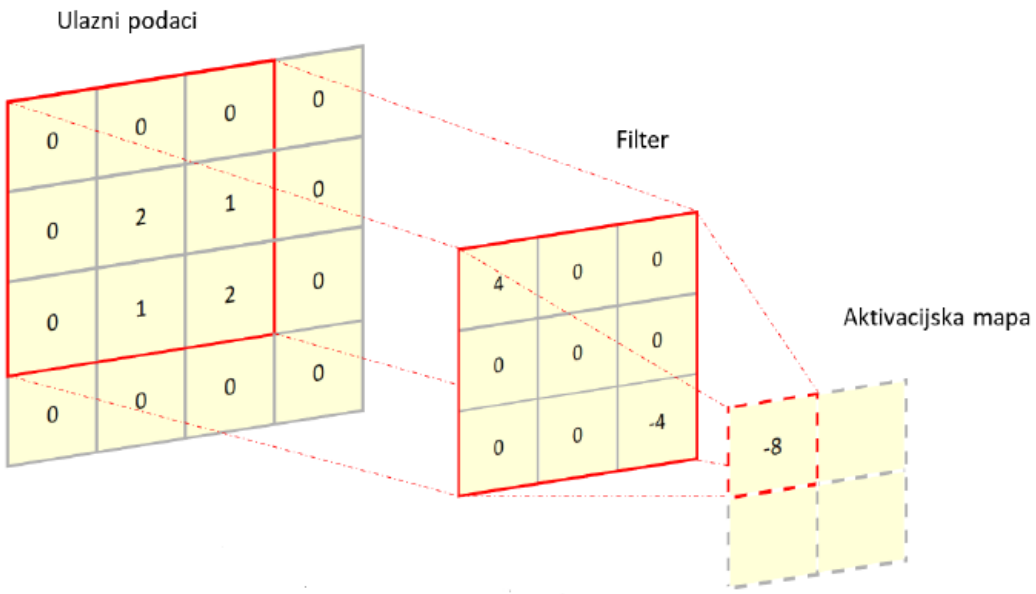
Konvolucijski sloj predstavlja prvi sloj konvolucijske neuronske mreže (CNN). Nakon početnog konvolucijskog sloja, mogu slijediti dodatni konvolucijski slojevi ili slojevi za udruživanje. Konačno, potpuno povezani sloj često predstavlja posljednji sloj u arhitekturi CNN-a. Svaki sloj CNN-a postaje sve složeniji kako procesira slikovne podatke. Raniji slojevi se fokusiraju na jednostavne značajke poput boja i rubova, dok kasniji slojevi prepoznaju sve veće elemente ili cjelovite oblike objekata, što dovodi do konačnog prepoznavanja ciljnog objekta. Na slici [Slika 2.4] prikazana je konvolucijska neuronska mreža.



Slika 2.4 Konvolucijska neuronska mreža [7]

Konvolucijski sloj je ključna građevna jedinica u arhitekturi CNN-a, gdje se odvija većina izračuna. Ovaj sloj sastoji se od nekoliko ključnih komponenti: ulaznih podataka, filtera i karte značajki. Ako se pretpostavi da je ulaz slika u boji, sastoji se od matrice piksela u 3D obliku, što znači da ima tri dimenzije: visinu, širinu i dubinu, što odgovara RGB komponentama slike. Također imamo detektor značajki, poznat kao kernel ili filter, koji se kreće kroz receptivna polja slike, provjeravajući prisutnost značajki. Ovaj proces poznat je kao konvolucija. Detektor obilježja je dvodimenzionalni niz težina koji predstavlja segment slike. Veličina filtra obično je matrica 3x3, što određuje i veličinu receptivnog polja, iako ta veličina može varirati. Filtar se primjenjuje na određeno područje slike, a zatim se računa točkasti produkt između ulaznih piksela i filtra. Dobiveni točkasti umnožak se zatim dodaje u izlazni niz. Postupak se ponavlja tako što se filtar pomakne korak po korak preko cijele slike, čime se generira konačna mapa značajki, poznata i kao aktivacijska mapa ili konvolvirana značajka. Konvolucijski sloj transformira sliku u numeričke vrijednosti, omogućujući neuronskoj mreži da izdvaja i analizira bitne uzorke. Uzorkovanje (padding) je tehnika koja se koristi kako bi se prilagodile dimenzije slike kada se filter ne uklapa savršeno u ulaznu sliku. Postoje dvije različite operacije uzorkovanja: prva je nula-uzorkovanje (zero-padding), gdje se podaci dopunjuju nulama, dok je druga valjano uzorkovanje (valid padding), gdje se izbjegava dopunjavanje i koristi samo valjani dio podataka koji su obuhvaćeni filterom. [7]

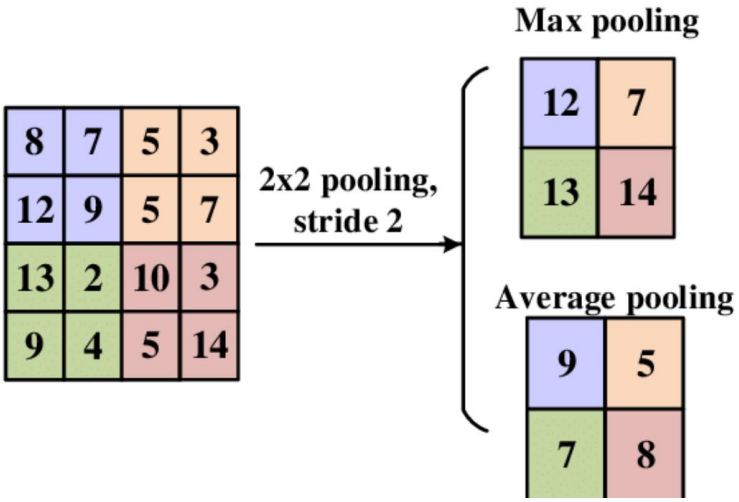
Na slici [Slika 2.5] je prikazan primjer konvolucijskog sloja s ulaznom mapom značajki, receptivnim poljima (engl. local receptive field), filterom te izlaznom mapom značajki.



Slika 2.5 Primjer konvolucijskog sloja [7]

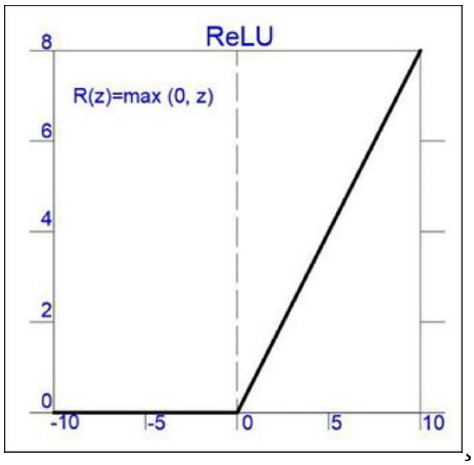
Sloj sažimanja reducira dimenzionalnost ulaza, smanjujući broj parametara. Slično kao kod konvolucijskog sloja, operacija sažimanja prelazi filterom preko cijelog ulaza, ali za razliku od konvolucijskog sloja, ovaj filter nema težine. Umjesto toga, kernel primjenjuje funkciju agregacije na vrijednosti unutar receptivnog polja kako bi generirao izlazni niz. Postoje dvije glavne vrste sažimanja:

1. Maksimalno sažimanje: kada se filter pomiče preko ulaza, odabire se piksel s maksimalnom vrijednošću za prenošenje u izlazni niz.
2. Prosječno sažimanje: kada se filter pomiče preko ulaza, izračunava se prosječna vrijednost unutar receptivnog polja za prenošenje u izlazni niz.

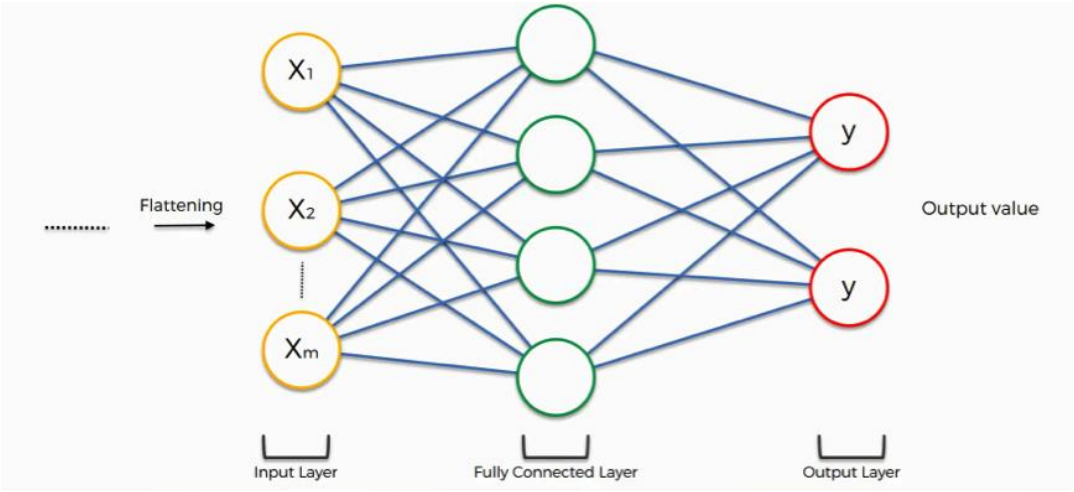


Slika 2.6 Prikaz sloja sažimanja [7]

Iako sloj sažimanja rezultira gubitkom mnogo informacija, donosi i niz prednosti za CNN, uključujući smanjenje složenosti i poboljšanje učinkovitosti. Ranije je spomenuto da vrijednosti piksela ulazne slike nisu direktno povezane s izlaznim slojem u parcijalno povezanim slojevima. Međutim, u potpuno povezanom sloju, svaki čvor u izlaznom sloju je izravno povezan s čvorom u prethodnom sloju. Ovaj sloj provodi klasifikaciju na temelju značajki izvučenih iz prethodnih slojeva i njihovih različitih filtera. Dok konvolucijski sloj i sloj sažimanja obično koriste ReLU funkcije, potpuno povezani slojevi koriste Softmax aktivacijsku funkciju za odgovarajuću klasifikaciju ulaza, što rezultira vjerojatnostima u rasponu od 0 do 1. Softmax funkcija je vrlo slična sigmoidnoj funkciji. Funkcija Softmax se koristi u klasifikaciji samo kada su klase međusobno isključive. [20] U posljednjem sloju CNN-a, odgovornom za konačnu predikciju klase ulazne slike, Softmax osigurava da ta predikcija bude vjerojatnosna distribucija. To znači da svaka klasa ima dodijeljenu odgovarajuću vjerojatnost. Takva distribucija vjerojatnosti omogućava preciznije predviđanje klase ulazne slike. [7]



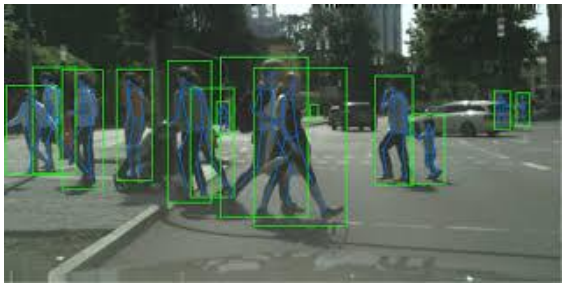
Slika 2.7 Prikaz ReLu funkcije [19]



Slika 2.8 Prikaz potpuno povezanog sloja [19]

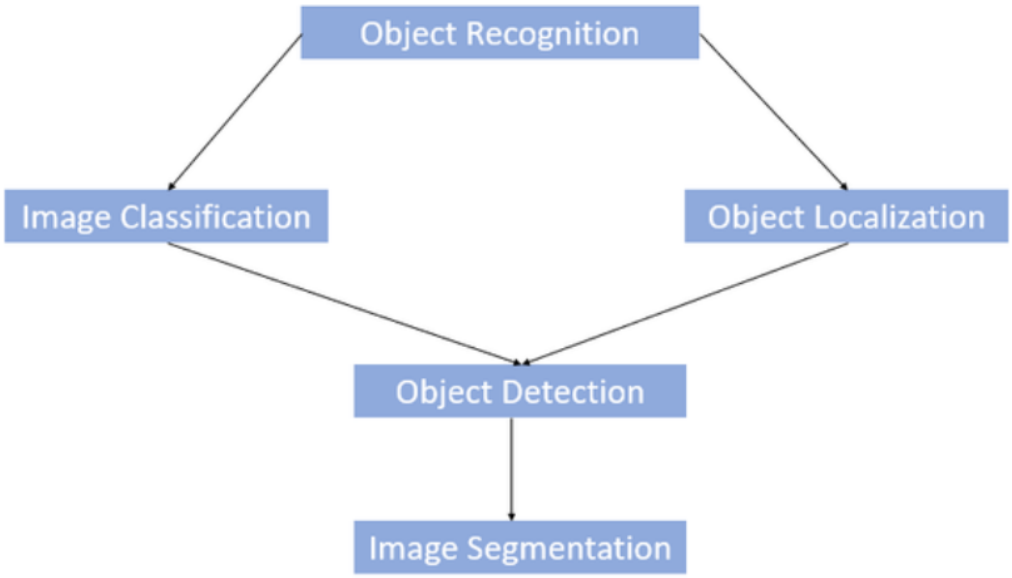
2.6. Računalni vid

Računalni vid je grana računalne znanosti koja se bavi replikacijom ljudskog vida, omogućujući računalima da prepoznaju i obrađuju objekte u video zapisima i slikama na način sličan ljudskom. Dok umjetna inteligencija omogućuje računalima razmišljanje, računalni vid im omogućuje vizualnu percepciju i razumijevanje. Za učinkovito funkcioniranje, računalni vid zahtijeva velike količine podataka. Kroz ponovljene analize podataka, računala uče prepoznavati slike identificiranjem razlika. Dvije ključne tehnologije koje se koriste za postizanje ovog cilja su duboko učenje i konvolucijske neuronske mreže (CNN). [9] Strojno učenje koristi modele algoritama koji pružaju mogućnost učenja iz vizualnih podataka. Kada se modelu pruži određena količina podataka, računalo ih može analizirati i naučiti prepoznati razlike između različitih slika. Algoritmi omogućuju strojevima da uče samostalno, bez potrebe za eksplicitnim programiranjem za prepoznavanje slika. Konvolucijske neuronske mreže (CNN) podržavaju modele strojnog učenja ili dubokog učenja na način da razlažu slike na piksele te im dodjeljuju oznake. Te oznake se koriste za provođenje konvolucija (matematičkih operacija koje stvaraju novu funkciju iz dvije ulazne funkcije) i za predviđanja o tome što "vide". Neuronska mreža provodi konvolucije i kroz niz ponavljanja provjerava točnost svojih predviđanja dok ne postanu precizna. Taj proces se naziva prepoznavanjem ili analizom slika na način sličan ljudskom viđenju. Kao što ljudi najprije primijete oštre rubove i jednostavne oblike, a zatim dodaju detalje, tako i konvolucijska neuronska mreža (CNN) prvo prepoznaje osnovne elemente slike, a potom ih nadopunjuje dodatnim informacijama kroz ponovljena predviđanja. CNN se koristi za razumijevanje pojedinačnih slika, dok se rekurentna neuronska mreža (RNN) primjenjuje u video aplikacijama kako bi pomogla računalima da razumiju povezanost slika u slijedu okvira. Računalni vid primjenjuje se u raznim industrijama, kao što su prijevoz, medicina, proizvodnja, građevina i mnoge druge. Primjeri njegove upotrebe uključuju autonomno upravljanje vozilima, prepoznavanje pješaka, analizu rendgenskih snimaka, otkrivanje karcinoma, inspekciju proizvoda radi otkrivanja grešaka, čitanje bar kodova i slično.[9]



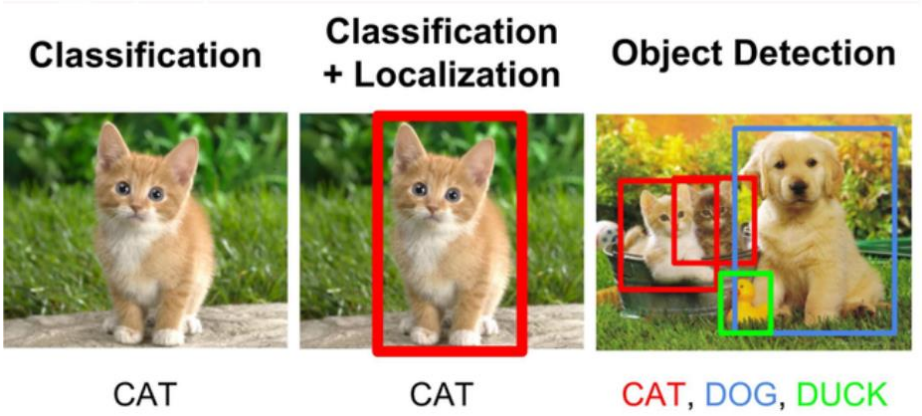
Slika 2.9 Jedna od primjena računalnog vida – detekcija pješaka [9]

Prepoznavanje objekata,jedan je od osnovnih zadataka računalnog vida, koji se bavi identifikacijom objekata različitih klasa unutar digitalnih vizualnih prikaza poput slika ili videozapisa. [9] Na slici su prikazani zadaci povezani s računalnim vidom.



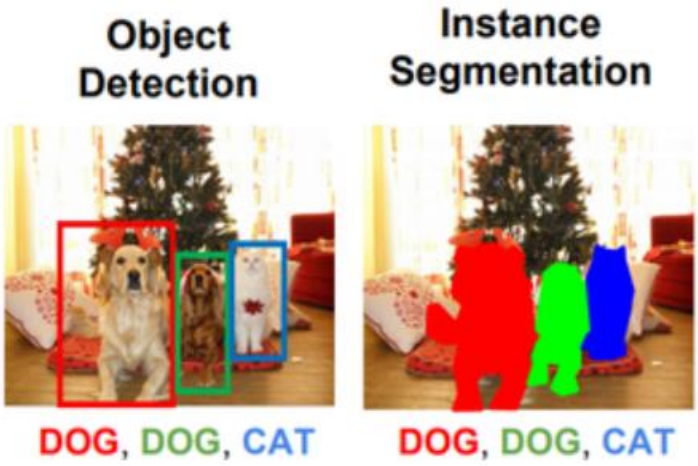
Slika 2.10 Zadaci računalnog vida [9]

Postoje tri zadatka računalnog vida: lokalizacija objekata, klasifikacija slika i detekcija objekata. U klasifikaciji slika, uzima sliku kao ulaz i ispisuje klasifikacijsku oznaku te slike s nekom metrikom (vjerojatnost, gubitak, točnost itd.). Na primjer: slika mačke može se s određenom vjerojatnošću klasificirati kao oznaka klase "mačka" ili slika psa može se klasificirati kao oznaka klase "pas". Kod lokalizacije objekta postoji algoritam koji locira prisutnost objekta na slici i predstavlja ga graničnim okvirom. Uzima sliku kao ulaz i ispisuje lokaciju graničnog okvira u obliku (položaj, visina i širina). Algoritmi za detekciju objekata djeluju kao kombinacija klasifikacije slike i lokalizacije objekta. Uzima sliku kao ulaz i proizvodi jedan ili više graničnih okvira s oznakom klase pričvršćenom za svaki granični okvir. Ovi algoritmi su dovoljno sposobni da se bave klasifikacijom i lokalizacijom više klasa, kao i da se bave objektima koji se višestruko pojavljuju. Na slici su prikazane razlike između ova tri zadatka računalnog vida. [8]



Slika 2.11 Razlike između zadataka računalnog vida[8]

Segmentacija slike daljnje je proširenje detekcije objekta gdje su prepoznati objekti označeni isticanjem specifičnih piksela tog objekta umjesto stvaranja okvira oko objekta. Ova tehnika je preciznija od generiranja graničnog okvira jer nam može pomoći u određivanju oblika svakog objekta prisutnog na slici jer umjesto crtanja graničnih okvira, segmentacija pomaže u određivanju piksela koji čine taj objekt. Ova granularnost nam pomaže u raznim područjima kao što su obrada medicinske slike, satelitsko snimanje itd. [12]



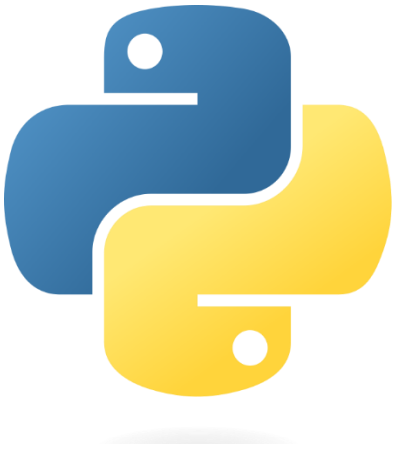
Slika 2.12 Razlika između detekcije objekata i segmentacije slike[12]

3. KORIŠTENE DATOTEKE

Za izradu rada korišten je Python verzije 3.10 jer je najbolje funkcionirao sa potrebnim bibliotekama za izvedbu aplikacije. Uvedene su biblioteke OpenCV, MediaPipe i NumPy te je također korišten Teachable Machine za treniranje odnosno učenje modela.

3.1. Python

Python je programski jezik visoke razine koji je interpretiran i objektno orijentiran. Privlačan je za brzi razvoj aplikacija te je idealan za korištenje kao skriptni jezik ili za povezivanje postojećih komponenti. Njegova jednostavna sintaksa i podrška za module i pakete potiču modularnost programa i ponovnu upotrebu koda. Pythonova standardna biblioteka je jako bogata i dostupna je za sve glavne platforme u binarnom ili izvornom obliku. Python omogućuje programerima upotrebu različitih načina programiranja, uključujući objektno orijentirano, strukturno i aspektno orijentirano programiranje. Ova fleksibilnost doprinosi sve većoj popularnosti jezika. Često se koristi za izradu softvera i web stranica, automatizaciju zadataka te analizu podataka. Kao jezik opće namjene, Python nije specijaliziran za određene probleme, već omogućuje stvaranje širokog spektra različitih programa. Njegova svestranost i pristupačnost za početnike učinile su ga jednim od najčešće korištenih programskih jezika danas. [10]



Slika 3.1 Python logo[10]

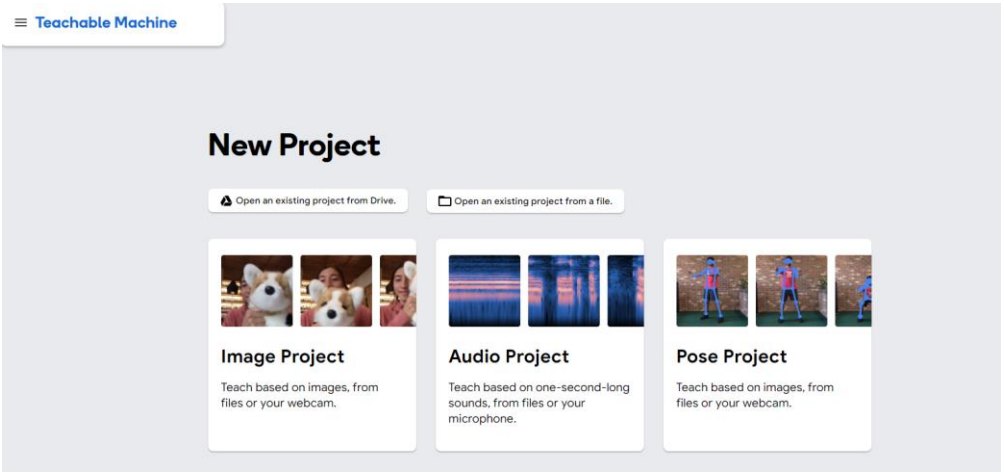
3.2. Teachable Machine

Teachable Machine je web-based GUI alat za stvaranje prilagođenih modela klasifikacije strojnog učenja (Machine Learning-ML) bez specijalizirane tehničke stručnosti. Stvoren je kako bi se pomoglo studentima, nastavnicima, dizajnerima i drugima da nauče o strojnom učenju stvaranjem i korištenjem vlastitih modela klasifikacije. Sadrži fleksibilno, pristupačno sučelje za ML klasifikacijske modele bez stručnosti u kodiranju.

Teachable Machine je alat koji se nalazi na webu te on omogućuje stvaranje modela strojnog učenja i uz to ga čini brzim i lakim. Funkcionira na način da se uvježba računalo da prepozna slike, zvukove i poze bez pisanja koda za strojno učenje. Zatim se može upotrijebiti u vlastitim projektima, aplikacijama i sl. U repozitoriju su sadržane dvije komponente Teachable Machine:

- Odjeljak biblioteka koji sadrži sav kod za strojno učenje koji se koristi u Teachable Machine. Teachable Machine koristi TensorFlow.js, biblioteku za strojno učenje u Javascriptu, za obuku i pokretanje modela koje korisnik napravi u svom web pregledniku.
- Odjeljak isječaka (snippets) koji sadrže kod i upute o tome kako koristiti izvezene modele iz Teachable Machine u jezicima kao što su Javascript, Java i Python.

Teachable Machine radi na način da korisnik može unijeti podatke putem kamere, mikrofona ili učitavanjem datoteka te ti podaci mogu biti slike, zvukovi ili pokreti. Zatim korisnik označava te podatke na način da svaki uzorak pripada određenoj klasi. Potom Teachable Machine koristi unaprijed definirane arhitekture neuronskih mreža za treniranje modela. Tijekom procesa treniranja modela, alat koristi označene podatke kako bi naučio prepoznati uzorke i razlike između različitih klasa. Nakon treniranja, korisnik može testirati model uživo koristeći novu kameru, mikrofon ili unos datoteka kako bi vidio kako model klasificira nove uzorke te alat daje povratne informacije o točnosti i performansama modela. [18] U ovom radu je Teachable Machine korišten za treniranje modela da prepozna različite poze u kojima se nalazi dlan te da te poze poveže sa slovima abecede znakovnog jezika. Na slici je prikazano sučelje Teachable Machine-a.



Slika 3.2 Sučelje Teachable Machine-a [23]

3.3. OpenCV

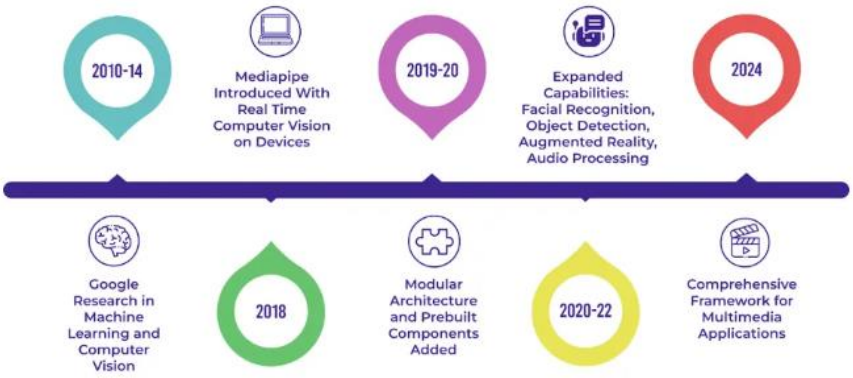
OpenCV (Open Source Computer Vision Library) je biblioteka softvera za strojno učenje i računalni vid otvorenog koda. OpenCV osmišljen je kako bi se osigurala zajednička podloga za aplikacije računalnog vida i ubrzalo korištenje strojne percepcije u komercijalnim proizvodima. OpenCv se primjenjuje u raznim situacijama, a neke od njih su: [11]

1. Prepoznavanje lica
2. Automatizirani pregled i nadzor
3. Brojanje ljudi
4. Brojanje vozila na autocestama zajedno s njihovim brzinama
5. Interaktivne umjetničke instalacije
6. Otkrivanje defekata u proizvodnom procesu
7. Pretraživanje i dohvaćanje videa/slike
8. Navigacija i kontrola automobila bez robota i vozača
9. Prepoznavanje predmeta
10. Analiza medicinske slike

Biblioteka sadrži više od 2500 optimiziranih algoritama, u što je uključen opsežan skup najsuvremenijih i klasičnih algoritama strojnog učenja i računalnog vida. Ovi algoritmi mogu biti korišteni za otkrivanje i prepoznavanje lica, identifikaciju objekata, klasificiranje ljudskih radnji u videozapisima, praćenje pokreta kamere, praćenje objekata u pokretu, izdvajanje 3D modela objekata, spajanje slika kako bi se proizvela visoka rezolucija slike cijelog prizora, prepoznavanje krajolika i postavljanje markera za prekrivanje s proširenom stvarnošću, itd. Ima C++, Python, Java i MATLAB sučelja i podržava Windows, Linux, Android i Mac OS. OpenCV je izdan pod BSD licencom i besplatan. [11]

3.4. MediaPipe

MediaPipe je biblioteka koja koristi princip strojnog učenja za obradu vremenskih serija podataka kao što su video, audio i dr. Namijenjena je za rad na više platformi te radi na stolnom računalu, Androidu, iOS-u i ugrađenim uređajima kao što su Jetson Nano i Raspberry Pi. Unutar MediaPipe grafa, čvorovi se nazivaju kalkulatori, a rubovi se zovu tokovi. MediaPipe radi kao okvir za programiranje protoka podataka. Podaci se kreću kroz niz povezanih 'kalkulatora', od kojih svaki obavlja određeni zadatak na podacima prije nego što ih proslijedi sljedećem. Ti kalkulatori su kao čvorovi povezani tokovima podataka. Svaki tok predstavlja niz paketa podataka. Korijeni MediaPipea sežu u rane 2010-e kada je Google radio na poboljšanju tehnologija strojnog učenja i računalnog vida. Prvi put je korišten 2012. za analizu videa i zvuka u stvarnom vremenu na YouTubeu. U 2018. MediaPipe je počeo rješavati probleme vezane uz korištenje složenih modela računalnog vida na uređajima poput pametnih telefona i malih računala. Do 2020. godine postojala je sve veća potreba za brzim i učinkovitim načinom obrade multimedije, pa je Mediapipe ažuriran istim. [21] Sada Mediapipe ostaje snažan okvir za programere koji žele stvarati inovativne multimedijske aplikacije koje stvarno dobro rade. Na slici je prikazan tok razvoja MediaPipe-a.



Slika 3.3 Tok razvoja MediaPipe biblioteke [21]

MediaPipe dolazi s mnogim značajkama. Jedna od njih je mogućnost korištenja ogromne snage grafičkih procesorskih jedinica (GPU) za bržu obradu. Korištenjem GPU-a za zadatke koji zahtijevaju puno računalne snage, MediaPipe može podnijeti čak i najzahtjevnije multimedijske zadatke u stvarnom vremenu. Zahvaljujući svojoj sposobnosti paralelne obrade, može istovremeno obavljati nekoliko aktivnosti, poput obrade mnogih videostreamova ili pokretanja više modela računalnog vida. MediaPipe također koristi OpenCV, moćnu biblioteku otvorenog koda za računalni vid, koja nudi brojne alate i algoritme za rad sa slikama i videozapisima. Uz to, MediaPipe se udružuje s TensorFlowom, Googleovim alatom za strojno učenje, što omogućava jednostavno dodavanje unaprijed obučenih ili prilagođenih modela. Ovo olakšava zadatke poput prepoznavanja lica ili razumijevanja govora. MediaPipe podržava popularne programske jezike kao što su C++, Java i Python, što ga čini jednostavnim za integraciju u različite projekte. Neke od ključnih značajki MediaPipe-a uključuju:

- Prethodno obučeni modeli: nudi modele spremne za rad kako bi se olakšala brza integracija u aplikacije.
- Prilagodba uz MediaPipe Model Maker: omogućuje prilagođavanje modela za rješenja s određenim podacima.
- Evaluacija i usporedna analiza: pomaže u vizualizaciji, evaluaciji i usporednoj analizi rješenja izravno u pregledniku.
- Učinkovita obrada na uređaju: MediaPipe je optimiziran za strojno učenje na uređaju, osiguravajući performanse u stvarnom vremenu bez oslanjanja na obradu u oblaku.

Rješenja ove biblioteke su unaprijed izgrađeni primjeri otvorenog koda koji se temelje na određenom unaprijed obučenom modelu. U nastavku je nabrojano svih 16 rješenja biblioteke: [21]

1. Face Detection
2. Face Mesh
3. Iris
4. Hands
5. Pose
6. Holistic
7. Selfie Segmentation
8. Hair Segmentation
9. Object Detection
10. Box Tracking
11. Instant Motion Tracking
12. Objectron
13. KNIFT
14. AutoFlip
15. MediaSequence
16. YouTube 8M

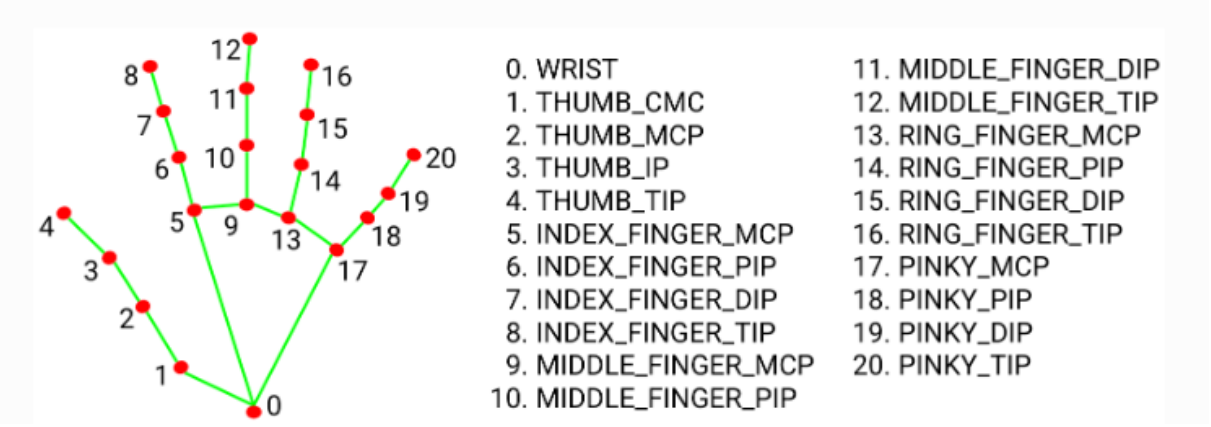
U sklopu ovog rada korišteno je rješenje pod brojem 4 pod nazivom MediaPipe Hands. MediaPipe Hands je rješenje za praćenje ruku i prstiju visoke točnosti. Koristi strojno učenje (ML) kako bi prikazao 21 3D orijentir ruke iz samo jednog okvira. MediaPipe Hands koristi ML sustav koji se sastoji od više modela koji rade zajedno. Model detekcije dlana radi na cijeloj slici i vraća orijentirani granični okvir ruke. Model orijentira ruke koji radi na izrezanom području slike definiranom detektorom dlana i vraća 3D ključne točke ruke. Ova je strategija slična onoj korištenoj u rješenju MediaPipe Face Mesh, koje koristi detektor lica zajedno s modelom orijentira lica. Pružanje točno izrezane slike ruke modelu orijentira ruke drastično smanjuje potrebu za povećanjem podataka (npr. rotacije i translacije) i umjesto toga omogućuje mreži da većinu svojih kapaciteta posveti točnosti predviđanja koordinata.

3.4.1. Model detekcije dlana

Dok lica imaju uzorke visokog kontrasta, npr. u području očiju i usta, nedostatak takvih obilježja u rukama čini relativno teškim njihovo pouzdano otkrivanje samo na temelju njihovih vizualnih obilježja. Umjesto toga, pružanje dodatnog konteksta, kao što su karakteristike ruke, tijela ili osobe, pomaže u preciznoj lokalizaciji ruke. MediaPipe metoda rješava gore navedene izazove koristeći različite strategije. Prvo, trenira se detektor dlanova umjesto detektora ruku, budući da je procjena graničnih okvira krutih predmeta poput dlanova i šaka znatno jednostavnija od otkrivanja ruku sa zglobnim prstima. Dlanovi se mogu modelirati pomoću kvadratnih graničnih okvira zanemarujući druge omjere širine i visine. [13]

3.4.2. Model orijentira ruku

Nakon detekcije dlana na cijeloj slici, naknadni model orijentira šake izvodi preciznu lokalizaciju ključne točke za 21 3D koordinate šake i zgloba unutar područja šake koja se detektiraju putem izravnog predviđanja koordinate. Model uči dosljedan interni prikaz položaja ruku i robustan je čak i za djelomično vidljive ruke. Kako bi se dobili istiniti podaci, ručno je označeno oko 30 tisuća slika stvarnog svijeta s 21 3D koordinatom. Kako bi se bolje pokrili mogući položaji ruku i pružio se dodatni nadzor nad prirodom geometrije ruku, također se renderira sintetički model ruke vioke kvalitete preko različitih pozadina i mapira se na odgovarajuće 3D koordinate. Na slici su prikazane karakteristične točke šake. [13]



Slika 3.4 Karakteristične točke šake[13]

4. IZVEDBA ZADATKA

Zadatak je izveden na način da su prvo prikupljeni podaci odnosno napravljene slike položaja dlana za svako slovo abecede te su zatim te slike dodane u Teachable Machine kako bi se istrenirao model i onda je sve skupa dodano u finalni programski kôd za prepoznavanje slova abecede znakovnog jezika.

4.1. Prikupljanje podataka

U nastavku je objašnjen programski kôd za prikupljanje podataka. Prvo je napravljen folder unutar projekta naziva 'data' unutar kojeg se nalaze folderi sa svim potrebnim slovima u koje će se spremati slike u sklopu prikupljanja podataka. Prvi korak je uvođenje biblioteke OpenCV te modula za detekciju ruku unutar nje i uvođenje NumPy biblioteke.

```
import cv2
from cvzone.HandTrackingModule import HandDetector
import numpy as np
```

Slika 4.1 Uvođenje biblioteka

Drugi korak je inicijalizacija web kamere te kreiranje glavne petlje koja će kontinuirano hvatati okvir slike prikazane na kameri. Nula u zagradi označava ID broj web kamere korištenog laptopa. Zatim se prikaže slika u prozoru pod nazivom 'Image' te je postavljeno da se čeka jedna milisekunda od prikaza jednog do drugog okvira.

```
cap = cv2.VideoCapture(0)

while True:
    success, img = cap.read()

    cv2.imshow("Image", img)
    key = cv2.waitKey(1)
```

Slika 4.2 Inicijalizacija kamere i prikaz slike

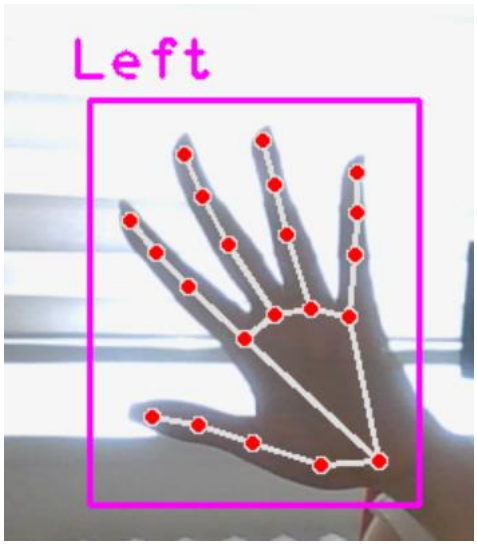
Zatim se želi postići da programski kod prepozna šaku te ključne točke na šaci i to se radi uz pomoć gore uvedenog detektora koji inicijalizira detektor ruku s maksimalno jednom rukom koju može detektirati. Detektor pronalazi šaku u zadanom okviru web kamere. Postavljamo da prepozna samo jednu ruku obzirom da ćemo koristiti jednoručnu abecedu znakovnog jezika. Također je navedeno prikazuje li se lijeva ili desna ruka.

```
detector = HandDetector(maxHands=1)

while True:
    success, img = cap.read()
    hands, img = detector.findHands(img)

    cv2.imshow("Image", img)
    key = cv2.waitKey(1)
```

Slika 4.3 Prikaz ključnih točaka šake



Slika 4.4 Prikaz ključnih točaka šake sa web kamere

Idući korak je izrezivanje područja šake kako bi se samo šaka zajedno s njenim točkama vidjela u zasebnom okviru. Prvo se detektira nalazi li se šaka u okviru te ako se nalazi stvara se okvir oko nje kako bi se samo ona prikazala. Postavlja se željena veličina okvira sa zadanim odstupanjem kako bi se stvorilo više prostora prikazanog oko šake jer inače okvir reže sliku točno na ključnim gornjim i donjim točkama pa se ne dobije cjelovita slika. Veličina okvira je zapravo matrica kojoj se zadaje početna i krajnja visina i širina.

```
offset = 20
```

Slika 4.5 Zadavanje odstupanja

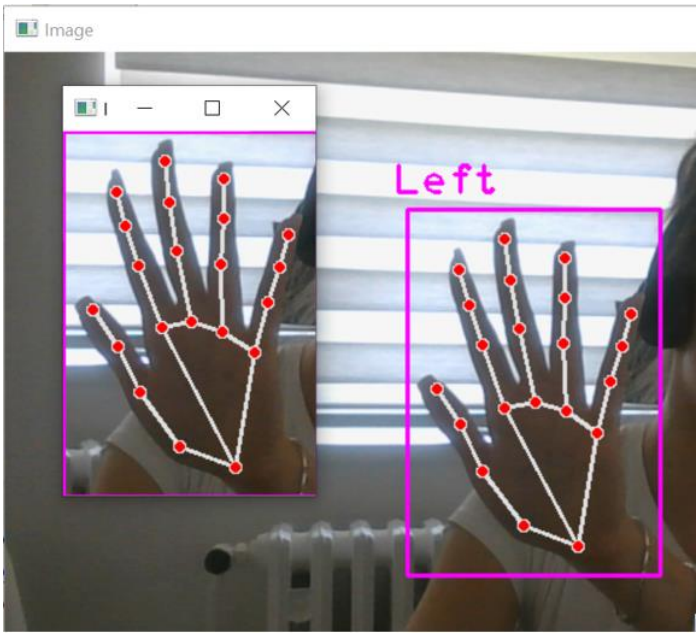
```
if hands:
    hand = hands[0]
    x, y, w, h = hand['bbox']

    imgCrop = img[y - offset:y + h + offset, x - offset:x + w + offset]
```

Slika 4.6 Pronalaženje šake te postavljanje okvira zadanih dimenzija oko nje

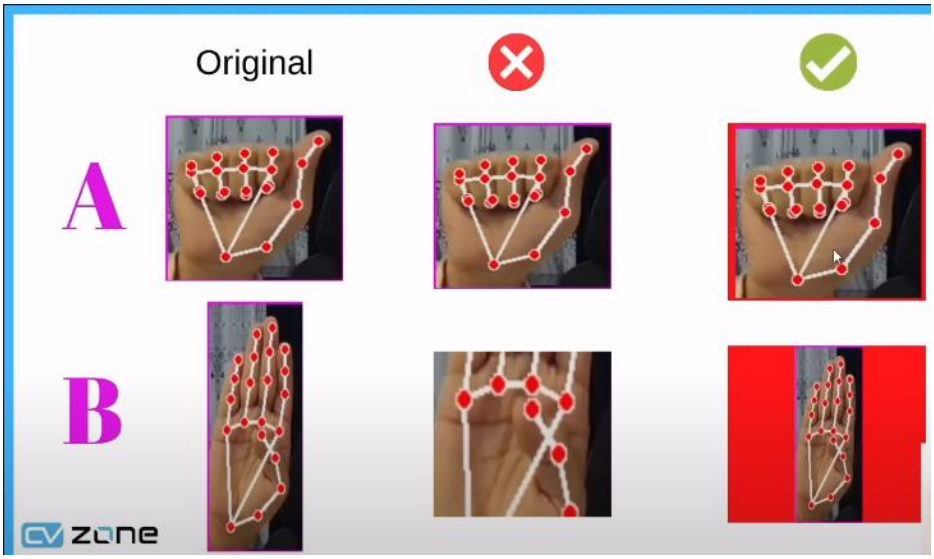
```
cv2.imshow("ImageCrop", imgCrop)
```

Slika 4.7 Linija koda za prikaz zadanog okvira na web kameri



Slika 4.8 Prikaz zadanog okvira na web kameri

Ovdje je prisutan jedan problem, svaki položaj ruke je drugačiji te se onda mijenja širina ili visina prikazane poze. Kada se šalju podaci klasifikatoru mora se slati taj izrezani okvir, a s obzirom da za klasifikator mora biti kvadratni okvir, izreže se dio šake za određene položaje što će onemogućiti točno prepoznavanje slova. Budući da sve slike moraju biti iste veličine, dodaje se fiskni okvir veličine 300 x 300 piksela na čiju sredinu će se onda postaviti prikaz šake bilo koje veličine. Ova problematika prikazana je na slici. [22]



Slika 4.9 Prikaz opisane problematike sa veličinom okvira [22]

Dakle, uvodi se nova ciljanua veličina slike koja će biti matrica veličine 300 x 300 piksela i bijele boje,odnosno na ekranu će se prikazati prozor bijele boje konstantne veličine.

```
imgSize = 300

imgWhite = np.ones((imgSize, imgSize, 3), np.uint8) * 255

cv2.imshow("ImageWhite", imgWhite)
```

Slika 4.10 Uvođenje varijable za bijeli prozor

Prozor koji prikazuje šaku sada se postavlja na bijeli prozor kako bi se dobile sve slike jednake veličine. Ako se samo postavi jedan prozor na drugi, ostaje puno neiskorištenog bijelog prostora koji će otežati klasifikaciju pa je cilj postići da se šaka nalazi na sredini bijelog prozora. Također, za poziciju šake gdje je visina dosta veća od širine i obrnuto, potrebno je izjednačiti te vrijednosti kako bi se što više iskoristio prostor. Linije koda za ovaj korak prikazane su u nastavku.

```
imgCropShape = imgCrop.shape
aspectRatio = h / w
if aspectRatio > 1:
    k = imgSize / h
    wCal = math.ceil(k * w)
    imgResize = cv2.resize(imgCrop, (wCal, imgSize))
    imgResizeShape = imgResize.shape
    wGap = math.ceil((imgSize - wCal) / 2)
    imgWhite[:, wGap:wCal + wGap] = imgResize
else:
    k = imgSize / w
    hCal = math.ceil(k * h)
    imgResize = cv2.resize(imgCrop, (imgSize, hCal))
    imgResizeShape = imgResize.shape
    hGap = math.ceil((imgSize - hCal) / 2)
    imgWhite[hGap:hCal + hGap, :] = imgResize
```

Slika 4.11 Postavljanje prikaza šake na središte bijelog prozora

Prvo se uvodi varijabla aspectRatio koja predstavlja omjer visine i širine prozora na kojem se prikazuje šaka. Zatim se postavlja uvjet: ako je visina veća od širine, visina se postavlja na konstantnu vrijednost 300 piksela, a širina se računa. Uvodi se konstanta k pomoću koje se onda računa širina tako da se množi konstanta sa trenutnom širinom prozora koji prikazuje šaku i zadana matematička funkcija taj broj uvijek zaokružuje na veću vrijednost. Zatim se pomoću varijable imgResize uzima slika imgCrop koja predstavlja prozor na kojem je prikazana šaka te se postavlja na dimenzije za izračunatu širinu te konstantnu visinu koja je veličine 300 piksela. Obzirom da se ta slika sada nalazi u gornjem lijevom kutu bijelog prozora, potrebno ju je postaviti na sredinu radi bolje klasifikacije. Uvodi se varijabla wGap koja će predstavljati prazan prostor prije i nakon slike koja treba biti na sredini. Dakle, taj prostor se računa tako da se od veličine bijelog prozora 300 piksela oduzme izračunata širina i podijeli se sa 2 kako bi se s obje strane dobio jednak prazan prostor. Također se koristi matematička funkcija za zaokruživanje na veću vrijednost. Zadnje što preostaje je postaviti sliku imgResize na imgWhite te definiranje početnih i krajnjih vrijednosti. Visina ostaje 300 piksela,a širina započinje zadanom prazninom, a završava izračunatom širinom i onda opet prazninom. Time se poboljšao omjer visine i širine te se postavio prikaz šake na središte bijelog prozora.

Isti taj proces se ponavlja za slike kod kojih je širina veća od visine.

Zadnje što preostaje je snimiti slike za svako slovo i spremiti ih u zasebne mape.

```
folder = "data/A"
counter = 0
```

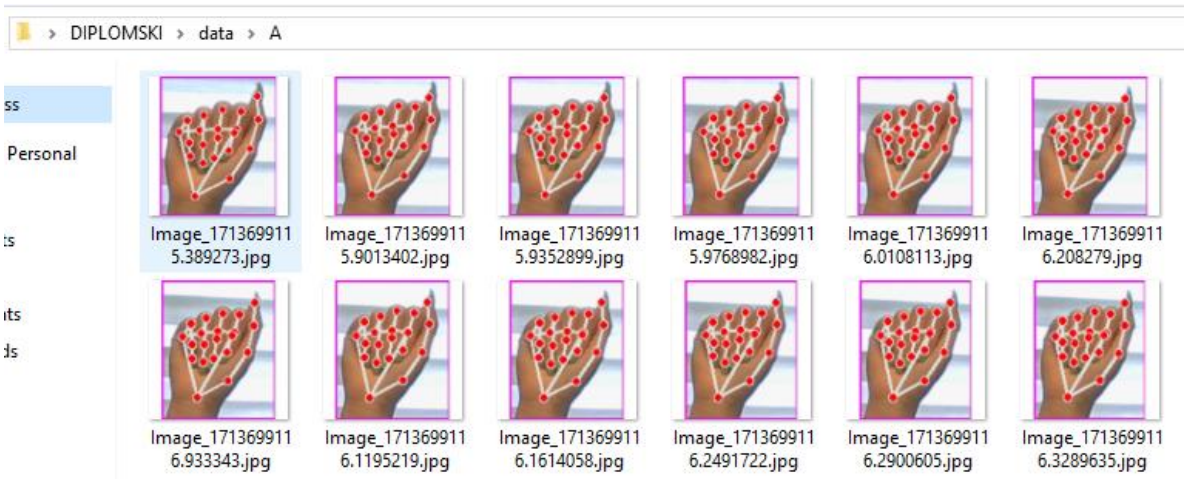
Slika 4.12 Uvođenje varijable datotke i postavljanje brojača

Uvodi se varijabla pod nazivom folder te se u njoj definira gdje će se slike spremati, i to se onda mijenja za svako slovo abecede. Brojač se postavlja na nulu te se on povećava s obzirom na broj spremljenih slika.

```
key = cv2.waitKey(1)
if key == ord("s"):
    counter += 1
    cv2.imwrite(f'{folder}/Image_{time.time()}.jpg',imgWhite)
    print(counter)
```

Slika 4.13 Prikaz linija koda za spremanje slika u datoteke

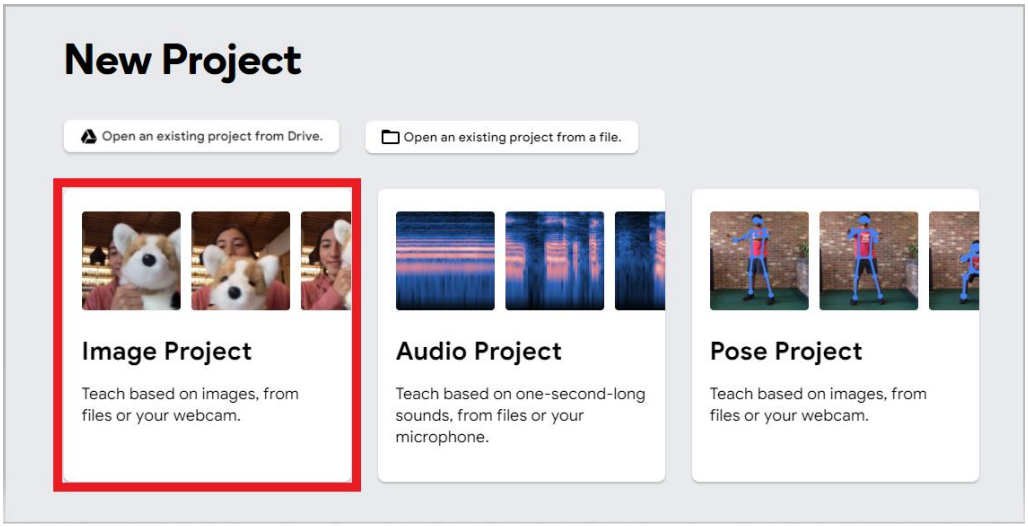
Obzirom da nije cilj da se slike cijelo vrijeme snimaju, namješta se da se krenu snimati tek kad korisnik pritisne tipku 's' na tipkovnici. Za svaku sliku se povećava broj na brojaču te će se prikazivati koliko je slika snimljeno. Postupak se ponavlja za svako slovo abecede, samo treba mijenjati ime mape u koju se sprema. Konačni rezultat pikazan je na slici. Za svako slovo snimljeno je 500 slika.



Slika 4.14 Prikaz snimljenih slika za slovo A

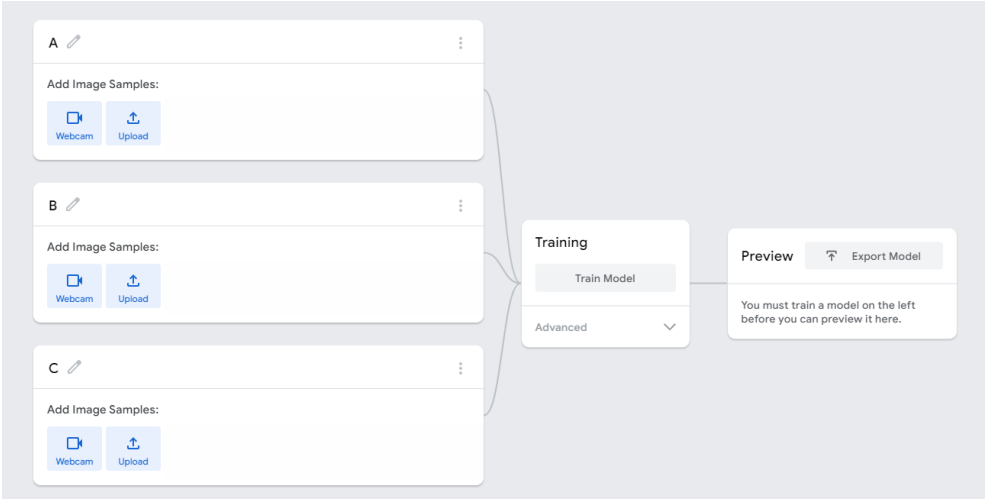
4.2. Treniranje modela

Idući korak je treniranje modela za koje će se koristiti Teachable Machine. Ulazi se u novi projekt te se odabire opcija za treniranje uz pomoć slika.



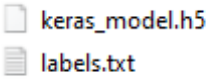
Slika 4.15 Odabir projekta unutar Teachable Machine-a [23]

Zatim se imenuje svaka klasa po slovima abecede te se za svako slovo dodaju odgovarajuće slike koje su prethodno snimljene. Na slici je prikazano kako taj dio izgleda.



Slika 4.16 Prikaz dodavanja klasa i njihovih odgovarajućih slika [23]

Nakon što se model istrenira se kao konačan rezultat dobije mapa sa istreniranim modelom te tekstualna datoteka sa labelama za slova. Model se konvertira u keras model koji je jedan od najčešće korištenih API-ja za neuronske mreže visoke razine kada je u pitanju razvoj i testiranje neuronskih mreža. [14] Te dvije datoteke se stavljaju unutar projekta u mapu pod nazivom 'Model'.



Slika 4.17 Prikaz dobivenog rezultata treniranja modela

4.3. Prepoznavanje abecede znakovnog jezika

Idući korak je testiranje modela da se provjeri hoće li uspjeti prepoznati slova znakovnog jezika. Potrebno je iskopirati kod koji se koristio za prikupljanje podataka te ga malo izmjeniti. Uvodi se modul za klasifikaciju slika te varijabla za klasifikator i određuju se parametri. Prvi parametar je istrenirani model,a drugi su oznake.

```
from cvzone.ClassificationModule import Classifier

classifier = Classifier("Model/keras_model.h5", "Model/labels.txt")
```

Slika 4.18 Uvođenje klasifikatora

Zatim je cilj dobiti predviđanje te indeks slova abecede koje klasifikator predviđa da se pojavljuje na temelju istreniranog modela. Dodaje se varijabla 'labels' u kojoj se nalaze imena klasa, tako da ako klasifikator npr. predvidi indeks 0, ispisat će se da je to slovo A i tako za ostala slova, odnosno postoji lista oznaka koje odgovaraju gestama koje se klasificiraju.

```
labels = ["A", "B", "C", "D", "E", "F", "G", "H", "I",

prediction, index = classifier.getPrediction(imgWhite, draw=False)
print(prediction, index)
```

Slika 4.19 Ispis predviđenog slova i njegovog indeksa

Obzirom da se na konačnom prozoru ne moraju prikazivati nužno sve točke šake, već samo okvir oko ruke i predviđeno slovo, mora se napraviti kopija originalnog prozora, jer je cilj postići da se na 'imgWhite' vide sve točke jer su tamo potrebne za klasifikaciju, a na konačnom prozoru kojeg vidi korisnik to nije potrebno prikazati. Zato se uvodi varijabla 'imgOutput' koja će biti kopija originalnog prozora samo bez prikaza tih točaka. Zatim se na 'imgOutput' stavlja željeni tekst koji je sadržan u varijabli 'labels', zadaje se ishodište i željeni font kojim će tekst biti ispisan, omjer, boja i debljina teksta. Dakle, na izlaznoj slici (imgOutput) iscrtavaju se pravokutnici oko ruke i područja za oznaku geste te se ispisuje oznaka geste pored ruke.

```
imgOutput = img.copy()

cv2.putText(imgOutput, labels[index], (x, y -26), cv2.FONT_HERSHEY_COMPLEX, 1.7, (255, 255, 255), 2)
```

Slika 4.20 Kreiranje kopije prozora te zadavanje željenog izgleda teksta

```
cv2.rectangle(imgOutput, (x-offset, y-offset),
              (x + w+offset, y + h+offset), (255, 0, 255), 4)
```

Slika 4.21 Postavljanje okvira oko šake

Da bi se prikazana slova bolje vidjela, zadaje se još ispunjeni okvir oko slova željene boje i debljine i na željenom položaju.

```
cv2.rectangle(imgOutput, (x - offset, y - offset-50),
              (x - offset+90, y - offset-50+50), (255, 0, 255), cv2.FILLED)
```

Slika 4.22 Postavljanje okvira oko slova

Na kraju se namješta da se prikaže konačni prozor, odnosno 'imgOutput'.

```
cv2.imshow("Image", imgOutput)
cv2.waitKey(1)
```

Slika 4.23 Prikaz konačnog prozora

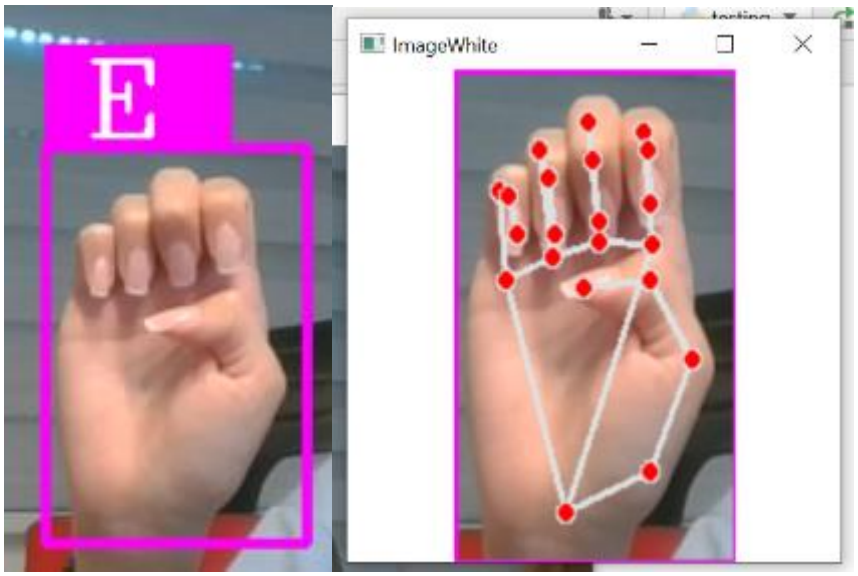
5. EVALUACIJA MODELA

Model koji je opisan u prethodnom poglavlju je testiran sa različitim modelima te različitim osvjetljenjem.

5.1. Prvi primjer – isti model,različito osvjetljenje

Prvo će se isprobati testirati model sa vlastitom šakom sa kojom je i treniran model,no testirat će se na sobnom osvjetljenju,dnevnom osvjetljenju i najgori slučaj kad se osvjetljenje nalazi iza šake te je time šaka puno lošije osvjetljena.

Kod sobnog osvjetljenja se vidi da program bez problema prepoznaje slovo E, te se to također može vidjeti na donjoj slici gdje je najveća vjerojatnost konstantno bila za slovo E.



Slika 5.1 Slučaj sa sobnim osvjetljenjem

[3.8980726e-08, 1.5384514e-05, 1.15114396e-07, 5.3747478e-09, 0.99962175, 0.0003601795, 3.8138957e-11, 8.3617413e-10, 1.835979e-07, 7.629099e-16, 1.8206127e-09, 4.5153898e-11, 3.0543266e-08, 1.3019215e-07, 1.0537514e-14, 8.6967145e-09, 1.26524425e-11, 1.135575e-08, 2.1755382e-06, 7.103372e-17, 3.4228282e-13, 8.914996e-17, 1.0033094e-09, 4.5506918e-11] 4

[1.4227031e-07, 1.2678499e-05, 1.7867889e-07, 8.666072e-09, 0.9999087, 7.4029485e-05, 2.0897575e-10, 8.580382e-09, 5.0758945e-07, 3.0627346e-16, 1.5764069e-09, 8.8094206e-11, 3.1153533e-07, 3.893055e-07, 1.3329885e-13, 1.4646903e-07, 1.2556729e-10, 1.7399567e-07, 2.630883e-06, 3.081494e-14, 1.8610577e-12, 1.8111731e-16, 2.1591773e-09, 6.166492e-11] 4

[2.1646918e-06, 0.0005708895, 1.0772367e-06, 1.580691e-08, 0.99921954, 0.00015537853, 3.6701426e-09, 1.4938082e-07, 1.6006557e-06, 7.382645e-14, 2.5499466e-08, 1.8812771e-09, 4.892278e-06, 4.693869e-06, 5.445974e-12, 7.4478125e-06, 3.0178157e-08, 8.840052e-06, 2.3289303e-05, 2.124327e-12, 4.0578014e-12, 1.0335297e-14, 2.5636744e-08, 9.830146e-10] 4

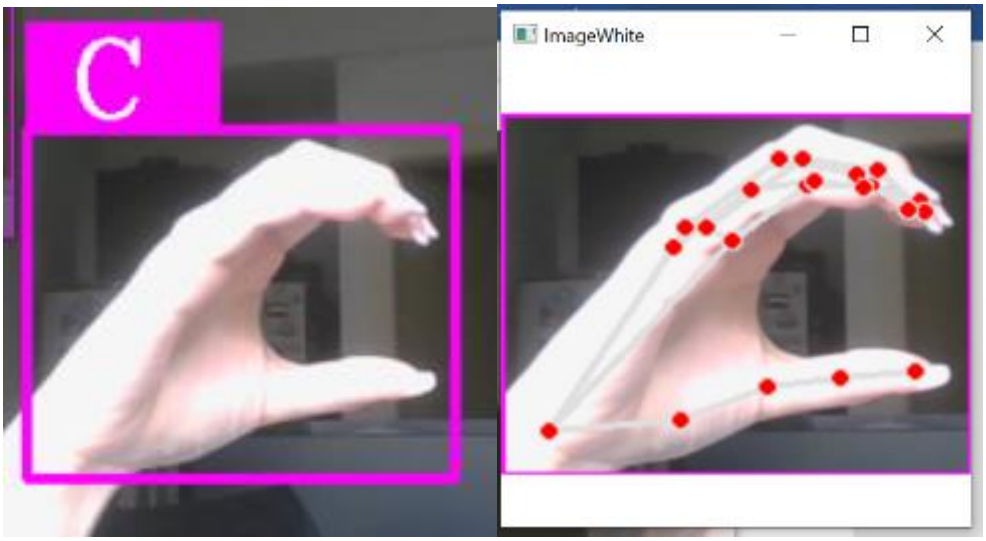
[3.1332706e-07, 0.0005428036, 5.0010584e-07, 1.6356269e-09, 0.99929607, 0.00014808141, 3.587346e-10, 1.16324834e-07, 4.1502796e-07, 2.9017485e-14, 1.907507e-08, 9.761994e-09, 7.651116e-06, 4.6631263e-07, 6.5252773e-13, 5.805037e-07, 8.484513e-11, 2.1049864e-06, 9.112265e-07, 3.9731287e-12, 2.1431992e-12, 1.1879124e-14, 1.0553903e-08, 7.674309e-10] 4

[8.2474095e-08, 1.7266579e-05, 1.0602434e-07, 2.2221027e-08, 0.9998702, 0.000105005965, 1.1815879e-10, 3.1755427e-09, 1.6307424e-06, 1.2664737e-15, 1.026713e-09, 2.1782557e-11, 6.2878695e-07, 5.884669e-07, 4.8290517e-14, 6.490924e-08, 2.346943e-10, 6.53364e-08, 4.3569776e-06, 2.2850615e-15, 2.5405668e-13, 5.564348e-17, 3.2522314e-09, 2.187315e-11] 4

[2.5307386e-07, 7.2659626e-05, 7.808367e-08, 2.1967413e-08, 0.9999021, 2.0676516e-05, 1.01303514e-10, 8.4807175e-09, 1.9332228e-06, 2.9555487e-15, 2.1966062e-09, 4.3127592e-11, 3.076954e-07, 6.853213e-07, 9.713045e-14, 4.4208483e-07, 3.0394462e-10, 3.9247305e-07, 3.4149357e-07, 9.08171e-14, 7.1723196e-12, 1.5326225e-15, 1.8298837e-09, 5.9000624e-11] 4

Slika 5.2 Prikaz vjerojatnosti za svako slovo te indeks predviđenog slova – sobno osvjetljenje

Drugi slučaj je pri dnevnom osvjetljenju te se može zaključiti da program također dobro radi i prepoznaje slovo C. To se također može vidjeti po vjerojatnostima koje su bile najveće za indeks 2 odnosno slovo C.



Slika 5.3 Slučaj sa dnevnim osvjetljenjem

[3.2079645e-13, 1.7003392e-08, 0.9999938, 1.3103928e-06, 7.7402734e-07, 8.761139e-11, 3.080331e-07, 1.4741039e-10, 2.0480076e-14, 2.4522641e-16, 2.9300253e-07, 2.4642885e-14, 6.2057325e-11, 2.9021144e-06, 9.510484e-12, 8.266407e-10, 1.3309489e-11, 8.8328655e-08, 5.540913e-08, 2.6863672e-14, 7.212854e-17, 1.6522918e-17, 5.2441885e-07, 4.674566e-08] 2

[5.1804763e-13, 3.4256036e-08, 0.99996424, 1.0257678e-06, 2.2672099e-05, 2.7754427e-10, 9.1673463e-10, 1.341945e-10, 1.7772972e-13, 3.833207e-17, 7.234713e-06, 3.0127485e-14, 9.390957e-10, 1.1227212e-06, 3.3174464e-13, 3.1718521e-09, 2.0514443e-12, 4.6221593e-10, 2.0952698e-10, 1.7161574e-14, 5.3339252e-20, 2.0506264e-16, 3.7123207e-06, 1.1758613e-08] 2

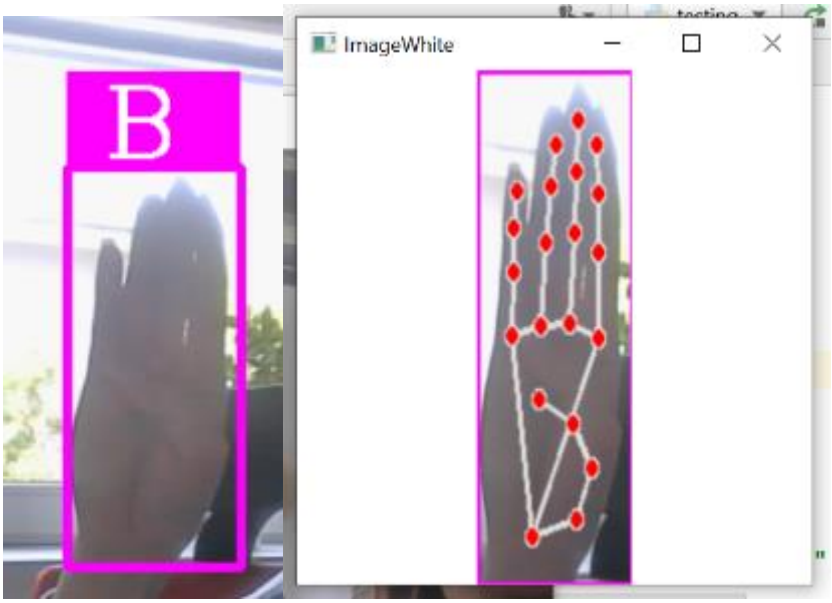
[4.1638032e-10, 7.574971e-07, 0.99691904, 1.3909059e-06, 0.0029022829, 1.1065826e-08, 3.918599e-08, 5.4985403e-09, 2.6720662e-10, 7.277079e-16, 8.811381e-05, 5.294058e-10, 1.6907252e-09, 1.4244276e-05, 6.723643e-12, 1.7470636e-07, 1.7484915e-13, 3.7279021e-06, 1.2708484e-09, 3.0075957e-12, 5.399662e-16, 2.3654505e-14, 6.972996e-05, 6.0873225e-07] 2

[6.323882e-09, 1.2398115e-06, 0.9998325, 1.7794741e-06, 5.520409e-05, 1.2560389e-07, 1.3461357e-07, 1.5514449e-07, 5.051054e-11, 6.5583094e-17, 9.493852e-05, 5.122513e-10, 5.2550536e-10, 5.8796595e-06, 2.3030158e-11, 2.364037e-06, 4.1217363e-13, 1.9157178e-06, 7.1903767e-09, 7.296552e-12, 3.981554e-15, 1.6627919e-14, 6.498065e-07, 3.0945712e-06] 2

[7.840217e-12, 1.8151563e-07, 0.999977, 8.269545e-07, 1.8558956e-05, 5.9474123e-09, 7.7444923e-07, 4.755337e-10, 2.530408e-12, 1.0182211e-18, 5.890666e-07, 1.1699828e-13, 1.1797205e-11, 1.9100653e-06, 5.6537623e-14, 5.3000565e-10, 1.1449163e-12, 5.2196913e-09, 1.430104e-09, 2.1330145e-14, 7.7348026e-16, 5.1895118e-17, 1.7450944e-07, 9.11817e-09] 2

Slika 5.4 Prikaz vjerojatnosti za svako slovo te indeks predviđenog slova – dnevno osvjetljenje

Kod slučaja sa najgorim osvjetljenjem je program uspio prepoznati slovo B, no prema donjim podacima se vidi da je na trenutke pretpostavljao slovo D koje predstavlja indeks 3, pa je zaključak da je s lošim osvjetljenjem programu teže prepoznati o kojem je slovu riječ. Također su na donjoj slici prikazane vjerojatnosti koje program vraća te se po tome vidi da je većinom najveća vjerojatnost bila za slovo B, a ponekad za D.



Slika 5.5 Slučaj sa lošim osvjetljenjem

[4.4550726e-13, 0.20465189, 1.2447917e-08, 0.7953444, 1.0761952e-06, 6.504238e-09, 5.514082e-11, 1.25567785e-11, 1.8454098e-06, 1.0860557e-12, 5.620009e-10, 3.452669e-16, 1.1662132e-11, 6.025783e-07, 7.897396e-15, 3.2159047e-08, 1.5291069e-07, 1.4440042e-10, 1.7185094e-11, 3.0826115e-08, 2.9767763e-10, 1.7966205e-16, 6.133744e-08, 9.238161e-13] 3

[1.9278223e-11, 0.7264026, 5.382119e-08, 0.27350208, 7.1251466e-06, 8.261164e-09, 7.365476e-10, 1.3425924e-09, 3.1268723e-06, 2.3522125e-14, 5.594039e-09, 2.0331705e-15, 1.4318382e-11, 1.5895188e-07, 1.2575055e-14, 1.2954817e-07, 3.0622388e-07, 3.8193151e-10, 1.8100882e-12, 8.440588e-05, 4.843722e-09, 4.66138e-16, 5.233218e-09, 9.2666924e-12] 1

[4.4704253e-12, 0.92745584, 1.78693e-08, 0.07247574, 6.352326e-08, 3.8013295e-09, 3.948332e-10, 3.0638056e-10, 1.668439e-07, 1.3840492e-12, 5.2766846e-10, 8.7107095e-17, 5.1015237e-11, 8.178171e-08, 4.8226265e-14, 5.5401642e-08, 1.203522e-05, 1.2910955e-11, 8.0151415e-12, 5.5974495e-05, 2.7320948e-09, 6.472191e-16, 2.9758318e-09, 1.4987943e-11] 1

[2.8052922e-12, 0.98833513, 1.4643445e-07, 0.011658195, 1.6197512e-06, 5.4423546e-09, 1.2758615e-09, 5.223995e-10, 7.1326e-07, 2.621075e-13, 4.3873496e-09, 1.9543765e-15, 6.190618e-11, 2.5970111e-08, 4.0330264e-14, 1.1473242e-08, 3.6937033e-06, 1.0717873e-10, 4.3528492e-13, 4.862317e-07, 7.66683e-09, 1.6025145e-14, 4.393127e-09, 5.715629e-12] 1

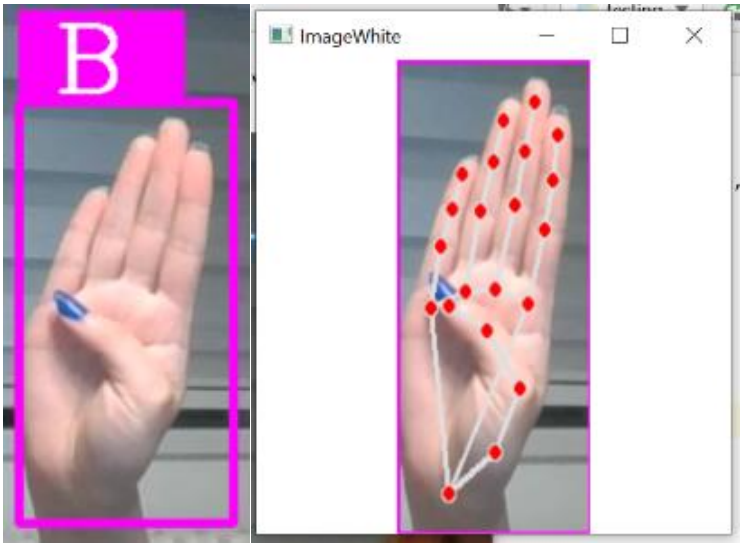
[7.417047e-13, 0.95768523, 1.7546178e-07, 0.042292625, 4.548006e-06, 1.8779043e-09, 7.1225323e-09, 3.349679e-10, 3.612861e-07, 1.61646e-13, 8.364987e-09, 9.771291e-16, 4.1441225e-12, 4.921191e-09, 4.286402e-15, 4.8640336e-09, 5.5902416e-07, 6.385224e-11, 2.8085844e-14, 1.648877e-05, 3.7807887e-08, 3.6016884e-14, 4.6186153e-09, 3.867355e-12] 1

[2.370681e-12, 0.6427275, 1.0279383e-07, 0.35726225, 1.9863437e-06, 1.7139167e-08, 8.3608326e-10, 2.4746555e-10, 3.6120403e-07, 1.2279147e-12, 4.1505115e-09, 1.524372e-15, 8.427175e-11, 1.7701977e-07, 7.5609203e-14, 4.5740464e-08, 7.133595e-06, 3.383871e-11, 6.071797e-12, 4.3115298e-07, 3.193296e-08, 1.6025521e-14, 1.7649223e-08, 6.9199533e-12] 1

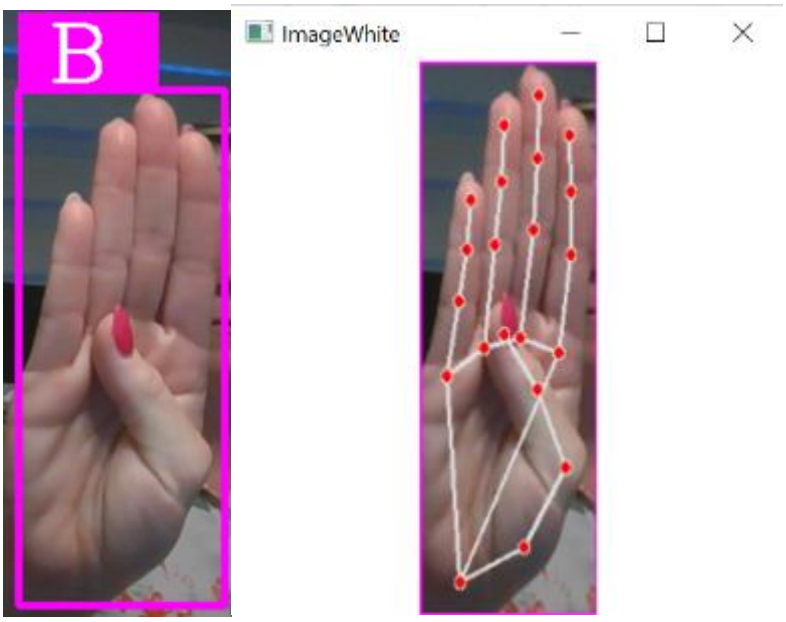
Slika 5.6 Prikaz vjerojatnosti za svako slovo te indeks predviđenog slova – loše osvjetljenje

5.2. Drugi primjer – različiti modeli,isto osvjetljenje

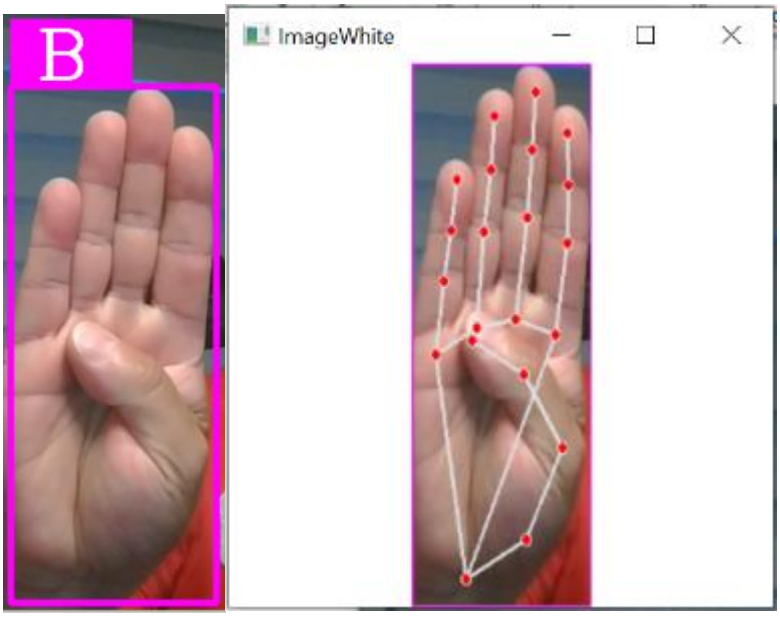
Ovdje će se provjeriti radi li model sa različitim modelima ruku. Koristit će se tri različita modela ruku na istom osvjetljenju.



Slika 5.7 Testiranje modela na različitim rukama – prvi model ruke



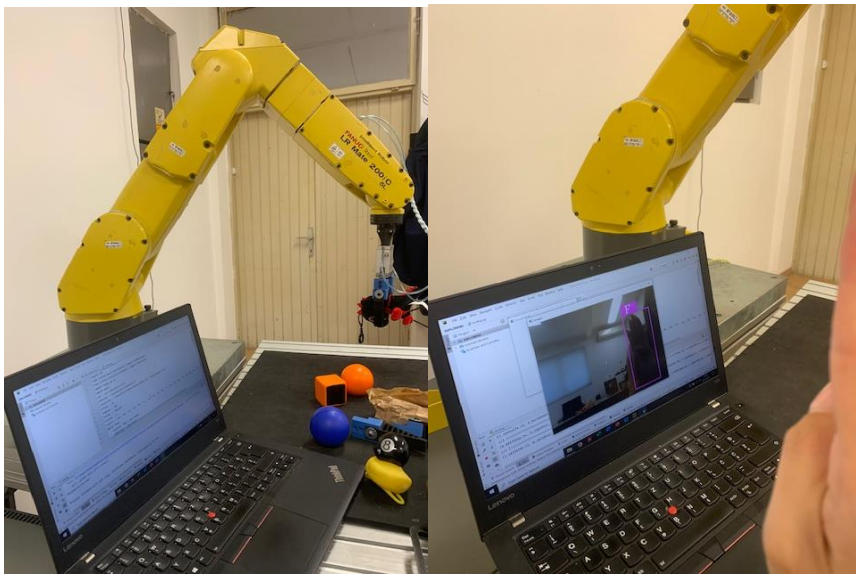
Slika 5.8 Testiranje modela na različitim rukama – drugi model ruke



Slika 5.9 Testiranje modela na različitim rukama – treći model ruke

Može se zaključiti da program dobro radi i prepoznaje slova i na različitim rukama.

Evaluacija modela je također napravljena još jednom u Laboratoriju za projektiranje izradbenih i montažnih sustava zajedno s mentorom i voditeljem laboratorija Dr.sc. Tomislavom Stipančićem, dipl. ing. Evaluacija se pokazala uspješnom te se slike iz laboratorija nalaze u nastavku.



Slika 5.10 Evaluacija u Laboratoriju za projektiranje izradbenih i montažnih sustava

6. KRITIČKI OSVRT NA RAD MODELA

Rezultatima iz prethodnog poglavlja se može zaključiti da model ispravno radi. U dnevnim i dobro osvjetljenim uvjetima prepoznaje sve točke šake te dobro prepoznaje koje slovo se prikazuje. Uviđeno je da pri lošem osvjetljenju model malo lošije radi,odnosno ponekad pomješa o kojem se slovu radi. Zato je bitno da prostorija bude dobro osvjetljena kako bi dobili točne rezultate. Međutim, uočeno je da model malo teže raspoznaje slova koja imaju jako sličnu pozu šake kao što su na primjer slova S,M i T gdje bi bilo potrebno prepoznati nalazi li se palac iza ili ispred prstiju što model ne može prepoznati pa nekad kao rezultat daje krivo slovo. To je zato što ne prepoznaje dubinu slike odnosno položaj palca iza ili ispred prstiju, no i to se određenim poboljšanjima može riješiti. Jedno od najpoznatijih dostignuća za prepoznavanje ljudske poze je pomoću Microsoft Kinect senzora. [16] Koristi se posebna vrsta značajki, izračunatih izravno na dubinskim slikama, za klasifikaciju dijelova ljudskog tijela. Iste značajke su nedavno primijenjene za problem procjene položaja ruke i klasifikaciju oblika. Također se mogu koristiti dubinske kamere koje su postale sve popularnije u posljednjih nekoliko godina. Dok su dubinske kamere vrlo atraktivne jer trenutačno daju 3D informacije, one se temelje na aktivnom osvjetljenju. [17] Izravna posljedica ovih tehnologija aktivnog očitavanja je da su 3D kamere ograničenog dometa i imaju nisku razlučivost. [15] Još jedan nedostatak modela je što ne može prepoznati slova koja uključuju pokret jer bi nam za to bili potrebni senzori koji prepoznaju pokret. Dva su pristupa naširoko prihvaćena u prepoznavanju znakovnog jezika: na temelju vida i na senzorima. Pristup temeljen na vidu koristi RGB kameru i senzor dubine i primjenjuje algoritme računalnog vida za analizu gesta ruku i izraza tijela i lica sa slika za prepoznavanje znakovnog jezika. S druge strane, pristup temeljen na senzorima izvodi obrasce kretanja prstiju i šake (kretanje, položaj i brzina) iz više senzora koji su pričvršćeni na ruke ili tijelo korisnika.

To su dvije vrste unaprjeđenja ovog modela kako bi mogao još bolje raditi. Također se može napraviti i sličan model za objeručnu abecedu znakovnog jezika.

7. ZAKLJUČAK

U ovom radu je pokazan način izrade modela za prepoznavanje abecede znakovnog jezika. U prvom dijelu je pokazana sva teorijska osnova potrebna za razumijevanje i izradu ovakvog modela. Za izradu modela korišten je Python te brojne biblioteke koje su podržane u Pythonu te je objašnjena njihova uloga. Prvi korak je bio sakupljanje podataka za treniranje modela što uključuje izradu slika karakterističnih za svaku pozu te njihova klasifikacija. Zatim se model trenirao koristeći Teachable Machine te na kraju testirao. Iz biblioteke Media Pipe su učitane karakteristične točke šake koje su ključne za prepoznavanje različitih poza karakterističnih za određeno slovo.

Model je testiran na različitim primjerima osvjetljenja te na različitim modelima kako bi se uvidjela njegova točnost. Donesen je zaključak da model dobro radi pri dobrom osvjetljenju, no da postoje načini kojima možemo poboljšati i unaprijediti model. Za unaprjeđenje se mogu koristiti tehnologije poput Microsoft Kinetic senzora te dubinske kamere za bolje određivanje poze te za prepoznavanje slova ili riječi znakovnog jezika koji uključuju pokret.

LITERATURA

[1] The Fascinating History of Sign Language, https://www.academyhearing.ca/blog/news/News/2016/11/16/50:the-fascinating-history-of-si%20gn-language, dostupno dana 15. svibnja 2024.

[2] The Impact of AI on Sign Language Interpreting- How Technology is Changing the Game for Deaf Communication, https://www.unspokenasl.com/aslblogs/the-impact-of-ai-on-sign-language-interpreting-how-technology-is-changing-the-game-for-deaf-communication/, dostupno dana 15.svibnja 2024.

[3] What is Machine Learning (ML)?, https://www.ibm.com/topics/machine-learning, dostupno dana 15.svibnja 2024.

[4] B.D. Bašić, M. Čupić, J. Šnajder, "Umjetne neuronske mreže", Zagreb: Fakultet elektrotehnike i računarstva, 2008.

[5] What Is Deep Learning? Definition, Examples, and Careers, https://www.coursera.org/articles/what-is-deep-learning, dostupno dana 15.svibnja 2024.

[6] Što je umjetna inteligencija, https://www.umjetnainteligencijaui.com/sto-je-umjetna-inteligencija/, dostupno dana 15.svibnja 2024.

[7] What are convolutional neural networks?, https://www.ibm.com/topics/convolutional-neural-networks, dostupno dana 16.svibnja 2024.

[8] I. Mihajlović, "Everything You Ever Wanted To Know About Computer Vision.", Towards Data Science, 2019.

[9] What is computer vision?, https://www.ibm.com/topics/computer-vision, dostupno dana 17.svibnja 2024.

[10] What is Python? Executive Summary, https://www.python.org/doc/essays/blurb/, dostupno dana 17.svibnja 2024.

[11] About OpenCV, https://opencv.org/about/, dostupno dana 17. svibnja 2024.

[12] Object Detection vs Object Recognition vs Image Segmentation, https://www.geeksforgeeks.org/object-detection-vs-object-recognition-vs-image-segmentation/, dostupno dana 19.svibnja 2024.

[13] MediaPipe Hands, https://mediapipe.readthedocs.io/en/latest/solutions/hands.html, dostupno dana 20.svibnja 2024.

[14] Remi M., "What is a Keras model and how to use it to make predictions", ActiveState, 2021.

[15] A.Kuznetsova, L.Leal-Taixe, B.Rosenhahn, "Real-time sign language recognition using a consumer depth camera", ´Institute fuer Informationsverarbeitung, Leibniz University Hannover Appelstr. 9A, Hannover, 30167, Germany

[16] B.G.Lee, T.W.Chong, W.Z.Chung, "Sensor Fusion of Motion-Based Sign Language Interpretation with Deep Learning", PubMed Central, 2020.

[17] S.Lee, "Consumer Depth Cameras and Applications", ResearchGate, 2013.

[18] A.Chen, A.Pitaru, B.Webster, I.Alvarado, J.Griffith, K.Phillips, M.Carney, N. Howell "Teachable Machine: Approachable Web-Based Tool for Exploring Machine Learning Classification" , 2020.

[19] Osnove neuronskih mreža, https://course.elementsofai.com/hr/5/1, dostupno dana 20. svibnja 2024.

[20] D.Radečić, "Softmax Activation Function Explained", Towards Data Science, 2020.

[21] MediaPipe: Google's Open Source Framework (2024 Guide), https://viso.ai/computer-vision/mediapipe/#:~:text=for%20Enterprise%20Teams-,What%20is%20MediaPipe%3F,currently%20in%20alpha%20at%20v0., dostupno dana 25.svibnja 2024.

[22] https://www.youtube.com/watch?v=wa2ARoUUdU8&t=3003s , dostupno dana 15.svibnja 2024.

[23] https://teachablemachine.withgoogle.com/

PRILOZI

I. Python kod – Prikupljanje podataka

II. Python kod – Prepoznavanje abecede znakovnog jezika

I. Python kod – Prikupljanje podataka

```
import cv2
from cvzone.HandTrackingModule import HandDetector
import numpy as np
import math
import time
cap = cv2.VideoCapture(0)
detector = HandDetector(maxHands=1)
offset = 20
imgSize = 300
folder = "data/W"
counter = 0
while True:
    success, img = cap.read()
    hands, img = detector.findHands(img)
    if hands:
        hand = hands[0]
        x, y, w, h = hand['bbox']
        imgWhite = np.ones((imgSize, imgSize, 3), np.uint8) * 255
        imgCrop = img[y - offset:y + h + offset, x - offset:x + w + offset]
        imgCropShape = imgCrop.shape
        aspectRatio = h / w
        if aspectRatio > 1:
            k = imgSize / h
            wCal = math.ceil(k * w)
            imgResize = cv2.resize(imgCrop, (wCal, imgSize))
            imgResizeShape = imgResize.shape
            wGap = math.ceil((imgSize - wCal) / 2)
            imgWhite[:, wGap:wCal + wGap] = imgResize
        else:
            k = imgSize / w
            hCal = math.ceil(k * h)
            imgResize = cv2.resize(imgCrop, (imgSize, hCal))
            imgResizeShape = imgResize.shape
            hGap = math.ceil((imgSize - hCal) / 2)
            imgWhite[hGap:hCal + hGap, :] = imgResize
        cv2.imshow("ImageCrop", imgCrop)
        cv2.imshow("ImageWhite", imgWhite)
    cv2.imshow("Image", img)
    key = cv2.waitKey(1)
    if key == ord("s"):
        counter += 1
        cv2.imwrite(f'{folder}/Image_{time.time()}.jpg',imgWhite)
        print(counter)
```

II. Python kod – Prepoznavanje abecede znakovnog jezika

```
import cv2
from cvzone.HandTrackingModule import HandDetector
from cvzone.ClassificationModule import Classifier
import numpy as np
import math
cap = cv2.VideoCapture(0)
detector = HandDetector(maxHands=1)
classifier = Classifier("Model/keras_model.h5", "Model/labels.txt")
offset = 20
imgSize = 300
folder = "Data/C"
counter = 0
labels = ["A", "B", "C", "D", "E", "F", "G", "H", "I", "K", "L", "M", "N",
"O", "P", "Q", "R", "S", "T", "U", "V", "W", "X", "Y" ]
while True:
    success, img = cap.read()
    imgOutput = img.copy()
    hands, img = detector.findHands(img)
    if hands:
        hand = hands[0]
        x, y, w, h = hand['bbox']
        imgWhite = np.ones((imgSize, imgSize, 3), np.uint8) * 255
        imgCrop = img[y - offset:y + h + offset, x - offset:x + w + offset]
        imgCropShape = imgCrop.shape
        aspectRatio = h / w
        if aspectRatio > 1:
            k = imgSize / h
            wCal = math.ceil(k * w)
            imgResize = cv2.resize(imgCrop, (wCal, imgSize))
            imgResizeShape = imgResize.shape
            wGap = math.ceil((imgSize - wCal) / 2)
            imgWhite[:, wGap:wCal + wGap] = imgResize
            prediction, index = classifier.getPrediction(imgWhite,
draw=False)
            print(prediction, index)
        else:
            k = imgSize / w
            hCal = math.ceil(k * h)
            imgResize = cv2.resize(imgCrop, (imgSize, hCal))
            imgResizeShape = imgResize.shape
            hGap = math.ceil((imgSize - hCal) / 2)
            imgWhite[hGap:hCal + hGap, :] = imgResize
            prediction, index = classifier.getPrediction(imgWhite,
draw=False)
        cv2.rectangle(imgOutput, (x - offset, y - offset-50),
                      (x - offset+90, y - offset-50+50), (255, 0, 255),
cv2.FILLED)
        cv2.putText(imgOutput, labels[index], (x, y -26),
cv2.FONT_HERSHEY_COMPLEX, 1.7, (255, 255, 255), 2)
        cv2.rectangle(imgOutput, (x-offset, y-offset),
                      (x + w+offset, y + h+offset), (255, 0, 255), 4)
        cv2.imshow("ImageCrop", imgCrop)
        cv2.imshow("ImageWhite", imgWhite)
    cv2.imshow("Image", imgOutput)
    cv2.waitKey(1) [22]
```