



UNIVERZITET U NOVOM SADU
FAKULTET TEHNIČKIH NAUKA
U NOVOM SADU




Лука Курељушић

Шах управљан говором

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 или Рачунарство и аутоматика
Руководилац студијског програма:	проф. др Мирослав Зарић (SW)

Студент:	Лука Курељушић	Број индекса:	SW 23/2018
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:

Шах управљан говором

ТЕКСТ ЗАДАТКА:

1. Анализирати стање у области.
2. Дизајнирати и имплементирати апликацију за гласовно управљање игром шах.
3. Изложити архитектуру решења, посебно агентског алгорита и архитектуру коришћеног модела машинског учења, као и структуру скупова података коришћених за његово тренирање
4. Евалуирати модел и дискутовати резултате и закључке евалуације.
5. Документовати (1), (2), (3) и (4)

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:	Лука Курељушић
Ментор, МН:	др Јелена Сливка, ванредни професор
Наслов рада, НР:	Шах управљан говором
Језик публикације, ЈП:	српски
Језик извода, ЈИ:	српски / енглески
Земља публикавања, 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:	10 поглавља / 45 страница / 0 цитата / 3 табеле / 9 слика / 0 графикона / 0 прилога / 1 формула
Научна област, НО:	Софтверско инжењерство и информационе технологије
Научна дисциплина, НД:	Софтверско инжењерство
Предметна одредница / кључне речи, ПО:	Шах управљан говором на српском језику
УДК	
Чува се, ЧУ:	Библиотека Факултета техничких наука, Трг Доситеја Обрадовића 6, Нови Сад
Важна напомена, ВН:	
Извод, ИЗ:	У раду је представљен систем управљан говором за игру шах, развијен за српско говорно подручје. Агента логика имплементирана је употребом минимакс алгоритма са алфа бета одсецањем. Евалуирана је кроз игру са ботом на <i>chess.com</i> . За гласовно управљање коришћена је конволутивна неуронска мрежа, обучавања и тестирана на ручно креираном скупу података. У погледу остварених резултата, алгоритам је успео да победи бота са ЕЛЮ рангом 1200, док класификациони модел постижу тачност од 94% за слова и 98,8% за бројеве.
Датум прихватања теме, ДП:	
Датум одбране, ДО:	
Чланови комисије, КО:	
председник	
члан	
ментор	
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 :	Luka Kureljušić
Mentor, MN :	Jelena Slivka, associate professor, PhD
Title, TI :	Voice Controlled Chess
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 :	10 chapters / 45 pages / 0 citations / 3 tables/ 9 images / 0 graphs / 0 attachments / 1 formula
Scientific field, SF :	Software Engineering and Information Technologies
Scientific discipline, SD :	Software Engineering
Subject / Keywords, S/KW :	Voice controlled chess in the Serbian language
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 voice-controlled chess game developed for the Serbian language. The agent's logic uses the minimax algorithm with alpha-beta pruning. It was evaluated through a series of games with a chess.com bot. The voice-controlled system uses a convolutional neural network, trained and tested on a manually created dataset. The algorithm beat a bot with an ELO rank of 1200, while the classification models achieved an accuracy of 94% for letters and 98.8% for number classification.
Accepted by sci. Board on, ASB :	
Defended on, DE :	
Defense board, DB :	
president	
member	
mentor	
Mentor's signature	

SADRŽAJ

КЉУЧНА ДОКУМЕНТАЦИЈСКА ИНФОРМАЦИЈА.....	4
KEY WORDS DOCUMENTATION	5
1. УВОД.....	9
2. ПРЕГЛЕД СТАЊА У ОБЛАСТИ	13
3. ТЕОРИЈСКИ ПОЈМОВИ И ДЕФИНИЦИЈЕ	17
3.1 Минимакс алгоритам	17
3.2 Обрада звука	19
3.2.1 Неуронска мрежа.....	20
3.2.2 Конволутивне неуронске мреже	23
3.2.3 Звук.....	24
4. МЕТОДОЛОГИЈА.....	27
4.1 Модул за логику игре и управљање потезима агента..	28
4.1.1 Агентски алгоритам	28
4.1.2 Хеуристичка функција.....	30
4.2 Модул за рад са звуком	31
4.2.1 Обрада улазног звука	31
4.3 Коришћени алати	33
5. ЕКСПЕРИМЕНТИ	35
5.1 Скуп података.....	35
5.2 Евалуација агентског алгоритма.....	36
5.3 Снимање и сегментација звука	37
5.4 Тестирање класификационих модела.....	38
5.5 Евалуација.....	38
6. РЕЗУЛТАТИ.....	39
6.1 Евалуација агентског алгоритма.....	39
6.2 Перформансе класификационих модела.....	41
7. ДИСКУСИЈА	45
8. ЗАКЉУЧАК.....	47
9. ЛИТЕРАТУРА.....	49
10. БИОГРАФИЈА.....	51

1. УВОД

Аутоматско препознавање говора (*Automatic Speech Recognition – ASR*) представља подобласт рачунарства и рачунарске лингвистике која се бави проучавањем и развојем методологија које омогућавају препознавање и превођење говорног језика у текст, или, у ширем смислу, придавањем семантике говорном језику. Ове технологије корисницима пружају могућност директне комуникације са рачунарима једноставним разговором, те своју примену налазе као подршка хендс-фри окружењима, као подршка за управљање уређајима у оквиру паметних кућа, подршка слепим и слабовидим особама и као помоћ у обављању једноставних задатака.

Популаризацији и широкој примена *ASR* технологија нарочито је допринео развој и пораст употребе алгоритама вештачке интелигенције (посебно неуронских мрежа) и развој интелигентних асистената, као што су Амазонова Алекса¹ (*Amazon Alexa*), Еплова Сири² (*Apple Siri*) и Мајкрософтова Кортана³ (*Microsoft Cortana*) и други.

У овом раду обрађена је употреба *ASR* технологија за имплементацију говором управљане игре шах, развијене пре свега као алтернатива конвенцијалном начину играња шаха, како на рачунарима тако и на мобилним уређајима. Ово решење своју примену могло би наћи као игра погодна слабо покретним и непокретним лицима, то јест, у свим случајевима када је говор преферирани начин комуникације са уређајем.

Развој овакве апликације може се разложити на три подпроблема:

1. Развој модула за играње шаха – обухвата функционалности које подржавају логику игре, правила, репрезентацију табле и фигура, рачунање могућих потеза, функционалности оцене стања на табли (хеуристика), функционалности агента, подршка за више нивоа тежине и слично.

¹ <https://alexa.amazon.com>

² <https://www.apple.com/siri/>

³ <https://www.microsoft.com/en-us/cortana>

2. Модул за обраду и препознавање звука – обухвата процес трансформације и даље обраде улазног звука чији резултат представља потез који корисник задаје игри.
3. Графички кориснички интерфејс

Основа логике за подршку самој игри јесте минимакс алгоритам са алфа-бета одсецањем, док се модул за препознавање звука ослања на трансформацију улазног звучног фајла у мел спектограм, који се даље класификује употребом конволутивне неуронске мреже.

Обзиром да се разлагањем основног проблема јасно издвајају две готово независне целине (функционалности везане за логику игре и функционалности везане за обраду и класификацију улазног звука), тако и евалуација решења мора независно да обухвати обе целине. Евалуација логике, односно агента, извршена је играњем против бота са дефинисаним ЕЛО рангом. На овај начин могуће је квантитативно одредити успех алгоритма дефинисањем конкретног ЕЛО ранга имплементираном агенту. Евалуација класификације звука извршена је помоћу тест скупа података, обзиром да се само решење заснива на употреби конволутивне неуронске мреже. Узевши у обзир да је основна намена апликације разонода и да није намењена професионалној употреби, извршена је и ручна евалуација која пружа увид у рад апликације као целине. Иако приликом ручне евалуације није могуће одређивање квантитативне оцене као у претходна два случаја, значај стицања општег утиска знатно доприноси крајњем квалитету апликације.

Након извршене евалуације, апликација је показала задовољавајуће перформансе. У погледу класификације остварена је тачност од 98.8% за бројеве и 94% за слова. У погледу квалитета самог агента остварен је ЕЛО ранг од око 1120 те је општи утисак да апликација у целини задовољава своју намену. Важно је напоменути да агент може да се употреби у игри са напреднијим почетницима и ентузијастима, али да је далеко од професионалног нивоа или нивоа на ком су савремена решења намењена професионалној употреби као што су *Stockfish*⁴ (ЕЛО 3514) или *Alphazero*⁵ (ЕЛО 4650).

⁴ <https://stockfishchess.org/>

⁵ <https://www.deepmind.com/open-source/alphazero-resources>

Остатак рада организован је кроз следеће целине: У поглављу 2 изнесен је преглед стања у области, односно, направљен је осврт на важне радове из области и на радове који су утицали на настанак овог решења. У поглављу 3 изнесене су теоријске основе потребне за даље разумевање рада. Поглавље 4 посвећено је опису методологије и опису основне проблематике којом се рад бави, чиме се даје увид у резонување приликом поставки експеримената, који су предмет поглавља 5. У поглављу 6 изнесени су и објашњени резултати претходних експеримената, док се дискусијом у поглављу 7 заокружује целина методологија-експерименти. На послетку, у поглављу 8 изложени су најважнији закључци рада.

2. ПРЕГЛЕД СТАЊА У ОБЛАСТИ

Обзиром да се игра агента заснива на претходно наведеном *минимакс* алгоритму, који не спада у нарочито сложене алгоритме, фокус овог поглавља биће пребачен на обраду, анализу и класификацију звука, односно на проблематику *ASR*.

Први модели са могућношћу препознавања релативно великог вокабулара (1000 речи) [1] заснивали су се на концепту *скривених марковљевих модела* (енг. *Hidden Markov Models – HMM*). Даљим развојем ове идеје јављају се моћнији статистички модели способни за препознавање већих вокабулара. Ови модели показали извесну робусност према варијацијама у говору у смислу различитог акцента, брзине изговора и слично, али су у значајној мери били склони преприлагођавању [2]. Такође, један од значајнијих проблема представљао је тешкоћу при анализирању узрока грешака у покушају побољшања перформанси система [3].

Даљи развој области кретао се у правцу развоја хибридних модела који се ослањају на *HMM*. Овим су уведене новине у смислу различитих класификатора способних да обухвате више обележја, као што је предлог [4]. Користи се метода потпорних вектора (енг. *support vector machine*), при чему аутори наводе да остварени резултати указују на потенцијал у примени изложене методологије, иако истичу потребу за даљим развојем и унапређивањем решења.

Развојем рачунара у погледу хардверских капацитета отвара се простор за употребу сложених алгоритама, нарочито неуронских мрежа, када моћ *ASR* система експоненцијално расте. Савремени *ASR* модели су компликовани системи који се састоје од различитих компоненти - акустичких модела, језичких модела, модела говора и сл. У раду [5] аутори предлажу ново решење под називом *LAS* (енг. *Listen, Attend and Spell*), у коме излажу сложено решење претежно засновано на *LSTM* рекурентним неуронским мрежама, које је способно за транскрипцију слово-по-слово без употребе језичких или граматичких модела *HMM* и сл. За потребе експеримента, аутори су користили скуп података сачињен од три милиона ручно преписаних узорака гугл гласовних претрага укупне дужине око две хиљаде сати, који је процесом аугментације повећан двадесет пута. Модел је остварио 14.1 *WER* скор, што представља импресиван резултат узевши у обзир одсуство језичких модела.

Како *ASR* примењен на шах у одређеној мери одступа од класичног проблема препознавања говора, важно је напоменути неке од специфичних карактеристика које представљају окосницу резона приликом одабира начина за решавање постављене проблематике.

1. Ограничен и веома мали вокабулар – основна разлика и олакшавајућа околност која сама по себи елиминише потребу за комплексним и вишекомпонентним решењем и отвара врата ка ефикасном решењу које ради брзо.
2. Вокабулар сачињен од кратких речи – отежавајућа околност представља чињеницу да су колоне у шаху означене словом, те су звучни записи веома кратки и као такви нису погодни за класификаторе.
3. Вокабулар сачињен од сличних речи - додатна отежавајућа околност, поред кратких, фонетски сличне речи.
4. Потреба да решење буде робусно – отпорност на спољни шум и слично.

У књизи [6] у поглављу посвећеном употреби конволутивних мрежа за препознавање говора, позивајући се на [7, 8, 9] аутори наводе конволутивне мреже као најбољи избор за рад са сиромашним скуповима података, што је свакако случај у овом раду.

Када се све претходно узме у обзир, решење у раду [10] може се оценити као веома погодно за конкретну проблематику овог рада. Централна идеја рада [10] своди се на увођење конволутивне уместо густо повезане неуронске мреже (*DNN*), која се користи за моделовање комплексних релација звучних обележја. Употребљена конфигурација хиперпараметара била је следећа: 150 *feature* мапа, филтер величине 8 и *pooling* величине 6. Предложено решење евалуирано је и упоређено са конкурентским решењима како на малом тако и на великом вокабулару (детаљније [10]), где су остварени резултати бољи за до 10% у односу на *DNN* решења, која су сама по себи донела значајан напредак у поређењу са претходним решењима. Поврх тога, конволутивне мреже имају могућност рада са великим бројем обележја што погодује кратким а, веома сличним звучним узорцима у шаху, раде релативно брзо и не захтевају имплементацију додатних компонената (односно помоћних алгоритама) за успешну класификацију улазног звука. Основни недостатак овог приступа огледа се у потреби за трансформацијом улазног звука у *мел спектограм*, односно у немогућности директног рада над сировим улазом.

Архитектура решења изложеног у овом раду ослања се рад [11], који се бави тематиком најсличнијом тематици овог рада,

односно, препознавањем изговорених цифри на енглеском језику. Архитектура изложеног решења обухвата конволутивну мрежу са пет конволутивних слојева са по 12, 24, 48, 48 и 48 филтера по сваком слоју и једним густо повезаним слојем на крају. На скупу података од око 38,900 узорака распоређених у 10 класа, аутор рада остварује прецизност од око 98%.

3. ТЕОРИЈСКИ ПОЈМОВИ И ДЕФИНИЦИЈЕ

У овом поглављу биће изложене теоријске основе потребне за даље разумевање рада. Поглавље ће бити организовано кроз две логичке целине – прво ће акценат бити стављен на теоријске основе везане за имплементирану логику агента (3.1), док ће касније бити изложене методе примењене за трансформацију и класификацију звука (3.2).

3.1 Минимакс алгоритам

Минимакс представља алгоритам претраге који се превасходно користи за доношење одлука у играма нулте суме два играча. **Игре нулте суме** односе се на појам из теорије игара који се односи на ситуацију у којој добитак једног учесника директно кореспондира се губитком противника у истој мери и обрнуто. Другим речима, збир добитака (односно губитака) свих учесника увек износи нула. Заснован на принципу нулте суме, минимакс је алгоритам који претражује **стабло игре**, при чему покушава да минимизира могући губитак агента разматрајући најнеповољнији случај (односно највећи губитак) под претпоставком да је циљ противника управо тај да оствари највећу добит за себе, односно, нанесе максимални губитак агенту.

Стабло игре представља структуру података у виду стабла, где коренски чвор одговара тренутном, текућем стању игре, а чворови дубине један представљају сва стања у која је могуће доћи из текућег (дакле, одговарају свим могућим потезима из дате позиције). Чворови на дубини два представљају сва могућа стања противника, а рачунају се развијањем претходно израчунатих стања на дубини један. Овај процес смене рачунања свих могућих потеза агента (непарне дубине) и противника (парне дубине) врши се, идеално, до тренутка када су обрађени сви могући потези. Како је ово процес експоненцијалне сложености, често (што је случај и са игром шах) није могуће развити све могуће потезе, обзиром на рачунарска и меморијска ограничења, те се стабло развија до неке дефинисане дубине, пожељно што веће.

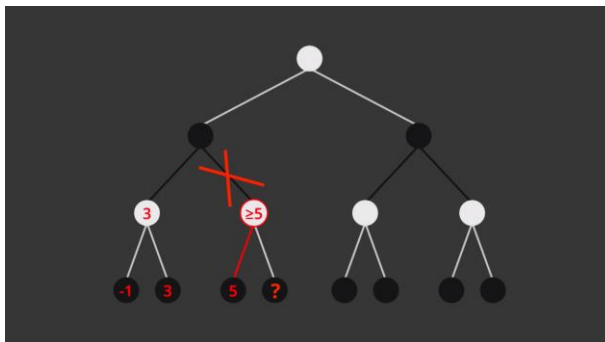
Важно је напоменути да се минимакс заснива на две претпоставке:

1. Противник агента игра оптимално, односно алгоритам покушава да предвиди противнички потез узимајући у обзир потез који ће противника довести у најповољнију ситуацију.
2. Игра не зависи од *фактора среће*, односно игра се може посматрати као искључиво стратешка, где је потез детерминисан искључиво доприносом који доноси играчу, без компоненте шансе.

За евалуацију потеза, односно стања игре, минимакс користи **хеуристичку функцију** која као аргумент узима стање игре и даје оцену колико је дато стање повољно за играча (односно неповољно за противника).

Сам минимакс је рекурзивни алгоритам који врши претрагу стабла игре у дубину. Доласком до терминалног чвора, односно до максималне дубине стабла, позива се хеуристичка функција која даје оцену стања табле. Циљ једног играча је да максимизира оцену табле, док је циљ противника да је минимизира. Обзиром на то, чворови на претпоследњем чвору добијају вредност у односу на то који играч је на потезу на тој дубини. Тако за случај када је максимизирајући играч на потезу на датој дубини, узима се максимална вредност од могућих стања у које је могуће доћи тим потезом јер ће право тај потез играча довести у најповољније стање (видети *претпоставку 1.* изнад). Аналогно, за случај минимизирајућег играча, алгоритам бира најмању вредност. На овај начин, вредности се пропадају од дна ка корену стабла, док се при предикцији потеза у обзир узима најповољнија ситуација за играча који је на потезу на датој дубини.

Обзиром да се одабир потеза одиграва наизменично искључиво на основу вредности чворова у стаблу, поставља се питање да ли је неопходно евалуирати све могуће потезе приликом доношења одлуке. Одговор на ово питање је не и заснива се на концепту под називом **алфа-бета одсецање** (енг. *alpha-beta pruning*). Алфа-бета одсецање представља алгоритам који проширује класични минимакс помоћу кога се зауставља даља евалуација дела минимакс стабла игре, када се у току евалуације пронађе најмање један потез гори од претходно евалуираног. Логика која лежи иза овог алгоритма биће укратко илустрована следећим примером (Слика 3.1).



Слика 3.1 – Приказ стабла игре са алфа бета одсецањем

На слици 3.1 потези максимизирајућег играча означени су белом, а потези минимизирајућег играча црном бојом. У ситуацији када је трећем терминалном чвору додељена вредност 5, јасно је да ће максимизирајући играч сигурно доћи у стање са вредношћу већом или једнаком са пет, али како том избору претходи избор минимизирајућег играча, јасно је да ће се он одлучити за леву страну стабла, односно за најмање неповољну позицију. Тим речено, није потребно рачунање чвора означеног упитником, обзиром да се у то стање никада неће доћи (резон вођен претпоставком 1 горе). Алфа-бета одсецање представља моћан алат приликом оптимизације минимакс алгоритма, обзиром да за стабла веће дубине долази до нарочитог изражаја, те се обезбеђује оптималан рад минимакс алгоритма уз изостављање евалуације често значајног дела стабла.

3.2 Обрада звука

Претходно је наведено да је за потребе класификације улазног звука имплементирана конволутивна неуронска мрежа. У овом сегменту биће изложене теоријске основе дубоких неуронских мрежа, које су основа за разумевање принципа рада конволутивних неуронских мрежа уопште. Након тога, биће изложени детаљи архитектуре и начин рада конволутивних мрежа, а на послетку и начин трансформације улазног звучног сигнала у формат погодан за ову мрежу.

3.2.1 Неуронска мрежа

Неуронске мреже (*енг. neural networks*) представљају једну од најпримењенијих метода машинског учења. Многобројне примене ових комплексних алгоритама у решавању софистицираних проблема, довеле су до тога да у лаичким круговима постоји тенденција поистовећивања целе области машинског учења, односно вештачке интелигенције са неуронским мрежама, што је наравно далеко од истине.

Неуронске мреже се могу посматрати као параметризоване структуре које могу послужити за апроксимацију других функција. Аналогно са другим методама машинског учења, проналажење оптималних параметара (који дају најбољу апроксимацију полазне функције) врши се математичком оптимизацијом неког критеријума квалитета апроксимације, што у зависности од контекста података и саме структуре мреже, може бити рачунарски врло изазовно.

Основну варијанту представљају потпуно повезане неуронске мреже (*енг. fully connected*), у наставку неуронске мреже.

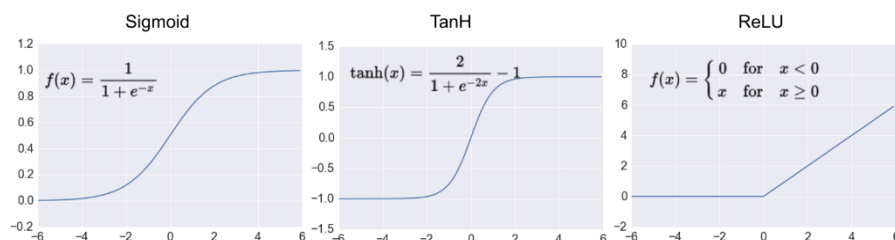
Основна јединица неуронских мрежа јесте **неурон**. Неурони представљају једноставне параметризоване функције. Улога сваког неурона јесте рачунање линеарне комбинације својих улазних аргумената. Над добијеном вредношћу даље се извршава одређена нелинеарна трансформација, такозвана **активациона функција** (*енг. activation function*).

Структура неуронске мреже организована је у **слојеве**, тако да неурони једног слоја примају као своје аргументе, односно улазе, вредности, односно излазе, свих неурона претходног слоја и сви неурони прослеђују своје излазне вредности наредном слоју, одакле и потиче назив потпуно повезана неуронска мрежа. Свака веза између неурона два слоја дефинисана је вредношћу која улази у претходно поменути линеарну комбинацију са конкретном излазном вредношћу неурона. Ове вредности називају се **тежине** и проблем оптимизације неуронске мреже се своди управо на оптимизацију вредности тежина.

Први, улазни слој неурона, који добија сирове податке назива се **улазом мреже**, док се неурони последњег слоја се називају **излазом мреже**. Сви слојеви чије јединице прослеђују своје излазе другим неуронима, не рачунајући улазни слој, називају се **скривеним слојевима**. Уколико неуронска мрежа има више од једног скривеног слоја, назива се **дубоком неуронском мрежом** (*енг. deep neural network*) у које, између осталог, спадају и **конволутивне неуронске мреже**. Основни допринос скривених слојева огледа се у томе што се вредности неурона скривених слојева могу сматрати новим

обележјима података над којима се неуронска мрежа учи. Тим речено, моћ неуронске мреже лежи управо у њеној могућности да кроз независан процес конструише нова обележја над улазним подацима, док је сваки следећи скривени слој способан да надограђује над претходним и тако издваја све суптилнија и сложенија обележја, што је особина нарочито значајна за конволутивне неуронске мреже о којима ће више речи бити у наредном сегменту.

Претходно је наведено да се по извршеној линеарној комбинацији улазних аргумената за сваки неурон, над добијеном вредношћу позива активациона функција у виду неке нелинеарне трансформације, међутим, није образложена улога активационе функције, осим што је наведено да резултат ове трансформације уједно представља и излазну вредност неурона која се даље пропагира кроз мрежу. У математичком смислу, без улажења у нарочите детаље, активациона функција представља функцију којом се арбитарна вредност ограничава, односно пресликава, у неки интервал (често $[-1, 1]$, $[0, 1]$ или $[0, +\infty)$). Овакво понашање инспирисано је потребом да се претходно поменутој линеарној трансформацији улаза, дода степен нелинеарности, односно да се спречи линеарност целокупног система. У супротном, сви подаци би пролазили кроз линеарне функције, чијом композицијом се опет добија линеарна функција. Слика 3.2 приказује неке од често коришћених активационих функција.



Слика 3.2– Најчешће коришћене активационе функције: ситмоидна, тангенс хиперболичка и исправљачка линеарна јединица.

Коначно, објаснићемо процес **тренирања** неуронских мрежа. Тренирање мреже представља процес оптимизације претходно поменутих **тежина**, којим се минимизује **функција губитка** (енг. *loss function*), односно, грешка апроксимације циљане функције неуронском мрежом. Сам процес одвија се тако што се мрежи прослеђују означени подаци, а излаз из мреже се пореди са конкретном ознаком, односно стварном вредношћу. Сви подаци се

мрежи прослеђују у више циклуса који се називају **епохе**, током којих се мрежа оптимизује односно **учи**. По прослеђивању конкретног податка мрежи у оквиру епохе, функцијом губитка на основу разлике излаза мреже и стварне вредности рачуна се вредност **губитка** (енг. *loss*). Како излаз из мреже зависи од вредности тежина, јасно је да и функција губитка управо зависи од истих вредности. Тим речено, конкретна оптимизација вредности појединачне тежине ослања се на израчунавање парцијалног извода функције тежине у односу на ту тежину, чиме се добија градијент.

Да би се избегло осциловање око глобалног минимума (обзиром да је вредност градијента за различите улазе различита), уведен је концепт **стопе учења** (енг. *learning rate*), која представља број мањи од 1, што омогућава да се ажурирањем тежине померимо ка оптимуму, за мали корак. Отуда потреба да се тренирање мреже врши кроз више епоха над истим подацима, када је за очекивати да се све тежине налазе близу (или идеално у) глобалном оптимуму. На овом концепту заснива се поступак којим се ажурирају вредности тежина који се назива алгоритам пропагације уназад (енг. *backpropagation*) у оквиру кога се креће од последњег ка улазном слоју и на претходно описан начин врши ажурирање тежина. У случају сложених дубоких неуронских мрежа, које подразумевају велики број неурона, а самим тим и велики број операција множења приликом пропагације вредности, алгоритам пропагације уназад није лако примењив обзиром да вредности парцијалних извода често умеју да буду практично нула (множењем бројева мањих од 1 резултат конвергира ка нули), или пак да буду огромне. Овај проблем у литератури познат је као проблем нестајућих односно експлодирајућих градијената (енг. *exploding and vanishing gradients*). Да би се избегло овакво понашање мреже, честа пракса је да се као активациона функција користи горепоменуто *ReLU*, обзиром на константан извод функције. Овај концепт примењен је и у овом раду, али ће о детаљима бити речи касније.

3.2.2 Конволутивне неуронске мреже

Конволутивне неуронске мреже представљају специјализацију дубоких неуронских мрежа које се најчешће користе за анализу слика, што је случај и у овом раду, те ће у наставку њихов рад бити описан кроз пример рада са улазном сликом.

Уобичајена структура конволутивне мреже подразумева смењивање две врсте слојева – **конволутивних слојева** (*енг. convolution layer*) и **слојева агрегације** (*енг. pooling layer*) при чему је могуће и да се иста врста слоја понови више пута. Као и код обичних густо повезаних мрежа, на вредност излаза конволутивног слоја примењује се активациона функција. Последњи, излазни слој конволутивне мреже обично представља бар један густо повезан слој или пак више густо повезаних слојева, који уче над атрибутима претходно конструисаним од стране конволутивних слојева.

Основна идеја у позадини конволутивних мрежа јесте да се при анализи дела улазне слике у обзир морају узети локалне зависности, односно да се вредности неког пиксела слике морају посматрати у контексту његове непосредне околине, да би се извукле битне информације, што се постиже процесом **конволуције**. Овакав приступ донекле је у супротности са приступом густо повезаних мрежа које као улаз узимају једнодимензиони низ вредности, при чему се у обзир не узима информација о непосредном окружењу пиксела, па самим тим није могуће остварити резултате упоредивим са онима који се добијају конволутивним мрежама.

Процес конволуције подразумева постојање две матрице – прве којом је представљена анализирана слика, и друге помоћне, која се назива филтер или кернел (*енг. kernel*), помоћу које се неко обележје издваја са слике. Конволуција се одвија тако што се филтер (димензија мањих од димензија слике) помера за одређени корак (*енг. stride*) по слици, те се рачунањем скаларног производа филтера и захваћеног региона анализиране слике, добија нова вредност која се уписује у одговарајуће поље резултујуће матрице. На основу овога може се закључити да ће по завршетку конволуције, број колона резултујуће матрице одговарати броју скаларних множења у једном реду, док ће број врста одговарати броју прелаза филтера у нови ред. У сваком случају, величина резултујуће матрице мања је од полазне, што није увек пожељно, па се може избећи **проширивањем** обода полазне матрице (*енг. padding*) нулама или репликацијом постојећих вредности на ободима, како би величина резултујуће матрице одговарала величини полазне матрице пре проширивања. Моћ конволутивних мрежа лежи у томе што филтери не морају бити претходно

дизајнирани, већ су научени. Сваки конволутивни слој састоји се од низа филтера, који омогућавају детекцију различитих **научених** обележја, на почетку једноставних, чијом се комбинацијом и даљом пропагацијом кроз мрежу, формирају све сложенија обележја.

Обзиром да је описани процес рачунарски захтеван, нарочито у случају улазне матрица великих димензија, пожељно је (некад и неопходно) извршити редукцију димензија резултујуће матрице. Ово се постиже претходно поменути **слојевима агрегације**. Овај слој се често налази непосредно иза конволутивног слоја. Агрегацијом се укрупњују претходно добијене информације путем неке једноставне агрегационе функције углавном **максимума** (енг. *max pooling*) или **просека** (енг. *average pooling*), примењеној на суседна поља полазне матрице. Бенефит овог поступка може се илустровати кроз следећи пример: применом 4×4 *max pooling* агрегације, издваја се максимална вредност, те је информација да ли је филтер у датом региону детектовао одређено обележје, очувана. Са друге стране, губи се прецизна информација где је тачно у датом региону то обележје и детектовано. На послетку, када се у обзир узме да је за наведену 4×4 агрегацију, величина полазне слике смањена чак 16 пута, а да прецизна информација о локацији неког обележја није увек неопходна, поступак агрегације можемо оценити као веома користан.

Укратко, суштина конволутивних мрежа могла би се формулисати као процес кроз који се врши редукција количине улазних информација издвајањем научених обележја процесом без губитка битних информација.

3.2.3 Звук

Звук се може дефинисати као механички талас у фреквенцијском опсегу који перципира људско ухо. Прецизније говорећи, звук представља механички талас у опсегу фреквенција 16 Hz до 20 kHz.

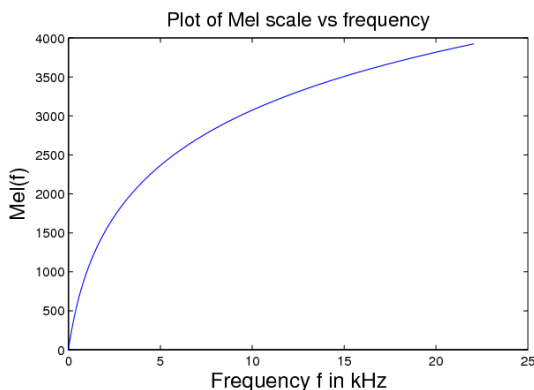
Обзиром да је звук по својој природи, континуалан сигнал, узорковањем звука потребно је редуковати овај континуални сигнал у низ дискретних вредности. Основни појмови везани за процес трансформације улазног континуалног звука у дискретни сигнал јесу **фреквенција узорковања** (енг. *sampling rate*), која представља учесталост узорковања сигнала и **битна дубина** (енг. *bit depth*), која представља ниво детаља, односно, квалитет самог узорка. Битно је напоменути да већа вредности фреквенције узорковања односно битне дубине дају квалитетнији запис, бележи се више информација, али се са друге стране повећавају, како меморијски тако и рачунарски захтеви у погледу његове обраде. Насупрот томе, ниске вредности дају

звучни запис ниже резолуције и квалитета, погодан за обраду, али овакав приступ резултује већим губитком информација. Наведена дилема нарочито је важна за овај рад, обзиром да је природа звучних записа који су предмет бављења рада таква да су они изузетно кратки, а веома слични, те да свака изгубљена информација представља потенцијални проблем при даљој класификацији. Детаљан коментар овог проблема биће изложен у наредним поглављима.

Обзиром да је претходно наведено да је за анализу звука употребљена конволутивна неуронска мрежа, те да је мрежа радила над сликама, остаје недоумица како и пре свега зашто, је сигнал, по природи секвенцијалан, трансформисан и потом анализиран у форми слике, што ће бити образложено у наставку.

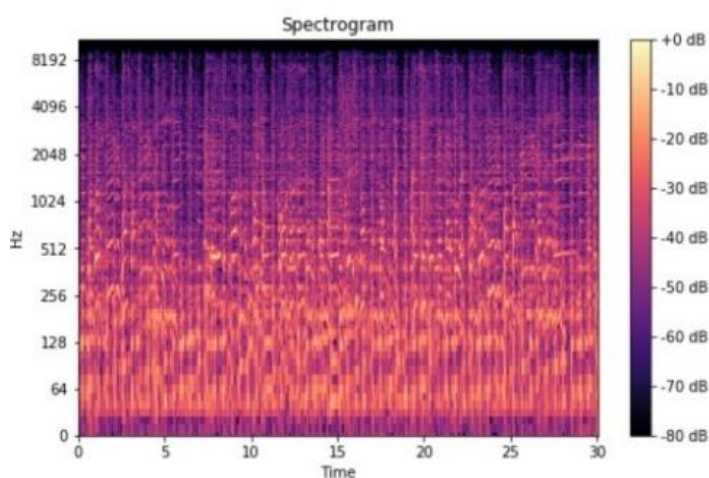
Први појам неопходан за разумевање претходног јесте **Мел скала**, која представља логаритамску трансформацију фреквенције улазног звучног сигнала. Основна идеја у позадини оваквог приступа јесте чињеница да људско ухо промене у фреквенцији звука не перципира линеарно, већ да се релативно мале разлике у нискофреквентном интервалу јасно уочавају, док се иста разлика (у апсолутној мери) у интервалу звука високих фреквенција веома тешко уочава. Тим речено, логаритамском трансформацијом звука, обезбеђен је механизам помоћу кога машина може да перципира звук на начин на који га перципирају људи. Формула 3.1 и Слика 3.3 дају илустрацију ове трансформације.

$$m = 2595 \log_{10} \left(1 + \frac{f}{700} \right) \quad (3.1)$$



Слика 3.3 – Мел скала

Мел спектрограм је спектрограм који обезбеђује визуелизацију звука, односно представља графички приказ промена у оквиру мел скале у зависности од времена. Дакле, x оса представља протекло време, y оса представља фреквенцију изражену на мел скали, док је боја на спектрограму дефинисана амплитудом, односно јачином звука. Наведеном графичком представом постиже се не само визуализација звука, већ визуализација звука на начин на који га перципира људско ухо, те представља оптималне податке за конволутивне неуронске мреже које су самостално способне да издвоје битна обележја са улазних слика. Слика 3.4 приказује пример мел спектрограма.

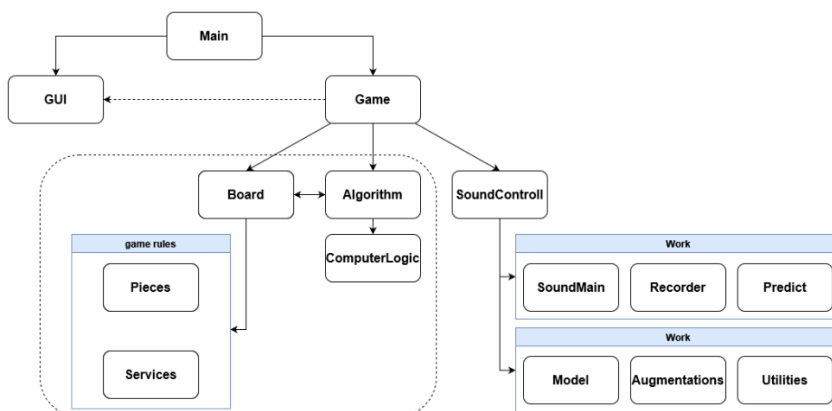


Слика 3.4 – Пример мел спектрограма

4. МЕТОДОЛОГИЈА

У овом поглављу биће представљена имплементација говором управљане игре шах. Решење је замишљено као интерактивна апликација која као улаз региструје изговорен потез, односно ознаку полазног, а потом одредишног поља, док излаз из система представља препознат и извршен кориснички потез уз играње одговарајућег потеза агента, када је корисник поново на потезу.

Након иницијалног излагања целокупне архитектуре система на вишем нивоу апстракције, поглавље ће бити организовано кроз две логичке целине у којима ће темељно бити представљена архитектура и имплементација самих модула. Прва целина бавиће се имплементацијом игре шах, што обухвата имплементацију правила саме игре, са акцентом на имплементацију агента. Друга целина обухвата рад са звуком. Архитектура система илустрована је сликом у наставку (Слика 4.1).



Слика 4.1 - Приказ архитектуре система

Централни модул апликације представља модул *Game* посредством кога се посредује комуникација како са графичким интерфејсом тако и са две најзначајније целине: модулима *Board* и *Algorithm*, који чине прву и *SoundControl* који представља другу целину. Модул *SoundControl* обухвата функционалности рада са звуком и као такав представља улаз у систем. Овде се врши регистровање, сегментација, обрада и препознавање улазног звука, те излаз чини одређени кориснички потез. Такође,

у оквиру овог модула налазе се функционалности за тренирање модела као и за рад са звуком генерално. Модул *Board* енкапсулира знање о правилима игре, фигурама, њиховом начину кретања и генерално садржи представу о текућем стању игре - табли. Модул *Algorithm* подржава функционалности агента, рачунања најбољег потеза, хеуристичке функције и слично. Као улаз добија кориснички потез одређен *SoundControll* модулом, разматра најбољи агентски потез у складу са задатом тежином игре и као излаз враћа ново стање табле, односно нову *Board* инстанцу. Наведени модули ће бити детаљније обрађени у наредном делу.

4.1 Модул за логику игре и управљање потезима агента

Као што је претходно наведено, улаз у овај модул представља дефинисани кориснички потез и он енкапсулира логику и правила шаха, чува тренутно стање табле и имплементира функционалности агента. Укратко, циљ ове компоненте је да уз подршку графичког интерфејса обезбеди функционалности играња шаха, независно од начина корисничке интеракције са апликацијом, односно начина задавања потеза. Свакако најзначајнији део модула представља подршка за управљање агентом, док је имплементација праћења стања игре, фигура и сличних помоћних функција тривијална, те ће бити ван фокуса поглавља.

4.1.1 Агентски алгоритам

Обзиром да шах представља игру нулте суме, као агентски алгоритам одабран је минимакс са алфа бета одсецањем. У претходном поглављу објашњена је интуиција и начин рада минимакса на апстрактном нивоу, те ће следећи део дати детаљнији опис конкретне имплементације. Исечак кода који приказује Слика 4.2 обухвата прву половину алгоритма односно максимизирајући део, док је минимизирајући део имплементиран аналогно.

```

@staticmethod
def minimax(board, depth, color_on_turn, alpha, beta, goal_depth, first_move, first_user_move):

    if (depth == goal_depth):
        return Heuristic.calculate(board)

    if (board.computer_color == color_on_turn):
        best_value = -100000
        moves = board.get_possible_moves(color_on_turn)
        if (len(moves) == 0):
            if (not board.check_if_it_is_check(color_on_turn)):
                return 0
            else:
                return best_value

    if (first_move == -1):

        move_to_make = -1
        Heuristic.possible_first_moves = moves
        for i in range(0, len(moves)):
            if (Heuristic.check_for_possible_tie_computer(moves[i])):
                value = 0
            else:
                value = Heuristic.minimax(moves[i], depth + 1, get_opponent_color(
                    color_on_turn), alpha, beta, goal_depth, i, first_user_move)
            if (best_value < value):
                best_value = value
                move_to_make = i
            alpha = max(alpha, best_value)
            if (beta <= alpha):
                break
        Heuristic.write_computer_move(
            Heuristic.possible_first_moves[move_to_make])

        return Heuristic.possible_first_moves[move_to_make]
    else:
        for i in range(0, len(moves)):
            value = Heuristic.minimax(moves[i], depth + 1, get_opponent_color(
                color_on_turn), alpha, beta, goal_depth, first_move, first_user_move)
            best_value = max(value, best_value)
            alpha = max(alpha, best_value)
            if (beta <= alpha):
                break

        return best_value

```

Слика 4.2 – Имплементација минимакс алгоритма

Аргументи који се прослеђују функцији су: текуће стање табле, играч на потезу, алфа и бета вредност, циљана дубина претраге стабла игре као и вредности да ли је реч о првом потезу играча или агента.

Обзиром да је приказан максимизирајући део, таргетирана вредност се поставља -100000 која се даље повећава. По рачунању свих дозвољених потеза и провере да ли је играч под шахом уколико нема дозвољених потеза (што указује на крај игре тј. терминални чвор стабла), за сваки потез рекурзивно се позива минимакс. Затим се алфа вредност поставља на максимум прослеђене алфе и тренутно најбољег потеза. Уколико је прослеђена бета вредност мања једнака од вредности алфе, даљи развој текућег минимакса није потребан.

4.1.2 Хеуристичка функција

Пре уласка у детаље имплементације хеуристичке функције важно је навести ограничења односно захтеве који се намећу. Наиме, идеална хеуристичка функција уме савршено да оцени таблу, што значи да ће поред пуког броја и врсте фигура, умети да произведе и оцену позиције оба играча. Са друге стране, пун потенцијал минимакса остварује се са обиласком комплетног стабла игре, а како је то у овом случају због превеликог броја могућих потеза немогуће, идеја је да извршена претрага буде што дубља, а да то буде реализовано у разумно кратком времену. Тим речено, превелики број стања које је потребно евалуирати намеће ограничење над комплексношћу хеуристике, те је неопходно наћи баланс између квалитета и комплексности функције. Узевши то у обзир, хеуристичка функција имплементирана је помоћу две помоћне функције: *calculate_piece_value* и *get_position_value*.

Да би се максимално смањила комплексност, хеуристичка функција пролази кроз сва поља табле, и за свако поље са фигуром позива ове две функције комплексности $O(1)$. Прва, *calculate_piece_value*, враћа предефинисану оцену искључиво на основу врсте тренутне фигуре (краљ је највреднији, затим краљица, топ, скакач и ловац...). Друга, *get_position_value*, даје оцену позиције на основу позиције текуће фигуре на табли. Имплементација ове функције сведена је на враћање вредности за дату позицију из предефинисане матрице позиција за дату фигуру. Слика 4.3 илуструје пример овакве матрице за скакача.

```
knight_positions = [
    [-50, -40, -30, -30, -30, -30, -40, -50],
    [-40, -20,  0,  5,  5,  0, -20, -40],
    [-30,  5, 10, 15, 15, 10,  5, -30],
    [-30,  0, 15, 20, 20, 15,  0, -30],
    [-30,  5, 15, 20, 20, 15,  0, -30],
    [-30,  0, 10, 15, 15, 10,  0, -30],
    [-40, -20,  0,  0,  0,  0, -20, -40],
    [-50, -40, -30, -30, -30, -30, -40, -50]
]
```

Слика 4.3 – Приказ матрице позиција скакача: идеја да је скакач најкориснији у центру док на периферији губи вредност обзиром на ограничено кретање

На овај начин, хеуристика ипак разматра позицију фигура, док је значајан недостатак у томе што се не оцењује утицај фигура једних на друге, односно релативна позиција како својих тако и противничких фигура. Глобално гледано, претходни исказ није у потпуности тачан, обзиром да развој стабла разматра све могуће потезе, те и потезе у којима једна фигура једе другу, те се на тај начин ипак у одређеној мери евалуира непосредан утицај фигура који произилази из позиције, али и даље нема суштинског знања о позицији, те је концепт жртвовања вредне фигуре зарад остварења позиционе предности, практично непознат алгоритму, осим у тривијалним случајевима, када води до сигурног завршетка игре кроз пар потеза.

4.2 Модул за рад са звуком

Модул за рад са звуком обухвата функционалности потребне за препознавање односно класификацију изговорених поља (што представља улазне податке), као и функционалности за снимање, обраду, манипулацију, аугментацију и обуку модела за класификацију. Детаљи о самом скупу података, аугментацији, тренирању модела и евалуацији биће изложени у наредном поглављу, док ће наредни сегмент обухватити, пре свега архитектуру модела за класификацију као и на начин снимања, сегментације и обраде улазног звука.

4.2.1 Обрада улазног звука

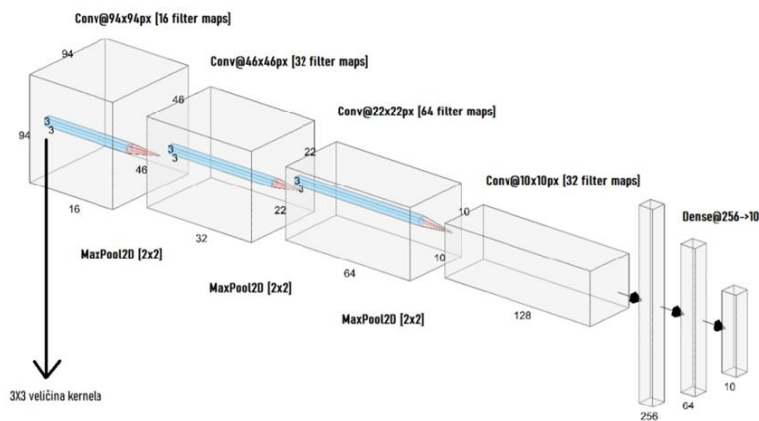
Проблем снимања улазног звука може се разложити у пар подпроблема. Поред конкретног снимања и одређивања што ниже фреквенција узорковања, која притом обезбеђује довољну резолуцију снимка, јављају се и следећи проблеми: утицај позадинског шума, паузе током говора, одређивање минималног интервала тишине пре започињања снимања, сегментација изговорених речи, недовољно гласан снимак... Обзиром на претходно наведено, употребљена је библиотека *speech_recognition*, односно њена класа *Microphone* искључиво у сврхе снимања и сегментације звука, не за сврхе препознавања изговореног. Методом *adjust_for_ambient_noise* решава се проблем издвајања звука и позадинског шума, обзиром да ова метода динамички одређује праг за издвајање звука „у првом плану“, на основу иницијалног слушања снимка у коме нема говора чиме се дефинише праг тишине.

Сегментација се врши тако што постављени праг тишине дефинише границу раздвајања односно минимални интензитет основног звука. Када год интензитет падне испод ове границе,

алгоритам претпоставља да корисник не говори, те може да издвоји реч из полазног звука. Проблем који се јавља при овом приступу је да у неком тренутку праг тишине може бити постављен ниже од оптималних вредности, те узроковати да се снимање настави и по завршетку говора, односно да се повећањем степена амбијенталног звука у међувремену, алгоритам „превари“, те да позадински шум региструје као говор. Да би се ово спречило, дефинисан је временски интервал у оквиру кога реч мора бити дефинисана, када се снимање прекида и звук евалуира. Овај сценарио дешава се у изузетно ретким случајевима, у ситуацијама када је присутан заиста јак ниво амбијенталног шума уз микрофон лошег квалитета, те је случај издвојене реч максималне дужине заиста реткост, али је овим путем опасност од грешке при раду отклоњена.

По снимању и сегментацији улазног звука, и провером да ли је улаз сегментиран на четири речи, претпоставља се да је корисник успешно одредио задати потез у формату полазно поље, одредишно поље, односно слово-цифра, слово-цифра. Над сировим снимцима извршава се појачавање тона (у случају микрофона лошијег квалитета), а потом се трансформишу у мел спектограме када су спремни за прослеђивање неуронској мрежи на анализу.

Модел за класификацију мел спектограма имплементиран је као конволутивна неуронска мрежа архитектуре приказане на слици (Слика 4.4).



Слика 4.4 - Архитектура класификационог модела

Мрежа, поред улазног и излазног слоја, садржи и четири конволутивна скривена слоја. Конволутивни слојеви редом садрже 16, 32, 64 и 128 филтера, величине 3×3 . Иза сваког конволутивног слоја додат је 2×2 *pooling* слој. На конволутивне слојеве надовезују се два густо повезана слоја, са по 256, односно, 64 неурона. Као активациона функција иза сваког слоја, конволутивног или густо повезаног, употребљена је *ReLU*. Излазни слој је класични густо повезани слој са *softmax* активацијом.

Као оптимизатор коришћен је *ADAM*, а као *loss* функција *sparse_categorical_crossentropy*. Приликом тренирања модела испробане су различите варијације наведене архитектуре, о којима ће више речи бити у наредном поглављу.

4.3 Коришћени алати

У овом поглављу биће специфицирани и објашњени алати и библиотеке коришћени приликом имплементације решења.

За решавање проблема снимања звука, уз прилагођавање позадинском амбијенталном шуму употребљена је класа *Microphone* библиотеке *speech_recognition* [12]. За потребе процеса обраде звука у смислу исецања сировог снимка на речи, појачавања улазног сигнала и слично, кориштена је класа *AudioSegment* библиотеке *pydub* [13].

Приликом аугментације снимака највећи део посла обављен је употребом библиотеке *librosa* [14]. Трансформација сировог сигнала у *mel-spectrogram* извршена је помоћу *librosa* библиотеке. За генерисање слике спектограма кориштена је *matplotlib* [15] библиотека, док је за даљу обраду генерисане слике употребљена *cv2* [16] библиотека.

За потребе мешања и поделе података на тренинг, валидациони и тестни скуп употребљене су функционалности библиотеке *scikit-learn* [17]. Неуронска мрежа имплементирана је помоћу *keras* [18] библиотеке, у позадини ослањајући се на *tensorflow* [19] радни оквир.

5. ЕКСПЕРИМЕНТИ

У овом поглављу биће изложени најзначајнији експерименти на основу којих је имплементирано финално решење. Такође, биће детаљно описан скуп података коришћених за тренирање, валидацију и евалуацију класификационог модела. Одређивање хиперпараметара модела и одређивање параметара генерално извршено је емпиријски и интуитивно комбинацијом много различитих вредности. Основна улога експеримената је да покаже и потврди перформансе система у реалном раду, те ће фокус пре свега бити на приказивању метрика којима може да се оправда рад, како компоненти тако и рад система као целине. У погледу хардвера, сви експерименти извршени су на машини са следећом конфигурацијом: процесор - *Intel i7-8750H*; графичка картица – *NVIDIA GeForce GTX 1050 Ti 4GB; 16 GB RAM*.

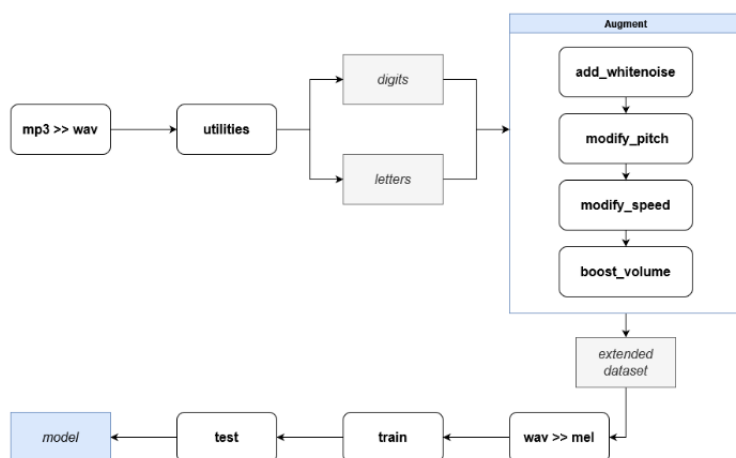
5.1 Скуп података

За потребе обучавања, валидације и тестирања модела ручно је прикупљен и креиран скуп података. У наставку ће бити детаљно описан како скуп података, тако и начин аугментације и претпроцесирања истог.

Сам скуп података се може поделити на два дела. Први, где су подаци бројеви са класама 1-8 и други - слова са класама А-Х. Први скуп броји 105 оригинална аудио записа по класи где је однос женских и мушких гласова око 40:60. Скуп података за слова садржи нешто више снимака по класи, односно 151 снимак са сличним односом мушких и женских гласова. Оба скупа података су строго балансирана и свака класа садржи једнак број елемената. Након поступка аугментације, класе су бројале по 1155 и 1661 елемент респективно, односно иницијални скуп података увећан је 11 пута. Класе су строго балансиране из два разлога. Након трансформације звучних записа у мел спектограме, извршен је поступак је поступак нормализације и скалирања на 128×128 пиксела, обзиром на хардверска ограничења машине на којој је обучавање мреже вршено.

Као један од разлога већег обима словног скупа података издваја се сличност појединих класа. За подстицање класификације, неколицина снимака, превасходно у најсличнијим класама („а“-„ха“ и „бе“-„де“) садржи кратке снимке сугласника („х“, „б“, „д“) који су омогућили мрежи јаснију екстракцију одговарајућих обележја снимака генерално.

Процес предпроцесирања и аугментације сирових звучних снимака илустрован је следећим дијаграмом (Слика 5.1).



Слика 5.1 – Предпроцесирање и аугментација

Процес аугментације обухватио је проширивање иницијалног скупа података додавањем позадинског шума три различита интензитета, затим смањивањем и повећањем висине тона снимака у четири нивоа интензитета (два смањења и два повећања). Такође, извршена је измена брзине са факторима 0.75, 1.5 и 1.75.

5.2 Евалуација агентског алгоритма

У овом експерименту извршена је евалуација агентског алгоритма односно имплементираног минимакса. Првобитно је одређена дубина претраге стабла са следећим испитаним вредностима: 8, 6, 5, 4 и 3. Као критеријум за одређивање вредности параметра узета је брзина играња потеза у најзахтевнијој фази игре, *касној средишњој партији*, где су позиције играча развијене. У овој фази постоји велики број опција у смислу дозвољених потеза, те је рачунарски најзахтевнија.

Да би се дала квантитативна оцена квалитета, алгоритам је тестиран у партијама са ботовима на *chess.com* са различитим ЕЛО рангом. Алгоритам је тестиран на дубини претраге 4. Дубина 4 је одређена као оптимална вредност при којој алгоритам ради довољно брзо. Више детаља о самој ЕЛО метрици као и резултатима на основу

којих је одабрана баш дубина 4, биће изложено у следећим поглављима.

5.3 Снимање и сегментација звука

Предуслов за успешан рад класификационог алгоритма представља квалитет снимања и обраде улазног звука. Претходно је наведено да решење овог проблема није имплементирано од нуле, већ да је употребљена библиотека. У овом сегменту биће прецизиран параметри којим су остварени најбољи резултати снимања и сегментације звука, као и детаљан опис и резон у позадини одабира самих вредности. Сви параметри приказани у табели (Табела 5.1) добијени су емпиријским путем, тестирани у различитим условима (у погледу квалитета микрофона, близине говорника микрофону, степена позадинског шума, гласноће и брзине изговора...).

Назив параметра	Вредност
<i>sample_rate</i>	22050
<i>duration</i>	0.6
<i>pause_threshold</i>	0.8
<i>non_speaking_duration</i>	0.25
<i>phrase_time_limit</i>	7
<i>min_silence_len</i>	100
<i>silence_thresh</i>	-16
<i>keep_silence</i>	400

Табела 5.1 - Параметри снимања звука

Методом *adjust_for_ambient_noise* решава се проблем издвајања звука и позадинског шума, динамичким одређивањем прага позадинског шума. Сваки забележени звук интензитета мањег од граничног, сматра се позадинским шумом, те се на основу тога врши издвајање речи. Да би се дефинисао овај праг, потребно је обезбедити снимање звука у неком што краћем интервалу када корисник неће говорити. Вредност најкраћег интервала при коме алгоритам успешно реагује на износи 0,6 секунди, те је ова вредност прослеђена као аргумент поменутој функцији. Обзиром на то, постављен је и параметар *pause_threshold* на 0,8 секунди, којим се обезбеђује да се сам звук неће снимати у току прилагођавања амбијенталном шуму. Да би се спречило одсецање почетка или краја изговорених речи (које су и онако кратке па је сваки забележени детаљ од велике важности),

постављен је параметар *non_speaking_duration* на 0,25 секунди, чиме снимак постаје „обложен“ са по 0,25 секунди додатно снимљене тишине, па је и резултујући мел спектограм веће резолуције. Како конволутивне мреже добро препознају транслирана обележја са слике, то се овим поступком не нарушава интегритет улазног снимка. Са друге стране, аугментацијом података која обухвата додавање различитог интензитета позадинског шума обезбеђена је робусност класификатора на додату тишину – која суштински представља амбијентални звук.

За решавање проблема варијабилног интензитета позадинског шума, подешавањем параметра *phrase_time_limit* одређен је интервал од 7 секунди, који представља максимално толерисану дужину речи. Овај параметар могао се подесити и на значајно мању вредност, али је као основни циљ постављено спречавање ситуације да алгоритам непрекидно слуша, а не да реч буде минималног трајања. Овим путем се врши минимално мешање у рад система, обзиром да динамичко постављање прага углавном уме успешно да одреди праг тишине, те да самостално прекине снимање.

Сегментација изговорених речи врши се на основу одређеног прага тишине.

5.4 Тестирање класификационих модела

Трећи експеримент односи се на тестирање класификационих модела, односно конволутивне мреже претходно изложене архитектуре (поглавље 4.2.1). Модел је трениран кроз девет епоха, са величином *batch*-а вредности 64, добијених емпиријским путем.

5.5 Евалуација

У наредном делу биће дефинисан начин евалуације сваког од претходно изложених експеримената, чији ће резултати бити дискутовани у наредном поглављу.

Експеримент за евалуацију агентског алгоритма састојао се из два дела. За одређивање дубине претраге стабла игре, било је потребно пронаћи највећу могућу вредност за коју алгоритам ради у прихватљивом времену (испод 10 секунди) у најзахтевнијем делу игре. Квалитет игре агента одређен је дефинисањем приближног ЕЛО скорa играњем 10 партија са ботом. ЕЛО систем оцењивања је метода за израчунавање релативних нивоa вештина играча у играма за два играча са нултим сумом. Овај алгоритам се налази у широкој употреби међу разним шаховским организацијама иако је готово свака

организација развила своју специфичну варијацију на основу оригиналног алгоритма. Конкретно, *chess.com* користи варијацију Аустралијске шаховске организације - *The Glicko System*. Детаљи ове варијације изложени су у раду на ког референцира документација *chess.com* [20].

Експеримент за одређивање параметара за снимање и сегментацију звука је евалуиран ручно, увидом у сегментисане снимке без конкретне метрике. Циљ овог експеримента био је да де обезбеди правилна сегментација звука, што је након проналаска оптималних параметара готово увек и био случај.

Тестирање класификационих модела вршено је аутоматски функционалношћу коју обезбеђује *keras* радни оквир. Како су класе оба тест скупа савршено балансиране, претпоставка је да мера прецизности (енг. *accuracy*) може поуздано да се употреби за оцену квалитета трениране мреже. Тест скуп добијен је издвајањем 10% укупних података *random sampling* методом са *random_state* параметром 42. По аугментацији, остатак података подељен је на тренинг и валидациони скуп у односу 85:15.

На послетку, рад система као целине, евалуиран је ручно, на различитим тежинама игре, при различитом позадинском шуму.

6. РЕЗУЛТАТИ

У наредном сегменту биће изложени резултати евалуације агентског алгоритма и класификационих модела. Снимање и сегментација звука није евалуирана квантитативно, већ се своди на квалитативну евалуацију након ручно подешавање параметара, по претходно објашњеном резону, те не постоји прецизна метрика за евентуално поређење резултата.

6.1 Евалуација агентског алгоритма

Први део експеримента односи се на одређивање дубине обиласка стабла игре за минимакс. Како је циљ пронаћи максималну дубину, при којој апликација ради релативно брзо, прва тестирана дубина била је 8, која није дала одговарајуће резултате. Приликом тестирања дубина 6 и 5, апликација је показала задовољавајуће перформансе у фази отварања и завршнице у случају малог броја фигура, међутим време рачунања потеза у најзахтевнијој средишњој фази је у неким моментима узимало неприхватљиво много времена (више од један минут). Оптимални резултати како у погледу квалитета потеза тако и у

погледу перформансе остварени су за дубину 4, стога је она узета као параметар за тестирање против бота. Након овога уведена је модификација у алгоритму, да се у завршници, када остају само пиони на табли, динамички повећава дубина претраге стабла за 2, обзиром на прихватљиво мали број потеза које је потребно евалуирати.

Овим проширењем, алгоритам у завршници ради на максималној дубини од 6. После одиграних 10 партија са ботом на *chess.com* остварени су следећи резултати (Табела 6.1):

ЕЛО ранг бота	Број партија	исход
800	2	++
1000	4	+ - + +
1200	4	- - - +

Табела 6.1 – Приказ добијених резултата. Знак плус означава добијену, а икс изгубљену партију.

Иако кроз само 10 партија није могуће прецизно утврдити ЕЛО ранг агента, циљ овог рада представља имплементацију звуком управљаног шаха намењеног ентузијастима, а не професионалним играчима, те ЕЛО ранг од око 1120 добијен експериментом представља задовољавајући резултат.

6.2 Перформансе класификационих модела

Резултати за сваку појединачну класу предикције слова представљени у табели испод (Табела 6.2).

Класа	Прецизност	Одзив	Ф-мера
A	0.97	0.98	0.974974
B	0.83	0.86	0.844734
C	1.00	1.0	1
D	0.86	0.90	0.879545
E	0.95	0.92	0.934759
F	1.00	0.95	0.974359
G	0.95	0.93	0.939894
H	0.97	0.97	0.97

Табела 6.2 – Резултати модела за предикцију слова

Резултати за сваку појединачну класу предикције цифара представљени у табели испод (Табела 6.3).

Класа	Прецизност	Одзив	Ф-мера
1	1.0	1.0	1
2	0.99	1.0	0.994975
3	0.95	0.99	0.969588
4	0.97	0.98	0.974974
5	0.99	0.94	0.964352
6	1.0	1.0	1
7	0.98	0.99	0.984975
8	0.99	0.99	0.99

Табела 6.3 – Резултати модела за предикцију слова

Приликом тестирања добијена је глобална тачност од 94.2% и 98.8% за слова и цифре, респективно. Овакви резултати су и очекивани обзиром да су снимци изговорених бројева и дужи и различитији, па је и за очекивати да модел за бројеве ради ефикасније. Са друге стране, модел за слова показује нешто лошије, али у сваком случају задовољавајуће резултате. Највише проблема јавља се приликом класификације класа Б и Д, а обзиром на њихову узајамну сличност, овакво понашање је и за очекивати.

Приликом ручног тестирања обе мреже, примећено је да изнесени резултати одговарају и раду система у реалним условима, са тим да је субјективни осећај такав да се класа Д ипак за нијансу слабије препознаје у односу на класу Б. Иако је ово само субјективни осећај, претпоставка је да разлика од 3% није толико статистички значајна, те да је просто могла да се испољи као последица конкретног тест скупа података.

У односу на рад [11], где аутор остварује глобалну прецизност од око 98% за предикцију бројева овај рад остварује нешто бољи резултат. Приликом евалуације експерименталних резултата рада [9], аутори износе закључак да се перформанса класификатора за решавање сличног проблема значајно побољшава додавањем више густо повезаних слојева на излаз крајњег конволутивног слоја, те је та идеја имплементирана у модел изложен у овом раду, што није случај са [11], те ово може да буде један од разлога за разлику у перформансама. Други разлог може се наћи у разлици тренинг скупа, односно чињеници да рад [11] класификује десет класа, док модел у овом раду класификује осам класа. Такође, модел у овом раду користи

значајно већи број филтера по конволутивним слојевима, што потенцијално представља други разлог за остваривање нешто бољег резултата.

7. ДИСКУСИЈА

У наредном делу биће коментарисан рад и перформансе апликације у целини у односу на друге сличне, те њена употребљивост у реалном окружењу односно продукцији.

На почетку, важно је напоменути да слична апликација развијана за српско говорно подручје није пронађена, те ће поређење рада апликације као целине бити извршено у односу на апликацију на енглеском. Такође, већина релевантних апликација у продукцији данас користи *chess engine* (нпр. *Stockfish*) у позадини, а у најмању руку комплексан алгоритам често заснован на неуронском мрежама, па нема смисла поредити такво решење са простим минимаксом, те ће фокус дискусије бити управо на препознавању звука. Као место окупљања највећих шаховских ентузијаста, форум *chess.com-a* показао се као веома користан приликом тражења апликације за поређење. Наиме како не постоји конкретна метрика за избор најбоље апликације, узета је апликација *Karuah Chess*, означена од стране више корисника као „једина пристојна“ шаховска апликација са подршком за говор. Детаљи овог чланка налазе се на [21], док се *Github* репозиторијум *Karuah Chess* налази на [22].

Основна разлика рада *Karuah Chess* и решења представљеног у овом раду јесте у управљању фигурама. Наиме, креатори *Karuah Chess* одлучили су се за елиминацију највећег проблема (сличности изговорених слова) тако што је уведен концепт фонетског алфавета коришћеног у авио превозу, где се уместо А изговара алфа, уместо Б-бета, Ц- чарли (енг. *Charlie*) итд... Овим путем се моћ класификације мреже знатно увећава чиме се знатно смањује простор за грешку. Увођење овог система наводи на претпоставку да креатори *Karuah Chess* ипак нису успели да обезбеде непогрешив рад мреже за конвенционални начин употребе, те су прибегли оваквом решењу. Апликација *Chess H5*, најбоље рангирана говором управљана апликација која ради са командама аналогним са овим решењем на *Google play* продавници има убедљиво највише оцена 1, при чему се у коментарима корисници жале управо на лоше препознавање слова. Тим речено, може се закључити да апликација изложена у овом раду остварује резултате у складу са сличним апликацијама тренутно у продукцији и да не заостаје значајно.

У погледу остварених перформанси, отежавајућу околност углавном представља нешто спорији рад него што би то било пожељно, односно процесирање звука у интервалу од 1 а у неким

случајевима и до 4 секунде. Претпоставка је да је овакво понашање узроковано комплексним моделом класификатора.

У целини, апликација се може оценити као реално употребљива, обзиром да у већини времена ради довољно брзо, са задовољавајућом прецизношћу.

8. ЗАКЉУЧАК

У овом раду представљен је предлог архитектуре и имплементације говором управљаног шаха на српском језику. Основа мотивације за развој овакве апликације лежи у чињеници да овакав систем још није развијен за српско говорно подручје, уз то да ни апликације развијене за препознавање речи на енглеском језику нису бројне, док своју примену заиста могу да пронађу као разонода слабо покретних лица или ентузијаста којима је овим путем омогућено да играња шаха без потребе да се конкретно налазе испред уређаја. Проширењем апликације које би обухватило изговарање агентских потеза, апликација би добила примену и у виду подршке игрању слабовидим лицима или пак ентузијастима који желе да играју партије „на слепо“.

Систем се може поделити у два основна модула:

1. Модул за шах
2. Модул за рад са звуком

Први модул обухвата правила и логику игре као и имплементацију агента. Тестиран је ручно против бота на познатој *chess.com* платформи где је агент остварио приближно 1120 ЕЛО оцену.

Централни део другог модула представља класификатор улазног звука, заснован на конволутивној мрежи. Модел је одвојено трениран на скупу са бројевима и на скупу са словима где је остварена укупна тачност од 98.8% и 94% респективно. Евалуација оба модула такође је реализована и ручно, чиме су потврђена појединачна тестирања и рад система као целине. У поређењу са конкурентским решењима на енглеском језику, систем је оцењен као задовољавајући у погледу препознавања звука, али се агентски алгоритам не може поредити са COA (*енг.* State of the art) решењима која се ослањају на *chess engine*.

Иако је наведено да апликација остварује задовољавајуће перформансе то ипак може да зависи од конкретног корисника односно његових захтева. Са једне стране, агентски алгоритам не показује довољну моћ, те је идеја да се постојећа логика замени или на неки начин интегрише са неким COA *chess engine*-ом као што је на пример *StockFish* које је *open source*. Ако се у обзир узму кориснички коментари конкурентских решења на продавницама апликација, јасно се издваја неспремност великог броја корисника на било коју врсту

грешке приликом препознавања команди. Тим речено, са становишта *UX*-а, и изложено решење потенцијално може да одвраћа корисника, обзиром да апликација не може да гарантује савршено препознавање речи, те се у току партије може десити случај погрешног препознавања команде, што за неке кориснике може да буде фрустрирајуће и самим тим одбојно. Један од начина да се ублажи овај недостатак је свакако имплементација механизма за враћање потеза, којим би корисник могао да исправи грешку, иако то ипак не представља дугорочно и потпуно решење.

9. ЛИТЕРАТУРА

- [1] Lowerre, Bruce T, 1976. *The HARPY Speech Recognition System*. Carnegie-Mellon University.
- [2] Ganapathiraju, Aravind, "Support Vector Machines for Speech Recognition" (2002). *Theses and Dissertations*. 4155.
<https://scholarsjunction.msstate.edu/td/4155>
- [3] Aida-zade, K., Xocayev, A., & Rustamov, S. (2016). *Speech recognition using Support Vector Machines. 2016 IEEE 10th International Conference on Application of Information and Communication Technologies (AICT)*.
- [4] Ganapathiraju, A., Hamaker, J., & Picone, J. (2004). *Applications of Support Vector Machines to Speech Recognition. IEEE Transactions on Signal Processing*, 52(8), 2348–2355.
- [5] Chan, W., Jaitly, N., Le, Q., & Vinyals, O. (2016). *Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. 2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.
- [6] Yu, D., & Deng, L. (2015). *Automatic Speech Recognition. Signals and Communication Technology*. Springer.
- [7] Abdel-Hamid, O., Deng, L., Yu, D.: In: Exploring Convolutional Neural Network Structures and Optimization Techniques for Speech Recognition, pp. 3366–3370 (2013).
- [8] Abdel-Hamid, O., Mohamed, A.r., Jiang, H., Deng, L., Penn, G., Yu, D.: Convolutional neural networks for speech recognition. *IEEE Trans. Audio, Speech Lang. Process.* (2014).
- [9] Deng, L., Abdel-Hamid, O., Yu, D.: A deep convolutional neural network using heterogeneous pooling for trading acoustic invariance with phonetic confusion. In: *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6669–6673 (2013).
- [10] Abdel-Hamid, O., Mohamed, A., Jiang, H., Deng, L., Penn, G., & Yu, D. (2014). *Convolutional Neural Networks for Speech Recognition. IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 22(10), 1533–1545.

- [11] Sharan, R. V. (2020). *Spoken Digit Recognition Using Wavelet Scalogram and Convolutional Neural Networks. 2020 IEEE Recent Advances in Intelligent Computational Systems (RAICS)*.
- [12] *Speech recognition* библиотека
<https://pypi.org/project/SpeechRecognition/> [датум приступа 1.9.2022.]
- [13] *PyDub* библиотека <https://pypi.org/project/pydub/> [1.9.2022.]
- [14] *Librosa* библиотека <https://librosa.org> [1.9.2022.]
- [15] *Matplotlib* библиотека <https://matplotlib.org/> [1.9.2022.]
- [16] *OpenCV* библиотека <https://opencv.org/> [1.9.2022.]
- [17] *Scikit-learn* библиотека <https://scikit-learn.org> [1.9.2022.]
- [18] *Keras* библиотека <https://keras.io/> [1.9.2022.]
- [19] *Tensorflow* библиотека <https://www.tensorflow.org> [1.9.2022.]
- [20] Mark E. Glickman, 1999. *The Glicko System*, Boston University.
- [21] Блог чланак, <https://www.chess.com/blog/Sawbonez/finally-a-decent-voice-controlled-chess-app>
- [22] *Github* репозиторијум *karuahChess*,
<https://github.com/karuahsoftware/karuahchess>

10.БИОГРАФИЈА

Лука Курељушић рођен је 22.септембра 1999. године у Суботици, Република Србија. У Суботици похађа основну школу „Јован Микић“ као природно-математичко одељење гимназије „Светозар Марковић“, коју завршава као носилац дипломе „Вук Караџић“. Од 2020. године носилац „Стипендије за изузетно надарене ученике и студенте“ министарства просвете и науке и технолошког развоја.