Primjena prijenosnog učenja za prilagodbu metoda računalnog vida blisko-infracrvenim kamerama

Kožul, Mihael

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 756

PRIMJENA PRIJENOSNOG UČENJA ZA PRILAGODBU METODA RAČUNALNOG VIDA BLISKO-INFRACRVENIM KAMERAMA

Mihael Kožul

Zagreb, veljača 2025.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 756

PRIMJENA PRIJENOSNOG UČENJA ZA PRILAGODBU METODA RAČUNALNOG VIDA BLISKO-INFRACRVENIM KAMERAMA

Mihael Kožul

Zagreb, veljača 2025.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 30. rujna 2024.

DIPLOMSKI ZADATAK br. 756

Pristupnik:	**Mihael Kožul (0016141878)**
Studij:	Računarstvo
Profil:	Znanost o podacima
Mentor:	izv. prof. dr. sc. Tomislav Petković

Zadatak: **Primjena prijenosnog učenja za prilagodbu metoda računalnog vida blisko-infracrvenim kamerama**

Opis zadatka:

Model naučen na nekom skupu primjera za rješavanje nekog problema se primjenom prijenosnog učenja može prilagoditi drugom srodnom problemu za kojeg ne postoji dovoljno podataka. Jedan od problema računalnog vida jest praćenje ljudskog pokreta u videu za što postoji više metoda koja pretpostavljaju video snimljen u vidljivom spektru (RGB). No profesionalni sustavi za praćenje pokreta često koriste kamere koje rade u blisko-infracrvenom području (NIR) pa je za njih potrebno prilagoditi postojeće metode za praćenje ljudskog pokreta. U diplomskom radu je potrebno dati kratki pregled metoda prijenosnog učenja i metoda praćenja ljudskog pokreta, te opisati razliku između kamera koje snimaju u vidljivom i u blisko-infracrvenom spektru. Zatim je potrebno odabrati metodu prijenosnog učenja koja je prikladna za prilagodbu postojećih metoda praćenja ljudskog pokreta u videu za rad s blisko-infracrvenim kamerama te je istu potrebno razraditi na konkretnom primjeru sustava OpenPose i OptiTrack koji su dostupni u fakultetskom laboratoriju. U fakultetskom laboratoriju je potrebno snimiti dovoljno video snimki potrebnih za prijenosno učenje, a osim toga potrebno je predložiti transformaciju koja RGB slike ili video približno pretvara u NRI slike ili video. Naposlijetku, potrebno je evaluirati uspješnost prijenosnog učenja za sve kombinacije korištenja NIR video snimki i predložene transformacije RGB u NIR.

Rok za predaju rada: 14. veljače 2025.

Prije svega, htio bih se zahvaliti svome mentoru izv. prof. dr. sc. Tomislavu Petkoviću bez kojega ovaj diplomski rad ne bi postojao. Hvala na pomoći pri smišljanju teme rada, hvala na savjetima te ukazanoj potpori i strpljenju.

Ponajviše, hvala svim članovima moje obitelji. Najveću zaslugu za ono što sam postigao i što ću tek postići pripisujem njima. Hvala im na neizmjernoj i nesebičnoj potpori u apsolutno svakom trenutku mojega života. Posebno hvala mojim roditeljima što su mi dali sve, a moje je bilo samo da učim. Hvala i na uzrečicama "Ne moram ja ništa, mora se samo umrijeti" i "Ako nisu plan A i B, bit će plan C i D" kojima sam se vodio, kojima se vodim i bez kojih bi moje školovanje i život bili znatno kompliciraniji.

Velika hvala svima, Mihael

Sadržaj

1. Uvod

U posljednjih nekoliko godina razvoj umjetne inteligencije i njenih poddomena ostvario je značajan napredak. Zbog toga se, uz umjetnu inteligenciju, mogu sve češće čuti pomodne riječi (engl. *buzzwords*) poput strojno učenje (engl. *machine learning*), neuronske mreže (engl. *neural networks*), duboko učenje (engl. *deep learning*) i prijenosno učenje (engl. *transfer learning*). Iako je razvoj ovih tehnologija započeo još sredinom prošlog stoljeća, značajniji iskorak i primjena, potaknuti primarno prepoznavanjem njihovog potencijala i razvojem računalnih resursa, postignuti su tek u posljednjem desetljeću.

Duboko učenje [1], odnosno duboka neuronska mreža, na valu nove popularnosti postaje zadana (engl. *default*) metoda u domeni strojnog učenja zbog svojih prednosti u pristupu složenijim problemima. Zato što je neke probleme jako teško u potpunosti definirati, prikazati sva pravila i programski ostvariti sve zahtjeve, koriste se duboke neuronske mreže kao metoda koja uči složene značajke iz velike količine podataka. Neki od tih problema su upravo prepoznavanje slika, prepoznavanje govora, obrada prirodnog jezika i analiza velikih skupova podataka [2, 3]. Nažalost, za njihovo učenje potrebne su ogromne količine podataka, velika računalna snaga i poprilična količina vremena što otežava njihovo istraživanje i razvoj [2].

Prijenosno učenje [4], kao što je prikazano u nastavku rada, pristup je strojnom učenju koji upravo pokušava riješiti navedene probleme. Prijenosno učenje koristi znanje modela nastalog iz jednog skupa podataka (npr. prepoznavanje slika) te ga primjenjuje na određenu drugu domenu (npr. radiološka dijagnostika). Ovakav pristup omogućuje smanjenje troškova učenja novih modela i skraćuje vrijeme razvoja, što je ključno u slučajevima gdje podaci nisu lako dostupni. Osim bržeg učenja, često se ostvaruje i veća efikasnost, bolja generalizacija te mogućnost učenja novog modela nad manjim skupom podataka. Međutim, treba uzeti u obzir i probleme među kojima su izbor prijenosnog

modela, nepoznata domena te problemi nastali tijekom prilagodbe. Usprkos tome, smatra se kako će uz daljnje razvijanje prijenosno učenje postati sve važniji alat u znanosti i industriji . Različite industrije, poput medicinske dijagnostike, autonomne vožnje i računalne sigurnosti već bilježe značajne koristi od primjene prijenosnog učenja što dodatno potvrđuje njegovu praktičnu vrijednost [5].

Jedna je od aktualnih tema u računalnom vidu praćenje ljudskog pokreta u videu, koje se najčešće temelji na videima snimljenim u vidljivom spektru ostvaren tzv. RGB kamerom. Profesionalni sustavi, poput onih korištenih u analizi biomehanike, virtualnoj stvarnosti ili jednostavno videonadzora, često se oslanjaju na blisko-infracrvene (engl. *near-infrared, NIR*) kamere zbog njihovih prednosti u specifičnim uvjetima [6, 7]. Kamere koje snimaju u NIR spektru bolje funkcioniraju u slabijim uvjetima osvjetljenja, imaju manju osjetljivost na sjene i refleksije te omogućuju preciznije praćenje u kontroliranim uvjetima, što ih čini idealnima u određenim domenama poput biomehaničke analize pri istraživanju ljudskog pokreta.

U ovom radu istražuje se mogućnost primjene prijenosnog učenja za prilagodbu metoda praćenja pokreta razvijenih na RGB kamerama da bi bile funkcionalne s NIR kamerama. Uz pregled metoda dubokog učenja, prijenosnog učenja te alata za praćenje pokreta, fokus je na prilagodbi sustava OpenPose [8]. Da bi se omogućila prilagodba, izrađeni su skupovi podataka korištenjem NIR kamera, a provedene su i transformacije RGB slika u NIR format. Pritom su testirana dva pristupa: simulacija NIR spektra korištenjem samo crvenog kanala te izračun težinske kombinacije RGB kanala. Na temelju kreiranih skupova podataka evaluirane su razlike u učinkovitosti modela pri korištenju izvornih NIR podataka u usporedbi s transformiranim RGB podacima. Cilj istraživanja je ne samo prilagoditi postojeće metode, već i procijeniti sposobnost modela da generalizira znanje stečeno na transformiranim podacima. Na taj način se istražuju mogućnosti uštede resursa i vremena u situacijama kada pravi NIR podaci nisu dostupni. Zaključci ovog istraživanja mogu poslužiti kao osnova za primjenu u srodnim domenama gdje je potrebna prilagodba metoda između različitih spektralnih ili modalnih podataka poput RGB i termalnih slika.

2. Prijenosno učenje i računalni vid

Razvoj umjetne inteligencije započeo je sredinom 20. stoljeća kada su Warren McCulloch i Walter Pitts predložili model umjetnog neurona koji je temeljen na jednostavnom binarnom sustavu stanja – uključen (engl. *on*) i isključen (engl. *off*) [9, str. 35]. Taj model, poznat kao perceptron, pokazao je da mreža povezanih perceptrona može izračunati bilo koju izračunljivu funkciju [10], što je postavilo temelje za današnje modele. Tijekom povijesti, područje umjetne inteligencije doživjelo je brojne uspone i padove, no kontinuirano je ostvarivalo napredak.

Umjetna inteligencija često se poistovjećuje s pojmovima poput strojnog učenja, neuronskih mreža i dubokog učenja, iako među njima postoje ključne razlike [11]. Umjetna je inteligencija širi pojam, a strojno učenje je njezina poddomena koja se bavi algoritmima sposobnim za samostalno učenje [12]. Neuronske mreže, inspirirane strukturom ljudskog mozga, dio su strojnih algoritama. S druge strane, duboko učenje predstavlja složenije mreže s više skrivenih slojeva što im omogućava rješavanje mnogo zahtjevnijih problema.

Svaki algoritam strojnog učenja može se podijeliti na tri ključne komponente. Prikazane jednadžbom one glase: **učenje = reprezentacija + evaluacija + optimizacija**. Drugim riječima, svaki algoritam temelji se na sljedećim elementima [13]:

- **Model** - skup hipoteza, odnosno funkcija parametriziranih vektorom θ
- **Funkcija gubitka** - kvantificirana razlika između predikcija modela i stvarnih rezultata
- **Optimizacijski postupak** - metoda za minimizaciju funkcije gubitka radi pronalaska optimalne hipoteze h^*.

Postoje tri glavna pristupa strojnome učenju: nadzirano (engl. *supervised*), nenadzirano (engl. *unsupervised*) i podržano (engl. *reinforcement*) učenje [9, str. 671].

Nadzirano učenje koristi označene podatke (engl. *labeled data*) za treniranje modela, pri čemu model uči povezivati ulazne podatke s njihovim odgovarajućim kategorijama [14]. S druge strane, nenadzirano učenje analizira neoznačene podatke da bi otkrilo skrivene obrasce ili strukture u podacima [14]. Podržano učenje inspirirano je bihevioristič-kom psihologijom i temelji se na sustavu nagrada i kazni da bi model učio uz pomoć sekvencijalnih odluka [15, str. 71].

Računalni vid polje je umjetne inteligencije koje omogućuje sustavima izvlačenje značajnih informacija iz digitalnih slika, videa i drugih vizualnih ulaza (engl. *input*) [16, str. 3-10]. Njegove primjene su brojne i uključuju detekciju bolesti u medicini, autonomnu vožnju, analizu pokreta (engl. *pose estimation*) i druge. S obzirom na temu rada posebno je značajan posljednji primjer - primjena računalnog vida u praćenju ljudskog kretanja.

Unatoč velikim mogućnostima računalni vid suočava se s nizom izazova. Izrada učinkovitih modela zahtijeva ogromne računalne resurse, mnogo vremena za treniranje te velik broj označenih podataka. Dodatno, složenost modela često ih čini „crnim kutijama", što otežava razumijevanje i pronalaženje grešaka u kodu (engl. *debugging*) [17]. Ključnu ulogu u razvoju računalnog vida odigrale su konvolucijske neuronske mreže (engl. *Convolutional Neural Networks, CNN*), koje su se pokazale iznimno učinkovitima u analizi vizualnih podataka zahvaljujući svojoj sposobnosti automatskog izvlačenja značajki iz slika [18]. Prijenosno učenje, kao tehnika koja koristi znanje stečeno na jednom zadatku za ubrzavanje učenja na drugom, javlja se kao rješenje za ove probleme. Ova metoda omogućuje učinkovitije korištenje resursa i smanjuje potrebu za velikim količinama novih podataka, čime se olakšava implementacija u raznim primjenama računalnog vida.

2.1. Osnove konvolucijskih neuronskih mreža

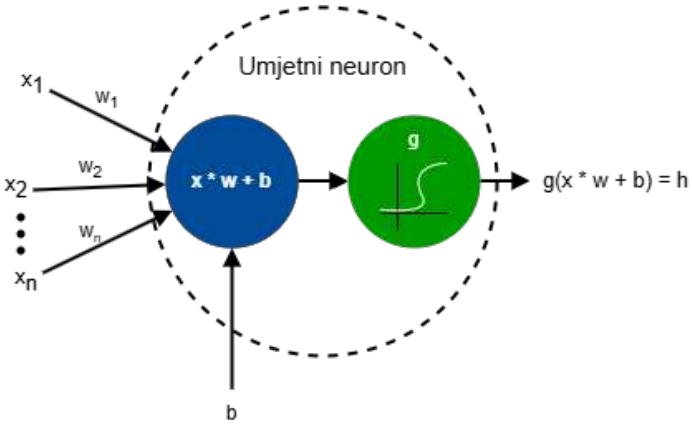
Tipična neuronska mreža sastoji se od ulaznog sloja (engl. *input layer*), jednog ili više skrivenih slojeva (engl. *hidden layer*) te izlaznog sloja (engl. *output layer*). Ulazni sloj

predstavlja sloj podataka nad kojima se vrši proces učenja mreže, a skriveni slojevi i izlazni sloj sastoje se od skupine neurona (perceptrona) koji su međusobno povezani. Jačina njihova povezanosti određena je težinama (engl. *weights*) te pristranosti (engl. *bias*), koje se prilagođavaju tijekom procesa učenja. Nakon izračuna linearne kombinacije ulaznih podataka i njima pripadajućih težina, izračun prolazi kroz tzv. aktivacijsku funkciju (engl. *activation function*) koja uvodi nelinearnost u mrežu [1, str. 172]. Jedna od najpoznatijih aktivacijskih funkcija, te ona koja se koristi u modelu OpenPose [19], jest funkcija ReLU (engl. *Rectified Linear Unit*). Aktivacijska funkcija ReLU koristi se često zbog svoje jednostavnosti i brzine učenja te u isto vrijeme sprječava problem zasićenja, što zna biti problem kod nekih drugih aktivacijskih funkcija, npr. sigmoidalne funkcije [20, str. 400-401]. Funkcija ReLU može se prikazati kao:

$$h_j = g(z_j) = \max(0, z_j), \tag{2.1}$$

gdje z_j predstavlja izlaz samog neurona prije primjene aktivacijske funkcije vidljive na slici 2.1. Ta funkcija definira se na sljedeći način:

$$z_j = \sum_{i=1}^{p} x_i w_{ij} + b_j. \tag{2.2}$$



Slika 2.1. Struktura umjetnog neurona; prema [15, str. 59]

Jednadžba 2.2 izračunava linearnu kombinaciju ulaznih vrijednosti x_i i pripadajućih težina w_{ij}, uz dodatak pomaka b_j. Vrijednosti i predstavljaju ulazne podatke, dok vrijednosti j predstavljaju neuron unutar kojeg se vrše kalkulacije. Mnogo češći prikaz formula u dubokom učenju je matrični prikaz zbog jednostavnosti, ali i praktičnosti u

prikazu i kalkulacijama. Stoga, uzimajući u obzir definirane formule za aktivacijsku funkciju ReLU te linearnu kombinaciju ulaznih vrijednosti u neuron, funkcija koja definira konačni model glasi [1, str. 171-173]:

$$\mathbf{h}(\mathbf{X}; \mathbf{W}, \mathbf{b}) = g(\mathbf{XW} + \mathbf{b}). \tag{2.3}$$

Konvolucijske neuronske mreže okosnica su razvoja modela računalnog vida i temelj su nad kojim je izgrađen model OpenPose [19] objašnjen u potpoglavlju *2.3.1. OpenPose*. Konvolucijske mreže definiraju se kao neuronske mreže koje koriste specijaliziranu vrstu linearne operacije, tzv. konvoluciju, na mjestu generalne matrice množenja u najmanje jednome svom sloju. Konvolucijski slojevi donose ključne prednosti poput smanjenja broja parametara zahvaljujući rijetkim interakcijama (engl. *sparse interactions*) i dijeljenju parametara (engl. *parameter sharing*), očuvanja prostorne strukture pomoću translacijske ekvivarijantnosti (engl. *equivariant representations*) te mogućnosti obrade ulaza različitih veličina, što ih čini učinkovitim i fleksibilnim za složene zadatke poput prepoznavanja slika i govora [1, str. 326-330].

Prvi argument konvolucijske operacije (u ovom slučaju x) zove se ulaz (engl. *input*), drugi se argument (u ovom slučaju funkcija w) zove jezgra (engl. *kernel*), a izlaz iz operacije zove se mapa značajki (engl. *feature map*). Ako na taj način definiramo operaciju konvolucije te argumente s prethodno opisanom terminologijom, onda se diskretna konvolucija u obliku matematičke funkcije definira kao [1, str. 328]:

$$s(t) = (x * w)(t) = \sum_{a=-\infty}^{\infty} x(a)w(t-a). \tag{2.4}$$

U slučaju slike, koja je temelj diplomskog rada, konvolucija se primjenjuje kroz jezgru koji se pomiče preko slike te se za svaki položaj računa skalarni produkt između elemenata filtra i odgovarajućeg dijela slike [1, str. 328]. To se matematički izražava na sljedeći način:

$$S(i,j) = (I * K)(i,j) = \sum_{m}\sum_{n} I(i-m, j-n) \cdot K(m,n), \tag{2.5}$$

gdje je $S(i, j)$ rezultat konvolucije na poziciji (i, j) nad ulaznim podatkom I, pri čemu je K konvolucijska jezgra [1, str. 329]. Slike nad kojima je treniran model OpenPose sastoje se od 3 kanala (RGB), stoga se konvolucija još dodatno provodi za svaki kanal zasebno.

2.2. Prijenosno učenje

Tijekom svoje povijesti strojno učenje postiglo je značajan napredak u mnogim područjima analize podataka te izvlačenja znanja iz njih. Bilo da je to regresija, klasifikacija ili neko drugo područje, problem koji nastaje jest da mnoge metode funkcioniraju samo pod pretpostavkom da se skup podataka za učenje, validiranje i testiranje izvlači iz istog prostora značajki i iste distribucije [4, str. 1]. Kada dođe do promjene u distribuciji, model postaje irelevantan te je potrebno izgraditi novi model iznova. S obzirom na to da je to često vrlo teško i zahtijeva značajnu količinu resursa, stvorila se potreba za prijenosom znanja s jednog na drugi model čime je nastalo prijenosno učenje.

Najšire moguće rečeno, prijenosno učenje pristup je strojnom učenju unutar kojega se pokušava prenijeti znanje modela naučenog na jednoj domeni, na neki drugi model koji se uči na sličnoj domeni. U literaturi se prijenosno učenje zna uspoređivati s adaptacijom domene (engl. *domain adaptation*), odnosno čak i smatrati jednakim pojmovima - istoznačnicama [1, str. 527]. Treba naglasiti kako je ipak češći slučaj u literaturi da su ta dva pojma različita, odnosno da je prijenosno učenje natpojam adaptaciji domene. U nastavku su dane dvije definicije prijenosnog učenja i adaptacije domene unutar kojih je vidljiv njihov međuodnos:

„Prijenosno učenje i adaptacija domene odnose se na situaciju u kojoj se ono što je bilo naučeno u jednoj postavci (npr. distribucija P1) iskorištava za poboljšanje generalizacije u drugoj postavci (recimo, distribucija P2).” ([1, str. 526])

„Ideja o obučavanju na jednom zadatku i zatim korištenju naučenog na drugom naziva se prijenosno učenje, dok se proces modificiranja konačne mreže za namjeravanu primjenu i statistiku naziva adaptacija domene.” ([16, str. 313])

Vidljivo je da je adaptacija domene podskup prijenosnog učenja. Razlika je u tome što je prijenosnom učenju primaran fokus na korištenju prednaučenih modela na jednoj

domeni za rješavanje problema u drugoj srodnoj domeni, a adaptacija domene fokusira se na prilagođavanje prethodno naučenog modela specifičnoj domeni primjene da bi se poboljšale performanse modela. Na primjer, u prijenosnom učenju, model za prepoznavanje objekata treniran na skupu podataka koji sadrži slike pasa može se iskoristiti za prepoznavanje mačaka, bez dodatnog prilagođavanja modela. Iako su psi i mačke različite vrste, dijele mnoge vizualne karakteristike, što omogućuje modelu da generalizira svoje znanje na novu domenu. S druge strane, u adaptaciji domene, isti taj model može se dodatno prilagoditi tako da preciznije prepoznaje određene pasmine pasa ili da bolje funkcionira s NIR kamerama, gdje su vizualne karakteristike slike značajno drugačije od onih u izvornom skupu podataka. Ovdje se model ne koristi samo u novoj domeni, već se i fino podešava kako bi se bolje prilagodio specifičnim uvjetima primjene.

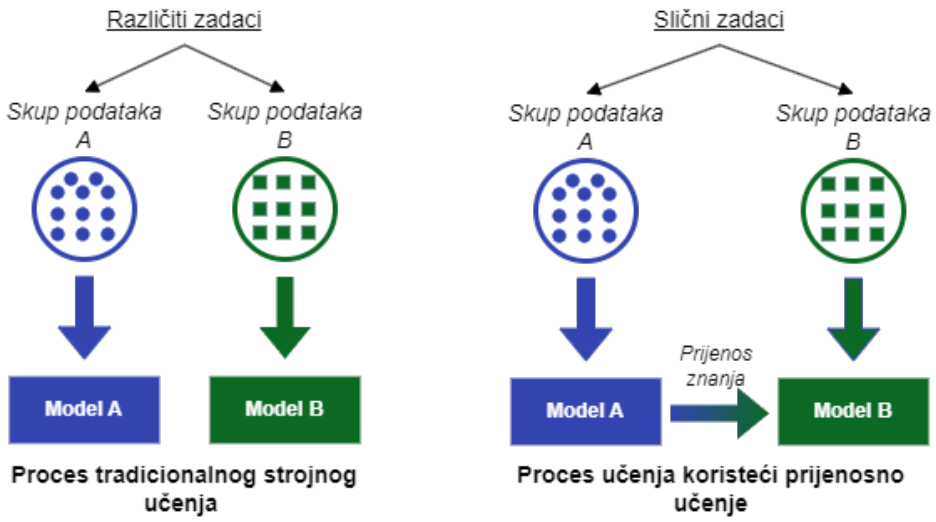
2.2.1. Tradicionalno nasuprot prijenosnom učenju

Tradicionalno strojno učenje skup je metoda učenja unutar strojnog učenja. S obzirom na to da su tradicionalne metode prvi pristupi učenju, često su jednostavnije za shvatiti te su razvijene i široko primijenjene prije samog nastanka prijenosnog učenja, dubokih modela i dr.

Idejna razlika između dva različita pristupa učenju ilustrirana je na slici 2.2. Proces tradicionalnog pristupa učenju izolirani je proces. Svaki novi model koji je učen za određenu svrhu, uči samostalno bez ikakvog predznanja [21]. Moderni pristupi, s druge strane, sastoje se od učenja s kraja-na-kraj (engl. *end-to-end*) uz prijenos znanja (prijenosno učenje). Učenje s kraja-na-kraj tehnika je učenja kod koje model uči sve korake između ulaznih podataka i očekivanog izlaza, a prijenosno učenje koristi znanje modela koji je ranije učen koristeći sličnu domenu.

Na primjer, u prijenosnom učenju, model za prepoznavanje objekata treniran na skupu podataka koji sadrži slike pasa može se prijenosnim učenjem naučiti za prepoznavanje mačaka. Iako su psi i mačke različite vrste, dijele mnoge vizualne karakteristike, što omogućuje iskorištavanje naučenog znanja iz prve domene u novoj domeni. S druge strane, u adaptaciji domene, isti taj model može se dodatno prilagoditi tako da preciznije prepoznaje određene pasmine pasa ili da bolje funkcionira s novim skupom podataka, primjerice onim snimljenim NIR kamerama gdje su vizualne karakteristike slike zna-

čajno drugačije od onih u izvornom skupu podataka.



Slika 2.2. Tradicionalno i prijenosno učenje; prema [4, str. 2]

Veliki je nedostatak tradicionalnih modela strojnog učenja što zahtijevaju modeliranje iznova za svaku novu primjenu, što je za složenije probleme računalno skupo, vremenski zahtjevno te podrazumijeva veliku količinu podataka za postizanje kvalitetnih rezultata. S druge strane, prijenosno učenje znatno je učinkovitije u gotovo svim aspektima u odnosu na tradicionalne modele. Njegova je najveća prednost mnogo manja računalna i vremenska zahtjevnost te postizanje boljih rezultata korištenjem malog skupa podataka [22]. Sama veličina skupa podataka ciljnog modela ovisi o tome koliki se udio znanja (značajke, težine, itd.) izvornog modela može prenijeti, no gotovo je uvijek manja u odnosu na tradicionalne pristupe. Osim toga, modeli nastali prijenosnim učenjem postižu i bolju generalizaciju. Treba naglasiti i da je nekad vrlo teško prikupiti veći skup podataka, stoga učenje iznova ne predstavlja prikladnu opciju [23, str. 2]. Razlog može biti taj da je prikupljanje i/ili označavanje podataka skupo, bilo zbog količine podataka, bilo zbog potrebe za angažiranjem visokokvalificiranih stručnjaka.

2.2.2. Strategije

Različite strategije prijenosnog učenja primjenjuju se s obzirom na domenu primjene i dostupnost skupa podataka. No, prije nego što se odabere strategija, potrebno je odgovoriti na sljedeća pitanja [4, str. 3]:

- **Što** prenijeti,

- **Kako** prenijeti i
- **Kada** prenijeti.

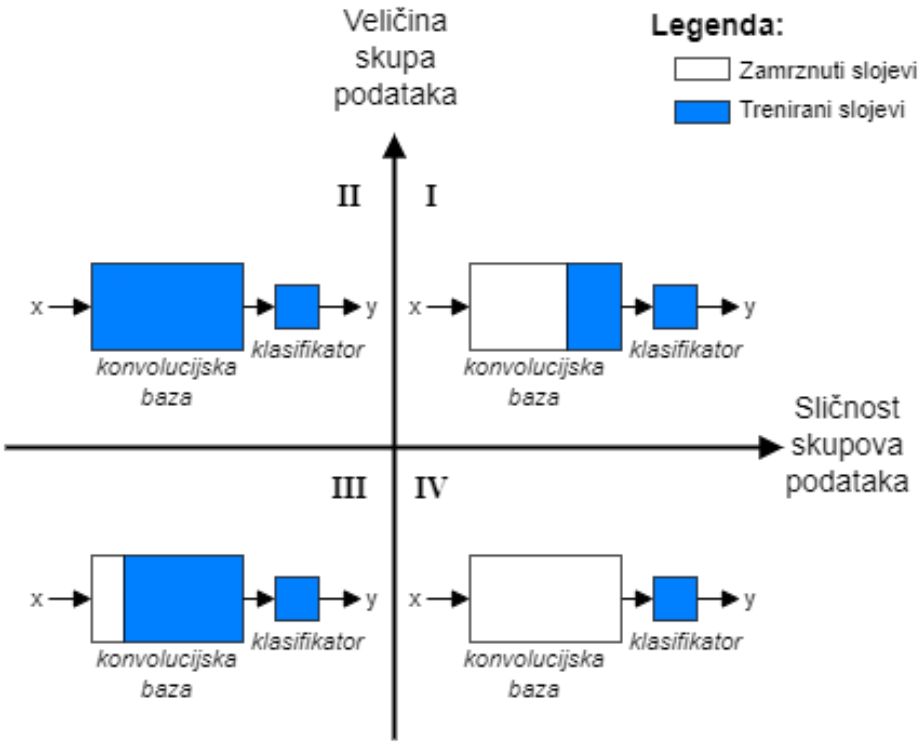
„Što prenijeti?" postavlja pitanje koji dio znanja prednaučenog modela je moguće prenijeti preko domena i zadataka. Potrebno je obratiti pažnju na to da su neka znanja vezana specifično uz pojedinačnu domenu, a neka znanja mogu biti zajednička i tako pomoći u poboljšanju izvedbe ciljne domene [4, str. 3]. Konkretno, postavlja se pitanje koje dijelove modela prenijeti, a to može biti cijeli model, samo određeni konvolucijski slojevi itd. „Kako prenijeti?" postavlja pitanje na koji način prenijeti znanje izvornog modela na temelju ciljne domene [21], odnosno kako prilagoditi model novom problemu. To obuhvaća podešavanje arhitekture, prilagođavanje hiperparametara te procesa učenja, validacije i evaluacije. Posljednje pitanje, „Kada prenijeti?", pokušava odrediti u kojim situacijama se znanje ne bi smjelo prenositi. U određenim situacijama, npr. kada izvorna i ciljna domena nisu povezane, prijenos znanja može ne samo biti neuspješan, nego i rezultirati tzv. negativnim prijenosom (engl. *negative transfer*) [4, str. 2-3]. Negativni je prijenos situacija u kojoj znanje stečeno u izvornoj domeni ometa učenje u ciljnoj domeni što rezultira slabijim performansama modela zbog pokušaja prijenosnog učenja.

Iako je strategije prijenosnog učenja moguće podijeliti na različite načine, tradicionalna podjela dijeli ih na tri kategorije s obzirom na danu izvornu domenu $\mathcal{D}_S$ i pripadajući zadatak učenja $\mathcal{T}_S$, ciljnu domenu $\mathcal{D}_\mathcal{T}$ i pripadajući zadatak učenja $\mathcal{T}_\mathcal{T}$:

- induktivno (engl. *inductive*),
  - Induktivno prijenosno učenje ima za cilj poboljšati učenje ciljane prediktivne funkcije $f_T(\cdot)$ u $\mathcal{D}_\mathcal{T}$ korištenjem znanja iz $\mathcal{D}_S$ i $\mathcal{T}_S$, gdje $\mathcal{T}_S \neq \mathcal{T}_\mathcal{T}$, neovisno vrijedi li $\mathcal{D}_S = \mathcal{D}_\mathcal{T}$ ili $\mathcal{D}_S \neq \mathcal{D}_\mathcal{T}$ [4, str. 5].
- transduktivno (engl. *transductive*) i
  - Transduktivno prijenosno učenje ima za cilj poboljšati učenje ciljne prediktivne funkcije $f_T(\cdot)$ u $\mathcal{D}_\mathcal{T}$ korištenjem znanja iz $\mathcal{D}_S$ i $\mathcal{T}_S$, gdje je $\mathcal{D}_S \neq \mathcal{D}_\mathcal{T}$ i $\mathcal{T}_S = \mathcal{T}_\mathcal{T}$. Osim toga, neki neoznačeni podaci ciljne domene moraju biti dostupni u vrijeme obuke [4, str. 8].

- nenadzirano (engl. *unsupervised*) prijenosno učenje.
  - Nenadzirano prijenosno učenje ima za cilj pomoći poboljšati učenje ciljne prediktivne funkcije $f_T(\cdot)$ u $\mathcal{D}_\mathcal{T}$ koristeći znanje iz $\mathcal{D}_\mathcal{S}$ i $\mathcal{T}_\mathcal{S}$, gdje je $\mathcal{T}_\mathcal{S} \neq \mathcal{T}_\mathcal{T}$ i $\mathcal{Y}_\mathcal{S}$ i $\mathcal{Y}_\mathcal{T}$ nisu vidljivi, odnosno ne postoje dostupne anotirane izlazne vrijednosti u izvornoj i ciljnoj domeni [4, str. 10].

Uzimajući u obzir veličinu skupa podataka te sličnost ciljne domene s izvornom domenom, postoje četiri različita slučaja odabira načina prijenosa znanja području računalnog vida za CNN-ove. Ti slučajevi prikazani su na grafički način na slici 2.3. pri čemu je prikazana pojednostavljena arhitektura modela temeljenog na CNN-u. Konvolucijska baza odgovara tzv. okosnici (engl. *backbone*) modela te predstavlja slojeve za ekstrakciju značajki, a klasifikator odgovara tzv. glavi (engl. *head*) modela i služi za provedbu klasifikacije na temelju izlaza konvolucijske baze [16, str. 304-305].



Slika 2.3. Mapa odluka za odabir načina prijenosa znanja; prema [22]

Prvi slučaj (prvi kvadrant slike 2.3.) odgovara trenutku kada je skup podataka velik, ali i sličan izvornom skupu podataka. U tom slučaju dopuštena je bilo koja opcija, no optimalno bi bilo zamrznuti velik dio izvornog modela te učiti samo vrhove konvolucijske baze i klasifikator [22]. Primjer takvog slučaja može biti prijenos znanja prednaučenog

modela za klasifikaciju slika pasa i mačaka na model koji se uči klasificirati slike divljih životinja.

Drugi slučaj (drugi kvadrant slike 2.3.) odnosi se na trenutak kada je dostupan velik skup podataka, drugačiji u odnosu na skup podataka izvornog modela. U tom slučaju preporučuje se učenje modela iz nule, no i dalje je dobra praksa inicijalizirati model prednaučenim modelom, odnosno preuzeti arhitekturu i težine [22]. Na primjer, kada je prednaučeni model učen za detekciju automobila, za pretpostaviti je da znanje takvog modela nije korisno ako je cilj kreirati model koji se „specijalizira" za radiološku dijagnozu.

Predzadnji slučaj (treći kvadrant slike 2.3.) jest slučaj kod kojega je skup podataka malen i različit od izvornog skupa podataka. U ovom slučaju koristi se strategija slična kao u prethodnom primjeru, no zamrznuti dio trebao bi biti manji, odnosno nije dovoljno učiti samo vrhove konvolucijske baze. Treba uzeti u obzir da ako se zamrzne samo manji dio izvornog modela, riskira se prenaučenost, a ako se zamrzne veći dio kao u prethodnom primjeru, riskira se podnaučenost [22]. Jednostavni primjer je slučaj kada se prednaučeni model za prepoznavanje objekata na fotografijama koristi kao model koji se uči za klasifikaciju vrsta vozila.

Posljednji slučaj (četvrti kvadrant slike 2.3.) je slučaj kada je skup podataka malen i sličan. U ovom kontekstu često je dovoljno samo promijeniti posljednji potpuno povezani sloj, pokrenuti prednaučeni model kao ekstraktor fiksnih značajki, a zatim upotrijebiti rezultirajuće značajke za obuku novog klasifikatora [22]. Primjer za ovu situaciju bio bi prijenos znanja modela koji se koristi za detekciju automobila na model koji se uči za detekciju modela automobila. Vidljivo je kako su domene vrlo slične, stoga je potrebno promijeniti samo klasifikator.

Učenje bez, s jednim ili s malo primjera

Nadovezujući se na prethodne slučajeve, postoji nekoliko pristupa prijenosnog učenja koji su se u određenim domenama pokazali izuzetno učinkovitima. Ljudska sposobnost učenja novih koncepata, posebno tijekom mladosti, često omogućuje usvajanje novih ideja uz minimalan broj primjera ili čak bez njih. Ljudi ne trebaju veliki broj primjera s različitim varijacijama da bi uspostavili poveznicu između naučenog i nepoznatog. Ins-

pirirani ovim prirodnim procesom, razvijeni su koncepti prijenosnog učenja poput učenja s malim brojem primjera (engl. *few-shot learning*), učenja s jednim primjerom (engl. *one-shot learning*) i učenja bez primjera (engl. *zero-shot learning*) [24, str. 177-179]. Ovi pristupi omogućuju modelima generalizaciju na nove zadatke ili klase uz minimalnu količinu podataka.

Literatura često na različite načine klasificira ove vrste prijenosnog učenja, no najčešće se učenje bez primjera, učenje s jednim primjerom, Bayesovsko programsko učenje (engl. *Bayesian Program Learning - BPL*), učenje s ograničenim resursima (engl. *poor resource learning*) i generalizacija domena (engl. *domain generalization*) svrstava pod učenje s malim brojem primjera. U usporedbi s ranije spomenutim postavkama prijenosnog učenja, pri učenju s malo primjera pretpostavlja se da ciljno područje (domena) općenito sadrži vrlo ograničene količine podataka, uključujući označene i neoznačene podatke. U ekstremnim slučajevima, može se pretpostaviti da unaprijed nisu dostupni nikakvi podaci iz ciljane domene. To je često slučaj kod problema generalizacije domena [24, str. 177-179].

Učenje bez primjera omogućuje sustavu da prepoznaje uzorke iz novih klasa koje nisu bile uključene u skup podataka za učenje. Ključna razlika u odnosu na klasično strojno učenje leži u potrebi za „mostom" koji povezuje znanje postojećih klasa s novim klasama. Taj most obično predstavljaju semantičke značajke, koje omogućuju prijenosno učenje. Semantičke značajke klase su atributi koji je opisuju. Umjesto mapiranja iz prostora značajki u prostor oznaka, u učenju bez primjera mapira se u semantički prostor. Baza znanja sadrži oznake svih klasa i njihove pripadajuće semantičke značajke. Tijekom testiranja, značajke primjera uspoređuju se sa značajkama u bazi da bi se odredila pripadnost klasi. Primjer algoritma za učenje bez primjera je algoritam DeViSE [25], koji kombinira vizualni model i jezični model. Semantičke reprezentacije oznaka dolaze iz unaprijed naučenog jezičnog modela, dok vizualne značajke dolaze iz klasičnog klasifikacijskog modela. Cilj je modela maksimizirati sličnost između vizualnih reprezentacija i semantičkih oznaka ispravnih klasa. [24, str. 178-184]

Učenje s jednim primjerom specifičan je izazov u strojnom učenju gdje se od modela očekuje učenje na temelju samo jednog uzorka po klasi. Ova situacija predstavlja velik izazov za tradicionalne algoritme, osobito duboke neuronske mreže, koje često imaju

problem s prenaučenosti (engl. *overfitting*) zbog premalog broja primjera. Razlikuju se dvije glavne strategije za rješavanje ovog problema [24, str. 184-195]:

- **Primjena prethodnog znanja** - Koriste se generativni modeli za prijenos prethodnog iskustva u zadatke s ograničenim brojem podataka. Na primjer, algoritmi poput BPL-a [24, str. 187-190] koriste generativne pristupe da bi modelirali uzorke na temelju apstraktnih komponenti, poput crtanja slova iz osnovnih poteza čime oponašaju ljudski način učenja.
- **Preoblikovanje zadatka** - Klasifikacija s jednim primjerom može se preoblikovati u zadatak provjere sličnosti. Modeli poput sijamskih neuronskih mreža [24, str. 184-187] uspoređuju ulazne primjere s prototipovima u skupu potpore. Takve mreže koriste identične arhitekture koje dijele parametre, omogućujući pretvorbu uzoraka u zajednički latentni prostor. Tako model procjenjuje vjerojatnost da uzorci pripadaju istoj klasi koristeći metričke funkcije za mjerenje udaljenosti između reprezentacija.

Iako su ovi primjeri učinkoviti, učenje s jednim primjerom se s razlogom rijetko koristi te je često potrebno proširiti učenje na više primjera da bi se ostvarili mnogo bolji rezultati.

2.2.3. Stanje tehnike unutar područja računalnog vida

U kontekstu dubokog učenja najčešći koraci provođenja prijenosnog učenja su [26]:

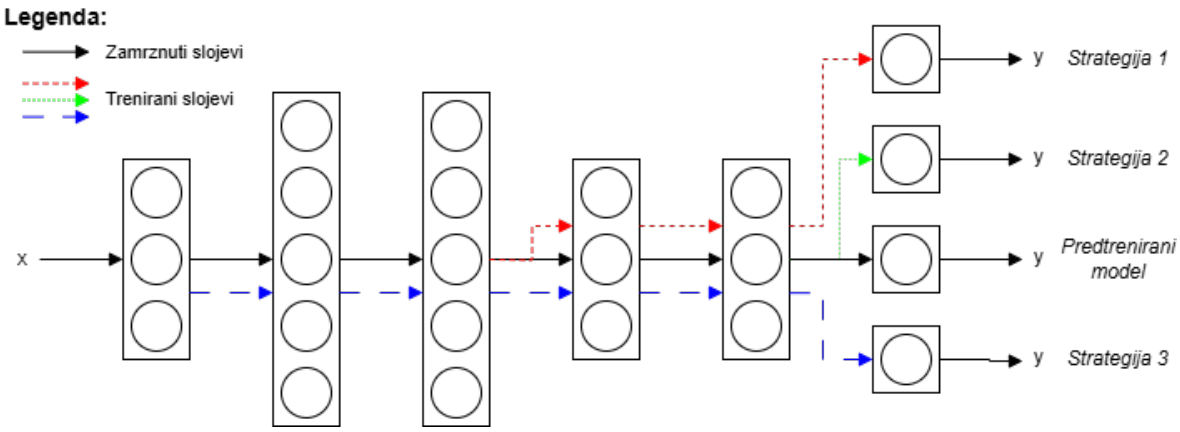
1. Uzeti željene slojeve prednaučenog modela.
2. Preuzete slojeve zamrznuti.
3. Dodati jedan ili više novih slojeva povrh zamrznutih slojeva.
4. Učiti nove slojeve drugim skupom podataka.
5. Fino podešavanje (opcionalno) (engl. *fine-tuning*).

Prije početka provedbe prvog koraka, potrebno je odgovoriti na prethodno objašnjena pitanja te odabrati odgovarajuću strategiju. U prvom koraku definiraju se željeni slojevi modela. Slojevi koji se odabiru uglavnom se uzimaju od ulaza do određenog skrivenog

sloja koji može biti manje ili više udaljen od izlaznog sloja. Nakon provedbe prvog koraka potrebno je zamrznuti preuzete slojeve što znači da se težine u tim slojevima neće ažurirati tijekom učenja novih slojeva, već će se zadržati nepromijenjene da bi se zadržalo prethodno naučeno znanje. Zatim se dodaju novi slojevi povrh smrznutih slojeva. Ovi slojevi učeni su na novom skupu podataka da bi naučili pretvarati stare značajke (znanje) u predviđanja. Posljednji, opcionalni korak je fino podešavanje.

Fino podešavanje sastoji se od otključavanja prethodno smrznutih slojeva i ponovnog učenja cijelog modela ili samo određenih dijelova modela. Tijekom procesa finog podešavanja postoje tri strategije koje se mogu koristiti da bi se optimizirao model. Navedene strategije vidljive su i u grafičkom obliku na slici 2.4. Crne strelice označavaju proces učenja izvornog modela koji se koristi u prijenosnom učenju.

Prva strategija označava odmrzavanje nekoliko slojeva s vrha konvolucijske baze te učenje samo tih slojeva uz novi klasifikator. Druga strategija pokriva učenje samo posljednjeg sloja, odnosno isključivo klasifikatora. Posljednja, treća strategija označava fino podešavanje čitavog modela. Treba uzeti u obzir da bi stopa učenja pri finom podešavanju čitavog modela trebala biti mala jer visoke stope učenja povećavaju rizik od gubitka znanja izvornog modela ili stvaranja modela koji loše generalizira [22].



Slika 2.4. Strategije finog podešavanja; prema [22, 27]

2.2.4. Pregled odabranih skupova podataka prednaučenih modela

Danas je ImageNet-1k [28] primarni je skup podataka za kreiranje prednaučenih modela dubokog učenja za zadatke računalnog vida. Razlog tome je što sadrži veliku količinu

slika, raznovrsnih kategorija i dovoljno je složen da bi modeli mogli naučiti općenite značajke koje se mogu primijeniti na druge, slične zadatke [29, str. 1]. Sastoji se od 1,4 milijuna slika raspodijeljenih u tisuću klasa, a jedan od razloga popularnosti ImageNeta-1K je upravo što je bio i jedan od prvih skupova podataka koji su omogućavali kreiranje prednaučenih dubokih modela strojnog učenja. Osim toga, postoji i velik broj istraživača i organizacija koje su sudjelovale u nastanku ImageNeta što je dovelo do velike količine istraživanja i razvoja novih tehnika prijenosnog učenja na ovom skupu podataka [30]. U razvoju skupa podataka ImageNet sudjelovale su mnoge organizacije poput Googlea, NVIDIA-e, Sveučilišta Princeton i Sveučilišta Stanford [30]. Veći i manje poznat skup podataka, ImageNet-21K, skup je podataka koji se sastoji od 14.197.122 slika podijeljenih u 21.841 klasa [30]. Cilj je industrije u budućnosti koristiti ImageNet-21K (nekad zvan *ImageNet-22K*) zbog ostvarivanja boljih rezultata. Danas se on često ne koristi zbog svoje složenosti, slabe pristupačnosti i podcjenjivanja dodane vrijednosti [29].

Drugi, malo manje poznati, ali i dalje vrlo kvalitetni skupovi podataka, koji su između ostaloga korišteni u projektu OpenPose su COCO [31], te MPII [32]. Microsoftov COCO (engl. *Common Objects in Context*) skup podataka postaje sve više popularan skup podataka za učenje i evaluaciju najmodernijih modela računalnog vida. To je zajednički projekt koji održavaju stručnjaci za računalni vid iz brojnih prestižnih institucija, uključujući Google, Caltech i Georgia Tech. Sastoji se od 330.000 slika, od kojih je preko 200.000 označeno s različitim tipovima anotacija. Sadrži 80 klasa objekata, s više od 1,5 milijuna anotacija objekata, te je poznat po raznolikim, često složenim scenama. Unutar samog skupa podataka postoji preko 250.000 osoba označenih ključnim točkama za zadatke prepoznavanja ljudske poze. Anotacije uključuju 17 ključnih točaka za svako označeno tijelo osobe, 68 ključnih točaka za lice osobe, po 21 ključnu točku za svaku ruku te 6 za stopalo, zajedno s informacijama o vidljivosti i koordinatama. [31, 33, 34, 35]

Skup podataka ljudskih poza MPII jedan je od vodećih standarda za ocjenjivanje metoda procjene artikuliranih ljudskih poza. Ovaj skup podataka, razvijen u Max Planck Institutu, sadrži oko 25 tisuća slika s više od 40 tisuća osoba na kojima su detaljno označeni ključni zglobovi tijela. Slike su prikupljene na sustavan način, prateći taksonomiju svakodnevnih ljudskih aktivnosti te pokrivaju ukupno 410 različitih aktivnosti. Svaka slika uključuje oznaku aktivnosti i potječe iz YouTube videa, uz dodatak nekoliko okvira

bez oznaka prije i nakon glavne slike. [36]

2.3. Metode praćenja pokreta u računalnom vidu

Istraživanja metoda praćenja pokreta započela su u ranim stadijima razvoja umjetne inteligencije, odnosno, konkretno, računalnog vida u kasnim 60-im godinama 20. stoljeća. Iako je početak obuhvaćao samo dijelove metoda praćenja poput razumijevanja slika, prepoznavanja objekata i analize oblika, nastavkom razvoja računalnog vida, praćenje pokreta razvilo se u zasebno područje. Mnogi problemi i dalje se javljaju pri procjeni poza, poput okluzija, neobičnih poza, nedostajućih ključnih točaka, promjena u osvjetljenju i odjeći, što rezultira stalnim naporima u razvoju preciznijih i robusnijih modela [37].

Tradicionalne metode računalnog vida, koje su dominirale od 1990-ih do ranih 2000-ih, temeljile su se na geometrijskim izračunima i pristupima temeljenima na značajkama za procjenu poza objekata [37]. Jedan od poznatih algoritama iz tog razdoblja je Direct Linear Transform (DLT), koji koristi 3D-2D korespondencije za izračunavanje pozicije kamere i njezine orijentacije [37]. Algoritam DLT zahtijeva mjerenje 3D pozicija nekoliko točaka u stvarnom svijetu i njihovo označavanje na 2D slici. Algoritam PnP (engl. *Perspective-n-Point*), sličan DLT-u, koristi minimalni broj 2D-3D korespondencija za procjenu pozicije kamere [38]. Oba algoritma omogućuju procjenu položaja drugih objekata u sceni.

Pristupi temeljeni na modelu (engl. *model-based*) razvijeni su zbog ograničenja tradicionalnih metoda računalnog vida, koje su bile osjetljive na šum, zaklonjene objekte i ovisile o parametrima kamere [37]. Ovi pristupi koriste unaprijed definirane modele objekata ili dijelova tijela za procjenu njihovih poza. Algoritam ICP (engl. *Iterative Closest Point*) koristi dvije skupine 3D točaka iz različitih perspektiva da bi uskladio njihove pozicije i orijentaciju objekta ili scene [39]. ICP široko se primjenjuje u 3D rekonstrukciji, registraciji podataka i proširenoj stvarnosti. ASM algoritam (engl. *Active Shape Models*) kombinira informacije o obliku i izgledu objekta za procjenu njegove poze ili oblika na slici [40]. ASM koristi se za preciznu analizu oblika i pozicioniranje objekata u računalnom vidu.

Metode temeljene na značajkama (engl. *feature-based*) fokusiraju se na identifikaciju i usklađivanje prepoznatljivih značajki slike da bi procijenile pozu objekata ili ljudi [37]. Te metode koriste napredne tehnike za detekciju značajki i njihovo usklađivanje, što omogućuje točnu i robusnu procjenu poza. Algoritam SURF (engl. *Speeded Up Robust Features*) je algoritam za ekstrakciju značajki koji se često koristi u računalnom vidu, uključujući procjenu poza. Iako SURF nije specifično dizajniran za procjenu poza, može se koristiti u procesu procjene radi detekcije i usklađivanja značajki [41]. Algoritam SIFT (engl. *Scale-Invariant Feature Transform*) je sličan algoritmu SURF, ali nudi veću robusnost prema promjenama u skaliranju i rotaciji te je računalno zahtjevniji [42].

Metode temeljene na dubokom učenju (engl. *deep learning-based*) u računalnom vidu koriste duboke neuronske mreže za direktno učenje povezanosti između slika i procjene poza, omogućujući visoku preciznost i robusnost [37]. Ove metode su značajno unaprijedile rezultate u mnogim područjima i nastavljaju se razvijati. Jedan od pionira ovog tipa metode bio je DeepPose [43], koji je ujedno i prethodnik OpenPose-u, a koji je uveo korištenje konvolucijskih neuronskih mreža za direktnu procjenu položaja ljudi iz slika. Modeli poput YOLOv7 [44] i YOLOv8 Pose [45] koriste napredne tehnike za detekciju ključnih točaka u slikama više osoba, s tim da YOLOv8 nudi bolje performanse i fleksibilnost. Model MediaPipe Pose [46] specijaliziran je za procjenu poza jedne osobe. On koristi seriju modela za detekciju tijela i lokalizaciju ključnih točaka, čime omogućuje brzo i efikasno praćenje.

2.3.1. OpenPose

OpenPose je sustav otvorenog koda (engl. *open source*) za procjenu poza koji su razvili istraživači sa Sveučilišta Carnegie Mellon (CMU), a koji omogućuje detekciju i praćenje ljudskog tijela u stvarnom vremenu, uz precizno određivanje njegove poze u 2D i 3D prostoru. Na 2D slikama i videima OpenPose ima mogućnost prepoznavanja više osoba, a u 3D formatu je to ograničeno na jednu osobu. Prepoznaje do 135 ključnih točaka tijela, lica, stopala te ruku osobe. Razvijen je koristeći biblioteku OpenCV [47], radni okvir Caffe (engl. *framework*) [48] te od 2020. godine platformu CUDA (engl. *Compute Unified Device Architecture*) [49]. Kao ulaz koristi sliku ili video, a dobiveni rezultati pohranjuju se u raznim formatima poput XML, YML i JSON datoteka. Tri su modela za tijelo razvijena na različitim skupovima podataka te nazvana prema istoimenim i pret-

hodno objašnjenim skupovima podataka - model MPII sadrži 15 ključnih točaka tijela, model COCO 18, a model BODY_25 njih 25. [19]

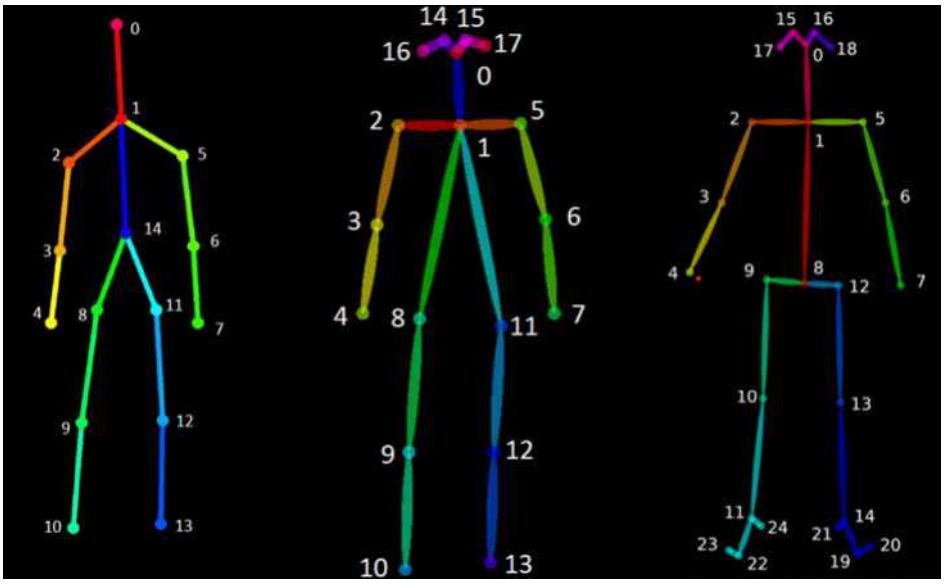
Redoslijed točaka koje modeli detektiraju, vidljive su na slici 2.5., a njihove vrijednosti, odnosno nazivi dani su u tablici 2.1.

Tablica 2.1. Ključne točke modela BODY_25, COCO i MPII [19]

Opis	**ID BODY_25**	**ID COCO**	**ID MPII**
Nos	0	0	0
Vrat	1	1	1
Desno rame	2	2	2
Desni lakat	3	3	3
Desni ručni zglob	4	4	4
Lijevo rame	5	5	5
Lijevi lakat	6	6	6
Lijevi ručni zglob	7	7	7
Središnji kuk	8	-	14
Desni kuk	9	8	8
Desno koljeno	10	9	9
Desni gležanj	11	10	10
Lijevi kuk	12	11	11
Lijevo koljeno	13	12	12
Lijevi gležanj	14	13	13
Desno oko	15	14	-
Lijevo oko	16	15	-
Desno uho	17	16	-
Lijevo uho	18	17	-
Lijevi nožni palac	19	-	-
Lijevi nožni mali prst	20	-	-
Lijeva peta	21	-	-
Desni nožni palac	22	-	-
Desni nožni mali prst	23	-	-
Desna peta	24	-	-
Pozadina	25	18	15

Prvi korak u procesu procjene poze uključuje prosljeđivanje slike kroz osnovnu CNN mrežu da bi se izvukle značajke slike. OpenPose za to koristi prvih 10 slojeva VGG-19 mreže [19]. Zatim se dobivene značajke obrađuju kroz višestupanjski CNN sustav da bi se generirali PCM-ovi (engl. *Part Confidence Maps*) i PAF-ovi (engl. *Part Affinity Fields*) [8, str. 1-3]:

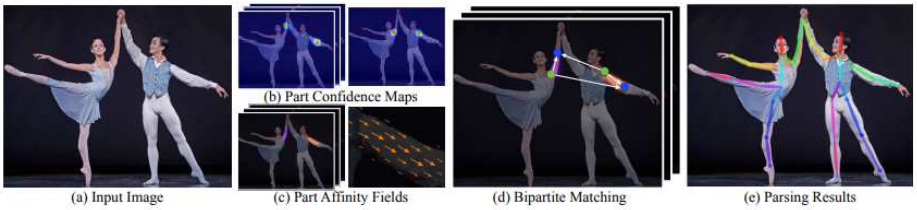
- **Part Confidence Maps**: Ove mape predstavljaju vjerojatnost da se određeni dio tijela može pronaći na svakom pikselu slike. Svaka karta predstavlja pouzdanost



Slika 2.5. Modeli detekcije položaja ljudskog tijela (MPII, COCO, BODY_25) [19]

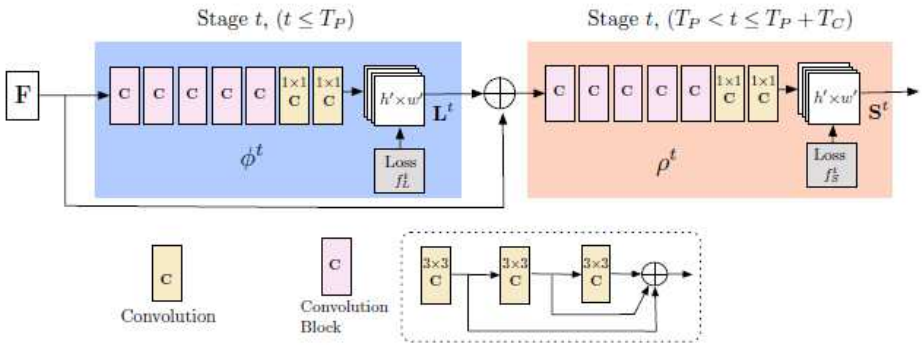
za lokaciju određenog dijela tijela na slici.

- **Part Affinity Fields**: Ova polja predstavljaju dvodimenzionalna vektorska polja koja kodiraju lokaciju i orijentaciju udova različitih osoba na slici. Ona povezuju dijelove tijela s odgovarajućim udovima, čime omogućuju preciznu detekciju i povezivanje dijelova tijela.



Slika 2.6. Prikaz cjelokupnog postupka procjene ljudske poze pomoću OpenPose sustava [8]

U završnom koraku PCM-ovi i PAF-ovi obrađuju se pomoću algoritma za bipartitno povezivanje (engl. *bipartite matching*) da bi se dobile točne poze svake osobe na slici. Funkcija gubitka koju koriste modeli je L2-gubitak i koristi se za izračunavanje razlike između PCM-ova i PAF-ova te stvarnih vrijednosti. Posredni nadzori (engl. *intermediate supervisions*) primjenjuju se na kraju svake faze da bi se riješio problem nestajućeg gradijenata tijekom treniranja. Pravilno povezivanje dijelova tijela kod više osoba omogućuju PAF-ovi, što je ključno za točnu procjenu poza u scenama s više ljudi. Ove veze između dijelova tijela omogućuju precizno praćenje i prepoznavanje tijela kada su osobe blizu jedna drugoj. [8]

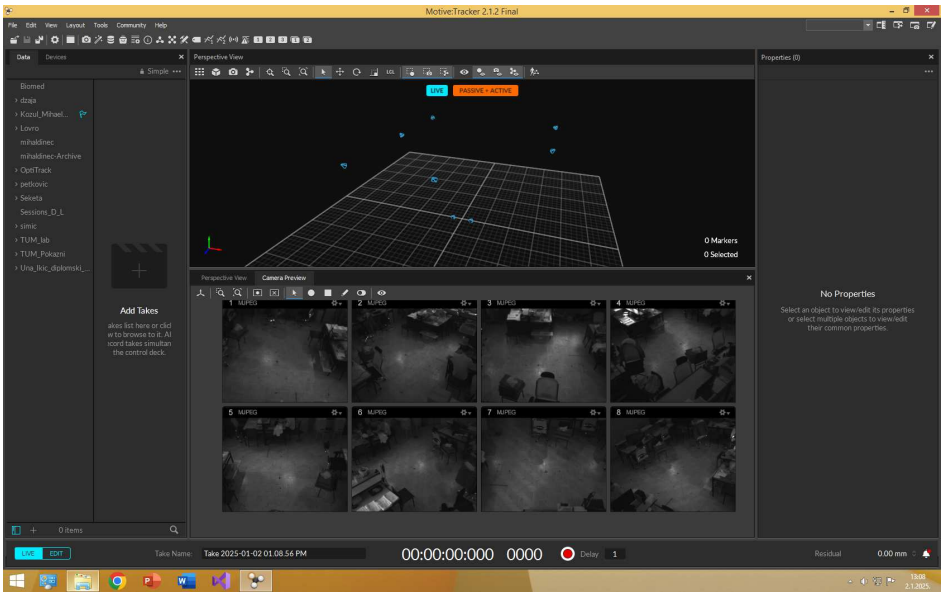


Slika 2.7. OpenPose arhitektura - prvi dio predviđa PAF-ove, a drugi dio generira PCM-ove [8]

2.3.2. OptiTrack

OptiTrack je jedan od svjetskih lidera u tehnologiji 3D praćenja (ljudskih) pokreta [7]. OptiTrackov sustav sastoji se od softverske platforme Motive i kamera koje se koriste za praćenje objekata. Platforma Motive omogućuje kalibraciju i konfiguraciju cijelog sustava te pruža sučelja za snimanje i obradu 3D podataka [50]. Podaci o pokretu prikupljaju se pomoću malih retroreflektivnih sfernih markera pričvršćenih za ljude ili objekte, a zatim se obrađuju da bi se osigurala visoka točnost u stvarnom vremenu, dakle na značajno drugačiji način u odnosu na OpenPose [51]. OptiTrackov sustav koristi se u razne svrhe poput biomehanike, robotike, animacije videoigara, filmske produkcije i virtualne stvarnosti.

Slika 2.8. Raspored OptiTrack Flex 3 kamera s prikazom subjekta s retroreflektivnim markerima [52]



Slika 2.9. Prikaz aplikacije Motive za rad sustavom OptiTrack

2.3.3. Blisko-infracrvena tehnologija u računalnom vidu

Infracrveni spektar obuhvaća širok raspon ljudskom oku nevidljivog dijela elektromagnetskog zračenja. Različite institucije različito definiraju same granice infracrvenog spektra i njegovih potpodručja. Stoga, izabrana definicija za podjelu i definiranje jest ona od Međunarodne organizacije za standardizaciju (engl. *International Organization for Standardization - ISO*), koja u svom standardu ISO 20473 dijeli infracrveno područje na 3 područja [53, str. 1]:

- **Blisko** infracrveno područje (engl. *near-infrared, NIR*) koje odgovara valnim duljinama od 780 nm do 3000 nm,
- **Srednje** infracrveno područje (engl. *mid-infrared, MIR*) koje odgovara valnim duljinama od 3,000 nm do 50,000 nm, te
- **Daleko** infracrveno područje (engl. *far-infrared, FIR*) koje odgovara valnim duljinama od 50,000 nm do 1,000,000 nm.

Za razliku od vidljivog spektra (380–780 nm [53, str. 1]), blisko infracrveni spektar ima sposobnost prodiranja kroz određene materijale, što ga čini idealnim za analizu struktura koje nisu vidljive RGB kamerama. Jedna od glavnih prednosti tehnologije NIR jest otpornost na promjene osvjetljenja, što je čini pouzdanom u uvjetima slabe vidljivosti. Zahvaljujući sposobnosti prodiranja kroz dim, maglu i određene površine, spektar

NIR ima široku primjenu u medicini, poljoprivredi, sigurnosnim sustavima i autonomnim tehnologijama. [54]

Općenito, NIR kamere koriste senzore osjetljive na valne duljine u rasponu od približno 700 nm do približno 1100 nm jer se u tom području silicijski senzori još uvijek mogu koristiti za detekciju infracrvenog svjetla [55, 56]. Većina komercijalnih NIR kamera, uključujući OptiTrack Flex 3, optimizirana je za snimanje u tom spektralnom području. OptiTrack Flex 3 koristi 850 nm infracrvenu iluminaciju za poboljšanje vidljivosti reflektirajućih objekata [52]. S obzirom na to da se u ovom radu koriste Flex 3 kamere, NIR spektar će se za potrebe istraživanja definirati u skladu s osjetljivošću tih kamera, tj. u rasponu 780–1100 nm.

3. Materijali i metode

Unutar poglavlja opisani su pristupi i alati korišteni tijekom istraživanja s posebnim naglaskom na pripremu i izradu skupa podataka te prilagodbu OpenPose modela za bliskoinfracrvene (NIR) podatke. Istražena su i uspoređena tri pristupa za kreiranje skupa podataka. Prvi, najjednostavniji pristup, temelji se na korištenju već postojećeg RGB skupa podataka, koji se pretvara u NIR skup podataka korištenjem isključivo crvenog kanala. Drugi, složeniji pristup, uključuje kombinaciju RGB kanala s odgovarajućim težinskim faktorima, a posljednji pristup podrazumijeva izradu potpuno novog skupa podataka pomoću NIR kamera. Osim toga, opisani su postupci implementacije i treniranja modela OpenPose pomoću prethodno opisanih, različitih skupova podataka. Rezultati tih postupaka prikazani su u narednom poglavlju 4. Rezultati i analiza.

3.1. Pretvorba RGB slika u NIR format

Izrada potpuno novog skupa podataka često je vremenski vrlo zahtjevan proces, a javno dostupni NIR skupovi podataka, u odnosu na RGB skupove podataka, iznimno su rijetki. Stoga su u nastavku razmotrene dvije alternativne metode za generiranje NIR podataka, koje omogućuju potencijalno učinkovitiji pristup ovom izazovu.

3.1.1. Simulacija NIR-a korištenjem crvenog kanala

Prva jednostavnija ideja kako doskočiti problemu rijetkih NIR podataka jest iskoristiti mnogo češće dostupne RGB skupove podataka poput skupova podataka COCO i MPII nad kojima su, između ostalog, naučeni različiti OpenPose modeli. RGB slike sastoje se od tri kanala - crvenog (engl. *red*), zelenog (engl. *green*) i plavog (engl. *blue*). S obzirom na to da je crveni kanal RGB slike najbliži infracrvenom području elektromagnetskog zračenja, najjednostavniji je pristup samo zadržati crveni kanal svake RGB slike i na taj

način simulirati NIR sliku.

Ovaj pristup ne zahtijeva dodatne kalkulacije težinskih faktora RGB kanala niti transformacije, čime se ubrzava obrada podataka i olakšava eksperimentiranje. Implementacija zadržavanja samo crvenog kanala pojednostavljuje proces, što može biti korisno u ranim fazama razvoja i evaluacije modela. U nekim slučajevima, ovaj pristup može biti dovoljno dobar sam po sebi pružajući prihvatljive rezultate bez dodatnih optimizacija. Time se štede resursi i vrijeme, osobito ako rezultati usporedive točnosti ne zahtijevaju složenije metode. Treba napomenuti da se ova metoda, ako se pokaže korisnom, također može kombinirati s postojećim NIR skupom podataka. Time se, na jednostavan način, proširuje skup za učenje obogaćujući ga većim rasponom pokreta, vremenskih uvjeta, svjetlosnih varijacija i drugih relevantnih karakteristika.

Isječak kôda 3.1: Primjer pretvorbe RGB slike u NIR sliku koristeći isključivo crveni kanal

```
def extract_r_channel(rgb_image):
    return rgb_image[:, :, 0]

image_path = os.path.join(test_folder, filename)

rgb_image = cv2.imread(image_path)
if rgb_image is None:
  print(f"Error image: {image_path}")
  continue
rgb_image = cv2.cvtColor(rgb_image, cv2.COLOR_BGR2RGB)

r_channel_image = extract_r_channel(rgb_image)
```

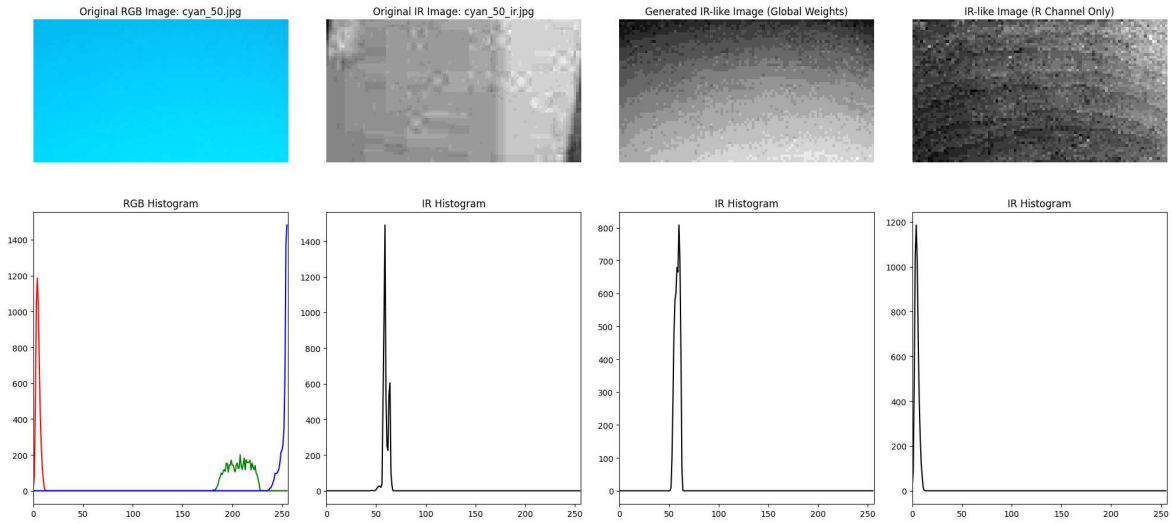
3.1.2. Kombinacija RGB kanala s težinskim faktorima

Za precizniju simulaciju NIR slika iz RGB podataka može se primijeniti metoda kombiniranja svih triju kanala (crvenog, zelenog i plavog) pomoću pažljivo podešenih težinskih faktora. Ovi faktori određuju doprinos svakog kanala u konačnoj transformaciji slike, oponašajući karakteristike NIR spektra. Ovakav pristup, u usporedbi s jednostavnom ekstrakcijom crvenog kanala, nudi veću fleksibilnost i potencijal za postizanje boljih rezultata.

Međutim, važno je napomenuti da simulacija NIR spektra korištenjem RGB podataka u ovom slučaju jest ograničena. Kao što je objašnjeno u potpoglavlju *2.3.3. Blisko-infracrvena tehnologija u računalnom vidu* NIR kamere koriste silicijske senzore osjet-

ljive na valne duljine do približno 1100 nm, a puni NIR spektar, prema definiciji ISO 20473 standarda, obuhvaća raspon od 780 nm do 3000 nm. Stoga, u ovome se slučaju zapravo simulira dio NIR spektra koji je moguće obuhvatiti s OptiTrack kamerama.

Jedna od ključnih prednosti korištenja težinskih faktora jest mogućnost finog prilagođavanja težinskih faktora da bi se bolje simulirale karakteristike NIR podataka. To omogućuje bolju preciznost u situacijama gdje je važno očuvati specifične detalje i značajke slike. Međutim, određivanje optimalnih težinskih faktora predstavlja izazov jer zahtijeva analizu kako bi se pronašla kombinacija koja najbolje oponaša NIR slike. Tome se može doskočiti koristeći procijenjene faktore (npr. R=0.7, G=0.2, B=0.1), preddefinirane faktore za klasičnu konverziju RGB slike u sivu sliku (R=0.299, G=0.587, B=0.114) ili optimizirane faktore poput onih koji su prikazani u nastavku.



Slika 3.1. Usporedba RGB, NIR i generiranih NIR slika s pripadajućim histogramima

Kako bi simulacija NIR slika ostvarivala što bolje rezultate, korišten je postupak optimizacije koji minimizira razliku između stvarnih NIR slika i njihovih simuliranih verzija. U tu svrhu definirana je funkcija gubitka koja računa srednju kvadratnu pogrešku (engl. *Mean Squared Error, MSE*) između stvarnih NIR slika i generiranih slika dobivenih linearnom kombinacijom RGB kanala s različitim težinskim faktorima. Početne vrijednosti težinskih faktora postavljene su prema klasičnim konverzijskim faktorima za sivu sliku (R=0.299, G=0.587, B=0.114), a zatim su optimizirane koristeći metode numeričke minimizacije.

Proces simuliranja NIR slika išao je sljedećim redoslijedom:

- Učitavanje NIR slika snimljenih u kontroliranim uvjetima te odabranog RGB skupa podataka.
- Normalizacija i prilagodba veličine RGB slika kako bi odgovarale dimenzijama NIR slika.
- Računanje simulirane NIR slike primjenom različitih težinskih faktora na RGB kanale.
- Evaluacija rezultata pomoću funkcije gubitka koja mjeri razliku između stvarne i simulirane NIR slike.
- Optimizacija težinskih faktora radi minimiziranja te razlike.

Dobiveni optimalni faktori nakon optimizacije su: R=0.353, G=0.504, B=0.143. Unatoč izazovima, kombinacija RGB kanala s težinskim faktorima predstavlja robusniji pristup koji može ponuditi značajna poboljšanja u točnosti simulacije NIR podataka.

Isječak kôda 3.2: Primjer pretvorbe RGB slike u NIR sliku koristeći težinske faktore

```
def simulate_nir(rgb_image, weights):
    red, green, blue = rgb_image[:, :, 0], rgb_image[:, :, 1],
   rgb_image[:, :, 2]
    return weights[0] * red + weights[1] * green + weights[2] * blue

image_path = os.path.join(test_folder, filename)

rgb_image = cv2.imread(image_path)
if rgb_image is None:
  print(f"Error image: {image_path}")
  continue
rgb_image = cv2.cvtColor(rgb_image, cv2.COLOR_BGR2RGB)

nir_image = simulate_nir(rgb_image, optimal_weights)
nir_image = np.clip(nir_image, 0, 255).astype(np.uint8)
```

3.2. Snimanje eksperimentalnih podataka

Eksperimentalni podaci snimljeni su u laboratorijskoj prostoriji FER-a, unutar Zavoda za elektroničke sustave i obradbu informacija (ZESOI). Za snimanje su korištene NIR kamere sustava OptiTrack. Opći raspored kamera i struktura okruženja prikazani su na slici 2.8., a izgled dijela kamera unutar laboratorija može se vidjeti na slici 3.2.

Ukupno je snimljeno 2480 slika, koje su podijeljene na skup za treniranje i skup za

testiranje u omjeru 75:25, što čini 1860 slika za treniranje i 620 za testiranje. Skup podataka uključuje snimke od nula do četiri osobe u kadru, pri različitim uvjetima osvjetljenja, kvalitetama slika, varijacijama u bojama odjeće i pozama. Zabilježene boje odjeće uključuju crvenu, plavu, smeđu, crnu, bijelu i druge, s ciljem povećanja raznolikosti podataka i poboljšanja generalizacije modela.

Slika 3.2. Položaj dijela OptiTrack kamera u prostoriji

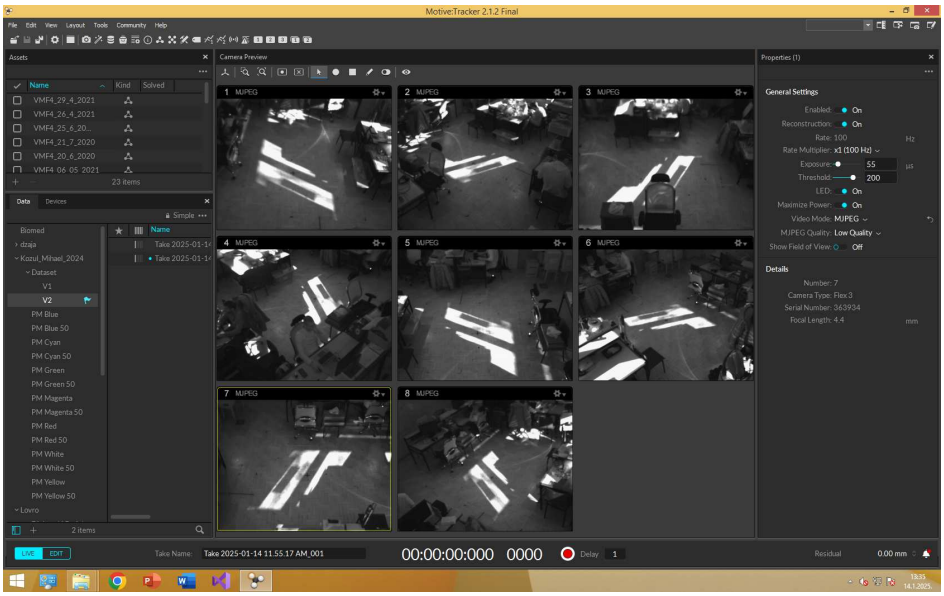
Platforma Motive, vidljiva na slici 3.3., korištena je za upravljanje i snimanje kamerama. Prilikom snimanja, Motive sprema videozapise u nativnom *.tak* formatu koji može korstiti isključivo Motive, no postoji mogućnost izvoza u formatu MJPEG, odnosno kao *.avi* datoteke. Tijekom izvoza moguće je odabrati broj sličica (engl. *frames*) koje će se izvesti, kao i razinu kvalitete putem ugrađenog izbornika unutar sustava Motive. Većina videozapisa izvezena je s najvišom dostupnom kvalitetom, a manji je dio izvezen s nižom kvalitetom radi raznolikosti skupa podataka. Snimanje je obavljeno pomoću osam kamera, koje su radile istovremeno pri rezoluciji od 3 MP.

3.3. Priprema eksperimentalnih podataka

Nakon što je skup podataka snimljen u laboratoriju i prenesen na lokalno računalo, bilo je potrebno izvući pojedinačne slike iz videozapisa za anotaciju i treniranje modela. Za ekstrakciju sličica iz videozapisa snimljenih u *.avi* formatu koristio se alat FFmpeg. FFmpeg je cjelovito, višeplatformsko rješenje za snimanje, obradu i pretvaranje audio i video sadržaja [57]. Kratica *FFmpeg* dolazi od naziva *Fast Forward MPEG*, u kojem *MPEG* označava skup standarda za kodiranje zvuka i videa. Primjer korištene naredbe:

Isječak kôda 3.3: Primjer FFmpeg naredbe

```
ffmpeg -i "<direktorij>\video.avi" -vf "fps=N" -q:v Q "C:\direktorij\
    slika_%03d.png"
```



Slika 3.3. Proces kreiranja skupa podataka iz perspektive sučelja sustava Motive

Objašnjenje korištenih parametara [57]:

- `-i "<direktorij> \video.avi"`: Putanja do ulaznog videozapisa.
- `-vf "fps=1"`: Video filtar koji definira učestalost odabira sličica. Vrijednost *N* određuje broj sličica koje se esktrahiraju po sekundi (npr. `"fps=1"` označava jednu sličicu po sekundi).
- `-q:v Q`: Parametar za kvalitetu izlaznih slika. Vrijednost Q predstavlja kvalitetu, gdje manji brojevi označavaju bolju kvalitetu (npr. `q:v 2` za visoku kvalitetu).
- `"<direktorij>\slika_%03d.png"`: Putanja i naziv izlaznih slika s numeracijom (npr. sličica_001.png). Oznaka %03d označava numerički format s tri znamenke (od 001 do 999).

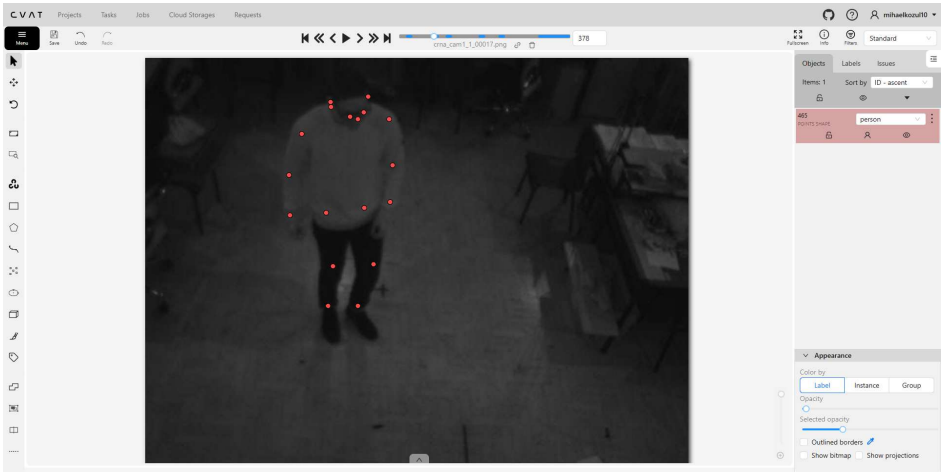
Osim izdvajanja sličica, FFmpeg nudi i niz drugih mogućnosti poput rotacije videa, dodavanja titlova, promjene načina reprodukcije i optimizacije kompresije. Sve opcije i parametri detaljno su opisani u službenoj FFmpeg dokumentaciji [57].

3.3.1. Anotacija podataka

U području anotacije podataka postoji nemali broj javno dostupnih alata kao što su VGG Image Annotator (VIA), LabelMe, LabelImg, MakeSense.ai, Diffgram, Roboflow i drugi.

Međutim, alat CVAT (engl. *Computer Vision Annotation Tool*) [58] se pokazao kao najpraktičnije rješenje za projektni dio diplomskog rada. Ostali su alati imali određene nedostatke poput greška u kodu (engl. *bug*), ograničenih funkcionalnosti, nepraktične anotacije ključnih točaka ili nekog drugog faktora koji je utjecao na konačnu odluku.

Platforma CVAT platforma je otvorenog koda za anotaciju podataka u području računalnog vida. Razvio ga je Intel, a nudi jednostavno sučelje za označavanje objekata na slikama i videozapisima, uključujući detekciju objekata, segmentaciju i definiranje ključnih točaka [59]. Alat CVAT podržava razne formate izlaznih datoteka koji su kompatibilni s popularnim bibliotekama za strojno učenje i računalni vid, no zbog greške u kodu format COCO se ne izvozi ispravno pa je potrebno izvršiti ručnu pretvorbu, odnosno pretvorbu na programski način [60]. Kod u programskom jeziku Python za pretvorbu dostupan je u dodacima koda 1.2 Moguće je koristiti ga u pregledniku ili lokalno preko Docker slike (engl. *Docker image*), a podržan je i način rada za više osoba te integracija s raznim alatima za upravljanje podacima. Ako se alat pokreće putem Dockera, potrebno je u terminalu pozicionirati se u direktorij projekta CVAT, pokrenuti Docker daemon te na kraju pokrenuti sliku naredbom `docker-compose up`.



Slika 3.4. Prikaz sučelja CVAT s dodanim ključnim točkama

3.3.2. COCO skup podataka

Format COCO (Common Objects in Context) jedan je od najčešće korištenih formata za anotaciju podataka u računalnom vidu, a razvijen je za potrebe skupa podataka COCO. Ovaj format koristi JSON datoteke za pohranu informacija o slikama, anotacijama i kategorijama. Službeni format COCO sastoji se od 5 dijelova: `info`, `licenses`, `images`,

`annotations` i `categories`. Tipičan izgled COCO JSON datoteke s primjerima atributa unutar svake sekcije i tipom podataka izgleda na sljedeći način [31]:

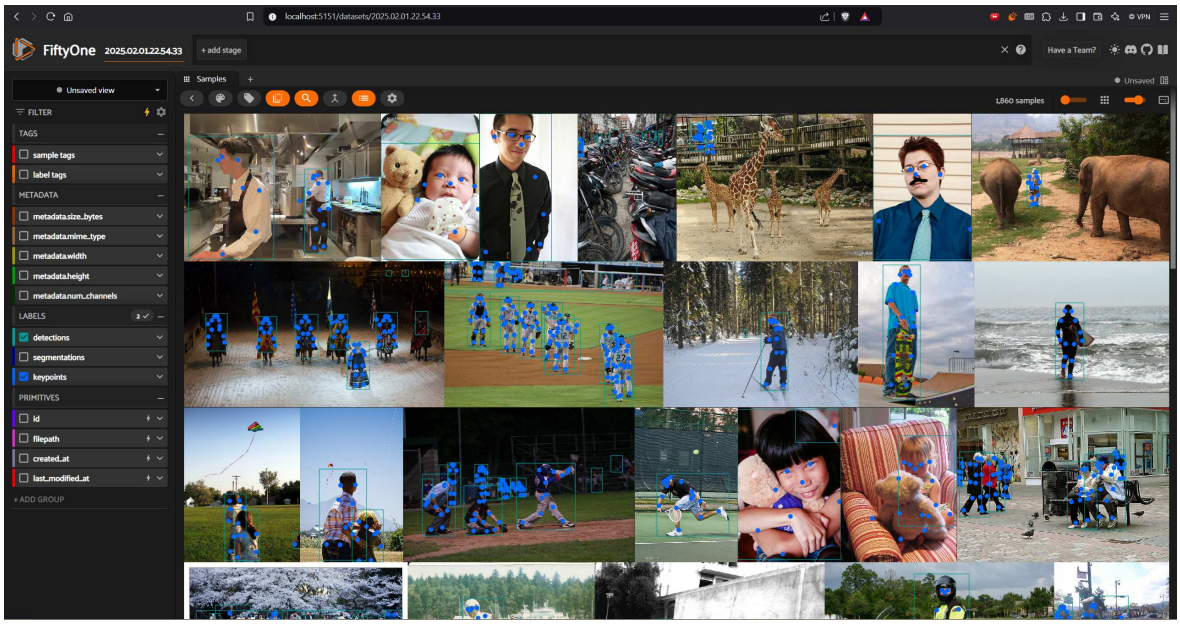
Isječak kôda 3.4: JSON datoteka COCO formata

```
{
info[{
    "year": int,
    "version": str,
    "description": str,
    "contributor": str,
    "url": str,
    "date_created": datetime,
}]
license[{
    "id": int,
    "name": str,
    "url": str,
}]
images[{
    "id": int,
    "width": int,
    "height": int,
    "file_name": str,
    "license": int,
    "flickr_url": str,
    "coco_url": str,
    "date_captured": datetime,
}]
annotations[{
    "id": int,
    "image_id": int,
    "category_id": int,
    "segmentation": RLE or [polygon],
    "area": float,
    "bbox": [x,y,width,height],
    "iscrowd": 0 or 1,
}]
categories[{
    "id": int,
    "name": str,
    "supercategory": str,
}]
}
```

Format COCO fleksibilan je i standardiziran te podržava različite zapise za zadatke kao što su detekcija objekata, segmentacija, prepoznavanje ključnih točaka i klasifikacija kao što je vidljivo u prethodno prikazanim atributima [31].

Za potrebe ovog rada, preuzet je uzorak od 2480 slika iz skupa podataka COCO korištenjem alata FiftyOne [61]. S obzirom na to da nije bilo moguće odabrati uzorak koji sadrži ključne točke, iz uzorka od 2480 slika filtrirano je njih 1860 s ključnim točkama. Izabrano je točno 1860 slika da bi svi skupovi za treniranje bili jednake veličine. Alat FiftyOne je alat otvorenog koda koji omogućuje jednostavno preuzimanje i upravljanje

skupovima podataka, kao i lakšu analizu podataka pomoću vizualizacije i filtriranja [31].



Slika 3.5. Sučelje alata FiftyOne s prikazom uzorka skupa podataka COCO

3.4. Prilagodba OpenPose modela za NIR podatke

U nastavku su poglavlja detaljno objašnjeni koraci prilagodbe, uključujući pokretanje OpenPose modela, pripremu eksperimentalnih podataka za treniranje i testiranje te implementaciju prijenosnog učenja. OpenPose modeli nisu izvorno trenirani na NIR podacima. Zbog toga, iako mogu prepoznati osobe u kadru, često dolazi do pogrešaka u detekciji dijelova tijela ili broja osoba. Neki od primjera lošijih rezultata COCO modela vidljivi su na slici 3.6.

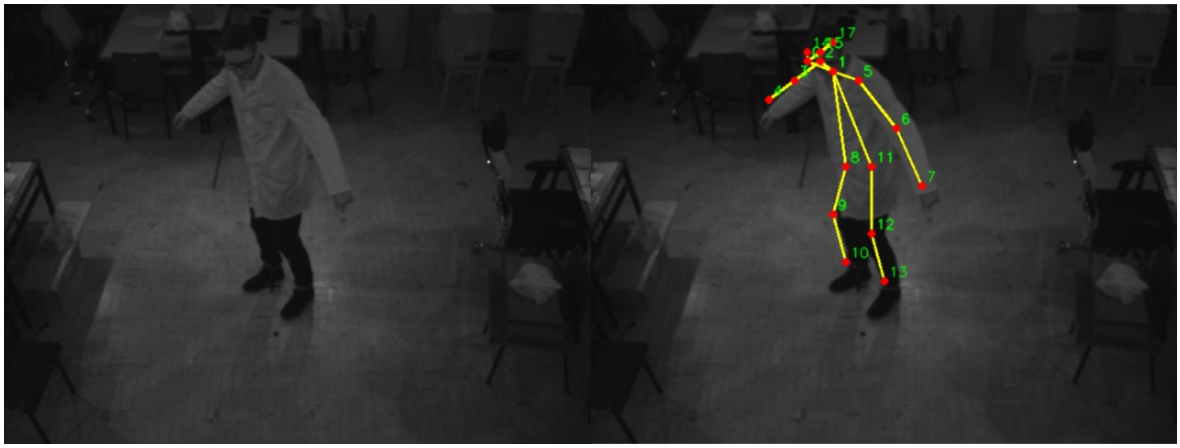
Slika 3.6. Prikaz lošijih rezultata OpenPose modela COCO_18

3.4.1. Pripema OpenPose projekta

Windows Portable Demo

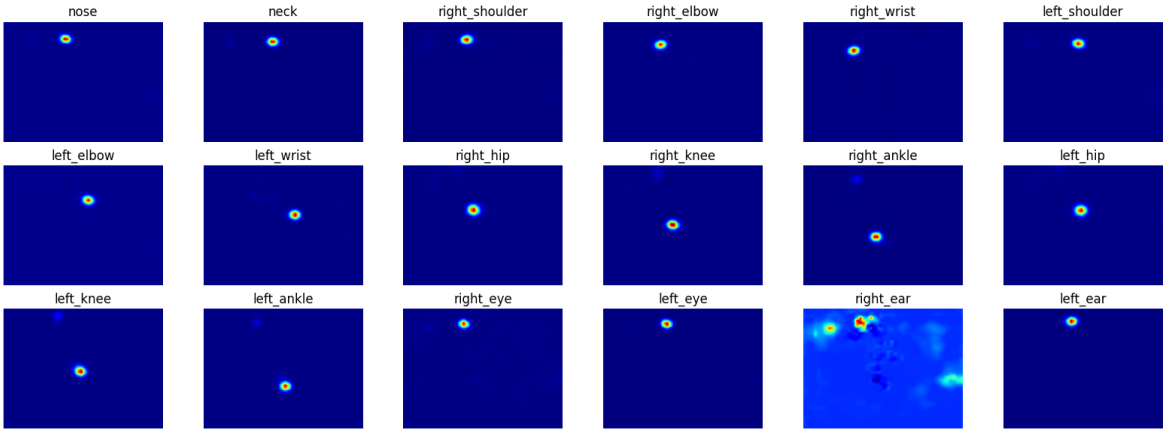
Za instalaciju i pokretanje projekta OpenPose, odnosno njegovih modela, postoje dva glavna pristupa. Prvi, jednostavniji način, uključuje korištenje unaprijed kompajlirane

inačice projekta OpenPose koja služi prvenstveno kao demonstracija mogućnosti modela. Ova inačica namijenjena je uglavnom operacijskom sustavu Windows i omogućuje pokretanje svih dostupnih modela na slikama i videozapisima, ali ne dopušta veću prilagodbu koda, testiranje modela i određene druge funkcionalnosti.



Slika 3.7. Ulaz i izlaz predtreniranog COCO modela

Primjer estimacije poze predtreniranog COCO modela prikazan je na slici 3.7., a slika 3.8. prikazuje probabilističke mape ključnih točaka koje model generira. Na slici nije prikazano desno uho jer model, nakon generiranja probabilističkih mapa, filtrira ključne točke odabirom najvjerojatnije lokacije za svaki dio tijela. Taj postupak uključuje usporedbu pouzdanosti predikcije (engl. *confidence*) s unaprijed definiranom graničnom vrijednošću (engl. *threshold*). Ako je pouzdanost ispod zadanog praga, ključna točka se ne detektira i ne prikazuje. U ovom slučaju, sve vrijednosti pouzdanosti bile su znatno iznad praga koji je iznosio 0.1, osim desnog uha, gdje je pouzdanost iznosila 0.01, zbog čega točka nije prikazana.



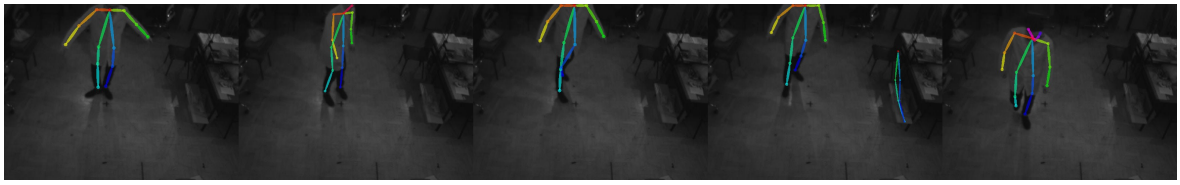
Slika 3.8. Probabilističke mape ključnih točaka COCO modela

Demo verzija može se pokrenuti pomoću izvršne datoteke u terminalu uz određene zastavice, kao što je prikazano u nastavku:

Isječak kôda 3.5: Primjer pokretanja OpenPoseDemo datoteke

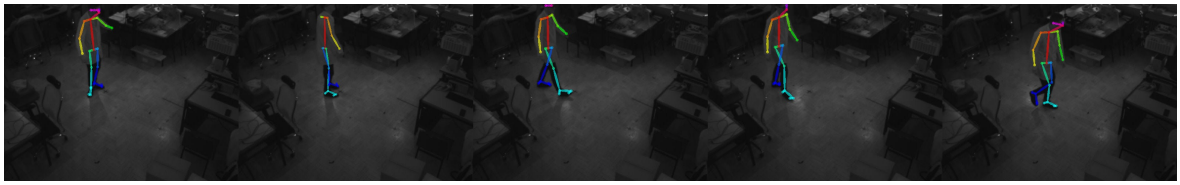
```
bin\OpenPoseDemo.exe --image_dir "<direktorij>\slika.png" --write_json
    "<direktorij>" --write_images "<direktorij>"
```

Rezultati predtreniranog COCO modela, odabranog za daljnje treniranje, prikazani su na slikama 3.9. i 3.10., koje prikazuju detekciju ključnih točaka iz različitih perspektiva kamere. Osim COCO modela, usporedbe radi, prikazan je i jedan primjer s modelom BODY_25, čiji su rezultati prikazani na slici 3.11.



Slika 3.9. Prikaz rezultata modela COCO_18 iz perspektive kamere broj 1

Slika 3.10. Prikaz rezultata modela COCO_18 iz perspektive kamere broj 6



Slika 3.11. Prikaz rezultata modela BODY_25 iz perspektive kamere broj 2

Pri pokretanju OpenPose modela moguće je pomoću zastavice definirati format zapisa izlaznih podataka. Zadani format prikazan je u nastavku, pri čemu se podaci za svaku sliku zapisuju u zasebnu JSON datoteku. S druge strane, postoji opcija generiranja izlaza u COCO formatu, što znači da su svi podaci za slike zapisani u jednoj JSON datoteci. Koordinate se zapisuju redoslijedno u listi u formatu $(x_1, y_1, v_1, x_2, y_2, v_2, \dots)$, gdje v predstavlja vidljivost (engl. *visibility*). Vidljivost može poprimiti tri vrijednosti: 0, 1 ili 2, gdje 0 označava da točka nije prisutna, 1 označava da je točka prisutna, ali skrivena, odnosno nije vidljiva, a 2 označava da je točka prisutna i označena [62]. U zadanoj verziji

treća varijabla nije vidljivost, već je ekvivalent vrijednosti `score` kod formata COCO te označava pouzdanost. Još jedna bitna razlika između zadanog formata i formata COCO jest da zadani format sadržava 18 ključnih točaka, a format COCO njih 17. Ključna točka koja definira lokaciju vrata nije podržana u formatu COCO.

Isječak kôda 3.6: Prikaz dva načina OpenPose izlaza ključnih točaka u obliku JSON datoteke

```
# Zadana verzija
{
  "version": 1.3,
  "people": [
    {
      "person_id": [],
      "pose_keypoints_2d":
   [275.903,163.678,0.370656,223.64,179.284,0.603284...],
      "face_keypoints_2d": [],
      "hand_left_keypoints_2d": [],
      "hand_right_keypoints_2d": [],
      "pose_keypoints_3d": [],
      "face_keypoints_3d": [],
      "hand_left_keypoints_3d": [],
      "hand_right_keypoints_3d": []
    }
  ]
}

# COCO verzija
[
{
    "image_id": 1,
    "category_id": 1,
    "keypoints": [275.903,163.678,1,-1,-1,0...],
    "score": 0.376613
},
{
    "image_id": 1,
    "category_id": 1,
    "keypoints": [346.346,78.8395,1,347.686,68.3703,1...],
    "score": 0.423883
}
]
```

Izvorni kod projekta OpenPose

Za sve dostupne funkcionalnosti modela potrebno je pokrenuti OpenPose iz izvornog koda (engl. *source code*). To, prije svega, zahtijeva instalaciju nužnih preduvjeta ovisno o operacijskom sustavu. Na operacijskom sustavu Windows potrebno je instalirati [19]:

- **CMake GUI** – alat za upravljanje procesom izgradnje softvera,
- **Microsoft Visual Studio** – integrirano razvojno okruženje (u sklopu rada instalirana verzija 2019 Community, no predlažu se verzije 2015, 2017 i 2019 Enterprise),

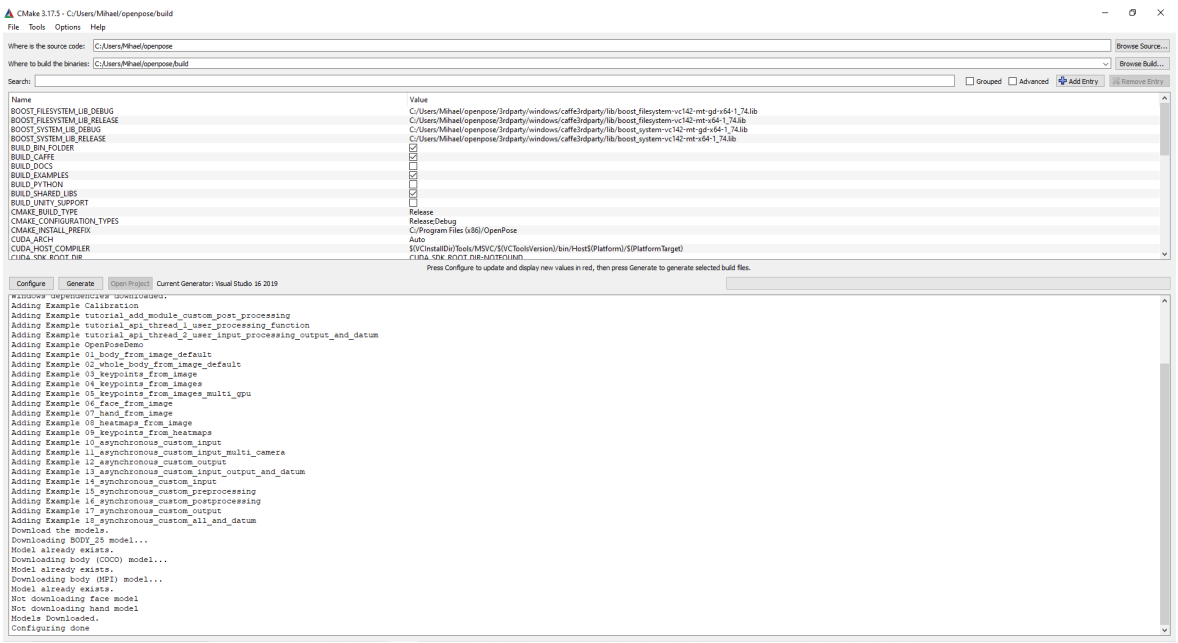
- **NVIDIA CUDA** – u sklopu rada instalirana je verzija 11.1.1 iako postoji podrška za više verzija, kao i za AMD GPU, i
- **Caffe, OpenCV i ostali paketi** – paketi bi se trebali automatski instalirati, no postoji greška, stoga je potrebno ručno instalirati, raspakirati i zamijeniti `3rdparty` direktorij [63].

Nakon instalacije svih preduvjeta, potrebno je klonirati OpenPose repozitorij i inicijalizirati njegove podmodule. No, kao i kod paketa Caffe i ostalih paketa, potrebno je prije inicijalizacije na internetu pronaći modele, instalirati ih, raspakirati i dodati u direktorij `models`:

Isječak kôda 3.7: Kloniranje repozitorija OpenPose

```
git clone https://github.com/CMU-Perceptual-Computing-Lab/openpose
## dodati datoteke koje nedostaju
cd openpose/
git submodule update --init --recursive --remote
```

Nakon što je repozitorij kloniran, potrebno je izgraditi projekt pomoću alata CMake-a. Prvi je korak kreiranje direktorija `build` unutar direktorija OpenPose. U grafičkom sučelji alata CMake zatim se unose odgovarajuće putanje, kao što je prikazano na slici 3.12. Nakon odabira generatora projekta (Visual Studio 2019, x64), konfiguracija se pokreće pritiskom na polje `Configure`, a zatim na polje `Generate`.

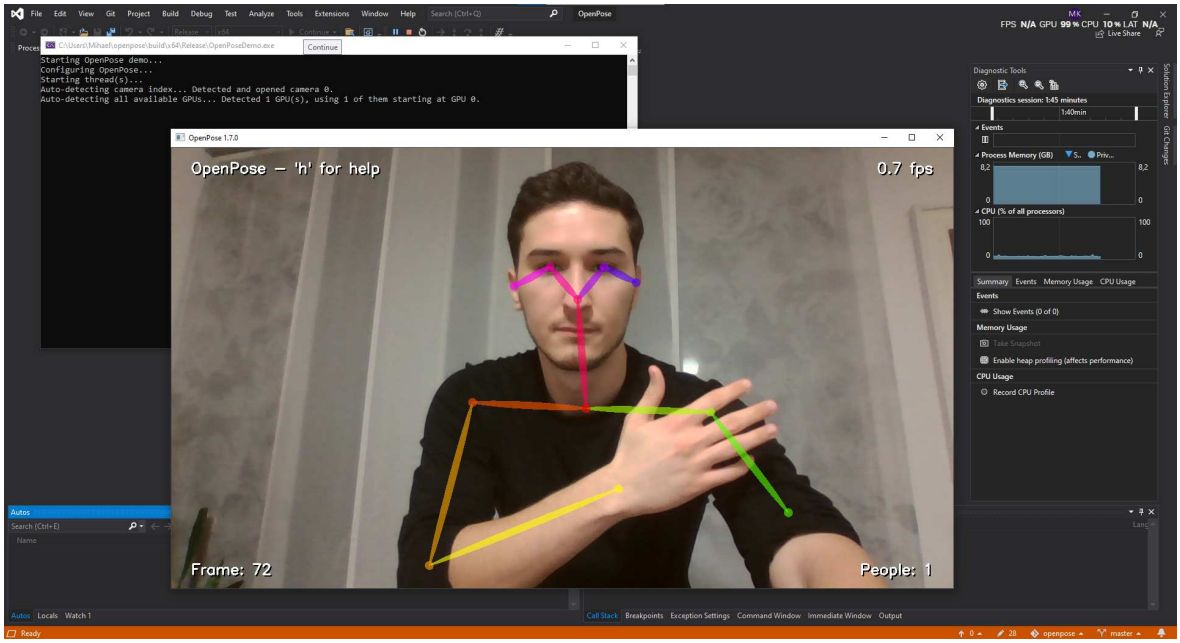


Slika 3.12. Generiranje projekta u alatu CMake

Projekt se zatim otvara u Visual Studiju, gdje je potrebno:

1. Promijeniti konfiguraciju iz `Debug` u `Release`
2. U izborniku `Build` odabrati `Build Solution`

Time je projekt spreman za pokretanje s lokalnom kamerom, što je prikazano na slici 3.13.



Slika 3.13. Detekcija ključnih točaka lokalnom kamerom u projektu OpenPose

Ako se OpenPose želi pokrenuti izvan razvojnog okruženja Visual Studio, potrebno je kopirati sve *.dll* datoteke iz `build_direktorij/bin`, u direktorij gdje se nalaze generirani *.exe* i *openpose.dll*, primjerice `build_direktorij/x64/Release` [19]. Primjer pokretanja modela nad videozapisom:

Isječak kôda 3.8: Primjer pokretanja OpenPose izvršen datoteke izvan razvojnog okruženja Visual Studio

```
build\x64\Release\OpenPoseDemo.exe --video examples/media/video.avi
```

Projekt OpenPose pokreće se pomoću izvršnih datoteka, pri čemu se koriste različite zastavice (engl. *flags*) za prilagodbu modela:

- `image_dir <direktorij>`: putanja do direktorija sa slikama koje se koriste kao ulaz,

- `video <video.datoteka>`: putanja do video datoteke koja služi kao ulaz,
- `write_images <direktorij>`: putanja za spremanje rezultata u obliku slika,
- `write_video <video.datoteka>`: putanja za spremanje rezultata u obliku videozapisa,
- `model_pose <model>`: odabir modela (COCO, MPII, BODY_25 – zadani model je BODY_25), i
- `number_of_gpu <broj>`: broj GPU-ova koji će se koristiti za obradu.

Osim ovih opcija, OpenPose podržava dodatne zastavice za fino podešavanje modela, prilagodbu preciznosti, postavke vremenskih parametara i kompatibilnost s različitim hardverskim platformama. Detaljan popis opcija može se vidjeti pozivom naredbe `-h` ili `-help`, a djelomično i u dokumentaciji [19].

3.4.2. Implementacija prijenosnog učenja

U sklopu diplomskog rada, za treniranje modela nije korišten originalni model OpenPose implementiran u radnom okviru Caffe [19], već njegova implementacija u PyTorch okruženju [64]. Razlog leži u činjenici da je OpenPose u Caffe okruženju mnogo složeniji za prilagodbu i primjenu prijenosnog učenja. Naime, prilikom pokušaja treniranja modela u Caffe okruženju pojavili su se isti problemi kao pri izgradnji projekta iz izvornog koda, ali u učestalijem obliku i bez alternativnih rješenja. Dodatni je izazov bio i to što OpenPose ne nudi treniranje modela izravno iz repozitorija, već je za tu svrhu potrebno koristiti eksperimentalni `openpose_train` repozitorij, koji za funkcioniranje zahtijeva uspostavljanje modificiranog Caffe repozitorija. Tu se javljaju problemi poput nedostatka, odnosno nemogućnosti pronalaska skripti, dijelova koda, datoteka i slično koje se dohvaćaju s različitih poveznica. Poveznice su zapisane u kodu, unutar skripti repozitorija, no nisu dostupne. Osim toga, dokumentacija ne nudi dovoljno jasno objašnjenje postupka pokretanja projekta. Dodatno, određeni problemi koji su se pojavili kod uspostave okruženja ostali su neadresirani zbog čega se čini kako je repozitorij u trenutnom obliku nefunkcionalan. Treba napomenuti i da je službeni *openpose_train* repozitorij dizajniran za treniranje modela isključivo od nule, koristeći COCO skup podataka koji se dohvaća i obrađuje putem specifičnih skripti. To predstavlja problem u kontekstu prije-

nosnog učenja jer bi omogućavanje prijenosnog učenja zahtijevalo značajnu promjenu velikog dijela projekta.

Zbog navedenih razloga, iskorištena je PyTorch implementacija OpenPose modela, koja je nastala konverzijom Caffe modela u PyTorch format pomoću skripte koja se nalazi u istoimenom projektu `caffemodel2pytorch.py` [65]. Osim svih problema vezanih uz Caffe OpenPose projekt, PyTorch verzija odabrana je i zbog veće fleksibilnosti prilikom eksperimentiranja, lakšeg definiranja hiperparametara te jednostavnijeg korištenja u Kaggle okruženju koje je korišteno za treniranje modela. Iako `pytorch-openpose` projekt implementira istu arhitekturu kao originalni model, rezultati prikazani u idućem poglavlju pokazuju da ova implementacija nije identična originalnom OpenPose modelu. Prilikom konverzije moguće je da su neki slojevi izostavljeni i/ili dodani, podaci drugačije normalizirani i sl. Ova preslikana verzija OpenPose modela u PyTorchu omogućuje klasičnu funkcionalnost detekcije ljudskih poza, ali ne dolazi s unaprijed definiranim alatima za treniranje. Zbog toga je bilo potrebno implementirati cijeli postupak treniranja COCO modela.

Za implementaciju prijenosnog učenja napisan je kod koji omogućuje treniranje PyTorch OpenPose COCO modela u Kaggle okruženju, koristeći dvije NVIDIA T4 GPU kartice. Parametri treniranja podešeni su na način da bi se omogućila stabilna konvergencija modela jer su mnogi parametri negativno utjecali na konačnu inferenciju modela. Koristio se mini-skup (engl. *batch*) veličine 8 te Adam optimizator s inicijalnom stopom učenja od 1e-5, a kao funkcija gubitka primijenjeno je srednje kvadratno odstupanje (engl. *mean squared error, MSE*). Da bi se sačuvale reprezentacije naučene na RGB COCO skupu podataka, a i uštedili resursi, zamrznuti su svi slojevi modela osim posljednjih šest koji su dotrenirani na novim podacima. Trening se odvijao kroz 10 epoha, pri čemu se u svakoj iteraciji slike i pripadajuće oznake učitavaju, prosljeđuju kroz model, računa se gubitak te se ažuriraju težine modela. Regularizacija kroz propadanje težina (engl. *weight decay*) nije primijenjena jer nije bilo potrebno s obzirom na veličinu skupa podataka i druge vrste regularizacije.

4. Rezultati i analiza

U ovom poglavlju prikazani su rezultati istraživanja provedenog u sklopu diplomskog rada. Glavni je fokus na uspješnosti treniranja COCO modela na NIR podacima, s naknadnom usporedbom tih rezultata s modelima treniranim na RGB podacima transformiranim pomoću crvenog kanala, kao i s modelima treniranim na RGB podacima transformiranim korištenjem težinskih faktora. Analiza rezultata provedena je klasičnim metodama detekcije ključnih točaka. Osim kvantitativne analize, u poglavlju su razmotreni i izazovi koji su se pojavili tijekom istraživanja, kao i mogući smjerovi za daljnji razvoj modela, odnosno cjelokupnog projekta.

4.1. Kvantitativni rezultati primjene prijenosnog učenja

Za evaluaciju modela korištena je popularna biblioteka za evaluaciju COCO podataka `pycocotools`, točnije `pycocotools.eval` evaluacijski modul. Ovaj modul, osim procjene točnosti prepoznavanja ključnih točaka, omogućuje kvantitativnu analizu performansi modela i za detekciju i segmentaciju objekata. Zbog toga je i često korišten uz COCO skup podataka jer omogućuje analizu svih vrsta anotacija unutar samog skupa. Za analizu ključnih točaka koristi standardne COCO metrike [66], poput prosječne preciznosti (engl. *Average Precision, AP*) i prosječnog odziva (engl. *Average Recall, AR*), koji se računaju za različite pragove metrike preklapanja područja (engl. *Intersection over Union, IoU*) i veličine objekata.

Glavne metrike koje `pycocotools.eval` izračunava uključuju:

- **IoU** – mjera preklapanja između predikcije modela i stvarne oznake (engl. *ground truth*) koja se koristi za procjenu točnosti modela. Metrika IoU računa se kao omjer

površine zajedničkog područja predikcije i stvarne oznake prema površini njihove unije. Standardni pragovi alata su:

  - **0.50** – gdje se predikcija smatra točnom samo ako pokriva barem polovicu stvarne oznake,
  - **0.75** – gdje se predikcija smatra točnom samo ako pokriva većinu stvarne oznake,
  - **0.50:0.95** – prosječna vrijednost IoU računana kroz različite pragove koja omogućuje procjenu performansi modela na širem rasponu točnosti preklapanja.

- **Površina (engl. *Area*)** – površina objekata za koje se računaju metrike. Ovisno o veličini objekata, razvrstavaju se u tri kategorije:
  - **all** – svi objekti u skupu podataka bez obzira na njihovu veličinu,
  - **medium** – objekti srednje veličine koji su u određenom rasponu dimenzija,
  - **large** – veliki objekti koji zauzimaju veću površinu u skupu podataka.
- **MaxDets** – maksimalan broj dozvoljenih detekcija po slici, ukodirana (engl. *hard-coded*) vrijednost za evaluaciju ključnih točaka.
- **AP - Average Precision** – mjeri preciznost modela (koliko su predikcije točne u odnosu na stvarne oznake).
- **AR - Average Recall** – mjeri odziv modela (koliko je stvarnih ključnih točaka uspješno detektirano).

Pri korištenju modula `pycocotools.eval` za evaluaciju ključnih točaka važno je napomenuti da su određeni parametri unutar implementacije ukodirani. Jedan od takvih parametara je `maxDets=20`, koji ograničava broj detekcija uzetih u obzir pri izračunu evaluacijskih metrika. Ova vrijednost definirana je unutar funkcije `_summarizeKps()` u službenoj COCO API implementaciji [67]. Konkretno, sve relevantne metrike za ključne točke, uključujući `AP`, `AR`, `iouThr=.5`, `iouThr=.75` te podjele prema veličini objekata računaju se s maksimalno 20 detekcija po slici. Ovaj limit utječe na rezultate, a pogotovo u situacijama gdje je veći broj predikcija.

Tablica 4.1. Rezultati evaluacije modela pomoću `pycocotools.eval`

Metrika	**IoU**	**Veličina područja**	**Max Dets**	**Bazni model**	**Pytorch model**	**NIR model**	**R-kanal model**	**Transformirani model**
AP	0.50:0.95	all	20	0.275	0.190	0.232	0.191	0.200
AP	0.50	all	20	0.504	0.404	0.435	0.415	0.419
AP	0.75	all	20	0.267	0.159	0.187	0.166	0.172
AP	0.50:0.95	medium	20	0.027	0.006	0.013	0.005	0.007
AP	0.50:0.95	large	20	0.448	0.324	0.361	0.337	0.339
AR	0.50:0.95	all	20	0.308	0.233	0.276	0.242	0.243
AR	0.50	all	20	0.516	0.431	0.460	0.434	0.447
AR	0.75	all	20	0.308	0.219	0.240	0.226	0.226
AR	0.50:0.95	medium	20	0.043	0.016	0.017	0.016	0.016
AR	0.50:0.95	large	20	0.504	0.391	0.444	0.399	0.405

Prije analize rezultata, važno je napomenuti da bi za kvalitetniji zaključak bilo potrebno ne samo optimizirati hiperparametre modela i trenirati ga na znatno većem skupu podataka, već i dodatno unaprijediti sam kod za treniranje kako bi se osigurala stabilnija konvergencija.

Rezultati prikazani u tablici 4.1. pokazuju da je model dotreniran na stvarnim NIR podacima postigao najbolje rezultate među prilagođenim modelima. Modeli trenirani na transformiranim podacima postigli su slabije rezultate, no i dalje vrlo korisne. Očekivano, model treniran na transformiranim RGB slikama težinskom kombinacijom kanala nije nadmašio model treniran na stvarnim NIR slikama. Međutim, ostvareni rezultati sugeriraju da bi ovakav pristup mogao biti dobra alternativa kada izvorni NIR podaci nisu dostupni. Nadalje, može poslužiti i kao dopuna skupu podataka koji sadrži NIR podatke da bi se na ovaj način dodatno proširio skup drugačijim scenarijima i na taj način poboljšala sposobnost modela za generalizaciju na različite uvjete nastalih snimki.

Zanimljivo je primijetiti da model treniran na težinskoj kombinaciji RGB kanala nije postigao značajno bolje rezultate od modela treniranog isključivo na crvenom kanalu. Ipak, postoje razlike, što upućuje na to da bi dodatna optimizacija težinskih faktora mogla poboljšati konačne performanse modela. U situacijama kada nije moguće pronaći optimalne težinske kombinacije, može se preporučiti korištenje provjerenih težinskih faktora. To mogu biti faktori izračunati u ovom radu ili klasični faktori za konverziju u sivu sliku, što se može pokazati boljim rješenjem od korištenja isključivo crvenog kanala.

4.2. Izazovi

Prilikom pripreme i primjene projekta, odnosno modela za praćenje ljudskih pokreta te njegovog treniranja nad NIR podacima, pojavilo se više različitih izazova.

Jedan od prvih izazova bio je OpenPose projekt koji je koncipiran na način da se veliki dio projekta poput modela i biblioteka kao što su Caffe i OpenCV moraju dohvatiti putem pred-definiranih skripti. Problem je nastao kada lokacija s koje su se dohvaćale ove komponente više nije bila dostupna. Iako je ista greška već ranije riješena promjenom poveznice za dohvaćanje u kolovozu 2024. godine, ubrzo je i nova poveznica prestala funkcionirati [68].

Drugi problem bio je pronalazak kvalitetnog alata za anotaciju ključnih točaka. Mnogi besplatni alati često su ograničeni u funkcionalnostima, osobito kada je riječ o anotaciji ključnih točaka. Primjer je toga nemogućnost definiranja vidljivosti ključne točke, zbog čega se ta informacija mora ručno dodavati nakon izvoza podataka. Dodatnu otežanost predstavljali su specifični formati izvoza, osobito kada bi alat samostalno generirao nasumične ID varijable i/ili mijenjao nazive slika, što je usporavalo povezivanje anotacija s izvornim podacima. Osim toga, neki alati imali su česte greške u kodu, a to je onemogućavalo očekivani proces anotacije.

S time je povezan i problem vremena potrebnog za anotaciju. Ručno označavanje ključnih točaka na velikom broju slika zahtijevalo je značajan vremenski angažman, posebno jer nije bilo dostupnih automatiziranih metoda anotacije ili mogućnosti uvoza OpenPose anotacija modela kao početne točke.

Računalni resursi i radni okvir Caffe predstavljali su još jedan bitan izazov. Lokalno pokretanje projekta OpenPose bilo je iznimno računalno zahtjevno, zbog čega lokalno dotreniravanje nije bilo izvedivo, a pristup dodatnim resursima nije bio dostupan. S druge strane, Caffe nije kvalitetno dokumentiran kao što su drugi, noviji radni okviri dubokog učenja poput TensorFlowa, PyTorcha ili Kerasa. Također, ovaj radni okvir nije niti toliko fleksibilan u usporedbi s novijim radnim okvirima, a to komplicira prilagodbu modela i eksperimentiranje s različitim postavkama. Problemi su nastajali i pri treniranju PyTorch implementacije modela. Javljali su se izazovi poput potpunog gubitka sposobnosti modela za prepoznavanje ključnih točaka nakon prvih nekoliko iteracija treniranja,

izuzetno sporog smanjenja funkcije gubitka te nepredvidivog ponašanja optimizatora.

Jedan od izazova bio je i ograničeni skup podataka. Skup podataka bilo je moguće snimiti samo u laboratorijskim uvjetima, a bio je ograničen i vremenom za anotaciju tih podataka, što smanjuje robusnost generalizacije modela. Također, skup je nastao OptiTrack kamerama koje snimaju u nižoj rezoluciji zbog čega je proces anotacije, koji je i sam po sebi dugotrajan, potrajao dulje.

4.3. Daljnji razvoj

Nekoliko je ideja kako nastaviti daljnji razvoj trenutnog projekta koji mogu značajno unaprijediti rezultate. Prvi je korak proširivanje NIR skupa podataka. Veći i kvalitetniji skup podataka donio bi mnogo bolju generalizaciju modela i učinio ga kvalitetnijim u stvarnim uvjetima. To bi uključivalo snimanje većeg broja uzoraka u različitim uvjetima kao što su više različitog broja i tipova ljudi, u različitim razinama osvjetljenja, kutovima gledanja, okolinama, bojama, itd. Iako je testiranje bazične verzije OpenPose COCO modela pokazalo kako model ostvaruje dobre rezultate, tek bi se u ovakvim, stvarnim uvjetima vidjelo koliko je model robustan.

Drugi važan aspekt daljnjeg razvoja bila bi optimizacija težinskih faktora. U ovome radu pronađeni su težinski faktori, međutim, moguće ih je kao i skup podataka dodatno optimizirati s većim rasponom boja, kamera, drugih optimizacijskih metoda i sl. Nastavno na optimizaciju težinskih faktora, optimizacija hiperparametara modela bila bi ključna za postizanje boljih rezultata. Iako su isprobani različiti hiperparametri, nije provedena njihova optimizacija uz skup za validaciju. Time bi se postigla bolja generalizacija, ali i robusnost modela pri detekciji ključnih točaka.

U konačnici, bilo bi korisno dotrenirati i druge OpenPose modele te ih međusobno usporediti, ali i usporediti s drugim modelima za detekciju ljudskih pokreta. Također, analiza bi mogla uključivati usporedbu s već predtreniranim modelima poput AlphaPose-a, PoseNet-a i MoveNet-a, kao i s modelima razvijenima od nule, koristeći alternativne arhitekture i različite pristupe treniranju.

5. Zaključak

U posljednjih nekoliko godina ostvaren je izuzetan napredak u razvoju umjetne inteligencije. Najveći napredak između raznih metoda strojnog učenja upravo je ostvaren u području neuronskih mreža, odnosno dubokih modela. Svoju popularnost, neuronske mreže ostvarile su zbog odličnih rezultata kod složenijih problema poput prepoznavanja govora, obrade prirodnog jezika i prepoznavanja slika. Konvolucijske neuronske mreže jedan su tip neuronskih mreža koji se pokazao sjajnim za probleme poput računalnog vida. No, za njihovo učenje potrebne su ogromne količine podataka, velika računalna snaga i poprilična količina vremena. Takav tip učenja, koji zahtijeva učenje iznova za svaku novu primjenu modela, naziva se još i tradicionalno strojno učenje. Zbog nedostataka takvog pristupa učenju javila se potreba za drugačijim načinom učenja čime je nastalo prijenosno učenje.

Cilj diplomskog rada bio je primijeniti prijenosno učenje na praktičnom primjeru. Koristeći metode prijenosnog učenja, ideja je bila unaprijediti model za prepoznavanje ljudskih pokreta - OpenPose, sa slikama u blisko infracrvenom spektru. Osim toga, ideja je bila vidjeti je li moguće te slike simulirati, stoga su obuhvaćena tri skupa podataka - skup podataka napravljen NIR kamerama u laboratorijskim prostorijama FER-a, RGB skup podataka (dio COCO skupa podataka) koji je transformiran koristeći samo crveni kanal te RGB skup podataka transformiran koristeći transformacije kanala da bi se bolje simulirao NIR spektar. Skup podataka sastojao se od 2480 slika raspoređenih u omjeru od 75:25, odnosno 1860 slika je korišteno za treniranje, a 620 za testiranje modela. Za analizu modela korištene su standardizirane metrike COCO formata poput AP i AR nad različitim pragovima IoU.

Implementacija prijenosnog učenja provedena je na PyTorch inačici OpenPose modela u Kaggle okruženju koristeći dvije NVIDIA T4 GPU kartice. Da bi se očuvale pret-

hodno naučene reprezentacije, svi slojevi osim posljednjih šest bili su zamrznuti. Model je treniran koristeći Adam optimizator i srednju kvadratnu pogrešku, uz stopu učenja od 1e-5 tijekom 10 epoha, pri čemu težinsko propadanje nije primijenjeno zbog prisutnosti drugih tipova regularizacije. Evaluacija je pokazala da je model treniran na stvarnim NIR slikama, očekivano, ostvario najbolje rezultate. Transformirani podaci također su se pokazali korisnima, posebno model treniran težinskom kombinacijom kanala, koji je nadmašio model treniran samo na crvenom kanalu. Ovo sugerira da se transformirane slike mogu koristiti kao alternativa u nedostatku NIR podataka, ali i kao dopuna postojećim skupovima za bolju generalizaciju modela na različite scenarije.

Literatura

[1] I. Goodfellow, Y. Bengio, i A. Courville, *Deep Learning*. The MIT Press, 2016.

[2] D. Katić, "Što je duboko učenje (deep learning)", 11 2024. [Mrežno]. Adresa: https://www.dalibor-katic.com/2024/11/01/sto-je-duboko-ucenje-deep-learning/

[3] A. Sushumna, "Exploring deep learning in natural language processing and image recognition", 5DATA INC., 6 2024. [Mrežno]. Adresa: https://5datainc.com/exploring-deep-learning-in-natural-language-processing-and-image-recognition/

[4] S. J. Pan i Q. Yang, "A survey on transfer learning", *IEEE Transactions on Knowledge and Data Engineering*, sv. 22, br. 10, str. 1345–1359, listopad 2010. https://doi.org/10.1109/TKDE.2009.191

[5] Anon, "Real-world transfer learning examples", restack, 6 2025. [Mrežno]. Adresa: https://www.restack.io/p/transfer-learning-answer-real-world-examples-cat-ai

[6] ——, "Nir cameras in embedded vision – advantages and applications", TechNexion. [Mrežno]. Adresa: https://www.technexion.com/resources/nir-cameras-in-embedded-vision-advantages-and-applications/

[7] ——. About optitrack. OptiTrack. [Mrežno]. Adresa: https://optitrack.com/about/

[8] Z. Cao, G. Hidalgo, T. Simon, S.-E. Wei, i Y. Sheikh, "Openpose: Realtime multi-person 2d pose estimation using part affinity fields", 2019. [Mrežno]. Adresa: https://arxiv.org/abs/1812.08008

[9] S. Russell i P. Norvig, *Artificial Intelligence: A Modern Approach, Global Edition*, 4. izd., S. Russell i P. Norvig, Ur. Pearson Education, 4 2021., sv. 1167.

[10] M. T. Augustine, “A survey on universal approximation theorems”, 2024. [Mrežno]. Adresa: https://arxiv.org/abs/2407.12895

[11] W. contributors, “Artificial intelligence — Wikipedia, the free encyclopedia”, 2025. [Mrežno]. Adresa: https://en.wikipedia.org/w/index.php?title=Artificial_intelligence&oldid=1274941297

[12] K. Casey. (2020., 11) How to explain machine learning in plain english. The Enterprisers Project. [Mrežno]. Adresa: https://enterprisersproject.com/article/2019/7/machine-learning-explained-plain-english

[13] P. Domingos, “A few useful things to know about machine learning”, *Commun. ACM*, sv. 55, br. 10, str. 78–87, listopad 2012. https://doi.org/10.1145/2347736.2347755

[14] K. Goel. Supervised learning vs unsupervised learning vs reinforcement learning. IntelliPaat. [Mrežno]. Adresa: https://intellipaat.com/blog/supervised-learning-vs-unsupervised-learning-vs-reinforcement-learning/

[15] G. N. Yannakakis i J. Togelius, *Artificial Intelligence and Games*, G. N. Yannakakis i J. Togelius, Ur. Springer International Publishing, 2 2018., sv. 337.

[16] R. Szeliski, *Computer Vision: Algorithms and Applications*, 2. izd. Springer International Publishing, 11 2021.

[17] T. Talaei Khoei i N. Kaabouch, “Machine learning: Models, challenges, and research directions”, *Future Internet*, sv. 15, br. 10, 2023. https://doi.org/10.3390/fi15100332

[18] Anon. (2024., 4) Kako se postupak izdvajanja značajki u konvolucijskoj neuronskoj mreži (cnn) primjenjuje na prepoznavanje slike? EUROPSKA AKADEMIJA ZA CERTIFIKACIJU INFORMACIJSKIH TEHNOLOGIJA. [Mrežno]. Adresa: https://hr.eitca.org/umjetna-inteligencija/eitc-ai-tff-tensorflow-osnove/tensorflow-js/pomo%C4%87u-tensorflow-a-za-klasificiranje-slika-odje%C4%87e/kako-se-proces-izdvajanja-zna%C4%8Dajki-u-konvolucijskoj-neuronskoj-mre%C5%BEi-cnn-primjenjuje-na-prepoznavanje-slike/

[19] G. Hidalgo, Z. Cao, T. Simon, S.-E. Wei, Y. Raaj, H. Joo, i Y. Sheikh, "Openpose", Carnegie Mellon University. [Mrežno]. Adresa: https://github.com/CMU-Perceptual-Computing-Lab/openpose

[20] G. James, D. Witten, T. Hastie, R. Tibshirani, i J. Taylor, *An Introduction to Statistical Learning with Applications in Python*, ser. Springer Texts in Statistics. Cham: Springer, 2023. https://doi.org/10.1007/978-3-031-38747-0

[21] P. Baheti, "A newbie-friendly guide to transfer learning", V7, 10 2021. [Mrežno]. Adresa: https://www.v7labs.com/blog/transfer-learning-guide

[22] P. Marcelino, "Transfer learning from pre-trained models", Towards Data Science, 8 2018. [Mrežno]. Adresa: https://towardsdatascience.com/transfer-learning-from-pre-trained-models-f2393f124751

[23] J. Plested i T. Gedeon, "Deep transfer learning for image classification: a survey", 5 2022. https://doi.org/10.48550/arXiv.2205.09904

[24] Q. Yang, Y. Zhang, W. Dai, i S. J. Pan, *Transfer Learning*. Cambridge University Press, 1 2020.

[25] A. Frome, G. S. Corrado, J. Shlens, S. Bengio, J. Dean, M. A. Ranzato, i T. Mikolov, "Devise: A deep visual-semantic embedding model", u *Advances in Neural Information Processing Systems*, C. Burges, L. Bottou, M. Welling, Z. Ghahramani, i K. Weinberger, Ur., sv. 26. Curran Associates, Inc., 2013. [Mrežno]. Adresa: https://proceedings.neurips.cc/paper_files/paper/2013/file/7cce53cf90577442771720a370c3c723-Paper.pdf

[26] fchollet, "Transfer learning and fine-tuning", Keras, 5 2020. [Mrežno]. Adresa: https://keras.io/guides/transfer_learning/

[27] DeepLearningAI, "Transfer learning (c3w2l07)", YouTube, 8 2017. [Mrežno]. Adresa: https://www.youtube.com/watch?v=yofjFQddwHE

[28] Anon, "Imagenet", Stanford University, Princeton University. [Mrežno]. Adresa: https://image-net.org/download.php

[29] T. Ridnik, E. Ben-Baruch, A. Noy, i L. Zelnik-Manor, “Imagenet-21k pretraining for the masses”, 2021. [Mrežno]. Adresa: https://arxiv.org/abs/2104.10972

[30] Anon, “About imagenet”. [Mrežno]. Adresa: https://www.image-net.org/about.php

[31] C. Consortium, “Coco”. [Mrežno]. Adresa: https://cocodataset.org//#home

[32] M. Andriluka, L. Pishchulin, P. Gehler, i B. Schiele, “2d human pose estimation: New benchmark and state of the art analysis”, u *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.

[33] Anon, “Coco-wholebody”. [Mrežno]. Adresa: https://github.com/jin-s13/COCO-WholeBody/tree/master

[34] J. Solawetz, “An introduction to the coco dataset”, roboflow, 10 2020. [Mrežno]. Adresa: https://blog.roboflow.com/coco-dataset/

[35] Microsoft, “Coco dataset computer vision project”, Robowflow. [Mrežno]. Adresa: https://universe.roboflow.com/microsoft/coco

[36] M. Andriluka, L. Pishchulin, P. Gehler, i B. Schiele, “2d human pose estimation: New benchmark and state of the art analysis”, u *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.

[37] R. Pytel. (2023., 1) Human pose estimation 2023 guide. SoftwareMill. [Mrežno]. Adresa: https://softwaremill.com/human-pose-estimation-2023-guide/

[38] Anon. Pose computation overview. OpenCV. [Mrežno]. Adresa: https://docs.opencv.org/4.x/d5/d1f/calib3d_solvePnP.html

[39] ——, “Iterative closest point”, Wikipedia. [Mrežno]. Adresa: https://en.wikipedia.org/wiki/Iterative_closest_point

[40] ——, “Algorithmic state machine”, Wikipedia. [Mrežno]. Adresa: https://en.wikipedia.org/wiki/Algorithmic_state_machine

[41] ——, “Speeded up robust features”, Wikipedia. [Mrežno]. Adresa: https://en.wikipedia.org/wiki/Speeded_up_robust_features

[42] ——, “Scale-invariant feature transform”, Wikipedia. [Mrežno]. Adresa: https://en.wikipedia.org/wiki/Scale-invariant_feature_transform

[43] ——. Deepposekit. [Mrežno]. Adresa: https://github.com/jgraving/DeepPoseKit

[44] ——. Official yolov7. [Mrežno]. Adresa: https://github.com/airhors/yolov7-pose

[45] ——. Ultralytics yolov8. [Mrežno]. Adresa: https://huggingface.co/Ultralytics/YOLOv8

[46] ——. Mediapipe pose. [Mrežno]. Adresa: https://github.com/google-ai-edge/mediapipe/blob/master/docs/solutions/pose.md

[47] ——. Opencv. OpenCV. [Mrežno]. Adresa: https://opencv.org/

[48] ——. Caffe. Caffe. [Mrežno]. Adresa: https://caffe.berkeleyvision.org/

[49] ——. Cuda toolkit. NVIDIA. [Mrežno]. Adresa: https://developer.nvidia.com/cuda-toolkit

[50] ——. Motive documentation. OptiTrack. [Mrežno]. Adresa: https://wiki.optitrack.com/index.php?title=Motive_Documentation

[51] ——. Motive optical motion capture software. OptiTrack. [Mrežno]. Adresa: https://optitrack.com/software/motive/

[52] ——. Flex 3. OptiTrack. [Mrežno]. Adresa: https://optitrack.com/cameras/flex-3/

[53] ——. (2007., 4) Optics and photonics — spectral bands. International Organization for Standardization. [Mrežno]. Adresa: https://cdn.standards.iteh.ai/samples/39482/bd35b5e41d7644e098b071636c05305b/ISO-20473-2007.pdf

[54] A. Salazar, “Nir spectroscopy: What it is, principles, advantages, and applications”, AZO Materials, 2 2024. [Mrežno]. Adresa: https://www.azom.com/article.aspx?ArticleID=23386

[55] P. Kumar, “What is near-infrared imaging and how do nir cameras work?” e-con Systems, 10 2024. [Mrežno]. Adresa: https://www.e-consystems.com/blog/camera/technology/what-is-nir-imaging-and-how-do-nir-cameras-work/

[56] P. A. Triolo, “Implementation of the diagnostic capabilities of the cmos sensor in the nir environment, using 1070 nm interference filter and a conventional ir-pass filters set”, *Journal of Cultural Heritage*, sv. 70, str. 54–63, 2024. https://doi.org/https://doi.org/10.1016/j.culher.2024.08.007

[57] Anon. Ffmpeg. [Mrežno]. Adresa: https://www.ffmpeg.org/

[58] ——, “Leading data annotation platform”, CVAT. [Mrežno]. Adresa: https://www.cvat.ai/

[59] ——, “Computer vision annotation tool”, Wikipedia. [Mrežno]. Adresa: https://en.wikipedia.org/wiki/Computer_Vision_Annotation_Tool

[60] F. T., “Train pose detection yolov8 on custom data | keypoint detection | computer vision tutorial”, YouTube, 4 2023. [Mrežno]. Adresa: https://www.youtube.com/watch?v=gA5N54IO1ko

[61] Anon, “Fiftyone”, VOXEL51. [Mrežno]. Adresa: https://docs.voxel51.com/

[62] T. Dutta i K. Dawn, “Object keypoint similarity in keypoint detection”, LearnOpenCV by Big Vision, 8 2023. [Mrežno]. Adresa: https://learnopencv.com/object-keypoint-similarity/

[63] Anon, “Openpose”, Carnegie Mellon University. [Mrežno]. Adresa: https://github.com/CMU-Perceptual-Computing-Lab/openpose/issues/1602

[64] ——, “pytorch-openpose”, Fudan University. [Mrežno]. Adresa: https://github.com/Hzzone/pytorch-openpose

[65] V. Kantorov, “caffemodel2pytorch”. [Mrežno]. Adresa: https://github.com/vadimkantorov/caffemodel2pytorch

[66] C. Consortium, “Coco”. [Mrežno]. Adresa: https://cocodataset.org/#keypoints-eval

[67] ——, “Coco api”. [Mrežno]. Adresa: https://github.com/cocodataset/cocoapi/blob/master/PythonAPI/pycocotools/cocoeval.py

[68] Anon, “Openpose”, Carnegie Mellon University. [Mrežno]. Adresa: https://github.com/CMU-Perceptual-Computing-Lab/openpose/commit/5c5d96523ef917bd30301245fdc8343937cae48d

Sažetak

Primjena prijenosnog učenja za prilagodbu metoda računalnog vida blisko-infracrvenim kamerama

Mihael Kožul

Praćenje ljudskih pokreta nalazi primjenu u raznim područjima poput biomehanike, sporta, virtualne stvartnosti i nadzora. U ovom radu istražuje se prilagodba modela OpenPose za analizu pokreta pomoću blisko infracrvenih (NIR) kamera. Standardni OpenPose model treniran je na RGB slikama, što može dovesti do smanjene preciznosti pri radu s NIR slikama. Da bi se to ispitalo, provodi se prijenosno učenje nad OpenPose modelom koristeći snimke dobivene NIR kamerama te RGB slike transformirane tako da oponašaju NIR spektar. Eksperimentalni dio uključuje usporedbu modela treniranog na RGB podacima, modela treniranog na stvarnim NIR podacima te modela treniranog na transformiranim RGB slikama. Transformacije uključuju uzimanje samo crvenog kanala RGB slike te korištenje težinske kombinacije kanala za NIR simulaciju. Rezultati pokazuju da model treniran na stvarnim NIR slikama postiže najbolje rezultate, no modeli s transformiranim podacima također ostvaruju solidne performanse. Model treniran težinskom kombinacijom RGB kanal pokazao se boljim od modela treniranog isključivo crvenim kanalom, ali nije mnogo lošiji od najboljeg modela, što sugerira njegovu primjenjivost kao alternativa ili dopuna NIR skupa podataka.

Ključne riječi: prijenosno učenje; OpenPose; Caffe; PyTorch; NIR format; fino podešavanje

Abstract

Application of Transfer Learning for Adapting Computer Vision Methods to Near-Infrared Cameras

Mihael Kožul

Tracking human movements finds application in various fields such as biomechanics, sports, virtual reality and surveillance. This paper explores the adaptation of the OpenPose model for motion analysis using near-infrared (NIR) cameras. The standard OpenPose model is trained on RGB images, which can lead to reduced accuracy when working with NIR images. To test this, transfer learning is performed on the OpenPose model using images obtained from NIR cameras and RGB images transformed to mimic the NIR spectrum. The experimental part includes a comparison of a model trained on RGB data, a model trained on real NIR data, and a model trained on transformed RGB images. The transformations include taking only the red channel of the RGB image and using a weighted combination of channels for the NIR simulation. The results show that the model trained on real NIR images achieves the best results, but models with transformed data also achieve solid performance. The model trained with the RGB channel weight combination performed better than the model trained exclusively with the red channel, but not much worse than the best model, suggesting its applicability as an alternative or supplement to the NIR data set.

Keywords: transfer learning; OpenPose; Caffe; PyTorch; NIR format; fine-tuning

Privitak A: Kod

Isječak kôda 1.1: FFmpeg python skripta za izvlačenje slika iz videa

```python
import os
import argparse
import subprocess

def extract_frames(input_path, output_path, fps=1, quality=2):
    if not os.path.exists(output_path):
        os.makedirs(output_path)

    for file_name in os.listdir(input_path):
        if file_name.endswith(".avi"):
            input_file = os.path.join(input_path, file_name)

            base_name = os.path.splitext(file_name)[0]
            output_file_pattern = os.path.join(output_path, f"{base_name}_%05d.png")

            command = [
                "ffmpeg",
                "-i", input_file,
                "-vf", f"fps={fps}",
                "-q:v", str(quality),
                output_file_pattern
            ]

            print(f"Running: {' '.join(command)}")
            subprocess.run(command, check=True)

if __name__ == "__main__":
    parser = argparse.ArgumentParser()
    parser.add_argument(
        "input_path",
        type=str
    )
    parser.add_argument(
        "output_path",
        type=str
    )
    parser.add_argument(
        "--fps",
        type=int,
        default=1
    )
    parser.add_argument(
        "--quality",
        type=int,
        default=1
    )

```

```
    args = parser.parse_args()

    extract_frames(args.input_path, args.output_path, args.fps, args.
    quality)
```

Isječak kôda 1.2: Python skripta za pretvorbu CVAT izvoza u skup podataka u COCO formatu

```
import os
import json
import xml.etree.ElementTree as ET

OPENPOSE_TO_COCO = {
    0: 0,   # nose -> nose
    2: 6,   # right_shoulder -> right_shoulder
    3: 8,   # right_elbow -> right_elbow
    4: 10,  # right_wrist -> right_wrist
    5: 5,   # left_shoulder -> left_shoulder
    6: 7,   # left_elbow -> left_elbow
    7: 9,   # left_wrist -> left_wrist
    8: 12,  # right_hip -> right_hip
    9: 14,  # right_knee -> right_knee
    10: 16, # right_ankle -> right_ankle
    11: 11, # left_hip -> left_hip
    12: 13, # left_knee -> left_knee
    13: 15, # left_ankle -> left_ankle
    14: 2,  # right_eye -> right_eye
    15: 1,  # left_eye -> left_eye
    16: 4,  # right_ear -> right_ear
    17: 3   # left_ear -> left_ear
}

def is_near_zero(x, y, threshold=5):
    return x <= threshold and y <= threshold

def parse_cvat_annotations(xml_file):
    tree = ET.parse(xml_file)
    root = tree.getroot()

    categories = [{
        "id": 1,
        "name": "person",
        "supercategory": "person",
        "keypoints": [
            "nose", "left_eye", "right_eye", "left_ear", "right_ear",
            "left_shoulder", "right_shoulder", "left_elbow", "
    right_elbow",
            "left_wrist", "right_wrist", "left_hip", "right_hip",
            "left_knee", "right_knee", "left_ankle", "right_ankle"
        ],
        "skeleton": [
            [0, 1], [1, 3], [0, 2], [2, 4], [5, 6], [5, 7], [7, 9],
            [6, 8], [8, 10], [5, 11], [6, 12], [11, 12], [11, 13],
            [13, 15], [12, 14], [14, 16]
        ]
    }]

    images = []
    annotations = []
    annotation_id = 1

    for i, image_elem in enumerate(root.findall("image")):
        image_id = i
```

```
        original_file_name = image_elem.attrib["name"]
        file_name = f"{i:04d}.png"
        width = int(image_elem.attrib["width"])
        height = int(image_elem.attrib["height"])

        images.append({
            "id": image_id,
            "file_name": file_name,
            "original_file_name": original_file_name,
            "width": width,
            "height": height
        })

        for points_elem in image_elem.findall("points"):
            points = list(map(float, points_elem.attrib["points"].
    replace(";", ",").split(",")))

            if len(points) < 36:
                print(f"Upozorenje: Slika '{original_file_name}' ima
    samo {len(points) // 2} tocaka! Preskacem...")

                continue

            keypoints_reordered = [0] * 51
            num_keypoints = 0
            x_min, y_min = float("inf"), float("inf")
            x_max, y_max = float("-inf"), float("-inf")

            for openpose_idx, coco_idx in OPENPOSE_TO_COCO.items():
                x, y = points[openpose_idx * 2], points[openpose_idx *
     2 + 1]
                if is_near_zero(x, y):
                    keypoints_reordered[coco_idx * 3 : coco_idx * 3 +
    3] = [0, 0, 0]
                else:
                    keypoints_reordered[coco_idx * 3 : coco_idx * 3 +
    3] = [x, y, 2]
                    num_keypoints += 1
                    x_min = min(x_min, x)
                    y_min = min(y_min, y)
                    x_max = max(x_max, x)
                    y_max = max(y_max, y)

            if x_min == float("inf") or y_min == float("inf") or x_max
     == float("-inf") or y_max == float("-inf"):
                bbox = [0, 0, 0, 0]
                area = 0
            else:
                bbox = [x_min, y_min, x_max - x_min, y_max - y_min]
                area = bbox[2] * bbox[3]

            annotations.append({
                "id": annotation_id,
                "image_id": image_id,
                "category_id": 1,
                "keypoints": keypoints_reordered,
                "num_keypoints": num_keypoints,
                "iscrowd": 0,
                "bbox": bbox,
                "area": area
            })

            annotation_id += 1

```

```
    return {
        "info": {
            "description": "Dataset with custom keypoint order (excluding neck)",
            "version": "1.0",
            "year": 2025
        },
        "licenses": [],
        "categories": categories,
        "images": images,
        "annotations": annotations
    }


def save_to_coco_format(xml_path, output_json_path):
    coco_data = parse_cvat_annotations(xml_path)

    with open(output_json_path, "w") as json_file:
        json.dump(coco_data, json_file, indent=4)
        print(f"COCO annotations saved to {output_json_path}")


if __name__ == "__main__":
    import argparse

    parser = argparse.ArgumentParser()
    parser.add_argument("xml_path", type=str)
    parser.add_argument("output_json_path", type=str)

    args = parser.parse_args()

    save_to_coco_format(args.xml_path, args.output_json_path)
```