|  |  |  |
| --- | --- | --- |
|  | UNIVERZITET U NOVOM SADU  **FAKULTET TEHNIČKIH NAUKA U NOVOM SADU** |  |

Иван Мршуља

**Праћење и препознавање геста шаке комбинацијом неуронских и традиционалних приступа**

Дипломски рад

- Основне академске студије -

Нови Сад, 2022.

|  |  |  |
| --- | --- | --- |
|  | УНИВЕРЗИТЕТ У НОВОМ САДУ  **ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА**  21000 НОВИ САД, Трг Доситеја Обрадовића 6 | Датум: |
|  |
| **ЗАДАТАК ЗА ИЗРАДУ ДИПЛОМСКОГ (BACHELOR) РАДА** | Лист: |
| 1/1 |

*(Податке уноси предметни наставник - ментор)*

| Врста студија: | **Основне академске студије** |
| --- | --- |
| Студијски програм: | **Софтверско инжењерство и информационе технологије** |
| Руководилац студијског програма: | **проф. др Мирослав Зарић** |

|  |  |  |  |
| --- | --- | --- | --- |
| Студент: | **Иван Мршуља** | Број индекса: | **SW 65/2018** |
| Област: | **Електротехничко и рачунарско инжењерство** | | |
| Ментор: | **Др Јелена Сливка, редовни професор** | | |
| НА ОСНОВУ ПОДНЕТЕ ПРИЈАВЕ, ПРИЛОЖЕНЕ ДОКУМЕНТАЦИЈЕ И ОДРЕДБИ СТАТУТА ФАКУЛТЕТА ИЗДАЈЕ СЕ ЗАДАТАК ЗА ДИПЛОМСКИ РАД, СА СЛЕДЕЋИМ ЕЛЕМЕНТИМА:   * проблем – тема рада; * начин решавања проблема и начин практичне провере резултата рада, ако је таква провера неопходна; * литература | | | |

**НАСЛОВ ДИПЛОМСКОГ (BACHELOR) РАДА:**

|  |
| --- |
| **Праћење и препознавање геста шаке комбинацијом неуронских и традиционалних приступа** |

**ТЕКСТ ЗАДАТКА:**

|  |
| --- |
| 1. Анализирати стање у области.  2. Израдити спецификацију захтева софтверског решења.  3. Израдити спецификацију дизајна софтверског решења.  4. Имплементирати софтверско решење према израђеној спецификацији.  5. Тестирати имплементирано софтверско решење.  6. Документовати (1), (2), (3), (4) и (5). |

|  |  |
| --- | --- |
| Руководилац студијског програма: | Ментор рада: |
|  |  |

|  |
| --- |
| Примерак за:  - Студента;  - Ментора |

# КЉУЧНА ДОКУМЕНТАЦИЈСКА ИНФОРМАЦИЈА

|  |  |
| --- | --- |
| Редни број, **РБР**: |  |
| Идентификациони број, **ИБР**: |  |
| Тип документације, **ТД**: | монографска публикација |
| Тип записа, **ТЗ**: | текстуални штампани документ |
| Врста рада, **ВР**: | дипломски рад |
| Аутор, **АУ**: | Иван Мршуља |
| Ментор, **МН**: | Др Јелена Сливка, редовни професор |
| Наслов рада, **НР**: | Праћење и препознавање геста шаке комбинацијом неуронских и традиционалних приступа |
| Језик публикације, **ЈП**: | српски |
| Језик извода, **ЈИ**: | српски / енглески |
| Земља публиковања, **ЗП**: | Србија |
| Уже географско подручје, **УГП**: | Војводина |
| Година, **ГО**: | 2022 |
| Издавач, **ИЗ**: | ауторски репринт |
| Место и адреса, **МА**: | Нови Сад, Факултет техничких наука, Трг Доситеја Обрадовића 6 |
| Физички опис рада, **ФО**: | бр. поглавља / страница / цитата / табела / слика / графикона / прилога |
| Научна област, **НО**: | Софтверско инжењерство и информационе технологије |
| Научна дисциплина, **НД**: | Софтверско инжењерство |
| Предметна одредница /  кључне речи, **ПО**: | Праћење шаке, препознавање геста шаке |
| **УДК** |  |
| Чува се, **ЧУ**: | Библиотека Факултета техничких наука, Трг Доситеја Обрадовића 6, Нови Сад |
| Важна напомена, **ВН**: |  |
| Извод, **ИЗ**: | апстракт – један пасус који добро описује суштину рада – проблем, мотивацију, назнаку решења и резултат. |
| Датум прихватања теме, **ДП**: |  |
| Датум одбране, **ДО**: |  |
| Чланови комисије, **КО**: |  |
| председник | др Име Презиме, звање |
| члан | др Име Презиме, звање |
| ментор | др Име Презиме, звање |
| Потпис ментора | |

# KEY WORDS DOCUMENTATION

|  |  |
| --- | --- |
| Accession number, **ANO**: |  |
| Identification number, **INO**: |  |
| Document type, **DT**: | monographic publication |
| Type of record, **TR**: | textual material |
| Contents code, **CC**: | bachelor thesis |
| Author, **AU**: | Ivan Mršulja |
| Mentor, **MN**: | Jelena Slivka, full professor, PhD |
| Title, **TI**: | Hand tracking and gesture recognition using neural and traditional model pipeline |
| Language of text, **LT**: | Serbian |
| Language of abstract, **LA**: | Serbian / English |
| Country of publication, **CP**: | Serbia |
| Locality of publication, **LP**: | Vojvodina |
| Publication year, **PY**: | 2022 |
| Publisher, **PB**: | author’s reprint |
| Publication place, **PP**: | Novi Sad, Faculty of Technical Sciences, Trg Dositeja Obradovića 6 |
| Physical description, **PD**: | br. poglavlja / stranica / citata / tabela / slika / grafikona / priloga |
| Scientific field, **SF**: | Software Engineering and Information Technologies |
| Scientific discipline, **SD**: | Software Engineering |
| Subject / Keywords, **S/KW**: | Hand tracking, gesture recognition |
| **UDC** |  |
| Holding data, **HD**: | Library of the Faculty of Technical Sciences, Trg Dositeja Obradovića 6, Novi Sad |
| Note, **N**: |  |
| Abstract, **AB**: | Prevod apstrakta na engleski |
| Accepted by sci. Board on, **ASB**: |  |
| Defended on, **DE**: |  |
| Defense board, **DB**: |  |
| president | Ime i prezime, zvanje na eng., PhD |
| member | Ime i prezime, zvanje na eng., PhD |
| mentor | Ime i prezime, zvanje na eng., PhD |
| Mentor's signature | |

**САДРЖАЈ**

[КЉУЧНА ДОКУМЕНТАЦИЈСКА ИНФОРМАЦИЈА 4](#_Toc94596412)

[KEY WORDS DOCUMENTATION 5](#_Toc94596413)

[1. УВОД 9](#_Toc94596414)

[2. ПРЕГЛЕД СТАЊА У ОБЛАСТИ 11](#_Toc94596415)

[3. ТЕОРИЈСКИ ПОЈМОВИ И ДЕФИНИЦИЈЕ 17](#_Toc94596416)

[3.1 Класична вјештачка неуронска мрежа 21](#_Toc94596417)

[3.2 Конволутивна неуронска мрежа 17](#_Toc94596418)

[3.3 Метода Потпорних Вектора (SVM) 17](#_Toc94596418)

[3.4 Стабло одлучивања (Decision Tree) 17](#_Toc94596418)

3.4 Random Forest 17

[3.5 Transfer Learning 17](#_Toc94596418)

[4. МЕТОДОЛОГИЈА 19](#_Toc94596419)

[4.1 Модул за праћење шаке и екстракцију кључних обележја 23](#_Toc94596420)

[4.2 Модул за класификацију геста 23](#_Toc94596421)

[5. ЕКСПЕРИМЕНТИ 25](#_Toc94596423)

[5.1 Skup(ovi) podataka 25](#_Toc94596424)

[5.2 Eksperiment 1 26](#_Toc94596425)

[5.3 Evaluacija 27](#_Toc94596426)

[6. РЕЗУЛТАТИ (И ДИСКУСИЈА) 29](#_Toc94596427)

[7. ДИСКУСИЈА 31](#_Toc94596428)

[8. ЗАКЉУЧАК 33](#_Toc94596429)

[9. ЛИТЕРАТУРА 35](#_Toc94596430)

[10. БИОГРАФИЈА 37](#_Toc94596431)

# УВОД

Могућност рачунара да перципира облик и кретање шаке може бити једна од виталних компоненти која побољшава *user experience-*ау широком спектру технолошких домена и платформи. Праћење шаке и препознавање гестикулације исте представљају један од основних проблема у овом домену истраживања [7].

На пример, решење горе поменутог проблема, може да представља основу за разумијевање знаковног језика и контролу покрета рукама, а такође може да омогући преклапање дигиталног садржаја и информација из физичког свијета у системима проширене стварности (augmented reality). Иако је људима природна, робусна перцепција руку у реалном времену је изразито изазован задатак у домену компјутерске визије, јер руке често заклањају саме себе или једна другу (нпр. оклузије прста/длана и дрхтање руку) и немају регионе високог контраста [1].

У овом раду, рађено је праћење и препознавање геста шаке у домену разумијевања знаковног језика, конкретно, разумијевања алфабета немачког језика.

За рјешавање овог проблема, имплементиран је јеноставан pipeline koji ima dva koraka. Sistem kao ulazne podatke dobija frejmove real-time снимка са веб камере које даје горе поменутом pipeline-у на обрађивање. Prvi korak u obradи слике јесте одређивање 21 кључног oбележја (landmark-а) шаке док други корак представља класификациони модел. Класификациони модел као улаз прима координате сваког од 21 кључног обележја и као излаз даје коначну предикцију геста шаке. Од технологија коришћене су MediaPipe радни оквир (framework) за детекцију кључних обележја шаке као и *scikit-learn* и *Keras* библиотеке за имплементацију класификационих модела.

Евалуација модела вршена је евалуацијом квалитета коначне предикције истог. Квалитет одређивања кључних обележја није евалуиран посебно. Разлог за ово је тај што први корак у проточној обради (детекција и локализација кључних обележја) служи као својеврсни feature extractor те се као сваки други вид претпроцесирања улазне слике не евалуира сам за себе већ у склопу читавог модела проточне обраде [9]. С обзиром да је улазни скуп података релативно балансиран, као мјере перформансе користе се тачност, прецизност, одзив као и микро Ф мјера. Такође, пошто је веома битно да овај модел ради што ближе реалном времену, као мјеру перформансе квалитета рада у реалним условима користи се однос фрејмова које модел може да обради у секунди (Frames Per Second) и микро Ф мјере. Главни закључци који су изведени из евалуације показују да овај приступ има најбољи однос између брзине обраде и квалитета предикције у односу на потпуно трдиционлне и потпуно неуронске приступе (поготово на слбијем хрдверу). Такође, имплементирани приступ се показао супериорно у односу на потпуно неуронске приступе и у домену обучавања јер му је потребно доста мање података за обучавање. На крају, треба напоменути да је мој приступ до одређене мјере отпоран код примјене на различитим нијансама коже као и заклањања шаке што није случај код традиционалних приступа.

Додати пасус са организацијом остатка рада по поглављима.

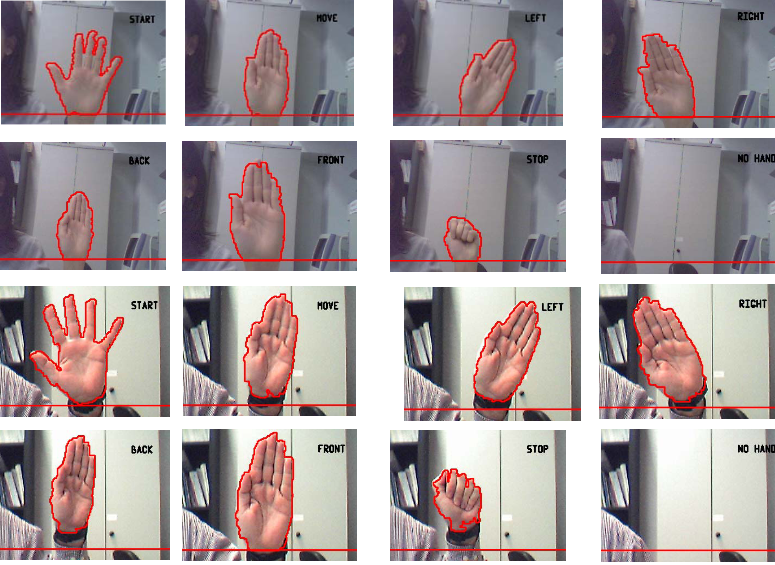
# ПРЕГЛЕД СТАЊА У ОБЛАСТИ

У овом поглављу је наведен историјат развоја приступа за праћење и препознаваље геста шаке као и тренутни *state-of-the-art* приступи код рјешавања овог проблемског домена. Код избора релевантних радова вођено је рачуна да су представљене методологије имале довољно добре перформансе приликом класификације како би се могли користити у пракси као и да су употребљиви из аспекта интеракције човека-рачунара тј. да се резултат могао добити у реалном времену или близу реалног времена као и да је погодан за реалну примјену. Такође, узети су у обзир само они радови који класификацију раде искључиво коришћењем фрејмова видеа као улаза те су сви *Skeleton-Based*, *Depth-Based* i *3D-Based* modeli otpisani u startu.

Први приступ за праћење и препознавање геста шаке кој је дао задовољавајућ резултат за реалну употребу био је заснован на *pipeline*-у који се састојао из три фазе [1]:

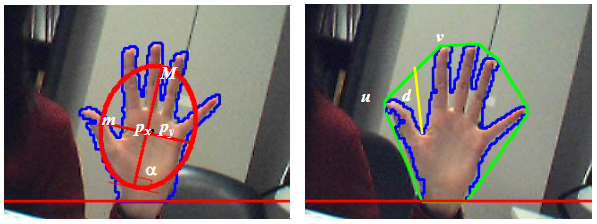
* традиционалне технике сегментације помоћу пробабилистичког модела
* ручног одређивања релевантних обележја
* систем заснован на правилима (*rule-based*) који је служио за одређивање геста

Аутори су развили овај систем за потребе контролисања кретања аватара у видео игрици те је коначна лабела имала 8 класа: *start*, *move*, *stop*, *left*, *right*, *front*, *back*, *no\_hand*. Прије почетка коришћења апликације (приликом сваког покретања) корисник треба да у исцртани квадрат на екрану услика свој длан како би сегментациони алгоритам могао да се обучи. Сви усликани пиксели длана пребацују се из RGB (*Red Green Blue*) у HSV (*Hue Saturation Value*) репрезентацију јер је она доста погоднија за даљу обраду. Функција густине вјероватноће боје коже моделована је гаусовом расподјелом а параметри се одређују стандардном методом максималне вјеродостојности (*maximum likelihood estimation*). Након процеса обучавања модел је спреман за употребу. За сваки пиксел улазне слике одређује се вјероватноћа да ли припада длану, а затим се користи CCA (*Connected Components Аlgorithm*) преко којег се пиксели длана групишу у један регион (*blob)*. Ивични пиксели овог *blob*-a представљају контуре шаке.



Слика 2.1. Приказ одређивања контура шаке употребом пробабилистичког модела

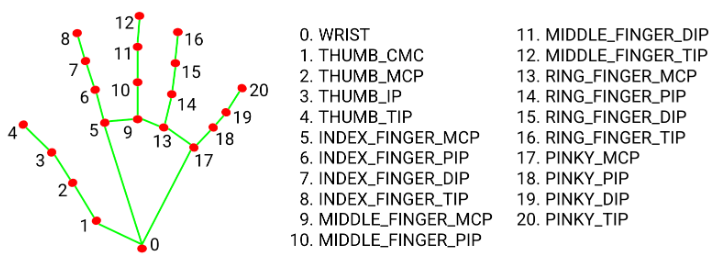
Када сегментациони алгоритам одреди контуре шаке, користи се једноставан хеуристички алгоритам који одређује центар длана. Циљ ове фазе је да се длан представи као елипса и да се одреде мала и велика оса (m, M респективно). С обзиром да као излаз из предходне фазе имамо ивичне пикселе региона, одређивање ових параметара је тривијално и своди се на одређивање линије која представља највеће растојање два пиксела унутар региона (велика оса) и линије која је нормална на њу и спаја два пиксела ивице који се налазе на супротним странама шаке (мала оса). Додатно, стандардним техникама компјутерске визије, рачуна се и конвексни овотач региона. На основу конвексног омотача и репрезентације путем елипсе могуће је одредити низ контурних тачака између два узастопна конвексна врха прста/длана. Ова секвенца формира такозвани дефект конвексности (*Convexity Defect*) и могуће је израчунати дубину и-тог дефекта конвексности di.



Слика 2.2 Приказ екстракције обележја са детектованог региона длана

Просјек дужина свих дефеката конвексности зједно са дужином мале и велике осе, координатама центра шаке као и орјентација шаке се узимају као обележја у финалној фази која се ослања на неколико ручно написаних *if-then-else* правила која, на крају, дају коначан резултат процеса класификације. Модел је евалуиран на ручно прикупљеном скупу података на коме су аутори успјели да постигну веома добре перформансе. За метрику перформансе коришћена је тачност која је израчуната за сваку класу те је након тога упросјечена. Просјечна тачност на свих 8 класа износила је одличних 98%. Главна мана овог приступа је та што се приликом сваког покретања мора поново обучавти пробабилистички модел који ради сегментацију слике. Истина је да се овим постигла робусност јер ће модел радити на различитим типовима коже корисника али остаје проблем коришћења у лоше освијетљеној просторији. Наиме, модел се мора користити у соби гдје је освијетљење константно јер промјеном освјетљења добијамо потпуно различите вриједности пиксела те се пробабилистички модел мора поново обучити. Такође, у сцени гдје је позадина сличне боје као и кожа корисника (поготово на камерама лошег квалитета) аутори наводе да се модел понаша непредвидиво. Још једна од великих мана овог приступа, коју сами аутори наводе, је та што је немогуће користити овај модел код гестова приликом којих нису виљиви сви прсти.

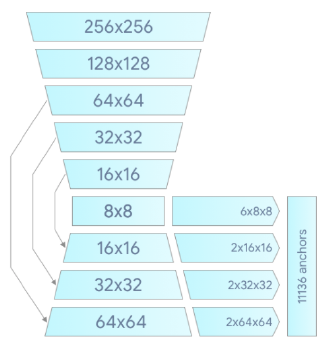
У раду [2] проблем са поновним обучавањем пробабилистичком модела код употребе на различитим нијансама коже као и проблем позадине са бојом сличној боји коже превазиђен је на начин што је као додатна опрема коришћена рукавица која је офарбана бојом која се јако ријетко може наћи у позадини (и природи уопште). Самим тим, једноставном примјеном сегментације помоћу прага је могуће извући регију од интереса. Поново се ручно имплементираним техникма извлче обележја која се у овом приступу убацују у *k-nearest neighbors* (KNN) класификатор. Аутори овог рада су добили скоро идентичне резултате као аутори рада [1] на јако сличном скупу података с тиме што су превазишли један од проблема увођењем додатне опреме. Са друге стране, идаље је остао велики проблем употребе модела у сцени која је неједнако освијетљена и генерално је остао проблем детекције геста код којег нису видљиви сви прсти. Рјешење за ове проблеме изложено је у [6] гдје *T. Maung* предлаже да, за разлику од свих претходних модела, немамо одвојене фазе за екстракцију обележја и класификацију већ је све било енкапсулирано у једну потпуно повезану неуронску мрежу са *hard-limit* активационом функцијом. Овај приступ дао је око 90% тачности на ручно прикуплјеном скупу података али је имао велику ману да ради само уколико се шака налази непосредно испред камере, стога није ни адресирао проблем праћења шаке. У радovima [3] i [10] предложено је коришћење SSD (*Single Shot Detector*) модела за локализацију шаке и класификацију геста (конкретно модела из YOLO породице) док је у раду [5] предложено коришћење мањих (*tiny*) YOLO модела како би се исти могао извршавати и на слабијем хардверу. Иако доста тежи за обучавање, SSD модели ријешили су све проблеме претходно набројаних приступа и дуго су се сматрали као *state-of-the-art* рјешење све до појаве *Mediapipe Hands* модела [4]. Аутори овог рада су увидјели да је главни проблем у овом проблемском домену управо екстракција обележја и ријешили су да направе pipeline модел који ће за улазну слику у RGB репрезентацији моћи да одреди 21 кључно обележје шаке које би касније било који модел машинског учења могао искористити као улаз за процес класификације геста.



Примјер 2.3 Визуализација сваког кључног обележја заједно са његовом ознаком

Њихов предложени модел има двије главне фазе. У првој фази користи се SSD модел за детекцију длана (*BlazePalm detector*). Разлог за ово је намјера да се ријеши проблем парцијалног или комплетног заклањања прстију шаке, такође, у њиховим експериментима се показало да је много мање података потребно за обучавање овог модела у односу на друге приступе који користе SSD модел за детекцију цијеле шаке. У другој фази, користи се Hand Landmark модел који врши прецизну локализацију сваког од 21 кључна обележја шаке путем регресије. Идеја је да модел научи конзистентну интерну структуру шаке како би могао да предвиди позиције прстију и самим тим био робустан на парцијално или потпуно заклањање истих.

Архитектура *BlazePalm* модела је јако слична архитектури YOLOv3 модела те се улазна слика прво скалира на 256х256 резолуцију и провлачи се кроз 5 конволутивних блокова, 3 *upscale* слоја и 4 блока за детекцију. Поједини конволутивни блокови имају стандардну *skip* конекцију са појединим *upscale* слојевима у циљу очувања својстава (*feature*) слике већег нивоа и подизања квалитета детекције. Излаз из детекционих блокова чине 11136 *anchor box*-a који на крају пролазе кроз стандардни *non maximum suppression* алгоритам у циљу добијања коначних граничних оквира дланова. Битно је нагласити да се овај модел може користити за детекцију произвољног броја дланова на снимку у исто вријеме од којих ће сваки бити обрађен. Истина је да детекција већег броја објеката може наштетити перформансе овог модела али јако се ријетко јавља потреба за више од 4 истовремено детектоване шаке те ово и не представља велики проблем.



Слика 2.4 Архитектура *BlazePalm* модела

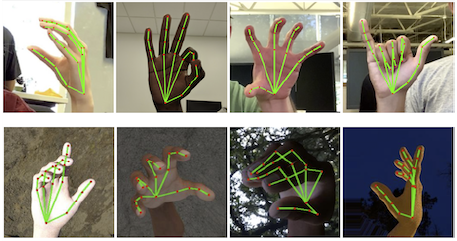
Архитехтура *Hand Landmark* модела је специфична за овај проблемски домен. У питању је *multi-head* модел који има 3 излазна слоја:

1. 21 кључно обележје гдје за свако обележје имамо триплет (*x* координата, *y* координата, релативна дубина у односу на камеру)
2. Вјероватноћа да је на слици детектована шака
3. Бинарна класификација типа шаке (да ли је шака лијева или десна)

Сви излазни слојеви дијеле зједнички *feature extractor*. Занимљива чињеница је то, да је свака од глава (излазних слојева) тренирана на различитим скуповима података (о томе више у наставку). Потребно је нагласити да сама имплементација овог модела јако варира од верзије до верзије радног оквира а постоји и „лакша“ верзија која се користи за праћење шаке на слабијем хардверу (телефону, таблету...).

Аутори овог рада наводе да је најизазовнији дио овог пројекта представљао управо креирање погодног скупа података који би омогућио квалитетно обучавање горе поменутих модела. Прикупљени подаци били су организовани у 3 различита скупа података који су адресирали различите аспекте проблемског домена:

1. *In-the-wild* скуп података који је садржао око 6000 слика са веома великом варијансом (географски диверзитет, различито освјетљење, различите величине и боје шаке)
2. *In-house* скуп података који је садржао око 10000 слика, прикупљан у лабораторијским условима који је покрио све познате гестове које је могуће направити шаком из различитих углова. Једна од мана скупа података је што је прикупљен на свега 30 лјуди у скоро истом окружењу те нема превелике варијације у позадини и тону коже, међутим, ово није направило превелики проблем приликом обучавања.
3. Синтетички скуп података од 100000 генерисаних слика компјутерски анотираног са 21 3D кључним обележјем шаке. Пажљиво је вођено рачуна да се у скупу података нађу различити тонови коже као и различите позадине.



Слика 2.5 Приказ неколико анотираних инстанци скупова података, слике из реалног свијета (горњи ред) и синтетичке слике (доњи ред)

Код обучавања *BlazePalm* модела коришћен је само in-the-wild скуп података док су се за обучавање *Hand Landmark* модела користили сви скупови података и то:

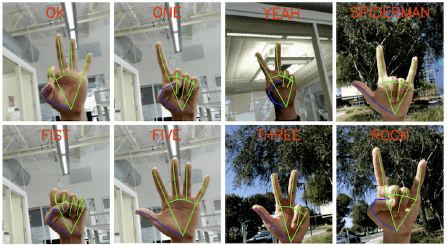
* Ручно анотирани *in-the-wild* скуп заједно са синтетичким скупом за обучавање главе за одређивање 21 кључног обележја
* Подскуп *in-the-wild* скупа заједно са *in-house* скупом за обучавање главе за детекцију присуства шаке као и главе за одређивање да ли је шака лијева или десна

Такође је битно напоменути да су се за евалуацију модела (и валидацију и тест) користили само подаци прикупљени из реалног свијета како би се дала што реалнија слика о перформансама модела. Приликом евалуације коришћена је *average precision* (AP) мјера перформансе за детекцију длана i *mean squared error* (MSE) нормализован према величини шаке за евалуацију квалитета локализације кључних обележја. Код детекције длана, аутори су пробали различите оптимизације са и без употребе декодера као и употребе различитих *loss* функција. Добијене резултате можете погледати у табели 2.1.

|  |  |
| --- | --- |
| **Варијација модела** | **Average Precision (AP)** |
| Без декодера + *cross entropy loss* | 86.22% |
| Декодер + *cross entropy loss* | 94.07% |
| Декодер + *focal loss* | 95.7% |

Табела 2.1 Добијени резултати детекције длана са употребом различитих оптимизација

Што се тиче евалуације квалитета локализације кључних обележја шаке, резултати су били веома добри гдје је нормализована MSE износила свега 16.1%. Резултати евалуације предикције да ли је шака на слици као и класификације шке на лијеву и десну нису дискутовани у овом раду. Autori rada [4] демонстрирају како се овај модел може користити као основа за класификацију геста шаке. Предложено је да се користи стандардна потпуно повезана неуронска мрежа са 5 потпуно повезаних слојева са 64, 128, 512, 64, 32 неурона респективно, заједно са *ReLu* активационом функцијом и *softmax loss*-ом као функцијом грешке. Резултати овог модела нису изнијети у раду али је дата слика која показује да овај приступ заправо ради што ме је и заинтересовало да детаљно експериментишем у овом пољу.



Слика 2.6 Демонстрација рада комплетног модела за праћење шаке и класификацију геста

У радовима [8] и [9] аутори износе своја рјешења за праћење шаке и детекцију знаковног језика, енглеског и вијетнамског респективно гдје се за енглески језик излаз из *Mediapipe* модела даје рекурентној неуронској мрежи на обраду док за вијетнамски језик аутори користе сличну архитектуру предложену у [4]. Аутори оба рада добијају веома добре резултате са преко 98% Ф мјере што ми је потврдило да овакав сличан приступ заправо и има смисла радити.

# ТЕОРИЈСКИ ПОЈМОВИ И ДЕФИНИЦИЈЕ

С обзиром да је методолигија праћења и препознавања геста шаке, у овом раду, подијељена на 2 истоимене цјелине. Као модел за праћење шаке и одређивање кључних обележја коришћен је *Mediapipe* модел чију срж представљају конволутивне неуронске мреже. Као модел за препознавање геста шаке тестирана су 3 приступа:

* Класична потпуно повезана вјештачка неуронска мрежа
* SVM (*Support Vector Machines*) класификатор
* *Random Forest* класификатор

У овом поглављу ће прво бити објашњене класичне вјештачке неуронске мреже (поглавље 3.1) које представљају основу за разумијевање конволутивних неуронских мрежа, представљених у поглављу 3.2. Након тога, у поглављу 3.3 биће објашњен SVM модел, док ће у поглављу 3.4 и 3.5 бити објашњени стабло одлучивања (*Decision Tree)* и *Random Forest* модели. Разлог за увођење поглавља посвећеном *Decision Tree* моделу је тај што овај модел представља срж *Random Forest* модела и увид у његово функционисање је умогоме олакшан схватањем рада позадинског модела. На крају, у поглављу 3.6, биће објашњен *transfer learning* приступ за обучавање неуронских мрежа.

## Класична вјештачка неуронска мрежа

Вјештачка неуронска мрежа функционише слично неуронској мрежи људског мозга. „Неурон“ у неуронској мрежи је математичка функција која прикупља и класификује информације према специфичној архитектури. Неуронска мрежа садржи слојеве међусобно повезаних чворова гдје је сваки чвор у литератури познат и као перцептрон. Битно је напоменути да постоје 3 различита типа слоја:

* улазни слој - садржи улазне информације потребне за одлучивање, односно класификацију
* скривени слој - на основу кога мрежа врши пропратне калкулације и ствара везе са потенцијалним излазима
* излазни слој - садржи резултат калкулације

Сваки неурон првог (улазног) слоја, комбинован са низом одређених коефицијента (тежинама) представља директан улаз у други (скривени) слој. Вредности чворова у овом слоју рачунају се као сума производа претходно поменутих тежина и вредности одговарајућих улазних чворова. Чвор доводи сигнал произведен овом математичком операцијом у функцију активације која је по правилу нелинеарна. Аналогно, претходно добијене вредности у комбинацији са тежинама представљају улаз у следећи скривени слој (може их бити произвољан број са произвољно неурона у сваком од њих) или излазни слој, а вредности индивидуалних чворова рачунају се на исти начин као и у претходном слоју. Из овога се може закључити да неуронска мрежа има велику сличност са статистичким методама као што су уклапање криве и регресиона анализа али да, велики број повезаних чворова праћених нелинеарним трансформацијама на крају, даје простора за далеко већу флексибилност него што је то случај код традиционалних модела на уштрб већег броја параметара и потребе за доста већим скупом податка како бисмо избјегли преприлагођавање.

## Конволутивна неуронска мрежа

У сфери дубоког учења, конволуционе неуронске мреже су класа дубоких неуронских мрежа (*Deep Neural Networks*, DNN) које се најчешће користе за анализу слика. Главни разлог за ово је чињеница да се коршћењем CNN умногоме смањује број параметара у односу на класичне потпуно повезане дубоке неуронске мреже. Узмимо на примјер слику димензија 1000 х 1000 х 3. Код обичне потпуно повезане DNN имали бисмо 3 милиона параметара на улазу. Чак иако би први следећи слој имао само 1000 неурона, димензије тежинске матрице W[1] би биле 1000 x 3 милиона. То би значило да је скоро немогуће наћи скуп података који би био довољно велик да не дође до преприлагођавања на тренинг скуп [11]. За исту димензију улазних података са 3 х 3 филтером, CNN би произвела свега 9 параметара. У наставку ће бити објашњени главни градиви блокови CNN-а.



Слика 3.2.1 Примјер архитектуре CNN

Главни градивни блокови који чине CNN (Слика 3.1) су:

* Конволуциони слој
* Операција испуњавања (*padding*)
* Активациона функција
* Слој сажимања
* Потпуно повезани слој
* Функција губитка (*loss function*).

Конволуциони слој се заснива на математичкој операцији конволуције. За улазну матрицу димензија N x N и филтер f x f спроводи се итеративна операција тако што се филтер поставља на почетак слике те се сваки елемент унутар матрице који се налази „испод“ филтера множи са својим кореспондентним елементом унутар филтера (Слика 3.2). Добијени производи се тада сабирају и ово представља један елемент резултујуће матрице (*feature map*). Након тога се филтер помијера за одређени корак (*stride*) те се операција понавља све док се не дође до краја слике. Код ове операције, важно је примијетити да су димензије резултујуће мтрице мње од димензија полазне матрице. Ово смањење у димензији се може изразити следећом законитошћу: за улазну матрицу димензија N x N и филтер f x f са кораком S добијамо матрицу димензија:

Ово може представљати проблем, поготово код слојева гдје је S велико, јер ће већ након неколико слојева полазна матрица бити превише мала и неупотребљива. Такође, пиксели (елементи матрице) који се налазе на крајевима ће доста мање бити узимани у обзир у односу на елементе који се нлазе у средини, јер филтер мање пута прелази преко њих [12]. Као рјешење, уз операцију конволуције најчешће се примјењује и операција испуњавања (*padding*). Идеја је да се матрица прошири ивицом за онолико редова и колона колико је то потребно да резултујућа матрица буде истих димензија као почетна. Елемент који се убацује је 0, одабран тако да се не наруши валидност операције конволуције [11]. Овим се претходно дефинисана законитост мијења, те ће се за *padding* ширине P димензије резултујуће матрице износити:

Након извршене операције конволуције, на feature мапу се, као и у класичним DNN, додаје вектор пристрасности (*bias vector*) и примјењује се активациона функција. Популарне активационе функције су *ReLu* (*Rectified Linear Unit*), *sigmoid* и *TanH* [12].

Поред конволуционих слојева, CNN често користе и сажимајуће (*pooling*) слојеве како би смањили димензије репрезентације, убрзали рачунање и повећали робусност приликом детекције одређених особина [13]. *Pooling* слој зависи од pooling операције. Као и код конволуционог слоја, филтер димензија N x N се превлачи преко слике са кораком f, само се умјесто операције коволуције примјењује pooling операција. Коју pooling операцију ћемо изабрати зависи од архитектуре нашег модела. Данас, најпопуларнији је *max pooling* код кога као излазну вриједност узимамо највећи елемент из региона захваћеног филтером (Слика 3.2.2). Могућа интуиција иза овога је та да желимо да само оне особине које су веома изражене уђу у даље разматрање, док се оне слабије изражене занемарују. Други популаран избор pooling операцији је *average pooling* који се историјски доста више користио, али је изгубио на популарности јер се max pooling показао као бржи и подједнако ефикасан у експериментима [14].

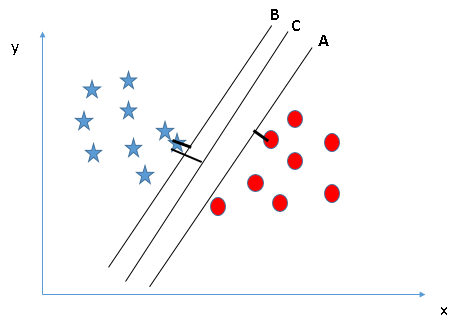


Слика 3.2.2 Разлика између различитих *pooling* алгоритама

Потпуно повезани (*fully connected*) слој представља идентичан слој као у класичној DNN. Ови слојеви се налазе на самом крају CNN. Прелаз између тродимензионалног тензора и једнодимензионалног потпуно повезаног слоја се у литератури назива и „*flattening*“. На самом крају СNN-а налази се *loss layer* који специфицира како тренинг „кажњава“ одступање између предвиђених (излазних) и реалних (лабелираних) ознака у тренинг скупу. Код проблема класификације типично се користи *Softmax loss* док се код предвиђања вјероватноће неког исхода најчешће користи *Sigmoid cross-entropy loss* [11]. На слици 3.2.1 приказан је примјер архитектуре CNN са свим горе поменутим градивним блоковима.

## Метода потпорних вектора (SVM)

Метода потпорних вектора (Support Vector Machine, SVM) је линеарни модел за рјешавање проблема класификације и регресије. Може да решава линеарне и нелинеарне проблеме и генерално добро ради на широком спектру проблема (нрочито класификационих) што га чини једним од најмоћних не-неуронских модела [14]. Идеја SVM-а је једноставна: алгоритам креира линију или хиперравну која раздваја податке у класе такву да маргина одлуке буде максимална тј. да подаци сваке класе буду удаљени што је више могуће од линије одлуке.



Слика 3.2.2 Формирање границе одлуке SVM класификатора

Овдје се може јавити проблем када подаци нису линеарно сепарабилни у њиховом првобитном облику. Постоје два рјешења за то. Прво је једноставно релаксирање границе одлуке, гдје дозвољавамо малу грешку у нади да ће алгоритам исконвергирати са адекватном границом одлуке. Друго је да се подаци трансформишу у други векторски простор у којем је могуће одредити адекватну хиперраван која ће раздвојити податке а затим примјеном математичких трансформација вратити и податке и хиперраван у оригинални векторски простор. Из самог поступка се може закључити да овај процес може бити изузетно рачунарски захтјеван код података који имају велику димензионалност али, на срећу, код SVM-a могуће је одрадити такозвани „кернел трик“ помоћу којег можемо одредити хиперраван у другом векторском простору без да морамо све податке експлицитно трансформисати у исти. Битно је напоменути да је SVM иницјално конципиран као бинарни класификатор али се проблем класификације у више класа може адресирати преко OVO (*one-vs-one*) или OVR (*one-vs-rest*) приступа.

## Стабло Одлучивања (Decision Tree)

Стабло одлучивања је алгоритам надгледаног машинског учења који се користи за класификацију или регресију на основу тога како је одговорено на научен скуп питања [15]. Сама чињеница да се у овом алгоритму не уносе никакве претпоставке о циљној функцији чини овај алгоритам веома флексибилним и робусним до те мјере да му нормализација података уопште није ни потребна за конвергенцију. Стабла одлучивања имитирају људско размишљање, тако да је научницима генерално лако да схвате и протумаче резултате једноставном визуелизацијом чворова и правила у њима. Овај алггоритам не мора увјек да даје егзактну вриједност класификације већ, умјесто тога, може представити опције тако да корисник може сам да донесе смислену одлуку. Градивни блокови овог алгоритма су:

* Коренски чвор (*root node*) – представља почетну тачку у одлучивању
* Операција раздвајања (*splitting*) – представља операцију разбијања једног чвора на више под-чворова
* Чвор одлуке (*decision node*) – чвор који може преусмјерити ток извршавања
* Лист стабла (*leaf node*) – представља могући резултат
* *Pruning* – операција уклањања чворова из стабла
* Грана одлуке (*decision branch*) – под-стабло стабла одлучивања

Основа стабла одлучивања је коренски чвор. Из коренског чвора тече низ чворова одлуке који описују одлуке које треба донијети. Потомци чворова одлуке могу бити други чворови одлуке или листови стабла. Сваки чвор одлуке представља питање или тачку раздвајања, а листови који потичу из чворова одлуке представљају могуће одговоре. Свако под-стабло стабла одлучивања називамо „граном“. Изградња стабла одлучивања се спроводи приликом тренирања модела, у којој се уче атрибути и услови који ће произвести стабло. Затим се стабло орезује (pruning) да би се уклониле небитне гране које би могле да негативно утичу на перформансе. *Pruning* укључује уочавање *outlier*-а, тачака података далеко изван норме, који би могли да доведу до одбацивања прорачуна дајући превелику тежину ретким појавама у подацима. У зависности који проблем рјешавају, стабла одлучивања се дијеле на категоричка и континуална, у овом раду бавићермо се само категоричким стаблима.

## Random Forest

Random forest, као што име модела имплицира, састоји се од великог броја појединачних стабала одлучивања која функционишу као *bagging* ансамбл [16]. Свако појединачно стабло даје своју предикцију класе, а класа са највише гласова се усваја као коначна предикција нашег модела. Основни принцип иза овог модела је једноставан: велики број некорелираних слабих предиктора (стабала одлучивања) ће имати боље перформансе од појединачних модела. Разлог за овај ефекат је што модели међусобно „штите“ један другог од својих појединачних грешака (све док не греше стално у истом правцу). Поменуо сам да овај ансамбл спада у bagging типове ансамбла. То значи да се приликом тренирања, сваком засебном стаблу изнова семплује тренинг скуп који је исте дужине као и оригинални скуп с тим што је креиран узимањем насумичних инстанци из оригиналног скупа на начин да једна инстанца може бити селектована више пута (комбинације са понављањем). Додатно, за сваки скуп податка се узима насумичан подскуп обележја што доводи до веће варијансе између појединачних модела осигурава међусобну некорелираност.

## Transfer learning

Архитектуре са великим бројем слојева, поготово кад су CNN у питању, могу потрошити доста времена и рачунарских ресурса при тренирању модела. Такође, за многе примјене не постоје велики, специјализовани, анотирани скупови података на којима би се могли тренирати дубоки модели.

Код transfer learning приступа, идеја је искористити претходно истрениран модел, који ради за сличан проблемски домен и прилагодити га за сопствене потребе [17]. Прилагођавање се врши тако што се отклони оригинални излазни слој и замијени се са неким од излазних слојева (секција 3.1 и 3.2) који одговарају нашем приступу. Затим је потребно истренирати нови излазни слој над нашим скупом података. У зависности од величине скупа, могуће је тренирати и неке додатне слојеве из модела како би се исти што боље прилагодио жељеном домену. Такође, уколико је доступан веома мали скуп података, могуће је све тежине сачувати на диску и користити их као одвојену функцију која служи као *feature extractor* чији излаз сервирамо неком другом моделу машинског учења.

# METODOLOGIJA

У овом поглављу је представљена имплементација система за праћење и препознавање геста шаке. Улаз у систем представља фрејм видео снимка са веб камере док излаз из система представља визуелизацију кључних landmark-ова шаке који повезани чине естимацију скелета шаке као и лабелу са препознатим гестом поред сваке шаке. Систем може независно обрађивати до 2 шаке истовремено на снимку.

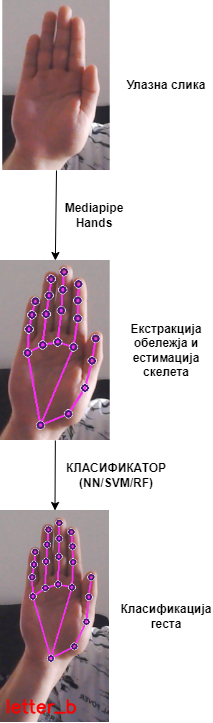
Систем се састоји из 2 одвојена модула:

1. Модул за праћење шаке и екстракцију кључних обележја
2. Модул за класификацију геста

Улаз у први модул представља фрејм са снимка камере у RGB репрезентацији а као излаз даје 21 кључно обележје шаке. Обележја се даље убацују у модул за класификацију чији излаз чини коначна класа која представља гест (једно од првих 12 слова немачког алфабета, конкретно: *a,b,c,d,e,f,g,h,i,j,k,l,m*). На крају, свако од 21 кључних обележја се исцртава на фрејму видеа и повезује линијама на начин да представи естимацију 2.5D скелета шаке. Такође, добијени гест (које је слово алфабета у питању) из 2. модула се исписује испод зглоба шаке како би лако могли да закључимо за коју шаку је који гест детектован.

Ток експеримента је такође једноставан:

1. Од скупа података [18] се узима подскуп чије излазне класе чине првих 12 слова алфабета
2. Из скупа података отклања се свака колона која представља Z координату
3. Добијени скуп података се дијели на тренинг, валидациони и тест скуп у односу 80/10/10
4. Дотренирава се модел машинског учења transfer learning приступом (само класификациони модел, моделе иѕ pipeline-а ѕа класификацију није потребно дотрениравати)
5. Врши се евалуација над валидационим скупом података и врши се селекција и оптимизација модела
6. Врши се евалуација класификационог модела над тестним скупом података
7. Врши се евалуација читавог pipeline-a за рад у реалним условима



Слика 4.1 Графички приказ претходно описане методологије



Слика 4.2 Дијаграм тока експеримента

## Модул за праћење шаке и екстракцију кључних обележја

Улаз у овај модул представља фрејм са снимка камере у RGB репрезентацији а као излаз даје 21 кључно обележје шаке. Ова обележја се касније могу визуелизовати на начин да осликавају 2.5D скелет шаке и иста се даље користе као улаз у модул за класификацију.

Имплементација овог дијела је веома једноставна, искоришћен је Mediapipe радни оквир, конкретно Mediapipe Hands pipeline чији је рад детаљно објашњен у поглављу 2. За рад са снимком са веб камере коришћена је OpenCV библиотека која тренутно представља state-of-the-art технологију за рад са сликом/видеом. Помоћу ове библиотеке, издваја се сваки фрејм видеа са веб камере у реалном времену и даје се на обраду Mediapipe Hands pipeline-u. Није вршено никакво додтно тренрање BlazePalm и HandLandmark модела с обѕиром да је имплементација са default-ним тежинама радила јако добро.

## Модул за класификацију геста

Улаз у овај модул представља 21 кључно обележје шаке а као излаз даје предикцију класе која означава који је гест шаком направљен на улазној слици. Овај модул је имплементиран на 3 начина тј. тестирана су 3 различита модела и то:

* Стандардна потпуно повезана неуронска мрежа
* Метода потпорних вектора (SVM)
* Random forest

Сва три модела су обучена и евалуирана над истим скуповима података. Важно је напоменути и да је одрађено предпроцесирање скупова података тако што су избачене све колоне које предстабљају Z координату (отприлике свака трећа колона), чиме је смањена димензионалност проблема, самим тим скраћено вријеме обучавања док су перформансе остале непромијењене. Нормализација података није рађена експлицитно с обзиром да је излаз из претходног модула такав да координате обележја нису представљена као апсолутно растојање од неке тачке већ као удео растојања од врха (лијеве стране) слике тако да су подаци нормализовани по default-у. Подаци који су се користили за обучавање, валидацију и тестирање су такође изворно били у истом формату те експлицитна нормализација није била потребна. За учитавање података и претпроцесирање коришћена је Pandas библиотека док су за трансформацију података из Pandas dataframe-а у облик погодан за обраду коришћене numpy и tensorflow библиотеке. За подјелу скупа података на тренинг, валидациони и тест скуп, коришћена је scikit-learn библиотека.

Неуронска мрежа је имплементирана у архитектури предложеној у [4] (*MpHandGesture* модел). Мрежа поред улазног и излазног слоја садржи и 5 скривених слојева са 42, 64, 128, 512, 64, 32, 12 неурона респективно. У оваквој конфигурацији мрежа садржи 112428 параметара који се могу научити. Излазни слој је стандардан softmax слој са 12 могућих катекорија (првих 12 слова немачког алфабета). Коришћен је adam оптимизациони алгоритам са categorical crossentropy функцијом грешке. Као активациона функција коришћена је ReLu функција. Покушане су разне варијације на овај приступ са другачијом архитектуром слојева, додавањем регуларизације и испробавањем другачијих активационих функција и оптимизационих алгоритама али је претходно наведени приступ давао најбоље резултате на валидационим скуповима (опширније у поглављу 5). Мрежа је тренирана transfer learning приступом гдје су за иницијалне тежине узете тежине које су дошле са иницијалном имплементацијом MpHandGesture модела. Имплементација као и процес тренирања одрађени су уз помоћ Keras библиотеке.

SVM модел је имплементиран коришћењем SVC (C-Support Vector Classification) из scikit-learn библиотеке. Коришћен је стандардни RBF кернел са свим default-ним параметрима.

Random Forest класификатор имплементиран је употребом RandomForestClassifier класе из scikit-learn biblioteke. Коришћено је 500 естиматора са највећом дубином естиматора од 100 нивоа. Одрађена је паралелизација тренинга и извршавања модела над свим расположивим језгрима процесора постављањем „n\_jobs“ параметра на -1 [19].

Битно је напоменути да је за одабир параметара модела за чију је имплементацију коришћена scikit-learn библиотека одрађен уз помоћ RandomizedSearchCV класе која бира најбољу комбинацију параметара уз помоћ унакрсне валидације. У мом случају коришћен је стандардни 5 fold cross validation. Одрађена је паралелизација читавог процеса над свим расположивим језгрима процесора постављањем „n\_jobs“ параметра на -1 [19].

# EKSPERIMENTI

**Cilj poglavlja**: U ovom poglavlju potrebno je opisati sve eksperimente koji su izvršeni u okviru diplomskog rada.

**Važno:** Postavka svakog eksperimenta trebalo bi da sadrži sve detalje koji su potrebni da bi neko drugi mogao da reprodukuje eksperiment.

**Razlika između poglavlja “Eksperimenti” i „Metodologija“**: u metodologiji opisujete arhitekturu sistema koji predstavljate u radu i detalje svakog od njegovih delova modula. Nasuprot toga, u ovom poglavlju opisujete na koji ste način(e) evaluirali taj sistem i/ili svaki od njegovih delova. Na primer, sistem ste možda evaluirali na više različitih skupova podataka ili sa više različitih kombinacija parametara ili ste hteli da vidite uticaj pred ili post-procesiranja pa ste evaluaciju radili sa i bez tih komponenti, itd.

**Organizacija:** U glavnom delu iskažite cilj poglavlja „Eksperimenti“ i pojasnite njegovu podelu na potpoglavlja koja je, tipično, ovakva:

1. Obično, eksperimenti u AI oblasti uključuju jedan ili više obučavajućih skupova. Predstavite ih u posebnom potpoglavlju „Skupovi podataka“.
2. Tipičan sledeći deo eksperimentalne postavke je opis jednog ili više modela mašinskog učenja. Napravite posebno potpoglavlje za svaki eksperiment koji ste izvršili, npr. „Eksperiment 1“, „Eksperiment 2“ (predložite specifičnija imena ako imaju smisla).
3. Opis postupka evaluacije.

**Napomena:** trudite se da ne ponavljate informacije – prvo utvrdite šta je zajedničko za sve vaše eksperimente. Tipično su to skupovi podataka i evaluacija, pa onda možete izdvojiti u dva potpoglavlja, dok ostale detalje pišete za svaki eksperiment posebno u potpoglavljima „Eksperiment 1“, „Eksperiment 2“, itd.

## Skup(ovi) podataka

**Cilj potpoglavlja:** opis skupova podataka korišćenih u svim eksperimentima.

**Organizacija:** Ako imate puno skupova podataka, možete ovo poglavlje podeliti na potpoglavlja koja im odgovaraju. Ako skupovi podataka imaju duža imena, u poglavlju 6.1 uvedite skraćenice. Na primer, K1 (korpus 1).

**Tipični podaci koje opis skupa podataka treba da sadrži:**

1. količina podataka,
2. distribucija klasnog obeležaja,
3. detalji vezani za prikupljanje i anotaciju (labeliranje),
4. da li ste skup podataka preuzeli (da li je javno dostupan, odakle ste ga preuzeli) ili sami kreirali za potrebe diplomskog rada,
5. detalji specifični za vaš skup podataka:
   * Na primer, ako su u pitanju slike onda: veličina, rezolucija, format, kvalitet slika, itd.
   * Na primer, ako je u pitanju tekst onda: prosečna dužina rečenica, dokumenata, entiteta, itd.
6. detalji skupa podataka značajni za problem koji rešavate:
   * Razmišljajte o tome koji detalj skupa vam je potreban da biste u diskusiji objasnili neki rezultat eksperimenta.
   * Na primer, za slike, rezultati mogu biti loši jer je rezolucija slika niska (i onda kod tih rezultata referencirate ovo poglavlje).
   * Na primer, za tekst, rezultati mogu biti loši jer je određena vrste entiteta slabo zastupljena u obučavajućem skupu, rečenice su previše dugačke, itd.

## Eksperiment 1

**Cilj:** prikazujete detalje vezane za ML model(e). Sam model je opisan u prethodnim poglavljima („Teorijske osnove“ i/ili „Metodologija“), dok je ovde cilj da opišete detalje koje ste podesili baš za eksperiment sa ciljem da eksperiment može da se kompletno reprodukuje.

Na primer, ovde prikazujete:

* Vrednosti hiper-parametara. Ako ih ima puno, navedite ih u tabeli.
* Način na koji ste odredili vrednosti hiper-parametara.
* Ako u vašem eksperimentu poredite više modela, onda ovde opisujete konkretne postavke za svaki takav model.
* Ako je cilj vašeg eksperimenta da utvrdite optimalne hiper-parametre ili skup atributa, onda je fokus na opisu načina na koji to radite.
* Hardver ili *cloud* servis koji je korišćen, itd.

## Evaluacija

**Cilj:** prikaz načina evaluacije rešenja. Tipično, ovo obuhvata:

1. Kako ste formirali skup podataka za evaluaciju:

* kako je formiran test skup,
* odnos udela podataka u *test/train* podeli,
* koja vrsta uzorkovanja (*sampling*) je upotrebljena pri podeli na *train*/*test* (npr. *random* ili *stratified*)
* ako je upotrebljena unakrsna validacija, opišite detalje vezane za taj postupak.

1. Mere performansi koje koristite za evaluaciju. Ako su mere neuobičajene i kompleksne onda bi bilo dobro da ih detaljnije objasnite na primeru. Jedan primer takve mere je *Intersection over Union* (IoU) koja se koristi za evaluaciju sistema za detekciju objekata.

# REZULTATI (I DISKUSIJA)

**Cilj:** prikazujete rezultate eksperimenata opisanih u prethodnom poglavlju.

**Napomena oko naslova poglavlja**: ponekad eksperimenti i rezultati mogu biti obimni i/ili zanimljivi pa se o njima može napisati duža diskusija. U tom slučaju je potrebno diskusiju odvojiti u posebno poglavlje „Diskusija“ koje ide nakon rezultata. U nastavku ovog teksta opisani su detalji vezani za poglavlje „Rezultati“, pa onda za poglavlje „Diskusija“, a na vama je da odlučite da li ćete ih pisati u jednom ili u dva odvojena poglavlja.

**Sadržaj poglavlja:**

1. Ovo poglavlje tipično sadrži jednu ili više tabela sa rezultatima eksperimenata. Najbolje je da tabele prikazujete redom po eksperimentima i onda na kraju da date jednu sumarnu tabelu ako to ima smisla. Na primer, ako eksperimenti uključuju evaluaciju nekoliko različitih ML modela, onda u sumarnoj tabeli pokazujete sve njih zajedno i komentarišete kako se porede.
2. Nakon prikaza tabele napišete jedan ili više pasusa u kojima ukratko saopštite i komentarišite rezultate.
   1. Ne ponavljate celu tabelu u tekstu, već istaknite nabolje, najlošije rezultate.
   2. Komentarišite da li su ovi rezultati očekivani (u skladu sa dosadašnjom literaturom) ili iznenađujući.
   3. Na primer, model X se pokazao kao najbolji, što je i očekivano jer je to SOTA model za ovaj problem ili rezultati za klasu X su najbolji, što je iznenađujuće jer je to klasa koje jako slabo zastupljena u obučavajućem skupu.
3. Važno je da donesete odluku o tome da li ćete diskusiju rezultata uključiti u ovo poglavlje ili ćete ih predstaviti u posebnom poglavlju „Diskusija“.
   1. Ako diskusije nema puno, onda ovde dajte svoja objašnjenja za rezultate. U tom slučaju, u nastavku kao posebne pasuse možete dodati delove poglavlja „Diskusija“ (sadržaj poglavlja „Diskusija“ je objašnjen u poglavlju 8).
   2. Razloge zašto su rezultati takvi kakvi su često nije lako utvrditi. Zato, pri diskusiji pišite „pretpostavljamo da su rezultati takvi zato što...“ umesto da čvrsto „tvrdimo da su rezultati...“.

# DISKUSIJA

**Cilj:** prikazujete svoje rezultate iz svih uglova važnih za cilj vašeg rada.

* Na primer, ako je cilj vašeg rada bio da kreirate novi sistem za neki problem onda je važno da diskutujete kako se vaši rezultati porede sa srodnim rešenjima, pogotovo onim najboljim.
* Ako ne postoje srodna rešenja koja baš rešavaju isti problem kao i vi, sigurno postoje rešenja koja rešavaju sličan problem sa kojima se možete uporediti.
* Na primer, razvijate određen NLP sistem za srpski jezik i niko do sada nije radio to baš za srpski jezik. U tom slučaju, poredite se sa sistemima koji to rada za engleski.

**Sadržaj poglavlja**:

* Komentarišite poređenja sa drugim sistemima.
  + Na primer, rezultati prikazani u radu [x] su bolji jer je korišćen bolji hardver ili veći skup podataka, što ovde nije moguće uraditi usled nedostatka *big data* infrastrukture.
  + Ako je teško porediti performanse usled korišćenja različitih skupova podataka i meri performanse, istaknite tu činjenicu.
* Komentarišite realnu upotrebljivost sistema.
  + Da li su performanse na nivou onih koje postižu ljudi?
  + Da li se rezultati direktno mogu koristiti u produkciji ili su tek prvi korak ka nekom boljem sistemu?
  + Ovde je takođe važno da komentarišete razloge za tvrdnje koje ste izneli.
  + Napomena: ponekad je samim anotatorima skupa podataka teško da se dogovore oko anotacije konkretnog primera. Ovo može biti uzrok loših performansi modela. Često u srodnim rešenjima postoje poglavlja koja govore o anotaciji skupa podataka gde su date mere (tipično F-mera) koje pokazuju koliko je teško da se više ljudi složi oko anotacije. U tom slučaju se možete porediti sa tim merama.
* Diskutujte prednosti i ograničenja svog sistema.
  + Na primer, prednost je da je sistem lako proširiv, ali je trenutno ograničenje da radi samo za srpski jezik.
  + Važno je da budete realni, nemojte sakrivati nedostatke. Svrha ovog dela je upravo to da pokažete da ste svesni ograničenja svog sistema.
  + Dajte predloge kako bi se ograničenja mogla prevazići. Na primer, bolji hardver, veći skup podataka, itd.
* (Opciono) u ovo poglavlje uključite i analizu grešaka sistema[[1]](#footnote-1). Za detalje vezane za analizu grešaka pogledajte prateći materijal.

# 

# ZAKLJUČAK

**Cilj**: (1) sumiranje glavnih poenti u radu i diskusija implikacija rešenja i (2) opis mogućih daljih pravaca za unapređenje iznetog rešenja.

**Sadržaj**:

* Sumiranje glavnih poenti u radu:
  + Rešavani problem i motivacija za njegovo rešavanje
  + Opis rešenja na visokom nivou apstrakcije
  + Opis evaluacije rešenja na visokom nivou apstrakcije
  + Podsetiti na sve izvučene zaključke i komentarišite njihovu važnost (glavne poente iz diskusije)
* Iznesite moguće pravce daljeg proširivanja/unapređenja/otklanjanje identifikovanih nedostataka rešenja

|  |
| --- |
| **(Rekapitulacija problema i motivacije za njegovo rešavanje:)** U ovom radu predstavljen je sistem za automatsku detekciju i prepoznavanje saobraćajnih znakova sa slike. Motivacija je bila što bi takav sistem mogao da se koristi prilikom vožnje i da koristi vozaču putem obaveštavanja o važećim saobraćajnim znakovima. Sistem je implementiram u vidu dva odvojena dela: modula za detekciju i modula za prepoznavanje saobraćajnih znakova. Oba modula oslanjaju se na modele konvolucionih neuronskih mreža.  **(Rezime rešenja, načina evaluacije i postignutih rezultata:)**  Modul za detekciju koristi *Faster* R-CNN [4] metod sa *AlexNet* [5] arhitekturom konvolucione mreže. Rezultat evaluacije istreniranog modela je 0.6756 mAP. Za treniranje i testiranje korišćen je skup belgijskih saobraćajnih znakova (engl. BTSD – *Belgium Traffic Sign Dataset*) [6], uz izbacivanje kategorija „nedefinisano“ i „ostalo“, i smanjenje broja negativnih uzoraka.  Modul za prepoznavanje saobraćajnih znakova vrši klasifikaciju znakova dobijenih iz faze detekcije. Za klasifikaciju treniran je model konvolucione neuronske mreže sa *ResNet* [8] arhitekturom od devet rezidualnih blokova. Korišćen je skup nemačkih saobraćajnih znakova za prepoznavanje (engl. GTSRB – *German Traffic Sign Recognition Benchmark*) [9], a preciznost modela nad test skupom iznosi 96.2%.  **(Predlozi za unapređenje rešenja:)** Preciznost konvolucione mreže za prepoznavanje mogla bi se dalje povećati boljim izborom hiper-parametara modela, što bi zahtevalo ponavljanje procesa treniranja. Što se tiče *Faster* R-CNN metoda korišćenog za detekciju, značajno bolje performanse mogu se postići prelaskom na *ResNet* arhitekturu konvolucione mreže. S obzirom na to da *Faster* R-CNN implementacija u korišćenom *Microsoft Cognitive Toolkit* (CNTK) [10] alatu ovo ne podržava, to bi zahtevalo izmenu implementacije ili korišćenje drugog alata. Skup podataka bi takođe mogao da se proširi spajanjem sa nekim drugim, kao što je na primer skup nemačkih saobraćajnih znakova za detekciju (engl. GTSDB – *German Traffic Sign Detection Benchmark*) [32]. |

Primer 9.1 Primer celokupnog zaključka uz komentare

# LITERATURA

|  |
| --- |
|  |
| 1. Manresa, C., Varona, J., Mas, R. and Perales, F.J., 2005. Hand tracking and gesture recognition for human-computer interaction. *ELCVIA Electronic Letters on Computer Vision and Image Analysis*, *5*(3), pp.96-104. 2. Wang, R.Y. and Popović, J., 2009. Real-time hand-tracking with a color glove. *ACM transactions on graphics (TOG)*, *28*(3), pp.1-8. 3. Sharp, T., Keskin, C., Robertson, D., Taylor, J., Shotton, J., Kim, D., Rhemann, C., Leichter, I., Vinnikov, A., Wei, Y. and Freedman, D., 2015, April. Accurate, robust, and flexible real-time hand tracking. In *Proceedings of the 33rd annual ACM conference on human factors in computing systems* (pp. 3633-3642). 4. Zhang, F., Bazarevsky, V., Vakunov, A., Tkachenka, A., Sung, G., Chang, C.L. and Grundmann, M., 2020. Mediapipe hands: On-device real-time hand tracking. *arXiv preprint arXiv:2006.10214*. 5. Yeo, H.S., Lee, B.G. and Lim, H., 2015. Hand tracking and gesture recognition system for human-computer interaction using low-cost hardware. *Multimedia Tools and Applications*, *74*(8), pp.2687-2715. 6. Maung, T.H.H., 2009. Real-time hand tracking and gesture recognition system using neural networks. *International Journal of Computer and Information Engineering*, *3*(2), pp.315-319. 7. Lugaresi, C., Tang, J., Nash, H., McClanahan, C., Uboweja, E., Hays, M., Zhang, F., Chang, C.L., Yong, M.G., Lee, J. and Chang, W.T., 2019. Mediapipe: A framework for building perception pipelines. *arXiv preprint arXiv:1906.08172*. 8. Halder, A. and Tayade, A., 2021. Real-time vernacular sign language recognition using mediapipe and machine learning. *Journal homepage: www. ijrpr. com ISSN*, *2582*, p.7421. 9. Duy Khuat, B., Thai Phung, D., Thi Thu Pham, H., Ngoc Bui, A. and Tung Ngo, S., 2021, February. Vietnamese sign language detection using Mediapipe. In *2021 10th International Conference on Software and Computer Applications* (pp. 162-165). 10. Le, T.H., Jaw, D.W., Lin, I.C., Liu, H.B. and Huang, S.C., 2018, May. An efficient hand detection method based on convolutional neural network. In *2018 7th International Symposium on Next Generation Electronics (ISNE)* (pp. 1-2). IEEE. 11. Rahul, C. and Ghansala, K.K., 2018. Convolutional neural network (CNN) for image detection and recognition. In First International conference on secure cyber computing and communication, IEEE. 12. Albawi, S., Mohammed, T.A. and Al-Zawi, S., 2017, August. Understanding of a convolutional neural network. In 2017 International Conference on Engineering and Technology (ICET) (pp. 1-6). Ieee. 13. Boureau, Y.L., Ponce, J. and LeCun, Y., 2010. A theoretical analysis of feature pooling in visual recognition. In Proceedings of the 27th international conference on machine learning (ICML-10) (pp. 111-118). 14. Hearst, M.A., Dumais, S.T., Osuna, E., Platt, J. and Scholkopf, B., 1998. Support vector machines. *IEEE Intelligent Systems and their applications*, *13*(4), pp.18-28. 15. Swain, P.H. and Hauska, H., 1977. The decision tree classifier: Design and potential. *IEEE Transactions on Geoscience Electronics*, *15*(3), pp.142-147. 16. Biau, G. and Scornet, E., 2016. A random forest guided tour. *Test*, *25*(2), pp.197-227. 17. Torrey, L. and Shavlik, J., 2010. Transfer learning. In *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques* (pp. 242-264). IGI global. 18. *German Sign Language* скуп података <https://www.kaggle.com/datasets/moritzkronberger/german-sign-language> (датум приступа 24.6.2022.) 19. *Scikit-Learn* библиотека <https://scikit-learn.org/stable/> (датум приступа 24.6.2022.) |
|  |  |

# BIOGRAFIJA

Иван Мршуља је рођен 31.01.2000. у Котору, гдје је стекао основно и средње образовање. Школске 2018/19 године се уписује на Факултет Техничких Наука на студијски програм Софтверско Инжењерство И Информационе Технологије. Положио је све испите предвиђене планом и програмом и стекао услов за одбрану завршног рада.

1. Analizu grešaka sistema možete opciono staviti i u poglavlje „Rezultati“. U tom slučaju, u diskusiji kratko prodiskutujte rezultate te analize. [↑](#footnote-ref-1)