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

Иван Мршуља

**Праћење и препознавање геста шаке комбинацијом неуронских мрежа 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, ванредни 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:**

|  |
| --- |
| Израдити систем за праћење и препознавање геста шаке, заснован на техникама компјутерске визије: 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, ванредни професор |
| Наслов рада, **НР**: | Праћење и препознавање геста шаке комбинацијом неуронских мрежа 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, рачунарска визија |
| **УДК** |  |
| Чува се, **ЧУ**: | Библиотека Факултета техничких наука, Трг Доситеја Обрадовића 6, Нови Сад |
| Важна напомена, **ВН**: |  |
| Извод, **ИЗ**: | Рад износи рјешење за овај проблем праћења и препознавања геста употребом *pipeline*-a који се састоји из *Mediapipe Hands* модела и класификационог модела (неуронска мрежа, SVM и *random forest*). Систем је тестиран на *German Sign Language* скупу података. |
| 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, associate professor, PhD |
| Title, **TI**: | Hand tracking and gesture recognition using neural networks 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**: | This paper presents a solution for hand tracking and gesture recognition using a pipeline consisting of the Mediapipe Hands model and a classification model (neural network, SVM, and random forest). The system was tested on the German Sign Language dataset. |
| 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](#_КЉУЧНА_ДОКУМЕНТАЦИЈСКА_ИНФОРМАЦИЈА)

[KEY WORDS DOCUMENTATION 7](#_KEY_WORDS_DOCUMENTATION)

[1. УВОД 11](#_УВОД)

[2. ПРЕГЛЕД СТАЊА У ОБЛАСТИ 13](#_ПРЕГЛЕД_СТАЊА_У)

[3. ТЕОРИЈСКИ ПОЈМОВИ И ДЕФИНИЦИЈЕ 23](#_ТЕОРИЈСКИ_ПОЈМОВИ_И)

[3.1 Класична вјештачка неуронска мрежа 23](#_Класична_вјештачка_неуронска)

[3.2 Конволутивна неуронска мрежа 24](#_Конволутивна_неуронска_мрежа)

[3.3 Метода Потпорних Вектора (SVM) 27](#_Метода_потпорних_вектора)

[3.4 Стабло одлучивања (Decision Tree) 29](#_Стабло_Одлучивања_(Decision)

3.4 Random [Forest](#_Random_Forest) 30

[3.5 Transfer Learning 30](#_Transfer_learning)

[4. МЕТОДОЛОГИЈА 31](#_METODOLOGIJA)

[4.1 Модул за праћење шаке и екстракцију кључних обележја 33](#_Модул_за_праћење)

[4.2 Модул за класификацију геста 33](#_Модул_за_класификацију)

[5. ЕКСПЕРИМЕНТИ 37](#_ЕКСПЕРИМЕНТИ)

[5.1 Скуп података 37](#_Скуп_података)

[5.2 Експеримент 1 39](#_Експеримент_1)

[5.3 Експеримент 2 39](#_Експеримент_2)

[5.3 Експеримент 3 39](#_Експеримент_3)

[5.3 Евалуација 40](#_Евалуација)

[6. РЕЗУЛТАТИ И ДИСКУСИЈА 41](#_РЕЗУЛТАТИ_И_ДИСКУСИЈА)

[6.1 Експеримент 1 41](#_Експеримент_1_1)

[6.2 Експеримент 2 42](#_Експеримент_2_1)

[6.3 Експеримент 3 43](#_Експеримент_3_1)

[7. ЗАКЉУЧАК 45](#_ЗАКЉУЧАК)

[8. ЛИТЕРАТУРА 47](#_ЛИТЕРАТУРА)

[9. БИОГРАФИЈА 51](#_БИОГРАФИЈА)

# УВОД

Могућност рачунара да перципира облик и кретање шаке може бити једна од виталних компоненти која побољшава корисничко искуство (*user experience*)у 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 датог проблема може да представља основу за разумијевање знаковног језика, 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. окулзије прста/длана и дрхтање руку) и немају регионе високог контраста [1].

У овом раду, вршено је праћење и препознавање геста шаке у домену разумијевања знаковног језика, конкретно, разумијевања алфабета немачког језика. Систем као улазне податке добија фрејмове *real-time* снимка са веб камере, а као излаз враћа коначну предикцију геста шаке.

За рјешавање овог проблема, имплементиран је једноставан *pipeline* koji ima dva koraka:

1. Први корак је одређивање 21 кључног обележја (*landmark*-а) шаке
2. Други корак представља класификациони модел који 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 класификационог модела као и евалуацијом рачунарских перформанси комплетног *pipeline*-a. 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 на то 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 приступ који комбинује *Mediapipe* *pipeline* са SVM класификатором има најбољи однос између брзине обраде и квалитета предикције у односу на потпуно неуронске приступе (поготово на слабијем хардверу). Такође, имплементирани приступ се показао супериоран у односу на приступе у целини засноване на неуронским мрежама и у домену обучавања јер му је потребно доста мање података за обучавање. На крају, треба напоменути да је овај приступ до одређене мјере отпоран код примјене на различитим нијансама коже као и заклањања шаке, што није случај код приступа заснованих на једноставним операцијама над сликом.

У поглављу 2 биће приказан преглед стања у области, са најутицајнијим радовима у овом проблемском домену као и детаљним описом рада на који се овај рад ослања. У поглављу 3 биће описани теоријски појмови и дефиниције неопходни за разумијевање овог рада. Поглавље 4 посвећено је опису методологије као и опису тока експеримента. Прије самог краја, у поглављу 5, представљене су поставке најбитнијих експеримената који су одрађени док се у поглављу 6 износе и дискутују резултати ових експеримената. На крају, у поглављу 7, даје се закључак на овај рад.

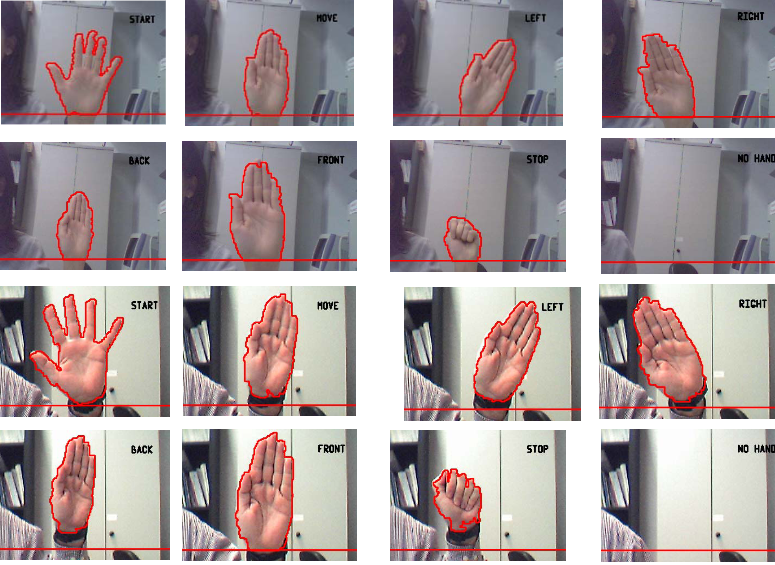
# ПРЕГЛЕД СТАЊА У ОБЛАСТИ

У овом поглављу је наведен историјат развоја приступа за праћење и препознавање 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. Такође, узети су у обзир само они радови који класификацију раде искључиво коришћењем фрејмова видеа као улаза. Стога *Skeleton-Based*, *Depth-Based* i *3D-Based* модели овде нису разматрани.

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

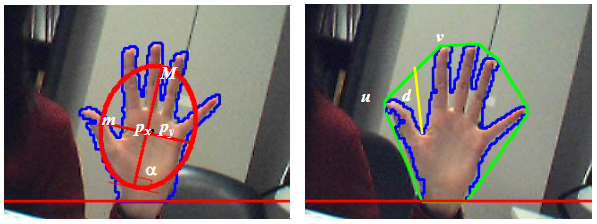
* традиционалне технике сегментације помоћу пробабилистичког модела
* ручног одређивања релевантних обележја
* систем заснован на правилима (*rule-based*) који је служио за одређивање геста

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 позиционира 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.).



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 на то да као излаз из претходне 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 омотач региона (слика 2.2). 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 осам класа износила је 98%.

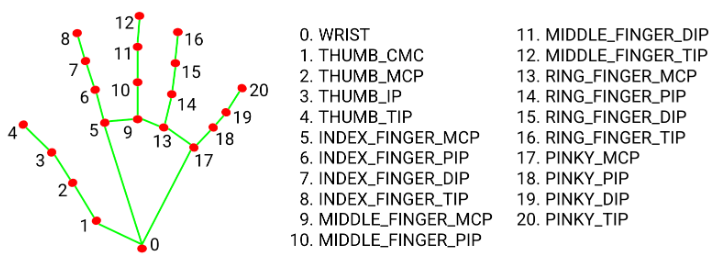
Главна мана овог приступа је та што се приликом сваког покретања мора поново обучавати пробабилистички модел који врши сегментацију слике. Применом пробабилистичког модела за сегментацију је постигнута робусност модела на различите типове коже корисника. Међутим, остаје проблем коришћења у лоше освијетљеној 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 аутори наводе, је немогућност коришћења овог модела код гестова приликом којих нису видљиви сви прсти.

У раду [2] проблем са поновним обучавањем пробабилистичког модела код употребе на различитим нијансама коже, као и проблем позадине са бојом сличној боји коже превазиђен је коришћењем рукавице која је офарбана бојом која се ријетко може наћи у позадини (и природи уопште). Самим тим, једноставном примјеном сегментације помоћу прага је могуће извући регију од интереса. И у овом раду се ручно имплементираним техникама извлаче обележја. У овом приступу, извучена обележја се прослеђују *k-nearest neighbors* (KNN) класификатору. 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, и даље је остао 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. Уместо тога, све је енкапсулирано у једну потпуно повезану неуронску мрежу са *hard-limit* активационом функцијом. Овај приступ дао је око 90% тачности на ручно прикупљеном скупу података, али је имао велику ману да ради само уколико се шака налази непосредно испред камере. Стога аутор није ни адресирао проблем праћења шаке.

У радовима [3] и [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 сматрали су се за *state-of-the-art* рјешење све до појаве *Mediapipe Hands* модела [4].

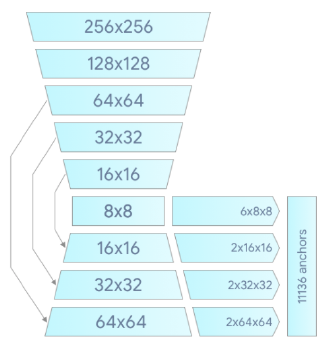
Аутори *Mediapipe Hands* модела [4] су увидјели да је главни проблем у овом проблемском домену управо екстракција обележја и ријешили су да направе *pipeline* модел који ће за улазну слику у RGB репрезентацији моћи да одреди 21 кључно обележје шаке (слика 2.3) које би касније било који модел машинског учења могао искористити као улаз за процес класификације геста.



Слика 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 (слика 2.4) је јако слична архитектури YOLOv3 модела. Улазна слика се прво скалира на 256 х 256 резолуцију и провлачи се кроз пет конволутивних блокова, три *upscale* слоја и четири 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 11 136 *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 перформансама овог модела, али јако се ријетко јавља потреба за више од четири истовремено детектоване шаке, те ово не представља велики проблем у пракси.



Слика 2.4 Архитектура *BlazePalm* модела

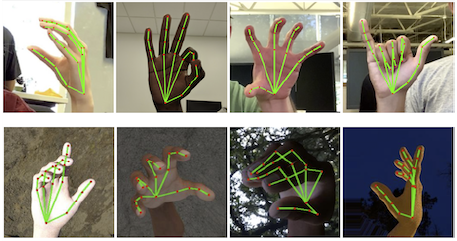
Архитехтура *Hand Landmark* модела је специфична за овај проблемски домен. У питању је *multi-head* модел који има три излазна слоја:

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 било креирање погодног скупа података који би омогућио квалитетно обучавање горе поменутих модела. Прикупљени подаци били су организовани у три различита скупа података који су адресирали различите аспекте проблемског домена (слика 2.5):

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 10 000 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 људи 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 100 000 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

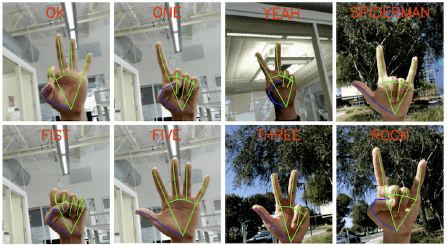
Такође је битно напоменути да су се за евалуацију модела (и валидацију и тест) користили само подаци прикупљени из реалног свијета, како би се дала што реалнија слика о перформансама модела. Приликом евалуације коришћена је *average precision* (AP) мјера перформансе за детекцију длана i *mean squared 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 шаке на лијеву и десну нису дискутовани у овом раду.

Аутори рада [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 пет 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 ради у пракси (слика 2.6).



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 потврђује успешност овог приступа у пракси.

# ТЕОРИЈСКИ ПОЈМОВИ И ДЕФИНИЦИЈЕ

С обзиром на то да је задатак рада праћење и препознавање геста шаке, методологија представљена у наредном, поглављу је подијељена на те две цјелине. У овом поглављу биће представљене теоријске основе неопходне за разумевање ових поступака.

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 три приступа:

* Класична потпуно повезана вјештачка неуронска мрежа
* Метод потпорних вектора (*Support Vector Machines,* SVM)
* Метод случајних шума (*Random Forest*)

У складу са тиме, 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 умногоме 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 три различита типа слојева:

* 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. Међутим, 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 коришћењем 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 градивни блокови CNN-а.



Слика 3.2.1 Примјер архитектуре CNN

Главни градивни блокови који чине CNN (Слика 3.2.1) 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. 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 матрице мање 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 налазе 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 конволуције примјењује *pooling* операција. Коју *pooling* операцију ћемо изабрати зависи од архитектуре нашег модела. Данас, је најпопуларнији *max pooling* код кога као излазну вриједност узимамо највећи елемент из региона захваћеног филтером (Слика 3.2.2). Могућа интуиција иза овога је 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].

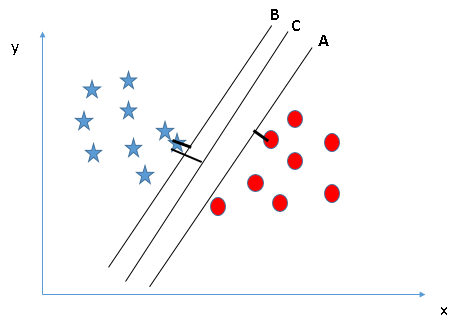


Слика 3.2.2 Разлика између различитих *pooling* алгоритама

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 (нарочито класификационих), што га чини једним од најмоћнијих не-неуронских модела [14]. Идеја SVM-а је једноставна: алгоритам креира хипер-раван која раздваја податке у класе, такву да маргина одлуке буде максимална. Односно, циљ је да подаци сваке класе буду удаљени што је више могуће од границе 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 хипер-раван која ће раздвојити податке а затим примјеном математичких трансформација вратити и податке и хипер-раван у оригинални векторски простор. Из самог поступка се може закључити да овај процес може бити изузетно рачунарски захтјеван код података који имају велику димензионалност. Међутим, код SVM-a могуће је применити такозвани „кернел трик“ помоћу којег можемо одредити хипер-раван у другом векторском простору без да морамо све податке експлицитно трансформисати у исти.

Битно је напоменути да је SVM иницијално 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

Стабло одлучивања (*Decision Tree*) је алгоритам надгледаног машинског учења који се користи за класификацију или регресију на основу тога како је одговорено на научен скуп питања [15]. У овом алгоритму се не уводе никакве претпоставке о циљној функцији, што чини овај алгоритам флексибилним и робусним. Стабла одлучивања имитирају људско размишљање, тако да је научницима генерално лако да схвате и протумаче резултате једноставном визуелизацијом чворова и правила у њима. Овај алгоритам 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, а у овом раду бавићемо се само категоричким стаблима.

## Метод случајне шуме

Метод случајне шуме (*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 и 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:

* визуелизација кључних *landmark*-ова шаке, који повезани чине естимацију скелета шаке, као и
* лабела са препознатим гестом поред сваке шаке.

Систем може независно обрађивати до 2 шаке истовремено на снимку.

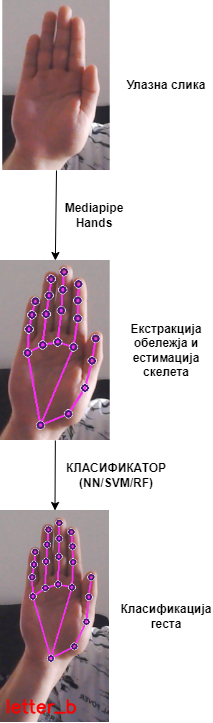
Систем се састоји из два одвојена модула (Слика 4.1):

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 следећи (Слика 4.2):

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 (предиктовану удаљеност од објектива камере)
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 из *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 модул враћа 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 додатно тренирање *BlazePalm* и *HandLandmark* модела с обзиром на то да је имплементација са подразумеваним тежинама радила јако добро.

## Модул за класификацију геста

Улаз у овај модул представља 21 кључно обележје шаке, а као излаз модул враћа предикцију класе која означава који је гест шаком направљен на улазној слици. Овај модул је имплементиран на три 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 представљају Z координату тј. Предиктовану удаљеност од објектива камере (отприлике свака трећа колона). Уклањањем ових колона је смањена димензионалност проблема и самим тим скраћено вријеме обучавања, док су перформансе остале непромијењене. Нормализација података није рађена експлицитно, с обзиром на то 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 подразумевано нормализовани. 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.

Неуронска мрежа је имплементирана у архитектури предложеној у [4] (*MpHandGesture* модел). Мрежа поред улазног и излазног слоја садржи и пет скривених слојева са 42, 64, 128, 512, 64, 32, 12 неурона, респективно. У оваквој конфигурацији мрежа садржи 112 428 параметара који се могу научити. Излазни слој је стандардан *softmax* слој са 12 могућих категорија (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 подразумеваним параметрима.

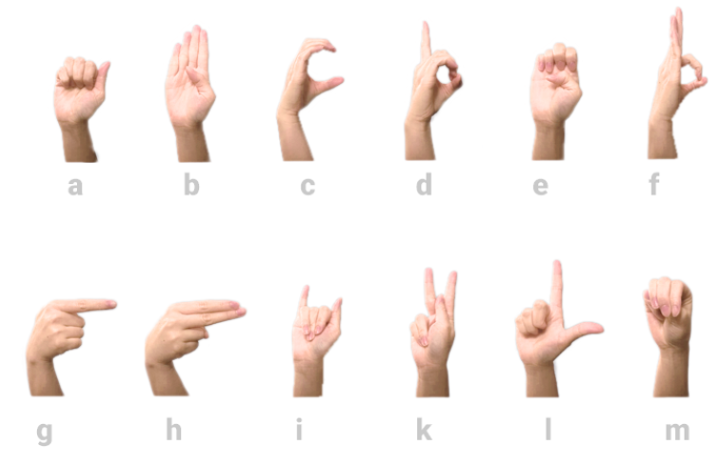
*Random Forest* класификатор имплементиран је употребом *RandomForestClassifier* класе из *scikit-learn* библиотеке. Коришћено је 500 естиматора са највећом дубином естиматора од 100 нивоа. Одрађена је паралелизација тренинга и извршавања модела над свим расположивим језгрима процесора постављањем „n\_jobs“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

# ЕКСПЕРИМЕНТИ

Ово поглавље се бави описом (главних) експеримената који су извршени у току развијања система. У потпоглављу 5.1 биће описан скуп података који је коришћен за тренинг, валидацију и евалуацију класификационих модела који чине други модул система. У 5.2, 5.3 и 5.4 биће описани експерименти који су спроведени у циљу евалуације сваког од три класификациона модела над тестним скупом података. Циљ ових експеримената је да се покажу реалне перформансе система, како за квалитет класификације тако и за рад у реалном времену. Експерименти у којима је експериментисано са различитим архитектурама/хипер параметрима модела нису укључени у овом поглављу како би се фокусирали на експерименте који врше евалуацију система. На крају, у потпоглављу 5.5, описан је процес евалуације како појединачних класификационих модела тако и читавог система. Битно је напоменути да је тренинг свих модела извршено на *Intel* *i7-8700K* процесору. Коришћена верзија *Python* програмског језика је 3.9.5.

## Скуп података

За потребе тренирања, евалуације и тестирања класификационих модела коришћен је подскуп јавно доступног *German Sign Language* скупа података [18]. Подскуп је садржао класе које су означавале првих 12 слова немачког алфабета (Слика 5.1.1).

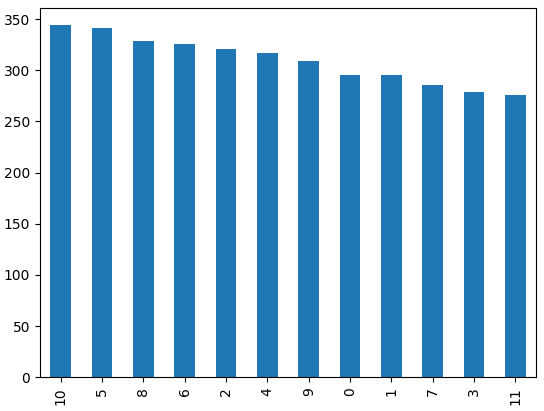


Слика 5.1.1 Класна обележја подскупа оригиналног скупа података

Одабрано је 12 класа која одговарају словима *a-m*. Разлог за смањење оригиналног скупа података (у смислу коришћења подскупа класа) је прије свега та што се овим скратило потребно вријеме обучавања модела, док је скуп података и даље репрезентативан да реално ослика перформансе модела за дати проблем. Takoђе, класе су одабране на начин да постоји добар однос између различитих и сличних класа како би се реално могле сагледати могућности система.

Добијени подскуп садржи 3 718 инстанцу, гдје се свака инстанца описује са 64 обележја. Обележја представљају координате сваког од 21 кључног обележја шаке тако што нпр. колоне 0, 1 и 2 осликавају корен шаке (формат обележја описан у поглављу 2), 3,4 и 5 најдоњи зглоб палца, и тако даље. Експерименталним путем и препоруком из [8] донешена је одлука да се координата Z избаци из разматрања, што је значило уклањање трећине колона у фази препроцесирања. Овим је смањена димензионалност проблема и олакшан процес обучавања модела.

Циљне лабеле су енкодоване за потребе обучавања техником *label encoding*-a. Дистрибуција класног обележја у овом скупу података је балансирана, гдје најзаступљенија класа има 345 а најмање заступљена 275 инстанци. *Frequency plot* за овај скуп података је приказан на слици 5.1.2.



Слика 5.1.2 Дистрибуција класног обележја која су *label* енкодована (0–а,..., 11-m)

Битно је напоменути да су подаци били претходно нормализовани (баш као и излаз *MediapipeHands* модела), тако да се није морао применити додатни вид нормализације.

## Експеримент 1

У овом експерименту тестирана је неуронска мрежа као класификациони модел. Архитектура неуронске мреже описана је детаљно у поглављу 4. Приликом тренирања је величина *mini batch*-a подешена на 32 и одрађено је 200 епоха. Вриједности ових параметара добијене су емпиријским путем на валидационом скупу података.

## Експеримент 2

У овом експерименту тестиран је SVM као класификациони модел. Коришћен је *RandomizedSearchCV* приступ (детаљно описан у 5.5) за одређивање оптималних хипер параметара модела на валидационом скупу података. Оптимални параметри се налазе у табели 5.3.1.

|  |  |
| --- | --- |
| Хипер параметар | Вриједност |
| *C* | 1.0 |
| *gamma* | ‘*scale*’ |
| *shrinking* | *True* |
| *probability* | *False* |
| *tol* | 1e-3 |
| *max\_iter* | -1 |
| *decision\_function\_shape* | ‘*ovr*’ |

Табела 5.3.1 Хипер-параметри за SVM класификатор

## Експеримент 3

У овом експерименту тестиран је *Random Forest* као класификациони модел. Коришћен је *RandomizedSearchCV* приступ за одређивање оптималних хипер параметара модела на валидационом скупу података.

|  |  |
| --- | --- |
| Хипер параметар | Вриједност |
| *criterion* | ‘*gini*’ |
| *min\_samples\_split* | 2 |
| *min\_samples\_leaf* | 1 |
| *min\_weight\_fraction\_leaf* | 0.0 |
| *max\_features* | ‘*sqrt*’ |
| *max\_leaf\_nodes* | *None* |
| *bootstrap* | *True* |

Табела 5.4.1 Хипер-параметри за *Random Forest* класификатор

## Евалуација

Како се систем састоји из два одвојена модула, главна питања која су се јављала су се тицала начина евалуације појединачних модула. Једноставном анализом система можемо примијетити да први део има улогу *feature extractor*-а, док други део представља класификациони модел који заправо даје коначни излаз. Како се *feature extractor*-и у већини случајева евалуирају у контексту крајњег циља, одлучено је да се први модул система не евалуира самостално (такође, коришћени модел је евалуиран у [4] и резултати су изнијети у поглављу 2). Зато је одлучено је да се евалуира само класификациони модел и то на два начина:

1. Евалуација квалитета предикције – тачност, прецизност, одзив и микро ф-мјера
2. Евалуација перформансе модела – однос између ф-мјере и броја фрејмова које систем може да одради у току једне секунде

Квалитет предикције се не евалуира у контексту цијелог система из разлога што је у овом систему *Mediapipe* модел својеврсно „уско грло“. Јер, уколико *Mediapipe* модел не препозна шаку, други модул неће ни добити податке на обраду.

Тест скуп је добијен *random* *sampling*-om оригиналног скупа података. Подјела је извршена на тренинг, валидациони и тест скуп у односу 80/10/10. Приликом финалног тестирања, тренинг и валидациони скуп су спојени у заједнички тренинг скуп (овим смо добили више података за тренинг у нади да ће се модел још боље обучити), поновљено је тренирање и потом је извршена евалуација на тест скупу. Подјела је извршена помоћу ручно-написаног алгоритма гдје је *random seed* за одабир псеудо-случајних инстанци био „12345“.

Такође, 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].

# РЕЗУЛТАТИ И ДИСКУСИЈА

У овом поглављу приказани су резултати експеримената описаних у претходном поглављу (5.2, 5.3 и 5.4).

## Експеримент 1

Резултати за сваку поејединачну класу предикције су представљени у табели 6.1.1.

|  |  |  |  |
| --- | --- | --- | --- |
| Класа | Прецизност | Одзив | Ф-мјера |
| а | 1.00 | 1.00 | 1.00 |
| b | 1.00 | 0.97 | 0.98 |
| c | 0.95 | 1.00 | 0.97 |
| d | 1.00 | 1.00 | 1.00 |
| e | 1.00 | 0.93 | 0.97 |
| f | 1.00 | 1.00 | 1.00 |
| g | 1.00 | 1.00 | 1.00 |
| h | 1.00 | 1.00 | 1.00 |
| i | 1.00 | 1.00 | 1.00 |
| k | 1.00 | 1.00 | 1.00 |
| l | 1.00 | 1.00 | 1.00 |
| m | 0.96 | 1.00 | 0.98 |

Табела 6.1.1 Резултати тестирања на свакој појединачној класи (неуронска мрежа)

Добијена тачност класификације износила је 0.99 на цијелом тест скупу, док је микро Ф-мјера износила 0.99. Ови резултати су били очекивани, с обзиром на то да је коришћена архитектура која је препоручена за коришћење у овом проблемском домену [4]. У поређењу са [9], добијени резултати представљају побољшање од око 1%, док са [8] не можемо извршити директно поређење с обзиром да је излаз из система доста другачији, као и методологија којом се до тог излаза дошло.

Тренирана неуронска мрежа се показала добро и при *bechmark* коришћењу у реалном времену, гдје је успевала да обрађује ~12 фрејма по секунди (FPS) за једну руку и ~9 FPS уколико су двије руке истовремено присутне на снимку. С обзиром на то да се све преко 5 FPS-a узима као задовољавајуће за реалну употребу [8], може се закључити да су добијени резултати добри.

Из теста рада у реалном времену уочена је потенцијална мана система да се у моменту преласка из једног геста у други добија као излаз *random* шум који не осликава реално који је гест детектован. Такође, уколико се направи гест на којем модел није обучаван, излаз ће поново бити нека класа из могућег излазног скупа класа, што може представљати проблем уколико би се систем користио у реалним условима.

## Експеримент 2

Резултати за сваку појединачну класу предикције су представљени у табели 6.2.1.

|  |  |  |  |
| --- | --- | --- | --- |
| Класа | Прецизност | Одзив | Ф мјера |
| a | 1.00 | 0.93 | 0.96 |
| b | 1.00 | 1.00 | 1.00 |
| c | 0.97 | 0.97 | 0.97 |
| d | 0.97 | 1.00 | 0.98 |
| e | 0.93 | 0.93 | 0.93 |
| f | 1.00 | 1.00 | 1.00 |
| g | 1.00 | 1.00 | 1.00 |
| h | 1.00 | 1.00 | 1.00 |
| i | 1.00 | 1.00 | 1.00 |
| k | 1.00 | 1.00 | 1.00 |
| l | 1.00 | 0.97 | 0.99 |
| m | 0.87 | 0.96 | 0.91 |

Табела 6.2.1 Резултати тестирања на свакој појединачној класи (SVM класификатор)

Добијена тачност класификације износила је 0.98 на цијелом тест скупу, док је микро Ф-мјера износила 0.98, што је ипак погоршање од ~2% у односу на неуронску мрежу.

SVM класификатор се показао одличан у погледу брзине рада приликом употребе – систем је обрађивао ~27 FPS са једном и чак ~25 FPS са обје руке истовремено присутне на видеу. Овакав однос квалитета предикције и брзине рада овај модел чини бољим за реалну употребу у односу на неуронску мрежу, поготово на слабијем хардверу.

Нажалост, овај модел инхерентно пати од истих проблема као и претходни, док се овдје јавља и слабија перформанса код правилне класификације слова „m“. Могући начин побољшања је да се конкретна класа *oversample*-ује, што може довести до побољшања перформанси. Други начин би могао бити тренирање још неколико SVM класификатора са различитим хипер-параметрима и увезати их у стандардни *boosting* (*hard* *voting*) ансамбл у нади да ће различито тренирани класификациони модели гријешити на различитим мјестима.

## Експеримент 3

Резултати за сваку појединачну класу предикције су представљени у табели 6.3.1.

|  |  |  |  |
| --- | --- | --- | --- |
| Класа | Прецизност | Одзив | Ф-мјера |
| a | 0.88 | 0.93 | 0.90 |
| b | 1.00 | 0.97 | 0.98 |
| c | 0.97 | 0.92 | 0.94 |
| d | 0.96 | 0.96 | 0.96 |
| e | 0.91 | 1.00 | 0.95 |
| f | 1.00 | 1.00 | 1.00 |
| g | 1.00 | 1.00 | 1.00 |
| h | 1.00 | 1.00 | 1.00 |
| i | 1.00 | 0.97 | 0.99 |
| k | 1.00 | 1.00 | 1.00 |
| l | 0.97 | 0.97 | 0.97 |
| m | 0.88 | 0.85 | 0.87 |

Табела 6.2.1 Резултати тестирања на свакој појединачној класи (*Random Forest* класификатор)

Добијена тачност класификације износила је 0.96 на цијелом тест скупу, док је микро Ф-мјера износила 0.96, што је горе од остала два приступа. Ово је било неочекивано с обзиром на то да за ансамбл моделе важи генерално прихваћена претпоставка да би на мањим, табеларним скуповима података требали радити боље од неуронских мрежа и осталих традиционалних модела.

Друга ствар која је изненадила је брзина рада која је износила ~10 FPS са једном и свега ~7 FPS са двије шаке на видеу, иако је алгоритам паралелизован на свих 12 хардверских нити које је мој рачунар имао на располагању. Могући разлог овако спорог рада је сама количина и максимална дубина искориштених естиматора (500 естиматора са дубином 100), али сваким значајнијим смањењем количине/дубине естиматора нарушавао се квалитет класификације. Због лошијих перформанси од осталих тестираних модела по сваком аспекту, није препоручена употреба сса овим класификационим моделом.

# 

# ЗАКЉУЧАК

У овом раду представљен је систем за праћење шаке и класификацију геста шаке на примјеру немачког знаковног језика (првих 12 слова алфабета). Мотивација је била што би такав систем могао да представља основу за разумијевање знаковног језика, као и основу за развијање неког сложенијег система који би користио класификацију геста шаке (нпр. за помоћ у физикалној терапији пацијената). Систем је имплементиран у два одвојена модула:

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

Први модул ослања се на моделе конволуционих неуронских мрежа. За други модул тестирани су и неуронски и традиционални модели који су обучавани и евалуирани над скупом података представљеним у [18].

Модул за праћење и екстракцију кључних обележја за имплементацију користи *Mediapipe Hands pipeline* који представља *state-of-the-art* технологију за овај проблемски домен [8]. За други модул система, експериментисано је са три различита приступа, конкретно:

1. Потпуно повезана неуронска мрежа
2. Метода потпорних вектора (SVM)
3. *Random Forest*

Модели су евалуирани у контексту квалитета класификације, као и у контексту рада у реалном времену. Што се квалитета класификације тиче, неуронска мрежа је постигла најбољу Ф-мјеру од 0.99, SVM је био на другом мјесту са 0.98, док је најгоре резултате имао *random forest* са Ф-мјером од 0.96.

У контексту рада у реалном времену најбоље се показао SVM помоћу којега је било могуће обрадити 25 FPS. Слиједи SVM са 12 FPS и, на крају, *random forest* са 7 FPS (наведени су резултати рада уколико су на снимку присутне обије шаке истовремено).

Уколико узмемо у обзир однос између метрика квалитета и брзине класификације, можемо закључити да се оптималан однос квалитета класификације и брзине рада добија коришћењем SVM модела, док се *random forest* најгоре показао на тестовима. Битно је напоменути да иако се *random forest* показао лошије од осталих тестираних модела, да је и његова употребљивост велика с обзиром на то да му Ф-мјера пребацује 0.95 и да је брзина рада је већа од 5 FPS, што представља тренутни стандард за рад у реалном времену [7], [8].

Треба се осврнути и на то да систем инхерентно пати од одређених проблема, а то је да у моменту преласка из једног геста у други добијамо као излаз насумичнишум који не осликава реално који је гест детектован. Такође, уколико се направи гест на којем модел није обучаван, излаз ће поново бити нека класа из могућег излазног скупа што може представљати проблем уколико би се систем користио у реалним условима. Проблем представља и коришћење система у условима екстремно лоше освијетљености, као и употреба са јако лошом камером гдје модул за праћење шаке и екстракцију обележја неће моћи да пронађе шаку на снимку.

Следећи корак у развоју овог система би био тестирање *boosting* ансамбл модела у улози класификационог модела, првенствено *XGBoost* модел за који важи да ће за табеларне податке (на мањим скуповима података) радити и брже и боље од неуронских мрежа и других традиционалних модела, што даље може побољшати перформансе система.

# ЛИТЕРАТУРА

|  |
| --- |
|  |
| 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.) |
|  |  |

# БИОГРАФИЈА

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.