Računalni model za klasifikaciju i brojanje sklekova

Njirić, Ana

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

Ana Njirić

Zagreb, 2023.

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

Mentor: Student:

izv. prof. dr. sc. Tomislav Stipančić, dipl. ing. Ana Njirić

Zagreb, 2023.

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

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

Također zahvaljujem svojoj obitelji i prijateljima koji su mi uvijek podrška.

Hvala Brunu i Maroju što su bili modeli i sudjelovali u izradi rada. Hvala Klaudii koja je bila kamerman za jedan od videa.

Ana Njirić

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 / 23 - 6 / 1**	
Ur.broj: **15 - 23 -**	

DIPLOMSKI ZADATAK

Student: **Ana Njirić** JMBAG: 0035216876

Naslov rada na hrvatskom jeziku: **Računalni model za klasifikaciju i brojanje sklekova**

Naslov rada na engleskom jeziku: **Computational model for classification and counting of push-ups**

Opis zadatka:

Računalni modeli temeljeni na algoritmima umjetne inteligencije omogućavaju povezivanje unutarnjeg svijeta računala s objektima, pojavama ili drugim fenomenima koji se pojavljuju u sklopu stvarne okoline. Između ostaloga, moguće je razviti računalni model za automatsko prebrojavanje sklekova kod osobe u treningu.
Takav se model potom može koristiti kod automatskog praćenja napretka osobe prilikom treninga te se može povezati u programsku aplikaciju koja bi mogla uključivati različite komponente za praćenje.

U radu je potrebno:

- proučiti MediaPipe Python programsku biblioteku,
- razviti programsku podršku koristeći Python programsku podršku za klasifikaciju poza, te
- razviti programsku podršku za brojanje ponavljajućih radnji prilikom izvođenja sklekova.

Model je potrebno eksperimentalno evaluirati koristeći vizijski sustav. U radu je potrebno navesti korištenu literaturu i eventualno dobivenu pomoć.

Zadatak zadan: 28. rujna 2023.

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

Datum predaje rada: 30. studenoga 2023.

Predviđeni datumi obrane: 4. – 8 . prosinca 2023.

Predsjednik Povjerenstva: Prof. dr. sc. Ivica Garašić

SADRŽAJ

POPIS SLIKA

POPIS TABLICA

POPIS OZNAKA

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

SAŽETAK

Računalni vid se svojim razvojem proširio i na korištenje za praćenje napretka u treningu. U ovom radu izrađen je računalni model za klasifikaciju i automatsko brojanje sklekova u Python programskom jeziku koristeći MediaPipe biblioteku. Prije izrade modela teorijski će te biti upoznati s procjenom ljudskog položaja pomoću računalnog vida. Programski kôd prikazuje tu procjenu u 2D prikazu i prebrojava sklekove. Nakon što se model evaluira na tri primjera, rezultati su analizirani i prikazani načini za poboljšanje rada.

Ključne riječi: Python, MediaPipe, procjena ljudskog položaja, brojač sklekova, računalni vid.

SUMMARY

The field of computer vision has expanded its scope to include tracking progress in training. In this paper, a computer model was developed for push-up classification and automatic counting using the Python programming language and the MediaPipe library. Before creating the model, you will be introduced with the theoretical aspects of human pose estimation using computer vision. The code displays this estimation in a 2D view and counts the push-ups. After evaluating the model on three examples, the results are analyzed, and methods for improving performance are presented.

Key words: Python, MediaPipe, human pose estimation, push-up counter, computer vision.

1. UVOD

Računalni modeli temeljeni na umjetnoj inteligenciji svakodnevno se razvijaju i povezuju računala sa stvarnom okolinom. Računala sve više aktivnosti obavljaju kao što to rade ljudi. Cilj umjetne inteligencije nije zamijeniti čovjeka u potpunosti već pomoći i olakšati čovjeku u svakodnevnom životu. Računalni vid omogućuje računalima da prima ulazne podatke kao što su slike i videozapisi. U ovom radu ulazni podatak je videozapis koji prikazuje osobu kako radi sklekove, a cilj modela je klasificirati i prebrojati sklekove. Potrebno je prikupiti ulazne podatke, napisati računalni model te ga testirati na podacima.

U teorijskoj osnovi rada biti će te upoznati s pojmovima dubokog učenja, računalnog vida te procjene ljudskih položaja. Zatim je kreiran kôd te učitan videozapis kao ulazni podatak. Korišten je programski jezik Python te biblioteke NumPy, OpenCV i MediaPipe. Pomoću MediaPipe učitavamo BlazePose koji predstavlja topologiju 33 ključnih točaka na tijelu. Pronalazimo ključne točke ramena, lakta i zapešća te računamo kut između njih. Na taj način model broji koliko sklekova je osoba napravila. Testiramo model na različitim videozapisima te usporedimo točan broj sa brojem koji model prebroji.

Promjenom par linija kôda u ovom modelu možemo pratiti i ostale tjelovježbe. Uz malo više truda može se i izraditi aplikacija te time olakšati i korištenje ovog modela. Ovako bismo mogli pratiti točnost tjelovježbi te broj ponavljanja. S obzirom da je procjena ljudskog položaja jedan od kompleksnijih problema, potrebno je još više razvijati računalni vid u tom području.

2. TEORIJSKA OSNOVA RADA

2.1. Strojno učenje

Strojno učenje je područje unutar umjetne inteligencije koje općenito opisuje sposobnost stroja da imitira inteligentno ljudsko ponašanje. Cilj umjetne inteligencije je stvaranje računalnih modela koji pokazuju "inteligentno ponašanje" slično onome kod ljudi. To uključuje strojeve koji mogu prepoznati vizualne scene, razumjeti tekst napisan prirodnim jezikom ili izvršavati radnje u fizičkom svijetu. [1]

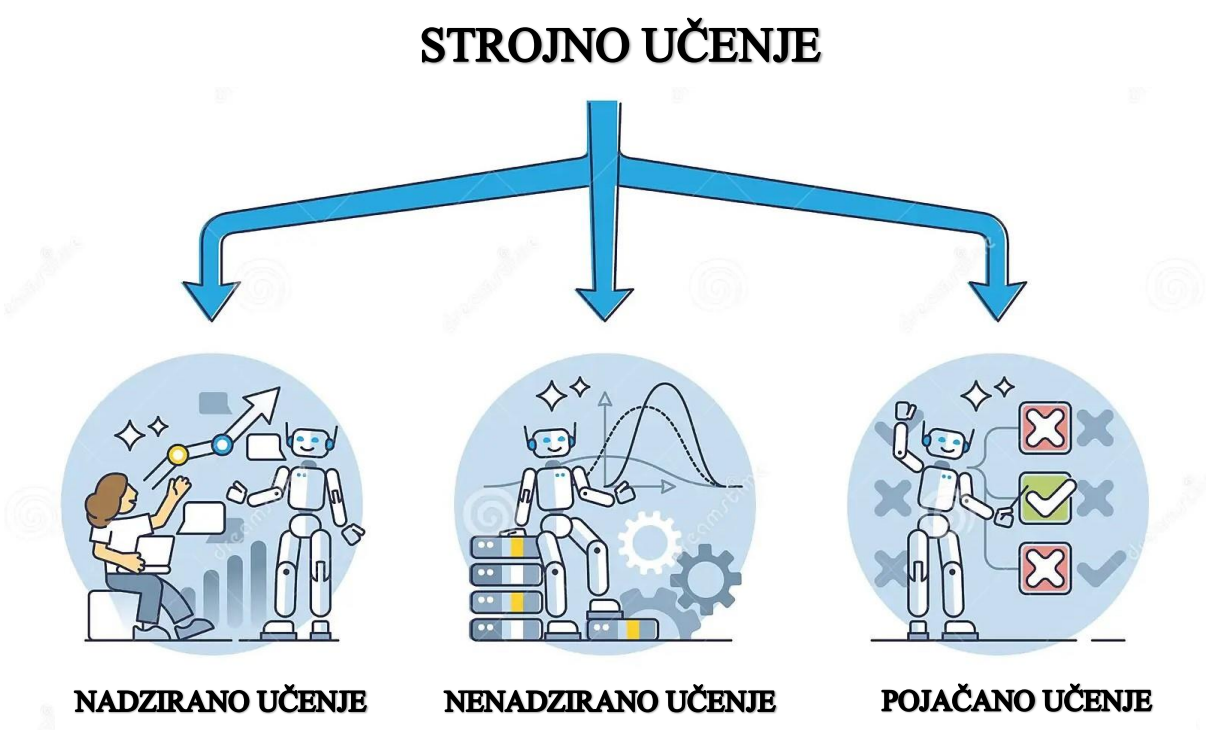
Strojno učenje je jedan od načina primjene umjetne inteligencije, 1950-ih ga je definirao pionir umjetne inteligencije Arthur Samuel kao "područje istraživanja koje računalima daje sposobnost učenja bez eksplicitnog programiranja". Primjenjuje pristup dopuštanja računalima da nauče programirati sebe kroz iskustvo.

Proces strojnog učenja započinje s podacima - brojevima, fotografijama ili tekstom. Podaci se prikupljaju i pripremaju za korištenje kao obuka za model strojnog učenja, odnosno informacije na kojima će se model trenirati. Podaci moraju biti pripremljeni i prilagođeni da što bolje predstavljaju fenomen koji opisuju.

Nakon toga, programeri odabiru model strojnog učenja, pružaju mu podatke i dopuštaju računalnom modelu da se samostalno trenira kako bi pronašao uzorke ili donio predviđanja. Tijekom vremena, ljudski programeri mogu prilagoditi model, uključujući promjenu njegovih parametara, kako bi postigli preciznije rezultate.

Nešto podataka se čuva izvan treniranja modela kako bi se koristilo kao skup podataka za evaluaciju, što testira koliko je precizan model strojnog učenja kada se koristi za nove podatke. Konačni rezultat je model koji se može koristiti u budućnosti s različitim setovima podataka.

Strojno učenje prema slici [Slika 2.1] možemo podijeliti u tri skupine: nadzirano, nenadzirano i pojačano učenje.



Slika 2.1 Podjela tipova strojnog učenja. [2]

Ako skup podataka uključuje ulazne i izlazne uzorke, sažeti opis podataka je funkcija koja može proizvesti izlaz na temelju unosa. Ovaj problem također je poznat kao problem nadziranog učenja (engl. *supervised learning*) jer su objekti pod nadzorom već povezani s ciljnim vrijednostima (klasama, stvarnim vrijednostima). Primjeri nadziranog učenja su klasifikacija ručno pisanih slova i brojeva, predviđanje vrijednosti dionica na burzi. Ovisno o vrsti izlaza, razlikuju se učenje klasifikacije, učenje preferencije i učenje funkcije. [3]

Ako prostor izlaza nema strukturu osim što se utvrđuje jesu li dvije stavke u prostoru izlaza jednake ili ne, to se naziva problemom učenja klasifikacije. Svaki element prostora izlaza naziva se klasom. Ovaj problem pojavljuje se u gotovo svakom zadatku prepoznavanja uzoraka. Primjeri ovog zadatka učenja uključuju klasifikaciju slika u klase ili klasifikaciju elemenata slike (piksela) u klase.

Ako je prostor izlaza uređeni prostor tj. možemo usporediti jesu li dvije stavke jednake tada se problem nadziranog učenja naziva i problemom učenja preferencije. Elementi prostora izlaza nazivaju se rangovima. Primjer je rasporediti web-stranice tako da su najrelevantnije stranice rangirane najviše. Moguće je povezivanje klasifikacije i učenja preferencije.

Ako je prostor izlaza metrički prostor, kao što su stvarni brojevi, tada se zadatak učenja naziva problemom učenja funkcija. Jedna od najvećih prednosti učenja funkcija je mogućnost korištenja tehnika gradijentnog spuštanja kad je funkcija f(x) različita funkcija objekta x samog.

Nenadzirano učenje (engl. *unsupervised learning*) koristi podatke koji su samo uzorak objekata bez povezanih ciljnih vrijednosti. Algoritmi klasterizacije otkrivaju sličnosti i razlike u informacijama te slične objekte grupiraju u iste klastere. To čini nenadzirano učenje idealnim za rješavanje problema segmentacije slike i teksta. [3]

Pojačano učenje (engl. *reinforcement learning*) je model učenja kod kojeg je cilj sustava učenje iz vlastitih postupaka i iskustva. Razlika između učenja pojačanja i nadziranog učenja je ta da u učenju pojačanja ne postoji optimalna akcija u danom stanju, već učeni algoritam mora identificirati akciju koja će maksimizirati očekivanu nagradu tijekom vremena. Kako bi se maksimizirala nagrada, algoritam za učenje mora birati akcije koje su isprobane u prošlosti i pokazale se učinkovitima u stvaranju nagrade. Primjer je igra šaha. Položaj svih figura na ploči predstavlja određeno stanje, akcije su mogući potezi u određenom položaju. Nagrada za određenu akciju (potez u šahu) je pobjeda, poraz ili postizanje neriješenog rezultata. [3]

2.2. Duboko učenje

Duboko učenje je područje unutar strojnog učenja koje se temelji na algoritmima za učenje na više razina i obično koristi umjetne neuronske mreže. Neuronske mreže pokušavaju simulirati ponašanje ljudskog mozga dopuštajući modelu da samostalno uči i donosi odluke. Dodatni skriveni slojevi u neuronskoj mreži pomažu u optimizaciji i poboljšanju radi točnosti predviđanja. Tri važna razloga za popularnost dubokog učenja danas su drastično povećane sposobnosti obrade čipova (npr. grafičkih procesorskih jedinica opće namjene), značajno povećana količina podataka korištena za obuku i nedavni napretci u istraživanju strojnog učenja i obrade signala. Razine u ovim naučenim statističkim modelima odgovaraju različitim razinama koncepata, pri čemu se viši koncepti definiraju od nižih, a isti niži koncepti mogu pomoći u definiranju mnogih viših koncepata. [4]

Na slici [Slika 2.2] prikazana je razlika između umjetne inteligencije, strojnog učenja i dubokog učenja.

Duboko učenje se razlikuje od strojnog učenja jer koristi neuronske mreže za prikaz podataka, dok strojno učenje koristi strukturirane podatke. Rješava složene probleme strojnog učenja s minimalno ljudske intervencije. Algoritmi dubokog učenja, za razliku od strojnog učenja,

zahtijevaju velike količine podataka za treniranje neuronskih mreža, ali mogu sami učiti i poboljšavati se kako obrađuju više podataka. [5]

Slika 2.2 Razlika između strojnog i dubokog učenja.

Duboko učenje se odnosi na prilično široku klasu tehnika strojnog učenja i arhitektura koje se odlikuju upotrebom više slojeva nelinearne obrade informacija koje su hijerarhijske prirode. Većinu rada u ovom području možemo široko kategorizirati u tri glavne klase: duboke mreže za nenadzirano, duboke mreže za nadzirano učenje i hibridne duboke mreže.

Duboke mreže za nenadzirano ili generativno učenje (engl. *deep networks for unsupervised or generative learning*), ne koriste informacije o nadzoru specifične za zadatak (npr. oznake ciljne klase) u procesu učenja. Mnoge duboke mreže u ovoj kategoriji mogu se koristiti za značajno generiranje uzoraka putem uzorkovanja iz mreža, kao što su generalizirani autoenkoderi s uklanjanjem šuma (engl. *denoising*). Ali neke mreže u ovoj kategoriji se ne mogu lako uzorkovati, poput mreža za rijetko kodiranje i originalnih oblika dubokih autoenkodera, te stoga nisu prirodno generativne. Duboki autoenkoder ima kao ciljeve izlaza same ulazne podatke umjesto oznaka klasa stoga je model učenja bez nadzora. Kada je treniran s kriterijem uklanjanja šuma, duboki autoenkoder je također generativni model i može se uzorkovati.

Duboke mreže za nadzirano učenje (engl. *deep networks for supervised learning*), koje su namijenjene izravno pružanju diskriminativne snage za svrhe klasifikacije obrazaca, često karakteriziranjem posteriori distribucija razreda uvjetovanih vidljivim podacima. Podaci o

ciljnim oznakama uvijek su dostupni u izravnom ili neizravnom obliku za takvo nadzirano učenje. Također se nazivaju diskriminativne duboke mreže. [4]

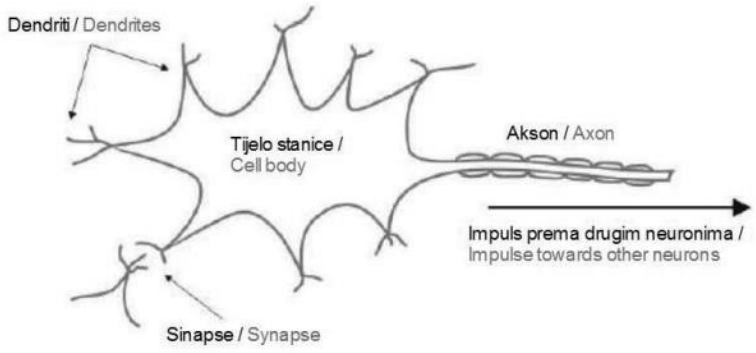
Hibridne duboke mreže (engl. *hybrid deep networks*) koriste rezultate generativnih ili nenadziranih dubokih mreža kao pomoć za postizanje diskriminacije (engl. *discrimination*). Cilj se može postići kada se diskriminativni kriteriji za nadzirano učenje koriste za procjenu parametara u bilo kojoj od dubokih generativnih ili nenadziranih dubokih mreža. Izraz "hibrid" odnosi se na duboku arhitekturu koja uključuje ili koristi i generativne i diskriminativne komponente modela [4]

2.3. Umjetne neuronske mreže

Umjetna neuronska mreža (engl. *artificial neural network*) je računalni model temeljen na neuronskoj strukturi mozga. Simulira način na koji ljudski mozak analizira i obrađuje podatke. Podskup je strojnog učenja i u središtu je algoritama dubokog učenja.

Umjetne neuronske mreže su izgrađene poput ljudskog mozga, sa neuronskim čvorovima isprepletenim poput mreže. Najosnovniji element ljudskog mozga je određena vrsta stanice koja pruža sposobnost pamćenja, razmišljanja i primjene prethodnih iskustava u svakoj našoj akciji. Te stanice, njih 100 milijardi, poznate su kao neuroni. Svaki od tih neurona može se povezati s do 200.000 drugih neurona, iako je uobičajeno da se povežu između 1.000 do 10.000 neurona. Informacije prenose putem raznih elektrokemijskih putanja. Postoji više od stotinu različitih klasa neurona, ovisno o korištenoj metodi klasifikacije. Ovi neuroni i njihove veze zajedno tvore proces koji nije binaran, nije stabilan i nije sinkroniziran.. Svaki neuron sastoji se od staničnog tijela (soma), dendrita, aksona i sinapse. [6]

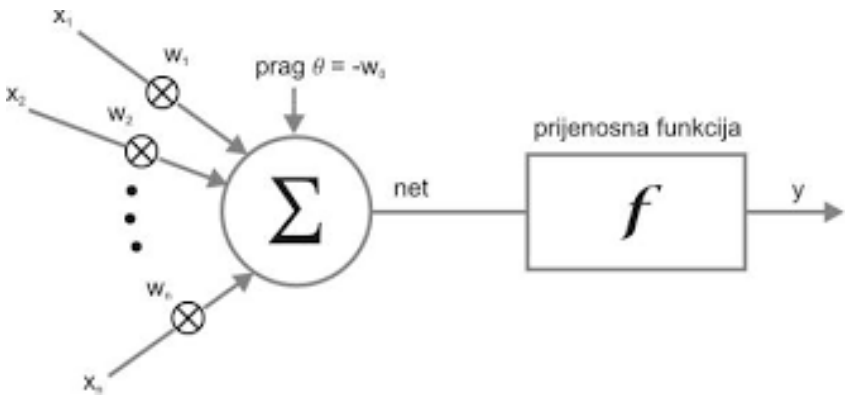
Na slici [Slika 2.3] prikazana je građa biološkog neurona.



Slika 2.3 Građa biološkog neurona. [7]

Dendriti su vlaknasta produženja some koja djeluju kao ulazni kanali koji primaju ulazni signal preko sinapsi drugih neurona, soma je odgovorna za obradu tih ulaznih signala i pretvara ih u izlaz koji se šalje drugim neuronima, akson je odgovoran za dobivanje obrađenih signala (impulsa) od some, a sinapsa je veza između aksona i drugih neuronskih dendrita i prenosi impulse s jednog neurona na drugi. [6]

Umjetna neuronska mreža ima stotine ili tisuće umjetnih neurona (perceptron). Umjetni neuron je matematička funkcija temeljena na modelu biološkog neurona. Na slici [Slika 2.4] je prikazan model umjetnog neurona. Neuroni se programiraju pomoću računala da djeluju kao povezane moždane stanice kako bi stvorili umjetnu neuronsku mrežu. Signali su opisani numeričkim iznosom (x) i na ulazu u neuron množe se odgovarajućim težinskim faktorom (w(n)), zatim se šalju u funkciju zbrajanja (∑) te ako je iznos iznad definiranog praga, ide dalje u funkciju prijenosa (f) te neuron pretvara ulazni broj u izlaz putem određenog algoritma i daje izlazni signal (y). [8]



Slika 2.4 Model umjetnog neurona. [9]

U tablici [Tablica 2.1.] su navedene sličnosti između biološkog i umjetnog neurona.

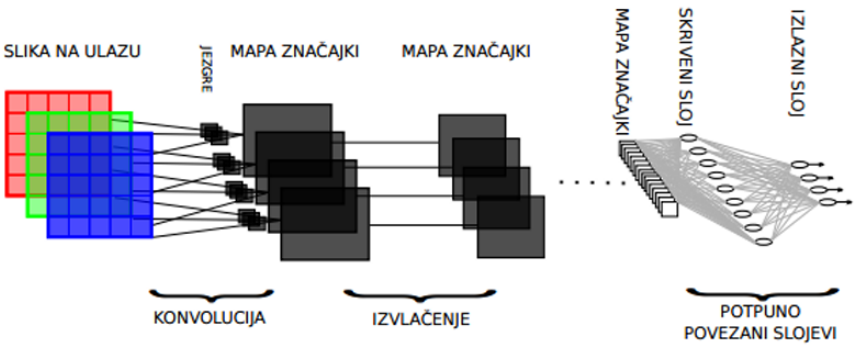
Tablica 2.1 Sličnosti između biološkog i umjetnog neurona.

Biološki neuron	Umjetni neuron
Ulazni signal prima putem dendrita	Ulazi su određeni težinskim faktorima
Signal se obrađuje u somi	Obrada ulaza; prag se dodaje sumi ulaza
Akson pretvara obrađeni ulaz u izlaz	Prijenosna funkcija pretvara ulaze u izlaz
Sinapsa prenosi impulse od neurona do ostalih neurona s kojima je povezan	Prenosi informacije do ostalih neurona

2.4. Konvolucijske neuronske mreže

Konvolucijske neuronske mreže (ConvNets ili CNN-s, engl. *convolutional neural networks*) su neuronske mreže koje se najčešće koriste za zadatke klasifikacije i računalnog vida. Mreža koristi matematičku operaciju konvolucija umjesto općeg matričnog množenja u barem jednom od svojih slojeva. Pružaju skalabilniji pristup klasifikaciji slike i zadacima prepoznavanja objekata. Oslanjaju se na principe linearne algebre, posebno množenja matrica, za identifikaciju uzoraka slika. Razlikuju se od ostalih neuronskih mreža po boljoj obradi ulaznih slika, govora ili audio signala. Sastoje se od ulaznog sloja, skrivenih slojeva i izlaznog sloja. Tri glavne vrste skrivenih slojeva su konvolucijski sloj (engl. *convolutional layer*), sloj sažimanja (engl. *pooling layer*) i potpuno povezani sloj (engl. *fully-connected layer*). [5]

Na slici [Slika 2.5] je prikazana konvolucijska neuronska mreža.



Slika 2.5 Prikaz konvolucijske neuronske mreže. [10]

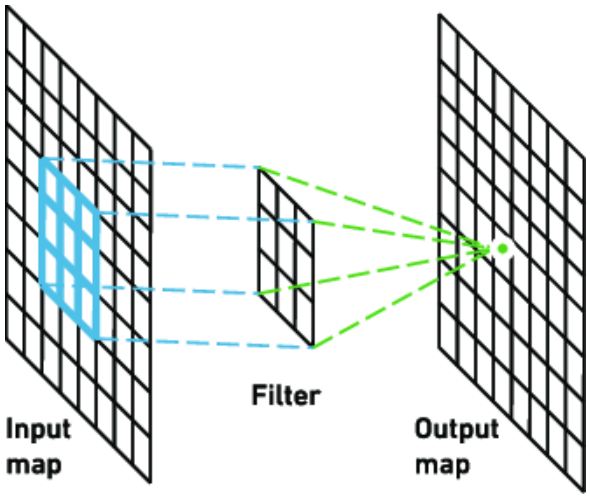
U mrežu ulazi ulazni volumen slike. Prvo je potrebno obraditi podatke. Ulazni podatci centrirani su na nulu oduzimanjem srednje vrijednosti izračunate na cijelom skupu podataka za

treniranje. Zatim prolaze proces normalizacije, dijele se sa standardnom devijacijom svake ulazne dimenzije (piksela ako je u slučaju slika) izračunate na skupu podataka za treniranje kako bi se normalizirala standardna devijacija na jediničnu vrijednost.

U konvolucijskom sloju filter (engl. *kernel*) se kreće po receptivnim poljima slike duž širine i visine te je skenira. Veličina filtra može biti matrica 2x2, 3x3, 4x4 itd. Filter se zatim primjenjuje na područje slike, množi se s područjem iste veličine unutar ulazne mape značajki (receptivnim poljima), a rezultirajuće vrijednosti se zbrajaju kako bi se dobila odgovarajuća vrijednost na izlaznoj mapi značajki (engl. *feature map*) u svakom koraku konvolucije. Konvolucijski sloj pretvara sliku u numeričke vrijednosti dopuštajući neuronskoj mreži da izdvaja i promatra bitne uzorke. [11]

Uzorkovanje (engl. *padding*) je tehnika koja se koristi kako bi se smanjile dimenzije slike kada se filter ne uklapa savršeno u ulaznu sliku. Postoje 2 različite operacije uzorkovanja: prva je kada dopunimo podatke nulama (nula-uzorkovanje, engl. *zero-padding*), a druga kada izbrišemo dio slike gdje filter nije potpuno prilagođen te ona uključuje samo valjane uzorkovane podatke (valjano uzorkovanje, engl. *valid padding*). [11]

Na slici [Slika 2.6] je prikazan primjer konvolucijskog sloja s ulaznom mapom značajki, receptivnim poljima (engl. *local receptive field*) označeno plavom na slici, filterom te izlaznom mapom značajki.

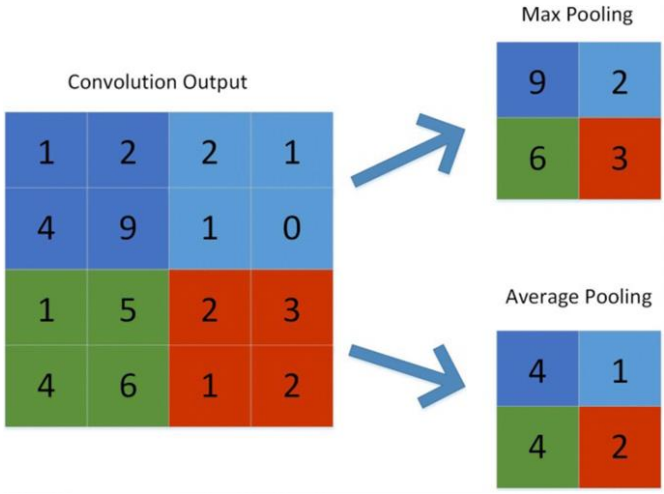


Slika 2.6 Primjer konvolucijskog sloja. [12]

Slojevi sažimanja ili slojevi smanjenja uzorkovanja (engl. *downsampling*) provode smanjenje dimenzionalnosti, smanjujući broj ulaznih parametara. Filter se kreće po cijeloj ulaznoj slici

kao i kod konvolucijskog sloja, ali ovaj filter koristi agregacijske funkcije (funkcije sakupljanja) na vrijednosti unutar receptivnog (prijemnog) polja popunjavajući izlazni niz. Postoje 2 vrste sažimanja : maksimalno sažimanje (engl. *max pooling*) i prosječno sažimanje (engl. *average pooling*). Kod maksimalnog sažimanja filter dok se pomiče po slici, odabire piksel s najvećom vrijednosti i šalje ga u izlazni niz, a kod prosječnog sažimanja filter izračunava prosječnu vrijednost unutar receptivnog polja te je šalje u izlazni niz. Ovaj sloj pomaže mreži u smanjenju složenosti i poboljšanju učinkovitosti. [11]

Na slici [Slika 2.7] je prikazan primjer sloja sažimanja.

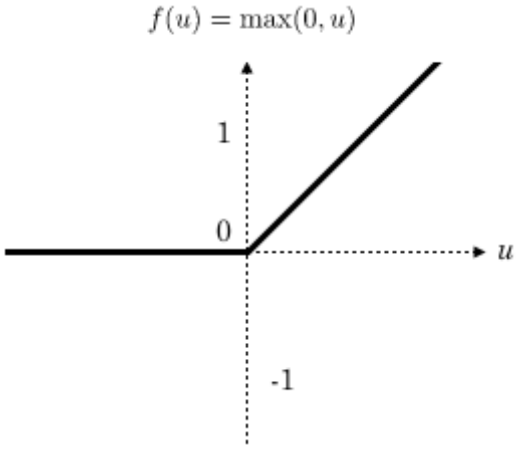


Slika 2.7 Prikaz sloja sažimanja. [13]

Težinski slojevi u CNN-u često su praćeni nelinearnom ili djelomično linearnom aktivacijskom funkcijom. Aktivacijska funkcija uzima stvarnu vrijednost ulaza i smanjuje je unutar malog raspona. Primjena nelinearne funkcije nakon težinskih slojeva važna je jer omogućava neuronskoj mreži učenje nelinearnih preslikavanja. Nelinearna funkcija također se može shvatiti kao mehanizam selekcije, koji odlučuje hoće li se neuron aktivirati ili ne, uzimajući u obzir sve njegove ulaze. Aktivacijske funkcije koje se često koriste u dubokim neuronskim mrežama su: sigmoidna funkcija, tanh funkcija (koristi hiperbolični tangens), algebarska sigmoidna funkcija te ispravljena linearna jedinična funkcija. [11]

„ReLU“ označava „ispravljenu linearnu jediničnu funkciju“ (engl. *rectified linear unit*). Primjenjuje se na mapi značajki, unoseći nelinearnost u model. Nelinearnost unosi na način da zamijeni negativne vrijednosti piksela s nulama, a pozitivne zadržava. [11]

Na slici [Slika 2.8] je prikazana ReLU aktivacijska funkcija.

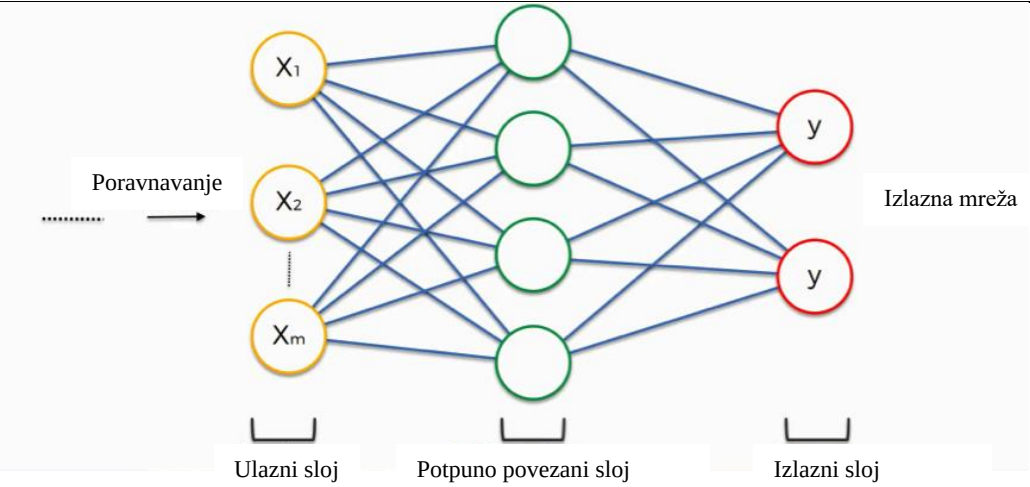


Slika 2.8 ReLU aktivacijska funkcija. [14]

Potpuno povezani slojevi (engl. *fully-connected layers*) u osnovi odgovaraju konvolucijskim slojevima s filterom veličine 1x1. Svaka jedinica prima izlaz od svake jedinice prethodnog sloja. Obavlja klasifikaciju na temelju značajki izdvojenih iz prethodnih slojeva i njihovih filtera. Proizvodi izlaznu matricu i konačni sloj za generiranje izlazne slike sa njenim imenom klase.[11]

Potpuno povezani slojevi koriste aktivacijsku funkciju *Softmax* za klasifikaciju ulaza koja stvara vjerojatnost između 0 i 1. Njena formula je jako slična sigmoidnoj funkciji. Funkciju *Softmax* možemo koristiti u klasifikaciji samo kada su klase međusobno isključive. Koristi se u posljednjem sloju CNN-a kako bi izlaz predstavljao vjerojatnosnu distribuciju mogućih klasa. Posljednji sloj CNN-a je odgovoran za konačnu prognozu o klasi ulazne slike. Softmax osigurava da konačna prognoza bude vjerojatnosna distribucija, što znači da je svakoj klasi dodijeljena odgovarajuća vjerojatnost. Ova distribucija vjerojatnosti zatim može biti korištena za precizniju prognozu klase ulazne slike. [15]

Na slici [Slika 2.9] je prikazan primjer potpuno povezanog sloja.



Slika 2.9 Prikaz potpuno povezanog sloja. [16]

2.5. Računalni vid

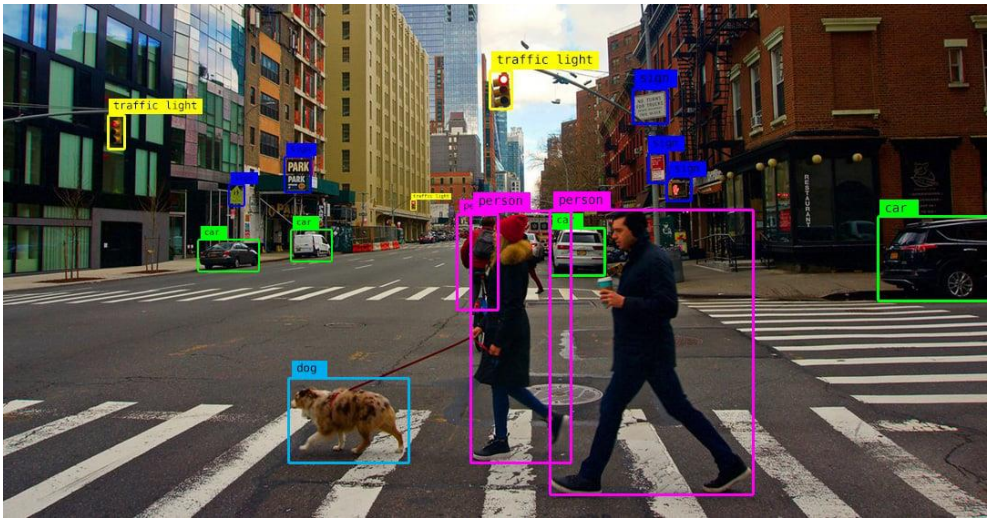
Računalni vid istražuje načine kako računalima omogućiti repliciranje ljudskog vizualnog sustava. Kao dio umjetne inteligencije, bavi se prikupljanjem i obradom informacija iz digitalnih slika ili videozapisa kako bi definiralo njihove karakteristike. Ovaj složeni proces uključuje akviziciju slike, njezino pregledavanje, analizu, identifikaciju i izdvajanje relevantnih informacija. Sve to omogućava računalima da razumiju vizualni sadržaj i poduzimaju odgovarajuće radnje.

Projekti računalnog vida transformiraju digitalni vizualni sadržaj u jasne opise radi prikupljanja višedimenzionalnih podataka. Ti se podaci potom prenose u oblik čitljiv za računala kako bi podržali proces donošenja odluka. Osnovni cilj ovog područja umjetne inteligencije jest poučiti strojeve kako učinkovito prikupljati informacije iz piksela.

Područje računalnog vida razvilo se od primjene klasičnih tehnika prepoznavanja oblika i obrade slika do naprednih primjena razumijevanja slika, vizije temeljene na modelima, vizije temeljene na znanju i sustava s mogućnostima učenja. Mogućnost zaključivanja i sposobnost učenja dvije su glavne sposobnosti povezane s tim sustavima. Web je postao ključno sredstvo za prijenos grafičkih informacija, potičući premještanje pretraživanja vizualnih podataka iz zasebnih radnih stanica u umreženo okruženje.

Na području intelektualnog pristupa vizualnim informacijama, interakcija između ljudskih i strojnih metoda indeksiranja slika počela je utjecati na razvoj sustava računalnog vida. Istraživanje i primjena od strane zajednice za razumijevanje slika sugeriraju da su najplodniji pristupi toj domeni analiza i učenje o vrsti informacija koja se traži, domeni u kojem će se koristiti, te sistematsko testiranje kako bi se identificirale optimalne metode. [17]

Na slici [Slika 2.10] je prikazan primjer računalnog vida.



Slika 2.10 Primjer računalnog vida. [18]

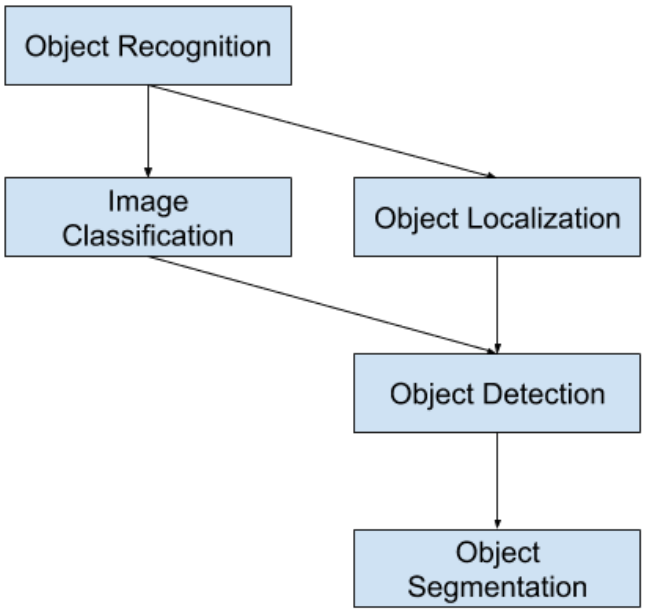
2.5.1. Primjena računalnog vida

Računalni vid obuhvaća vrlo široko područje s različitim načinima obrade slika i iznimnom raznolikošću primjena. Primjene računalnog vida kreću se od repliciranja ljudskih vizualnih sposobnosti, poput prepoznavanja lica, do stvaranja potpuno novih kategorija vizualnih sposobnosti. Većina istraživanja dubokog učenja u računalnom vidu usmjerena je ne na egzotične primjene koje proširuju područje mogućnosti slika, već na temeljne ciljeve umjetne inteligencije usmjerene na repliciranje ljudskih sposobnosti. Duboko učenje u računalnom vidu uglavnom se koristi za prepoznavanje objekata ili detekciju, što znači izvještavanje o prisutnosti određenog objekta na slici, označavanje slike okvirima oko svakog objekta, transkripciju simbola slike ili označavanje svakog piksela na slici identitetom pripadajućeg objekta. Budući da je generativno modeliranje bilo vodilja istraživanja dubokog učenja, postoji i obimna studija o sintezi slika pomoću dubokih modela. Modeli koji su sposobni za sintezu slika često su korisni za obnavljanje slika, što je zadatak računalnog vida koji uključuje popravljanje nedostataka na slikama ili uklanjanje objekata s njih. [5]

2.6. Prepoznavanje objekata

Prepoznavanje objekata (engl. *object recognition*) jedan je od temeljnih zadataka računalnog vida koji se odnosi na identifikaciju objekata različitih klasa unutar digitalnih vizualnih reprezentacija poput slika ili videa. Kao jedan od najvažnijih zadataka u računalnom vidu, pronašlo je primjenu u brojnim područjima, od autonomnog upravljanja i prepoznavanja lica do procjene položaja ljudskog tijela i daljinskog osjetljivog snimanja. [19]

Na slici [Slika 2.11] prikazana je podjela zadataka računalnog vida.

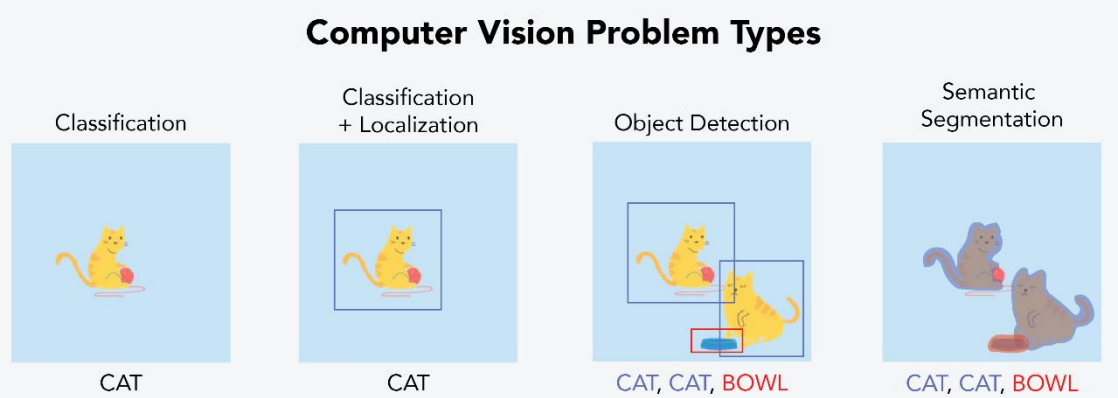


Slika 2.11 Podjela zadataka računalnog vida. [21]

Pojam prepoznavanje objekata obuhvaća klasifikaciju slika (engl. *image classification*) i detekciju objekata (engl. *object detection*). Tako možemo razlikovati tri zadatka računalnog vida: klasifikacija slika, lokalizacija objekata (engl. *object localization*) i detekcija objekata. Klasifikacija slika uključuje predviđanje klase objekta na slici. Lokalizacija objekata je lociranje prisutnosti i položaja objekata te crtanje okvira oko njih. Detekcija objekata kombinira klasifikaciju i lokalizaciju, locira prisutnost objekata s okvirom i predviđa vrste ili klase.

Jedno daljnje proširenje ove podjele zadataka računalnog vida je segmentacija objekata (engl. *object segmentation*), također nazvana "segmentacija instanci objekata" ili "semantička segmentacija", gdje su instance prepoznatih objekata označene isticanjem specifičnih piksela objekta umjesto grubog okvira. [20]

Na slici [Slika 2.12] prikazana je razlika između klasifikacije, lokalizacije, detekcije te segmentacije objekata na primjeru.



Slika 2.12 Razlika između klasifikacije, lokalizacije, detekcije i segmentacije objekata. [22]

2.7. Procjena ljudske poze

Procjena ljudske poze (engl. *human pose estimation*) jedan je od najvažnijih zadataka računalnog vida posljednjih nekoliko desetljeća. Bavi se automatskim predviđanjem i praćenjem ljudskog držanja lokalizacijom *K* tjelesnih zglobova (poznatih i kao ključne točke, poput lakta, zapešća) u RGB slikama ili videozapisu, kao i definiranjem orijentacije njegovih udova. Potencijal ovog zadatka vidljiv je u raznim primjenama, bilo da se radi o praćenju ljudskog kretanja ili sportskoj analizi za automatsko praćenje ili procjenu preciznosti ljudskog kretanja (kao što je u ovom radu primijenjeno na prebrojavanju sklekova), te služi kao temeljno sredstvo u mnogim drugim područjima, poput interakcije čovjeka s računalom i proširene stvarnosti.

U računalnom vidu postoji velika razlika između procjene poze u 2D i 3D. [23]

2.7.1. Procjena poze u 2D

Procjena poze u 2D (dvodimenzionalnom) prostoru sastoji se od predviđanja položaja ključnih točaka tijela u 2D prostoru. Model procjenjuje X i Y koordinate za svaku lokalizaciju zglobova. Za procjenu 2D poze pojedinca, slijedile su se dvije smjernice. Neki pristupi razmatraju procjenu kao problem detekcije (pristupi temeljeni na detekciji, engl. *detection-based approaches*), dok se drugi usredotočuju na izravno regresiranje lokalizacije tjelesnih zglobova (pristupi temeljeni na regresiji, engl. *regression-based approaches*).

Pristupi 2D procjeni više osoba mogu se podijeliti u dvije glavne kategorije: pristupi odozgo prema dolje i pristupi odozdo prema gore. Kod pristupa odozgo prema dolje mreža prvo primjenjuje detektor osobe kako bi definirala okvir oko svake osobe, a zatim locira ključne točke unutar svakog područja koje je definirano. Pristupi odozdo prema gore lokaliziraju sve prisutne ključne točke i zatim pokušavaju grupirati te ključne točke po pojedinim osobama. Bez obzira na odabranu kategoriju, većina modela dubokog učenja prvo uvodi enkoder za izdvajanje značajki iz ulazne slike kroz niz konvolucijskih slojeva. [23]

Na slici [Slika 2.13] je prikazan primjer procjene poze u 2D.

Slika 2.13 Primjer procjene poze u 2D prostoru. [24]

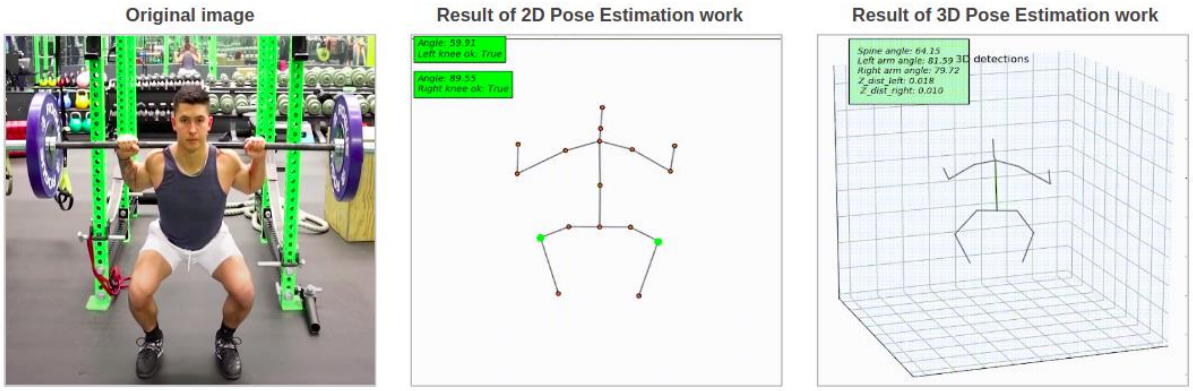
2.7.2. Procjena poze u 3D

Procjena poze u 3D prostoru sastoji se od generiranja trodimenzionalnog izlaza koji ukazuje na prostorni položaj zglobova osobe. Postoje dvije različite kategorije procjene poze u 3D prostoru. Prva kategorija, nazvana jednostupanjski pristupi (engl. *one-stage approaches*), regresira 3D položaj izravno iz slikovnih značajki bez dodavanja bilo kakvih međukoraka. Nasuprot tome, druga kategorija, nazvana dvostupanjski pristupi (engl. *two-stage approaches*), proizlazi iz činjenice da 2D i 3D položaji mogu dijeliti zajedničke prikaze. Uključuje dvije faze, pri čemu prva faza obuhvaća procjenu 2D položaja zglobova, a druga faza obuhvaća obnavljanje 3D položaja iz rezultirajućih 2D zglobnih točaka. Obično ovi pristupi koriste i 2D i 3D skupove podataka kako bi poboljšali izvedbe modela. [23]

Na slici [Slika 2.14] prikazana je razlika između procjene poze u 2D i 3D prostoru.

Područje procjene 3D položaja ljudi još uvijek je ograničeno. Glavni razlog tome je kompliciranа rekonstrukcija stvarnog 3D položaja zglobova ljudi na otvorenom, a time i

nedostatak velikih skupova podataka koji su označeni s 3D položajima zglobova ljudi. Većina 3D skupova podataka snimljena je u kontroliranom laboratorijskom okruženju koristeći sustave za snimanje pokreta.



Slika 2.14 Prikaz razlike između procjene poze u 2D prostoru i 3D prostoru. [25]

3. KORIŠTENE DATOTEKE

Za izradu rada korišten je programski jezik Python izdanja 3.11 jer je jednostavan za korištenje te ima velike mogućnosti. U Python su uvođene biblioteke NumPy, MediaPipe i OpenCV kako bi se izradio računalni model procjene položaja tijela promatrane osobe dok radi sklekove.

3.1. Python

Python je interpretiran, interaktivan, objektno orijentiran programski jezik. Ima široku primjenu u području softvera, web razvoja, znanosti o podacima i automatizacije. Uključuje dinamična semantiku jezika, visokorazinske ugrađene podatkovne strukture, dinamičko tipiziranje i dinamičko povezivanje.

Python je izuzetno svestran programski jezik. Omogućuje jednostavno automatiziranje procesa putem skriptiranja, što ga čini ključnim za testiranje softvera, otklanjanje poteškoća i praćenje pogrešaka. [26]

Postoje različiti Python okviri i biblioteke, poput NumPy i MediaPipe, koje dodaju dodatnu snagu i prilagođene mogućnosti za određene procese.

Python je besplatan te multiplatforman, tj. podržava ga više operacijskih sustava. Sve to dovodi do njegove popularnosti i rasprostranjenosti u programerskom svijetu.

Na slici [Slika 3.1] je prikazan logo programskog jezika Python.

Slika 3.1 Python logo. [27]

3.2. NumPy

NumPy (numerical Python) je projekt otvorenog koda (engl. *open source*) koji omogućuje numeričko računanje s Pythonom. Razvija se uz sudjelovanje raznolike skupine suradnika. Besplatan je za sve korisnike. Koristi se za rad s višedimenzionalnim poljima (engl. *array*).

Polje u NumPy-u je tablica elemenata (obično brojeva), svi istog tipa, indeksiranih tupleom (n-torkom) pozitivnih cijelih brojeva. U NumPy-u, broj dimenzija polja naziva se redom polja. Tuple cijelih brojeva koji daju veličinu polja duž svake dimenzije poznati su kao oblik polja. Klasa polja u NumPy-u naziva se ndarray. Elementima u NumPy poljima pristupa se pomoću uglatih zagrada. [28]

Na slici [Slika 3.2] je prikazano uvođenje NumPy biblioteke u Python.

```
import numpy as np
```

Slika 3.2 Uvođenje biblioteke NumPy.

3.3. OpenCV

OpenCV (Open Source Computer Vision Library) je open-source biblioteka za računalni vid i strojno učenje. Biblioteka ima više od 2500 optimiziranih algoritama, što uključuje opsežan skup i klasičnih i najmodernijih algoritama za računalni vid i strojno učenje. Ti se algoritmi mogu koristiti za detekciju i prepoznavanje lica, identifikaciju objekata, klasifikaciju ljudskih radnji u videima, praćenje kretanja kamere, praćenje pokretnih objekata, izradu 3D modela objekata, pronalaženje sličnih slika iz baze podataka slika, uklanjanje crvenih očiju sa slika snimljenih pomoću bljeskalice, praćenje pokreta očiju, prepoznavanje krajolika i postavljanje markera za preklapanje s proširenom stvarnošću, itd. OpenCV ima više od 47 tisuća članova korisničke zajednice i procijenjeni broj preuzimanja prelazi 18 milijuna. Biblioteka se široko koristi u tvrtkama, istraživačkim grupama i od strane vladinih tijela.

OpenCV koriste poznate tvrtke poput Google-a, Yahoo-a, Microsoft-a, Intela, IBM-a, Sony-a, Honde, Toyote. Primjene OpenCV-a obuhvaćaju širok spektar, od spajanja slika ulica, praćenja opreme u rudnicima u Kini, pomoći robotima u navigaciji i podizanju objekata u tvrtki Willow Garage, detekcije nesreća utapanja u bazenima u Europi, pokretanja interaktivne umjetnosti u Španjolskoj i New Yorku, do brze detekcije lica u Japanu. [29]

Ima sučelja za C++, Python, Java i MATLAB te podržava Windows, Linux, Android i Mac OS.

Na slici [Slika 3.3] je prikazano uvođenje biblioteke OpenCV u Python.

```
import cv2
```

Slika 3.3 Uvođenje biblioteke OpenCV.

OpenCV je izdan pod BSD licencom i besplatan. U ovom radu je korišteno izdanje OpenCV2, objavljeno 2009. godine.

3.4. MediaPipe

Mediapipe je open-source okvir (engl. *framework*) za izgradnju vrhunskih rješenja strojnog učenja tvrtke Google, trenutno u alfa fazi. Otvoren je od 2019. godine, ali je vjerojatno bio u razvoju znatno duže. Besplatan je i kod je napisan u C++, ali ga se lako može implementirati na bilo kojoj platformi.

Prilikom prvog puštanja, Mediapipe je imao samo nekoliko demo aplikacija, ali sada se na njihovoj GitHub stranici može pronaći gotovo desetak različitih demo aplikacija, od praćenja trajnih objekata, do praćenja položaja.

MediaPipe postiže svoju brzinu zahvaljujući upotrebi GPU ubrzanja i višedretvenosti (engl. *multi-threading*). Takve tehnike su složene, ali MediaPipe ih preuzima i radi umjesto vas. Višedretvenost znači da se program sastoji od više jedinica koje se samostalno mogu izvoditi.

Korištenje grafova, podgrafova i kalkulatora u Mediapipe-u znači da rad jednog projekta lako može preći na rad drugog. Mediapipe već dolazi opremljen s mnogo "primjera kalkulatora" koje možete slobodno koristiti, uključujući višeplatformske renderere, višeplatformski TensorFlow Lite i već izrađene neuronske mreže. Renderer je softverski ili hardverski proces koji generira vizualni prikaz iz modela. [30]

MediaPipe nije jednostavan za korištenje, pogotovo zbog nedostatka dokumentacije. Taj nedostatak proizlazi iz toga što se još uvijek nalazi u alfa fazi. Mnoge značajke još uvijek mogu biti podložne promjenama.

Unatoč svojim alfa ograničenjima, MediaPipe je trenutno najbolje rješenje za implementaciju neuronskih mreža na mobilnim uređajima i dobar je za implementaciju neuronskih mreža na računalima.

Knjižnice i resursi koji pružaju osnovne funkcionalnosti za svako rješenje MediaPipe su:

- MediaPipe zadaci - univerzalna API-ji i knjižnice za implementaciju rješenja
- MediaPipe modeli - unaprijed trenirani, modeli spremni za upotrebu sa svakim rješenjem

Ovi alati omogućuju prilagodbu i evaluaciju rješenja:

- MediaPipe Model Maker - prilagodi modele za rješenja prema svojim podacima
- MediaPipe Studio - vizualizira, evaluira i uspoređuje rješenja u svom pregledniku.

MediaPipe rješenja (engl. *MediaPipe Solutions*) dostupna su na više platformi. Svako rješenje uključuje jedan ili više modela, a neka rješenja omogućuju prilagodbu modela. [31]

U tablici [Tablica 3.1] se nalazi popis koji prikazuje koja rješenja su dostupna za svaku podržanu platformu i informacija o mogućnosti prilagodbe modela pomoću Model Maker-a.

Tablica 3.1 Popis MediaPipe rješenja i podržanih platformi.

Rješenje	**Android**	**Web**	**Python**	**iOS**	**Prilagodba modela**
Detekcija objekta	■	■	■	■	■
Klasifikacija slika	■	■	■	■	■
Segmentacija slika	■	■	■		
Interaktivna segmentacija	■	■	■		
Detekcija točaka na ruci	■	■	■	■	
Prepoznavanje gesta	■	■	■	■	■
Ugrađivanje slike	■	■	■		
Detekcija lica	■	■	■	■	
Detekcija točaka na licu	■	■	■		
Stilizacija lica	■	■	■		■
Detekcija točaka na tijelu	■	■	■		
Generacija slika	■				■
Klasifikacija teksta	■	■	■	■	■
Ugrađivanje teksta	■	■	■		
Detektor jezika	■	■	■		
Klasifikacija zvuka	■	■	■		

Na slici [Slika 3.4] prikazano je uvođenje MediaPipe u Python.

```
import mediapipe as mp
```

Slika 3.4 Uvođenje okvira MediaPipe.

3.4.1. Detekcija točaka na tijelu

Zadatak MediaPipe Pose Landmarker omogućuje detekciju ključnih točaka ljudskog tijela na slici ili videozapisu. Može se koristiti kako bi identificirali ključne lokacije tijela, analizirali držanje i kategorizirali pokrete. Ovaj zadatak koristi modele strojnog učenja koji rade s pojedinačnim slikama ili videozapisima. Izlazni podaci o značajkama su prikazani u koordinatama slike i u 3D svjetskim koordinatama.

Značajke ovog zadatka su:

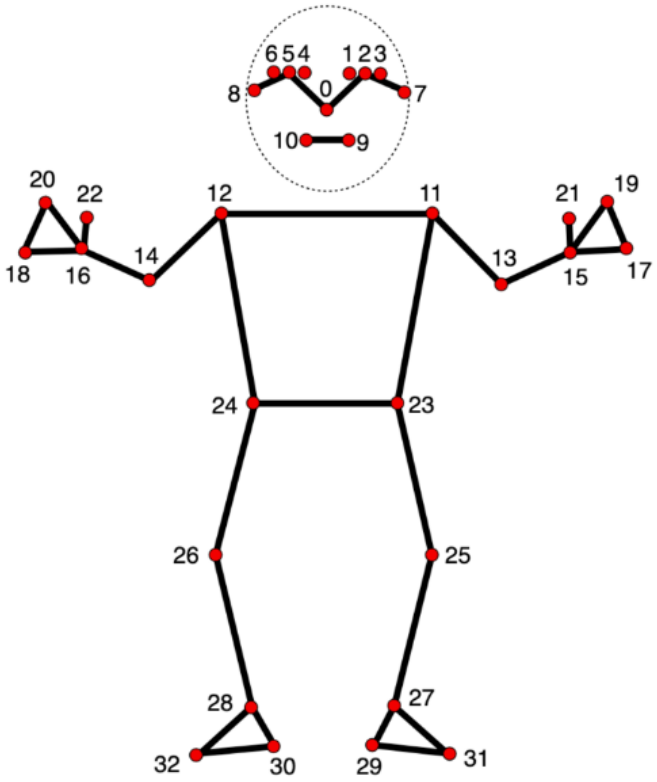
- obrada ulazne slike – rotacija slike, promjena veličine, normalizacija i konverzija prostora boje
- bodovni prag – filtriranje rezultata na temelju predviđenih ocjena/bodova.

Ulazni podaci mogu biti statične slike, dekodirani video okviri ili video uživo. Izlazni podaci koje Pose Landmarker daje su točke položaja/poze u normaliziranim koordinatama slike i točke položaja u svjetskim koordinatama.

Pose Landmarker koristi niz modela za predviđanje točaka položaja. Prvi model detektira prisutnost ljudskog tijela unutar slike, a drugi locira točke na tijelu. Model za otkrivanje položaja i model za označavanje položaja dolaze skupa u paketu Pose Landmarker. Model za otkrivanje položaja detektira prisutnost tijela s nekoliko ključnih značajki položaja, a model za označavanje položaja dodaje potpuno preslikavanje položaja.

Ovaj paket koristi konvolucijsku neuronsku mrežu sličnu MobileNetV2 i optimiziran je za primjenu na uređajima za fitness. Ova varijanta modela BlazePose koristi GHUM (engl. *Generative 3D Human Shape and Articulated Pose Models*), cjevovod (engl. *pipeline*) za modeliranje ljudskog oblika u 3D-u, kako bi procijenila potpuni 3D položaj tijela pojedinca na slikama ili videozapisima. [32]

BlazePose je prikazan na slici [Slika 3.5] te prati 33 lokacije ključnih točaka tijela, predstavljajući približnu lokaciju dijelova tijela prikazanih na slici [Slika 3.6].



Slika 3.5 BlazePose - topologija 33 ključnih točaka. [32]

```
0 - nose
1 - left eye (inner)
2 - left eye
3 - left eye (outer)
4 - right eye (inner)
5 - right eye
6 - right eye (outer)
7 - left ear
8 - right ear
9 - mouth (left)
10 - mouth (right)
11 - left shoulder
12 - right shoulder
13 - left elbow
14 - right elbow
15 - left wrist
16 - right wrist
17 - left pinky
18 - right pinky
19 - left index
20 - right index
21 - left thumb
22 - right thumb
23 - left hip
24 - right hip
25 - left knee
26 - right knee
27 - left ankle
28 - right ankle
29 - left heel
30 - right heel
31 - left foot index
32 - right foot index
```

Slika 3.6 Popis 33 ključnih točaka i dijelova tijela koje prikazuju. [32]

U ovom radu je korišten ovaj model BlazePose te ključne točke 11 (lijevo rame), 13 (lijevi lakat) te 15 (lijevi ručni zglob).

BlazePose je lagana arhitektura konvolucijskih neuronskih mreža za procjenu ljudskog položaja. Pogodna je za primjene u stvarnom vremenu poput praćenja fitnessa i prepoznavanja znakovnog jezika. Glavni doprinos BlazePose-a je inovativno rješenje praćenja položaja tijela i lagano neuronsko mrežno rješenje za procjenu položaja tijela koje koristi i toplinske mape i regresiju do koordinata ključnih točaka. Većina modernih rješenja za detekciju objekata oslanja se na algoritam ne maksimalnog suzbijanja (NMS – Non Maximum Suppression) za njihov posljednji korak nakon obrade. Taj način dobro funkcionira za krute objekte s malo stupnjeva slobode, ali ne za visoko artikulirane položaje poput onih ljudi koji mašu ili se grle. Zbog toga se BlazePose fokusira na detekciju okvira za relativno kruti dio tijela poput ljudskog lica ili trupa. Koristi brzi detektor lica na uređaju kao zamjenu za detektor osobe. Ovaj detektor lica predviđa dodatne parametre poravnanja specifične za osobu: srednju točku između kukova osobe, veličinu kruga koji obuhvaća cijelu osobu i nagib (kut između linija koje spajaju dvije srednje točke ramena i kukova). Ovo nam omogućuje dosljednost s odgovarajućim skupovima podataka i mrežama za zaključivanje. Za razliku od topologija OpenPose i Kinect, koristi se minimalno potreban broj ključnih točaka na licu, rukama i nogama kako bismo procijenili rotaciju, veličinu i položaj područja interesa za daljnji model. Skup podataka za treniranje sastoji se od 60 000 slika s jednom ili nekoliko osoba u sceni u uobičajenim položajima i 25 000 slika s jednom osobom u sceni tijekom vježbanja. [33]

4. IZVEDBA ZADATKA

U nastavku će biti objašnjen programski kôd kojim je razvijen model za klasifikaciju i prebrojavanje sklekova.

Prvi korak je učitavanje biblioteka koje su nam potrebe, u ovom slučaju NumPy, OpenCV i MediaPipe koji su prethodno opisani. Na slici [Slika 4.1] je prikazan kod.

```
import cv2
import mediapipe as mp
import numpy as np
```

Slika 4.1 Uvođenje biblioteka.

Zatim dodavamo dvije nove varijable. Prva je prikazana na slici [Slika 4.2] i ona pokreće alat za crtanje kako bi označili ključne točke na tijelu u videu.

```
mp_drawing = mp.solutions.drawing_utils
```

Slika 4.2 Varijabla za alate za crtanje.

Druga varijabla je prikazana na slici [Slika 4.3] i ona uvodi model procjene položaja/poze.

```
mp_pose = mp.solutions.pose
```

Slika 4.3 Varijabla za model procjene položaja.

Postavljamo brojač sklekova na 0 na slici [Slika 4.4].

```
count = 0
```

Slika 4.4 Postavljanje brojača sklekova na 0.

Videozapise koje ćemo koristiti za evaluaciju koda učitavamo pomoću kôda na slici, a mogla bi se i koristiti kamera računala kao ulazni podatak ako bi u zagradi pisalo 0 umjesto naziva videozapisa [Slika 4.5].

```
cap = cv2.VideoCapture('Bruno_22.mp4')
```

Slika 4.5 Kod za učitavanje videozapisa.

S obzirom da će program prepoznati kada je sklek napravljen po kutu koji zatvaraju rame, lakat i zglob, ispisujemo kod za izračun kuta na slici [Slika 3.6]. Prvo definiramo funkciju *calculate_angle* i pridružujemo joj vrijednosti a, b, c koje predstavljaju prvu, srednju i krajnju točku. Zatim računamo kut u radijanima te ga pretvaramo u stupnjeve da bi kasnije lakše

pretpostavili koji će nam kut trebati za sklekove. U zadnjem dijelu ovog kôda zaključujemo da kut koji ruka zatvara ne može biti veći od 180 i zato se uvodi naredba da ako je kut veći od 180 oduzmemo ga od 360 stupnjeva.

```
def calculate_angle(a,b,c):
            a = np.array(a)
            b = np.array(b)
            c = np.array(c)

            radians = np.arctan2(c[1]-b[1], c[0]-b[0]) - np.arctan2(a[1]-b[1], a[0]-b[0])
            angle = np.abs(radians*180.0/np.pi)

            if angle >180.0:
                  angle = 360-angle
            return angle
```

Slika 4.6 Kod za izračun kuta.

Na slici [Slika 4.7] je prikazana varijabla *Pose* koju postavljamo. Njoj postavljamo minimalna pouzdanost otkrivanja (engl. *minimum detection confidence)* na 50% te minimalna pouzdanost praćenja (engl. *minimum tracking confidence*) na 50%. Ako bi povećali ove brojeve detekcije bi bile točnije, ali povećalo bi se i vrijeme čekanja dok se podaci pojave. Pokušala sam i povećati te brojeve, ali s obzirom da su videozapisi snimani mobilnom kamerom, rezultati ne ispadnu točniji.

```
with mp_pose.Pose(min_detection_confidence=0.5, min_tracking_confidence=0.5) as pose:
```

Slika 4.7 Postavljanje varijable Pose.

While petlja se vrti kroz kod, čita podatke te pomoću varijabli *ret* i *frame* vraća sliku iz videozapisa. Sliku prvo pretvaramo iz BGR (Blue – Green - Red) u RGB (Red – Green – Blue). OpenCV koristi BGR format slika, a mi smo učitali videozapis OpenCV naredbom koja onda našu sliku prikazuje u BGR formatu. Suprotno od toga, MediaPipe koristi RGB format te zbog toga smo sliku pretvorili u RGB format. U idućem koraku MediaPipe obavlja detekciju slike. Kada završi, sliku pretvaramo nazad u BGR format. Ovaj postupak je prikazan na slici [Slika 4.8].

```
while cap.isOpened():
    ret, frame = cap.read()


    image = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    image.flags.writeable = False


    results = pose.process(image)


    image.flags.writeable = True
    image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)
```

Slika 4.8 Podešavanje formata ulazne slike.

Slijedi nam dio kôda koji detektira položaj tijela te označava ključne točke koje su nam potrebne za prebrojavanje sklekova. Na slici [Slika 4.9] prikazan je blok *try* koji omogućuje testiranje bloka kôda. Ako se pojavi pogreška u kôdu, on ne zaustavlja program nego samo prolazi kroz taj dio bez da se petlja uništi.

```
try:
    landmarks = results.pose_landmarks.landmark
    print(landmarks)
```

Slika 4.9 Blok try i učitavanje orijentira.

Odabiremo ključne točke na položaju tijela koje ćemo koristiti za izračun kuta. Uzeli smo lijevo rame, lijevi lakat i lijevi ručni zglob. Slika [Slika 4.10] prikazuje da smo odabrali položaje lijevog ramena, lijevog lakta i lijevog ručnog zgloba. Na prethodnoj slici [Slika 3.5] gdje je prikazana topologija ključnih točaka, možemo vidjeti da ti naši položaji odgovaraju točkama 11, 13 i 15. U kôd smo mogli umjesto npr. LEFT_ELBOW upisati broj 13.

```
landmarks[mp_pose.PoseLandmark.LEFT_SHOULDER.value].visibility
landmarks[mp_pose.PoseLandmark.LEFT_ELBOW.value]
landmarks[mp_pose.PoseLandmark.LEFT_WRIST.value]
```

Slika 4.10 Položaji odabranih točaka na tijelu.

Određujemo 3 nove varijable nazvane *shoulder*, *elbow* i *wrist* koje predstavljaju rame, lakat i zglob. Pomoću naredbi na slici [Slika 4.11] uzete su x i y koordinate ključnih točaka te pohranjene u niz. Kada pozovemo te varijable ispisat će se njihove točne koordinate.

```
shoulder = [landmarks[mp_pose.PoseLandmark.LEFT_SHOULDER.value].x,
            landmarks[mp_pose.PoseLandmark.LEFT_SHOULDER.value].y]
elbow = [landmarks[mp_pose.PoseLandmark.LEFT_ELBOW.value].x,
         landmarks[mp_pose.PoseLandmark.LEFT_ELBOW.value].y]
wrist = [landmarks[mp_pose.PoseLandmark.LEFT_WRIST.value].x,
         landmarks[mp_pose.PoseLandmark.LEFT_WRIST.value].y]


shoulder, elbow, wrist
```

Slika 4.11 Prikazivanje koordinata ključnih točaka.

Definiramo varijablu *angle* koja pomoću prethodno objašnjene funkcije uzima koordinate točaka te računa kut između njih [Slika 4.12].

```
angle=calculate_angle(shoulder, elbow, wrist)
```

Slika 4.12 Računanje kuta.

Izrađeni model na kraju je testiran na 3 različita videozapisa. Ispisivanjem kutova koji zatvaraju ramena i ruke te analizom koji raspon daje najbolje rezultate, odredila sam koji raspon kutova ću koristiti. Ako je kut jednak ili manji od 105 stupnjeva, osoba je u donjem položaju skleka. Ako je kut jednak ili veći od 145 stupnjeva, osoba je u gornjem položaju skleka te tada brojač dodava 1 sklek kao što je prikazano na slici [Slika 4.13]. Možemo ispisati koliko sklekova je model prebrojao, ali u nastavku kôda ćemo ispisati taj broj na ekranu sa slikom.

```
if angle<=105.0:
    position="down"
if angle>=145.0 and position=="down":
    position="up"
    count+=1
    print(count)
```

Slika 4.13 Prebrojavanje sklekova.

Odabir fonta, veličine fonta, boje te debljine slova za vizualni prikaz prebrojavanja su prikazani na slici [Slika 4.14].

```
org = (50, 100)
fontScale = 2
color = (255, 0, 0)
thickness = 3
font = cv2.FONT_HERSHEY_SIMPLEX
```

Slika 4.14 Podaci za estetiku vizualizacije.

Naredbom *cv2.putText* određujemo kako će izgledati video okvir koji će nam prikazivati prebrojavanje sklekova. Tekst će ispisivati koliko sklekova je prebrojano te njemu dodjeljujemo podatke određene na slici [Slika 4.14] te još vrstu linija. Konačan izgled naredbe je prikazan na slici [Slika 4.15].

```
image = cv2.putText(image,"Broj sklekova:  " + str(count),
                    org, font, fontScale, color, thickness, cv2.LINE_AA)
```

Slika 4.15 Naredba za vizualni prikaz brojanja sklekova.

Za prikaz topologije ključnih točaka na položaju tijela koristi se naredba sa slike [Slika 4.16]. Linijama se spoje točke te se pomiču kako se i tijelo pomiče.

```
mp_drawing.draw_landmarks(image, results.pose_landmarks,
                          mp_pose.POSE_CONNECTIONS,
                        mp_drawing.DrawingSpec(color=(245,117,66),
                                               thickness=2, circle_radius=2),
                        mp_drawing.DrawingSpec(color=(245,66,230),
                                               thickness=2, circle_radius=2)
                         )
```

Slika 4.16 Naredba za crtanje linija koje spajaju ključne točke.

Naredbom sa slike [Slika 4.17] otvori se novi prozor u kojem se prikaže videozapis te tekst na njegovom okviru.

```
cv2.imshow('Prebrojavanje sklekova', image)
```

Slika 4.17 Prikaz videozapisa.

5. EVALUACIJA MODELA

Model, koji je opisan u prethodnom poglavlju, je testiran na 3 različita videozapisa.

5.1. Prvi primjer

Prvo je učitan videozapis 'Bruno_11.mp4'. Po njemu sam prilagođavala kôd.

Na slici [Slika 5.1] prikazane su koordinate ključnih točaka u gornjem položaju, te na slici [Slika 5.2] kut koji zatvaraju ruke.

```
[0.6342424750328064, 0.14499086141586304] [0.7202197313308716, 0.31372135877609253] [0.7269573211669922, 0.5193696618080139]
```

Slika 5.1 Prvi primjer - koordinate ključnih točaka u gornjem položaju.

```
174.8421663601317
```

Slika 5.2 Prvi primjer - kut u gornjem položaju.

Na slici [Slika 5.3] prikazane su koordinate ključnih točaka u donjem položaju, te na slici [Slika 5.4] kut koji zatvaraju ruke.

```
[0.6420485377311707, 0.36904335021972656] [0.7397487759590149, 0.3305700719356537] [0.7312710285186768, 0.5214524269104004]
```

Slika 5.3 Prvi primjer - koordinate ključnih točaka u donjem položaju.

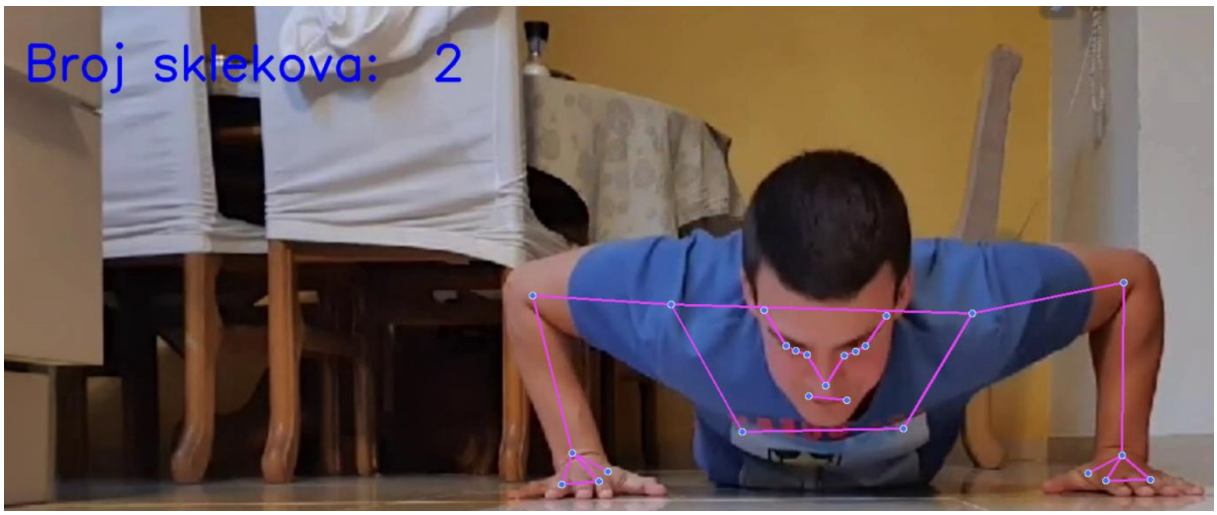
```
65.96299273572646
```

Slika 5.4 Prvi primjer - kut u donjem položaju.

Na osnovu tih kutova odredila sam raspon koji je korišten u funkciji na slici [Slika 4.6].

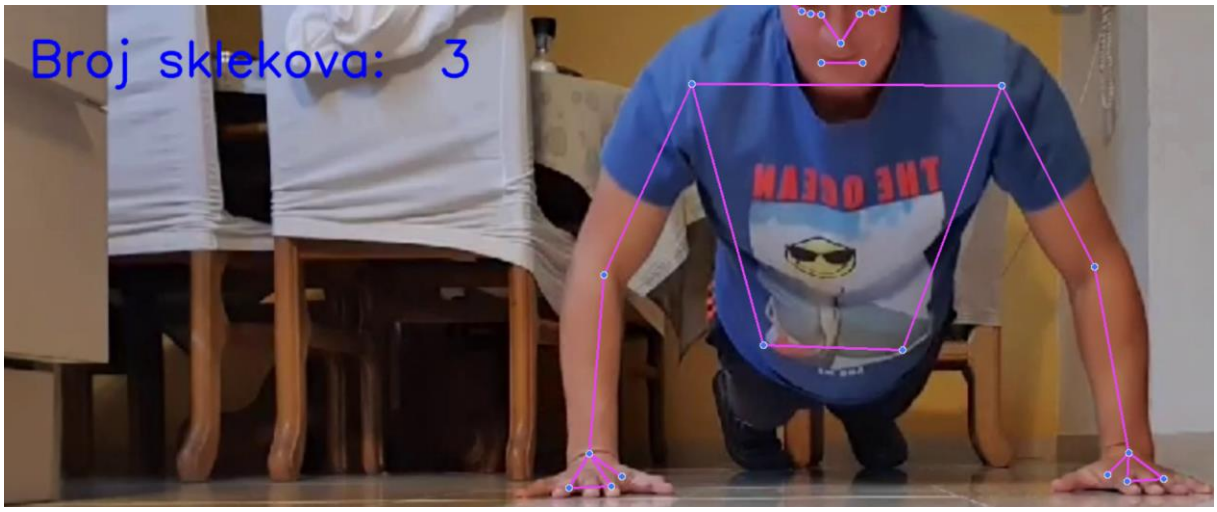
U nastavku ćemo na par slika vidjeti kako izgleda videozapis kada pokrenemo model.

Možemo vidjeti na slici [Slika 5.5] donji položaj kada osoba kreće raditi treći sklek. U okviru se nalazi i tekst koji nam prikazuje koliko je sklekova dosad prebrojano.



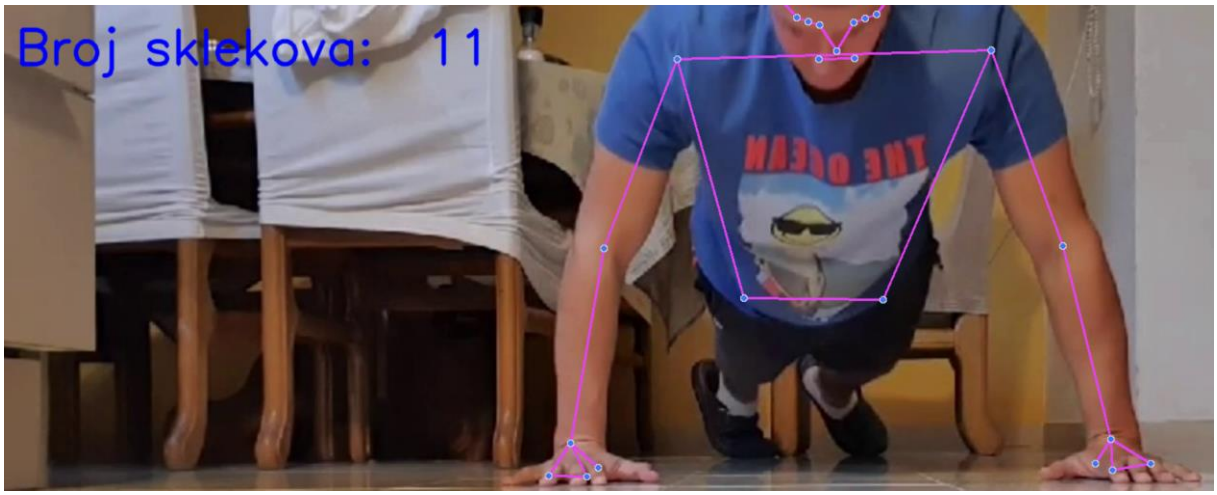
Slika 5.5 Prvi primjer - vizualni prikaz donjeg položaja.

Na slici [Slika 5.6] vidimo da je tijelo u gornjem položaju i brojač je dodao još jedan sklek, broj je povećan na 3.



Slika 5.6 Prvi primjer - vizualni prikaz gornjeg položaja.

Konačan broj sklekova je 11 i model je prebrojao također 11, kao što je i prikazano na slici [Slika 5.7]. Time smo dokazali da nam model daje točne rezultate, ali ćemo još provjeriti na dva primjera.



Slika 5.7 Prvi primjer - ukupan broj sklekova.

5.2. Drugi primjer

U ovom primjeru učitan je videozapis 'Maroje_16.mp4'.

Na slici [Slika 5.8] prikazane su koordinate ključnih točaka u gornjem položaju, te na slici [Slika 5.9] kut koji zatvaraju ruke.

```
[0.6347200870513916, 0.16905450820922852] [0.6537121534347534, 0.42090466618537903] [0.662653923034668, 0.6817343235015869]
```

Slika 5.8 Drugi primjer - koordinate ključnih točaka u gornjem položaju.

```
177.65092368733318
```

Slika 5.9 Drugi primjer - kut u gornjem položaju

Na slici [Slika 5.10] prikazane su koordinate ključnih točaka u donjem položaju, te na slici [Slika 5.11] kut koji zatvaraju ruke.

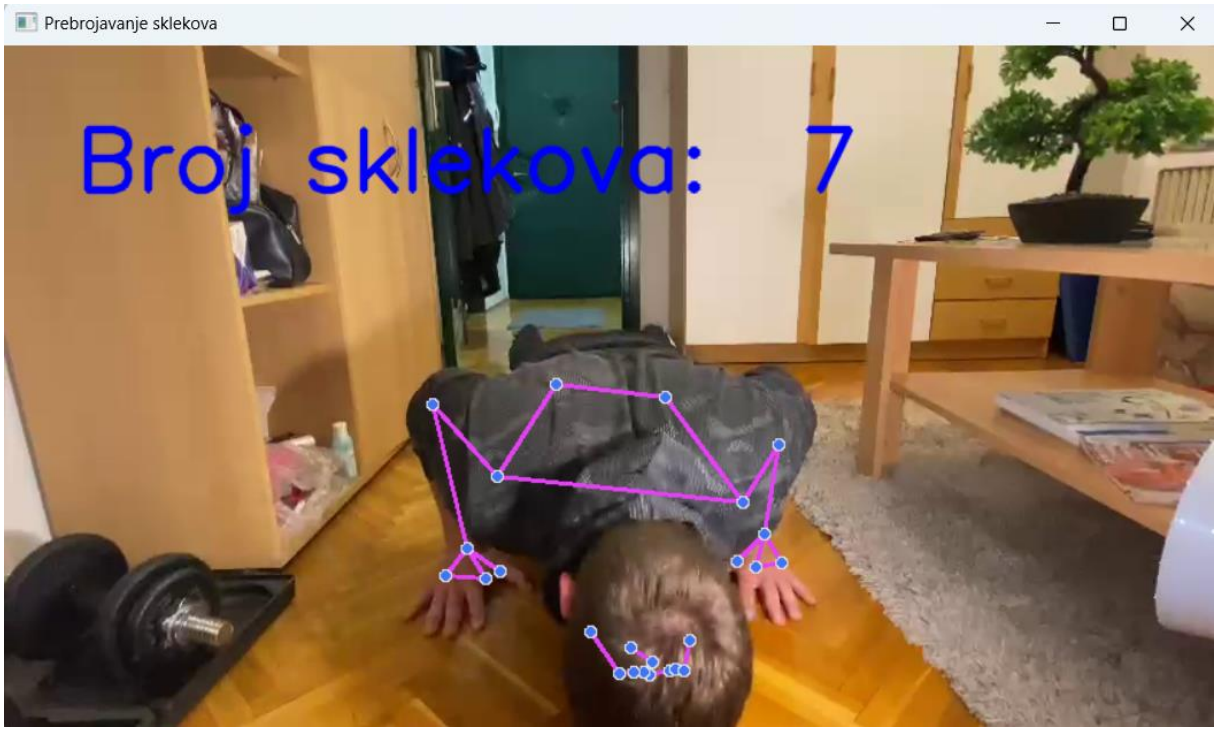
```
[0.6063197255134583, 0.6527166366577148] [0.6338538527488708, 0.5745912790298462] [0.6226651072502136, 0.7044637799263]
```

Slika 5.10 Drugi primjer - koordinate ključnih točaka u donjem položaju.

```
14.490277863425602
```

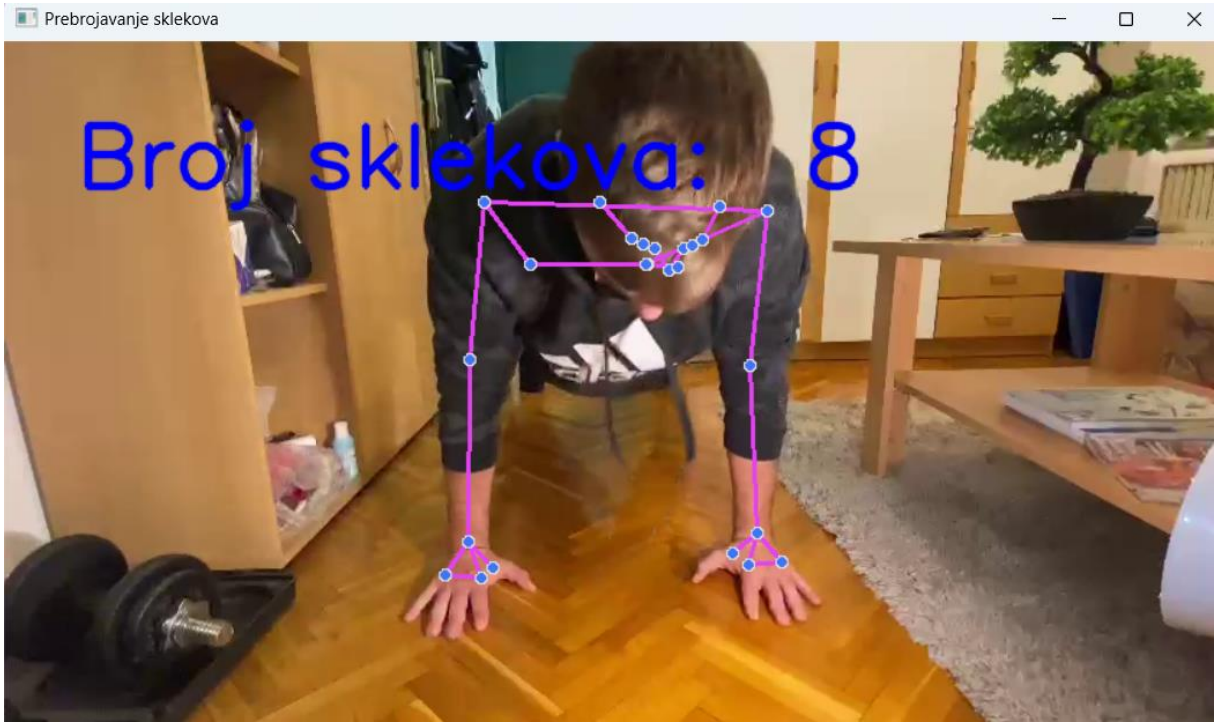
Slika 5.11 Drugi primjer - kut u donjem položaju.

Prikazan je na slici [Slika 5.12] donji položaj kada osoba kreće raditi osmi sklek.



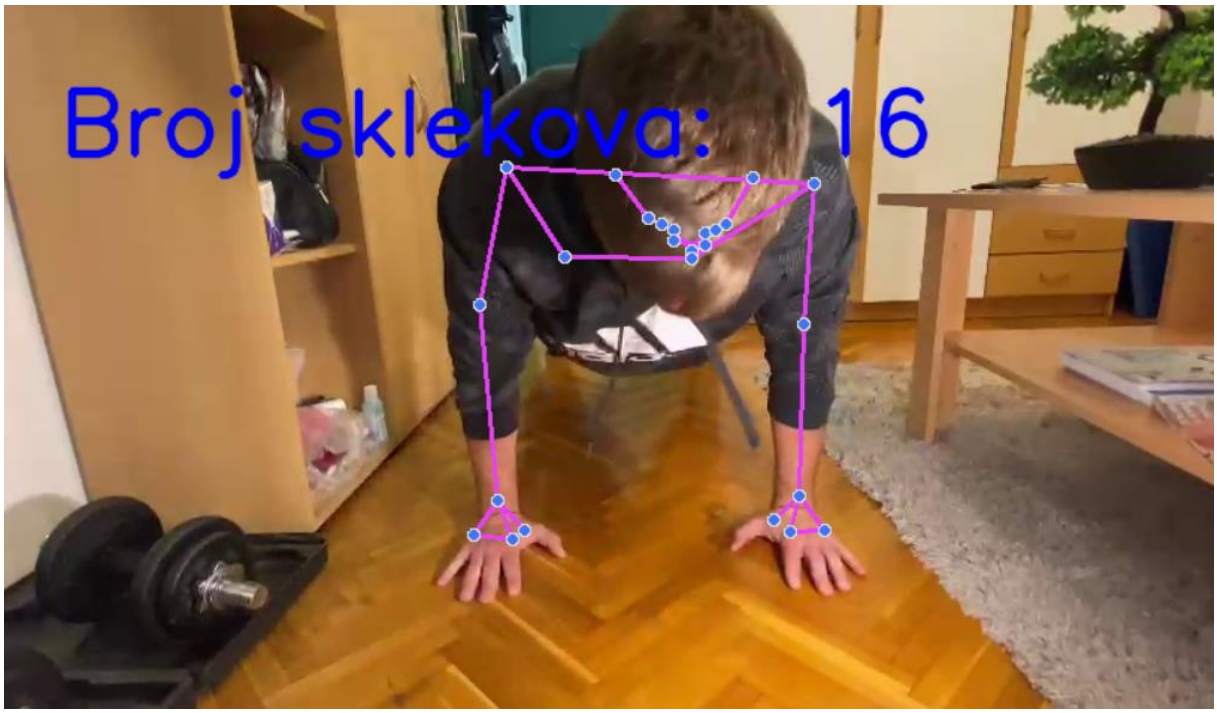
Slika 5.12 Drugi primjer - vizualni prikaz donjeg položaja.

Kada se osoba podigla u gornji položaj i završila osmi sklek brojač je dodao još jedan broj, kao što vidimo na slici [Slika 5.13].



Slika 5.13 Drugi primjer - vizualni prikaz gornjeg položaja.

Ukupan broj sklekova je 16 i model je prebrojao također 16, kao što je i prikazano na slici [Slika 5.7]. Opet je dokazano da je model točno prebrojao.



Slika 5.14 Drugi primjer - ukupan broj sklekova.

5.3. Treći primjer

U ovom primjeru učitan je videozapis 'Bruno_22.mp4'.

Na slici [Slika 5.15] prikazane su koordinate ključnih točaka u donjem položaju, te na slici [Slika 5.16] kut koji zatvaraju ruke.

```
[0.6274621486663818, 0.5146265625953674] [0.7118493318557739, 0.4736971855163574] [0.7385122179985046, 0.6930973529815674]
```

Slika 5.15 Treći primjer - koordinate ključnih točaka u donjem položaju.

```
65.93645398702078
```

Slika 5.16 Treći primjer - kut u donjem položaju.

Na slici [Slika 5.17] prikazane su koordinate ključnih točaka u gornjem položaju, te na slici [Slika 5.18] kut koji zatvaraju ruke.

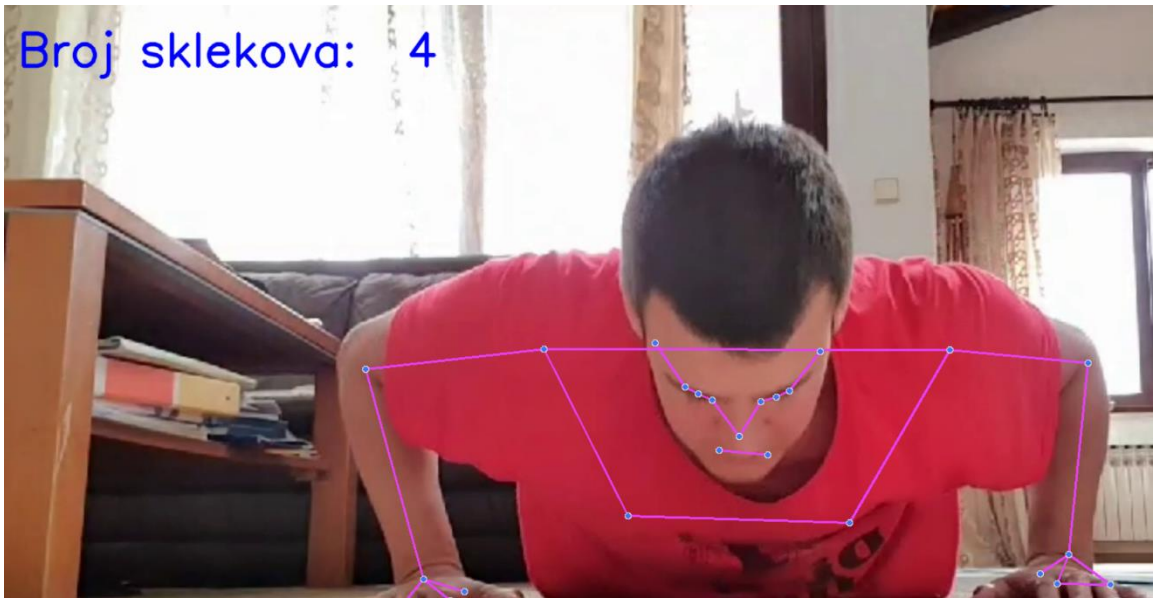
[0.6250985860824585, 0.12687958776950836] [0.6968260407447815, 0.4112307131290436] [0.7352133989334106, 0.6999784708023071]

Slika 5.17 Treći primjer - koordinate ključnih točaka u gornjem položaju.

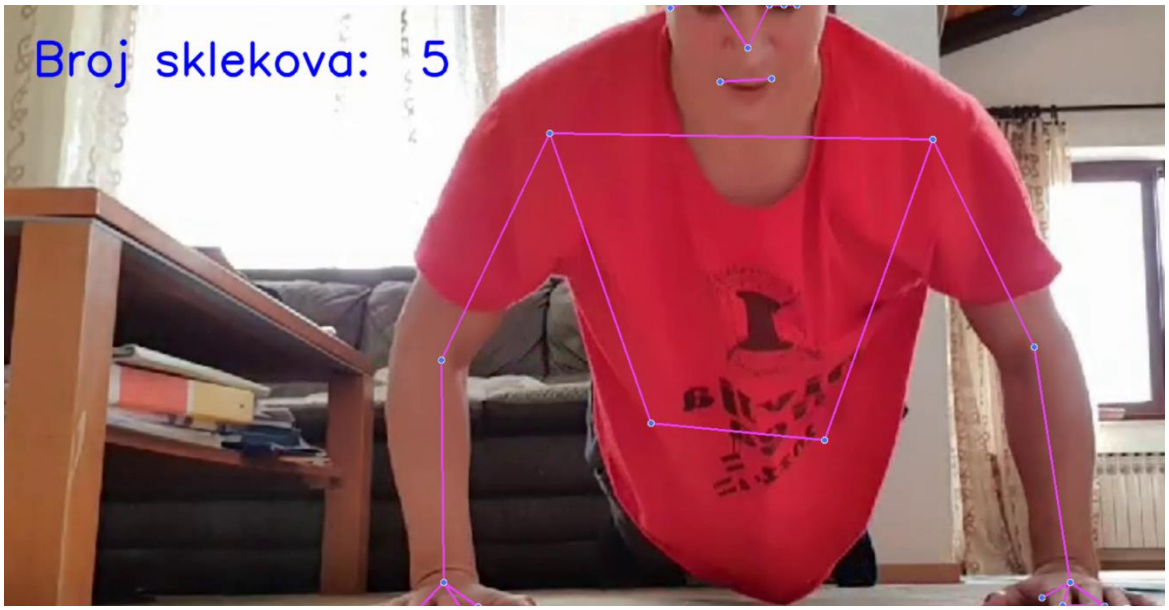
173.4152503999028

Slika 5.18 Treći primjer - kut u gornjem položaju.

Donji položaj kada osoba krene raditi četvrti sklek je prikazan na slici [Slika 5.19], a gornji položaj kada završi četvrti sklek na slici [Slika 5.20].

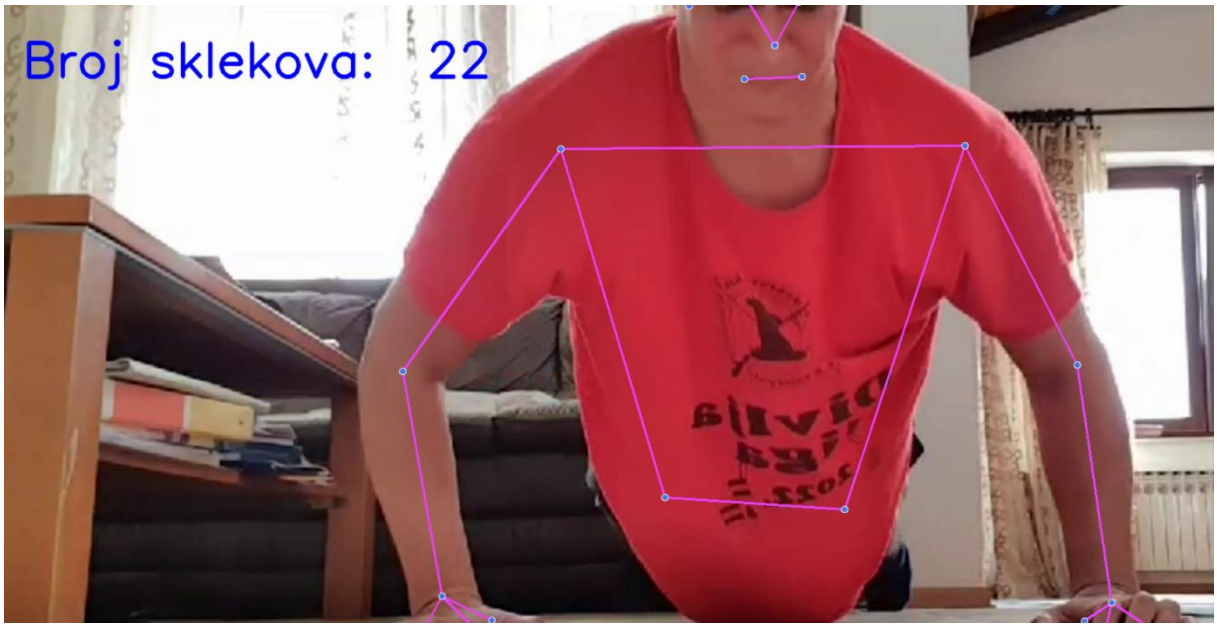


Slika 5.19 Treći primjer - vizualni prikaz donjeg položaja.



Slika 5.20 Treći primjer - vizualni prikaz gornjeg položaja.

Ukupan broj sklekova je 22, kao što je i prikazano na slici [Slika 5.21]. Opet je dokazano da je model točno prebrojao.



Slika 5.21 Treći primjer - ukupan broj sklekova.

Ovim primjerom je završena evaluacija modela jer smo dokazali da u ovom položaju tijela, kada je kamera postavljena ispred lica i u kadru se nalaze ramena, laktovi i ručni zglobovi, model točno broji sklekove.

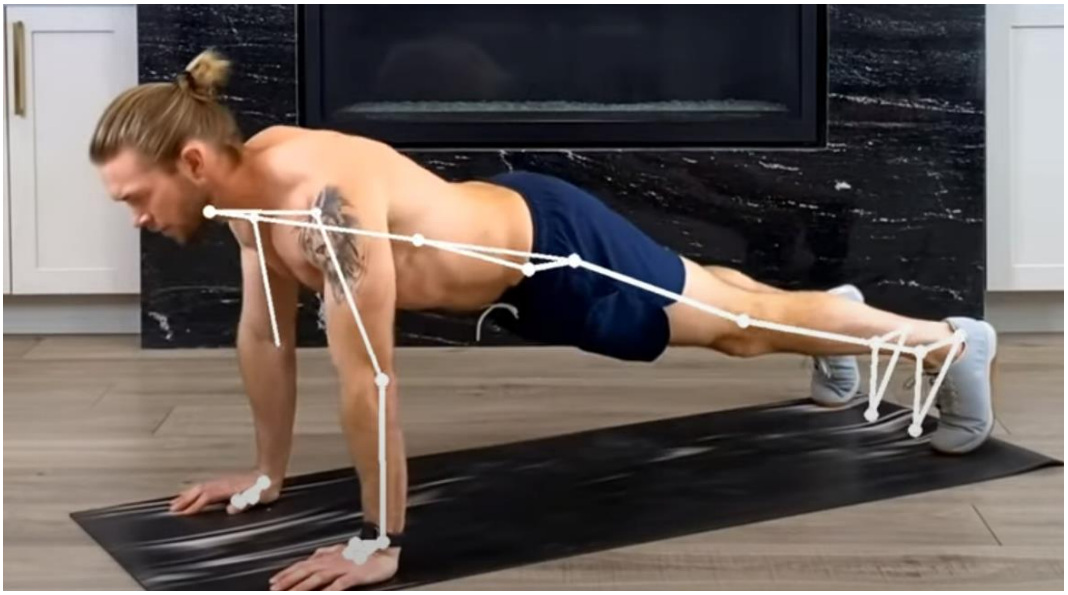
6. KRITIČKI OSVRT NA RAD MODELA

Rezultatima iz prethodnog poglavlja možemo zaključiti da ovaj računalni model ispravno radi. U dnevnim uvjetima dobro prepoznaje i prati sve ključne točke koje se pomiču. Model može prepoznati položaj osobe neovisno o udaljenosti od kamere, jedino što može smetati je ako se u kadru nalazi neka druga osoba ili živo biće. Zbog toga je bolje da se osoba nalazi bliže kameri te da oko nje nema stvari koje bi mogle ometati model.

Svjetlost utječe na rad modela. Iako model dobro prepoznaje ključne točke i u mraku, ipak neće raditi stopostotno točno. Bitno je da prostorija bude osvijetljena tako da smo sigurni u rezultate modela.

Model također jako dobro radi neovisno o rezoluciji slike. Naravno da ako je slika jako mutna, neće moći prepoznati gibanje svake točke i pomicanje poze. Ali kada uzmemo u obzir kamere mobitela i ostalih uređaja koji se danas koriste za snimanje, model će ispravno raditi.

Zadovoljna sam ovim računalnim modelom, ali uvijek može biti poboljšan. Ovim modelom prepoznaje se sklek samo po položaju ramena i ruka, te njihovim pomicanjem. Kako bi mogli procijeniti ispravnost položaja tijela dok osoba radi sklek, u model bi mogli uključiti još ključnih točaka tijela. Kao što je prikazano na slici [Slika 6.1], model bi prepoznavao ključne točke laktova, ramena, kukova, koljena te gležnjeva. Na osnovu njih bi modificirali kod koji bi prebrojavao sklekove samo ako su skroz ispravno odrađeni. Moj cilj je bio izraditi model koje će prepoznati i „muški“ i „ženski“ sklek, neovisno o položaju koljena.



Slika 6.1 Prikaz ključnih točaka na tijelu u drukčijem računalnom modelu. [34]

7. ZAKLJUČAK

U ovom radu je prikazana izrada računalnog modela za brojanje sklekova. Opisana je teorijska osnova potrebna za razumijevanje izrade modela. Model je izrađen na temelju Python biblioteka NumPy, OpenCV, te MediaPipe koje su pojednostavnile izradu. Računalni model je napravljen na način da osoba mora biti dovoljno udaljena od kamere da bi se vidjela njena ramena, laktovi i ručni zglobovi. Iz biblioteke MediaPipe su učitane 33 ključne točke na tijelu te na slici videozapisa prikazane točke lijevog ramena, lakta i zgloba povezane linijama. Pomoću izračunatog kuta u zglobu lakta, napisana je funkcija koja određuje kada je tijelo u donjem ili gornjem položaju skleka. Pomoću te funkcije brojač prebrojava sklekove.

Model je testiran na tri primjera, te na sva tri izbroji točan broj sklekova. Uz promjenu par linija kôda, na primjer odaberemo druge ključne točke, ovaj model se može primijeniti i na druge tjelovježbe. Dinamički prati te točke kroz pokret u stvarnom vremenu. Može se primijeniti na prethodno snimljenom videozapisu, ali i na kameri uživo.

Model se može unaprijediti i stvaranjem aplikacije koja bi brojala sklekove i pratila ostale tjelovježbe. Pomogao bi osobi u lakšem praćenju napretka. Za unaprjeđenje je bitno paziti na kvalitetu ulazne slike (svjetlost, izoštrenost, kutovi kamere, prepreke ispred kamere) te varijabilnost položaja tijela kako bi model mogao funkcionirati u svim slučajevima. Potrebno je model testirati na što većem broju primjera koji sadrže razne varijante položaja tijela i uvjeta oko osobe u okviru slike.

LITERATURA

[1] Machine learning, explained, https://mitsloan.mit.edu/ideas-made-to-matter/machine-learning-explained, dostupno dana 9.listopada 2023.

[2] Types of machine learning with algorithms classification outline diagram, https://www.dreamstime.com/types-machine-learning-algorithms-classification-outline diagram-types-machine-learning-algorithms-classification-image260201316, dostupno dana 10.listopada 2023.

[3] Ralf Herbrich, „Learning Kernel Classifiers“, MIT Press, 7 Dec. 2001.

[4] Deng, Li, and Dong Yu, „Deep Learning : Methods and Applications“, Boston, Now Publishers Inc, 2014.

[5] Bengio, Yoshua; Courville, Aaron; Goodfellow, Ian J, „Deep Learning“, Cambridge, Massachusetts, The Mit Press, 2016.

[6] DARPA Neural Network Study (U.S.), „Artificial neural networks technology“, Afcea Intl Pr, Year: 1988

[7] Tačković, K., Nikolovski, S., „Kratkoročno prognoziranje opterećenja, Energija“, god. 57(2008), No. 5, pp. 560-579

[8] Goel Akash, Kumar Goel Amit, Kumar Adesh, “The Role of Artificial Neural Network and Machine Learning in Utilizing Spatial Information.” Spatial Information Research, vol.31, 18 Nov. 2022,

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

[10] Mislav Larva, „Vrednovanje raspoznavanja znamenki i slova konvolucijskim neuronskim mrežama“, Zagreb: Fakultet elektrotehnike i računarstva, 2015.

[11] Salman Khan, Hossein Rahmani, Syed Afaq Ali Shah, Mohammed Bennamoun, „A Guide to Convolutional Neural Networks for Computer Vision“, Morgan & Claypool, Year: 2018

[12] Yakura, Hiromu; Shinozaki, Shinnosuke; Nishimura, Reon; Oyama, Yoshihiro; Sakuma, Jun, „Malware Analysis of Imaged Binary Samples by Convolutional Neural Network with Attention Mechanism“ 127-134, 2018.

[13] Jiang, X., Lu, M. & Wang, SH. „An eight-layer convolutional neural network with stochastic pooling, batch normalization and dropout for fingerspelling recognition of Chinese sign language“, Multimed Tools Appl 79, 15697–15715 (2020)

[14] H. Sultan, Hossam; Salem, Nancy & Al-Atabany, Walid. (2019), “Multi-Classification of

Brain Tumor Images Using Deep Neural Network“, IEEE Access, 2019 PP. 1-1. 10.1109/ACCESS.2019.2919122.

[15] Introduction to Softmax for Neural Network, https://www.analyticsvidhya.com/blog/2021/04/introduction-to-softmax-for-neural-network/ , dostupno dana 23. listopada 2023.

[16] Convolutional Neural Networks (CNN): Step 4 - Full Connection, https://www.superdatascience.com/blogs/convolutional-neural-networks-cnn-step-4-full-connection, dostupno dana 23. listopada 2023.

[17] N. Sebe, Ira Cohen, Ashutosh Garg, Thomas S. Huang, „Machine Learning in Computer Vision“, Springer Netherlands, Year: 2005

[18] What is computer vision and how does it work with artificial intelligence?, https://www.algotive.ai/blog/what-is-computer-vision-and-how-does-it-work-with-artificial-intelligence, dostupno dana 24. listopada 2023.

[19] Aria Salari, Abtin Djavadifar, Xiangrui Liu, Homayoun Najjaran, „Object recognition datasets and challenges: A review“, Neurocomputing, Volume 495, 2022, Pages 129-152

[20] Yali Amit, „2D object detection and recognition: models, algorithms and networks“, The MIT Press, Year: 2002

[21] A Gentle Introduction to Object Recognition With Deep Learning, https://machinelearningmastery.com/object-recognition-with-deep-learning/, dostupno dana 26. listopada 2023.

[22] Object Detection vs. Image Classification vs. Keypoint Detection, https://blog.roboflow.com/object-detection-vs-image-classification-vs-keypoint-detection/, dostupno dana 26. listopada 2023.

[23] Miniar Ben Gamra, Moulay A. Akhloufi, „A review of deep learning techniques for 2D and 3D human pose estimation“ Image and Vision Computing, Volume 114, 2021

[24] AI Pose Estimation and Analysis Software Development, https://usmsystems.com/human-pose-estimation-and-analysis-software-development/, dostupno dana 28. listopada 2023.

[25] Using Human Pose Estimation in Fitness & Rehab Therapy Apps, https://mobidev.biz/blog/human-pose-estimation-technology-guide, dostupno dana 30. listopada 2023.

[26] What Is Python?, https://builtin.com/software-engineering-perspectives/python, dostupno dana 31. listopada 2023.

[27] What is Python Coding?, https://junilearning.com/blog/guide/what-is-python-101-for-students/, dostupno dana 31. listopada 2023.

[28] Ivan Idris, „NumPy 1.5 Beginner's Guide“, Packt Publishing, Year: 2011

[29] About OpenCV, https://opencv.org/about/, dostupno dana 1. studenog 2023.

[30] A Review of Google’s New Mobile-Friendly AI Framework: Mediapipe, https://medium.com/swlh/a-review-of-googles-new-mobile-friendly-ai-framework-mediapipe-25d62cd482a1, dostupno dana 1. studenog 2023.

[31] MediaPipe Solutions guide, https://developers.google.com/mediapipe/solutions/guide, dostupno dana 1. studenog 2023.

[32] Pose landmark detection guide, https://developers.google.com/mediapipe/solutions/vision/pose_landmarker, dostupno dana 3. studenog 2023.

[33] Valentin Bazarevsky, Ivan Grishchenko, Karthik Raveendran, Tyler Zhu, Fan Zhang, Matthias Grundmann, „BlazePose: On-device Real-time Body Pose tracking“, 2020

[34] Realtime Push Up Counter using Python and Mediapipe, https://www.youtube.com/watch?v=KCC4QmzPoGg&ab_channel=WhatTheAI, dostupno dana 22. studenog 2023.

PRILOZI

I. Python kod

I. Python kod

```
import cv2
import mediapipe as mp
import numpy as np
mp_drawing = mp.solutions.drawing_utils
mp_pose = mp.solutions.pose

count = 0

cap = cv2.VideoCapture('Bruno_22.mp4')

def calculate_angle(a,b,c):
            a = np.array(a)
            b = np.array(b)
            c = np.array(c)

            radians = np.arctan2(c[1]-b[1], c[0]-b[0]) - np.arctan2(a[1]-b[1], a[0]-b[0])
            angle = np.abs(radians*180.0/np.pi)

            if angle >180.0:
                  angle = 360-angle
            return angle


with mp_pose.Pose(min_detection_confidence=0.5, min_tracking_confidence=0.5) as pose:
    while cap.isOpened():
        ret, frame = cap.read()


        image = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        image.flags.writeable = False
```

```
 results = pose.process(image)


 image.flags.writeable = True
 image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)

 try:
     landmarks = results.pose_landmarks.landmark
     print(landmarks)



     landmarks[mp_pose.PoseLandmark.LEFT_SHOULDER.value].visibility
     landmarks[mp_pose.PoseLandmark.LEFT_ELBOW.value]
     landmarks[mp_pose.PoseLandmark.LEFT_WRIST.value]




     shoulder = [landmarks[mp_pose.PoseLandmark.LEFT_SHOULDER.value].x,
                 landmarks[mp_pose.PoseLandmark.LEFT_SHOULDER.value].y]
     elbow = [landmarks[mp_pose.PoseLandmark.LEFT_ELBOW.value].x,
              landmarks[mp_pose.PoseLandmark.LEFT_ELBOW.value].y]
     wrist = [landmarks[mp_pose.PoseLandmark.LEFT_WRIST.value].x,
              landmarks[mp_pose.PoseLandmark.LEFT_WRIST.value].y]


     shoulder, elbow, wrist
```

```
        angle=calculate_angle(shoulder, elbow, wrist)


        if angle<=105.0:
            position="down"
        if angle>=145.0 and position=="down":
            position="up"
            count+=1
            print(count)



        org = (50, 100)
        fontScale = 2
        color = (255, 0, 0)
        thickness = 3
        font = cv2.FONT_HERSHEY_SIMPLEX


        image = cv2.putText(image,"Broj sklekova:  " + str(count),
                            org, font, fontScale, color, thickness, cv2.LINE_AA)



    except:
        pass
```

```
          mp_drawing.draw_landmarks(image, results.pose_landmarks,
                                    mp_pose.POSE_CONNECTIONS,
                                   mp_drawing.DrawingSpec(color=(245,117,66),
                                                          thickness=2, circle_radius=2),
                                   mp_drawing.DrawingSpec(color=(245,66,230),
                                                          thickness=2, circle_radius=2)
                                    )


      cv2.imshow('Prebrojavanje sklekova', image)

      if cv2.waitKey(10) & 0xFF == ord('q'):
          break

cap.release()
cv2.destroyAllWindows()
```