Универзитет у Београду

Машински и Математички факултет

Мастер академске студије - Индустрија 4.0

Предмет: Машинско учење

Број регистра: 4003/2020

Стефан М. Ковач

Примена машинског учења за препознавање знаковног језика

МАСТЕР РАД

Београд, 21.04.2025.

Комисија за преглед и одбрану

1. Проф. др Младен Николић
2. Проф. др Живана Јаковљевић
3. Проф. др Јована Ковачевић

Датум одбране: _____

Оцена: _____

Овера студентског одсека:.....

Предметни наставник

Проф. др Младен Николић

За руководиоца студијског програма

Проф. др Живана Јаковљевић

Продекан за наставу МФ

Проф. др Живана Јаковљевић

Задатак мастер рада

- Увод у област теме мастер рада.

С обзиром на то да знаковни језици представљају визуелно-просторну природу језика, они су већ дуже време предмет интересовања и истраживања у области вештачке интелигенције и комуникације између човека и машине. Динамична и вишеструка невербална комуникација између човека и машине представља комплексан изазов за информатичку заједницу, чије решавање често доводи до иновативних и значајних решења у овој области.

- Шта се тражи да кандидат конкретно уради.

Кандидат треба да изучи и објасни теоретску и практичну страну три модерне архитектуре вештачких неуронских мрежа које се примењују у области рачунарског вида. Кандидат треба да јасно и прецизно објасни целокупан процес прикупљања и обраде података, њихове примене у тренирању модела, као и подешавање хиперапараметара у циљу одабира оптималних модела. Од студента се очекује да користи значајне референтне академске радове за сваки од модела који ће бити јасно и прецизно наведени.

- Смернице шта би рад требало да садржи.

Од кандидата се очекује да објасни мотивацију за коришћење изабраних модела, као и због чега је њихова примена у области знаковних језика значајна за даљи развој и имплементацију вештачке интелигенције. У представљању модела, кандидат ће јасно и прецизно објаснити теоретски оквир и разлоге за избор конкретних метода у процесу тренирања и евалуирања сваког модела. Кандидат би требало да укратко прикаже постигнуте резултате, изазове са којима се суочио током имплементације, као и начине на које је те изазове превазилазио. Рад треба да садржи уводно поглавље, опис примењених метода обраде и претпроцесирања података, приказ теоријских аспеката модела, поглавље са резултатима, закључак и библиографију.

Београд, 21.04.2025.

Предметни наставник

Проф. др Младен Николић

Захвалност

Осећам потребу да се овом приликом захвалим професору Младену Николићу на изузетно квалитетним предавањима из предмета „Машинско учење“, као и на подршци и помоћи током израде овог рада.

Такође се захваљујем родитељима Весни и Миловану на стрпљењу и разумевању, као и ујаку Саши и остатку породице.

Садржај

1 Увод

1.1 О знаковним језицима

Знаковни језик је визуелни језик који се изражава манифестацијом физичких покрета, уместо кроз изговорене речи или гласове као што је случај у обичном говорном језику, а то значи да он зависи од знакова руку, очију, израза лица и покрета тела како би преносио поруке. Као и било који говорни језик, знаковни језик има властита правила граматике и структуре реченице. Баш као и код говорних језика не постоји један универзални знаковни језик, већ различите земље углавном користе властите верзије знаковног језика који је јединствен за регион и културу из којих је проистекао. Тако, на пример, амерички знаковни језик - *ASL* (енгл. *American Sign Language*) је различит од аустралијског знаковног језика – *AUSLAN* (енгл. *Australian Sign Language*), који је опет различит од британског знаковног језика – *BSL* (енгл. *British Sign Language*), без обзира што сва три долазе из енглеског говорног подручја.

Слика 1.1 - Др. Петерсон вежба знаковни језик са Коко

Иако је знаковни језик превасходно користан и ефектан алат који олакшава комуникацију људима са оштећеним слухом, интересантно је да се он може користити и за комуникацију са горилама. Професор лингвистике доктор Пени Петерсон (енгл. Dr. *Penny Peterson*) је од средине 1970-их до краја 1990-их година радила на експерименту са горилом Коко учећи је знаковни језик и на коме је Коко током живота научила преко хиљаду *ASL* знакова. Иако постоје контроверзе везане за ово истраживање, оно је, без сумње, показало

да гориле имају способности лингвистичког разумевања на нивоу мале деце, као и то, да имају одређену способност да и саме комуницирају знаковним језиком ако се истренирају да га науче. Осим тога истраживање је показало и то да гориле вероватно поседују одређени ниво емоција, осећаје, ниво смисла за хумор, али и способност перцепције једноставнијих форми апстракције.

Слика 1.2 - Гориле демонстрирају знакове

Подаци Светске федерације глувих (енгл. *World Federation of the Deaf*) указују да у свету тренутно живи 70 милиона особа са оштећеним слухом који се користе са једним од три стотине постојећих знаковних језика [1]. Уједињене нације процењују да око 5% светске популације, односно неких 430 милиона људи имају оштећење слуха које захтева медицинску рехабилитацију, а да ће до 2050. године овај број нарасти на 700 милиона људи. Око 80% људи са оштећеним слухом живе у слабо развијеним земљама, тако да побољшање могућности комуникације овим људима има знатну глобалну етичку, социјалну и економску важност. Оштећења слуха често имају огроман негативан утицај на животе људи на индивидуалном нивоу. Чести негативни аспекти који утичу на животе људи са оштећеним слухом су тешкоће у свакодневној комуникацији, друштвена изолованост, усамљеност, а у одређеним друштвима и стигматизација од заједнице у којој живе. У слабије развијеним земљама деца са оштећеним слухом често не добијају формално образовање због недостатка адекватног просветног кадра. У њиховим одраслим годинама ово обично значи да су стопе незапослености особа са оштећеним слухом у неразвијеним земљама систематски веће него је то случај у развијеним земљама. Светска здравствена организација процењује да глобални трошак који се на овај начин ствара, односно који је асоциран за људе са оштећеним слухом, износи 980 милијарди америчких долара годишње.

Упркос уобичајеној перцепцији знаковни језици нису вештачки језици на начин као што су то „конструисани језици“ (на пример есперанто, клингон итд.) или формални језици (нпр. програмски језици *Python*, *Java*, итд.), већ су настали природно и постепено од стране група људи који су их користили за комуникацију у свакодневном животу. Развој и еволуција модерних знаковних језика траје преко 300 година, а њихов почетак је настао спонтано у заједницама у којима је било знатно присуство глувонемих. Неки од познатијих језика из којих су еволуирали модерни знаковни језици су стари кентски знаковни језик (енгл. *Old Kentish Sign Language*) који је настао крајем 16. века у Кенту у Великој Британији, а који је један од темеља модерног британског знаковног језика - *BSL* (енгл. *British Sign Language*) и стари француски знаковни језик - *VLSF* (фр. *Vieille langue des signes française*) из 18. века. Поготово је интересантан случај знаковног језика са острва Мартини виногради *MVSL* (енгл. *Martha's Vineyard Sign Language*) који је почео да се користи крајем 17. века. *MVSL* је настао спонтано на острву државе Масачусетс као последица рецесивног гена у популацији острва због којег се рађао велики број глувих особа готово у свим породицама острва, што је изнедрило потребу за знаковним језиком. У смислу развоја модерних знаковних језика, поготово је битна улога *VLSF* који је делимично први извршио систематизацију и кодификацију језика, а то је почетком 19. века имало директан, велики утицај на развој *ASL* који јесте у правом смислу први потпуно кодификован модерни знаковни језик.

Треба приметити и то да, како у практичном тако и у правном смислу, знаковни језици су легално матерњи језик за људе са оштећеним слухом који се активно служе овим језицима, а не говорни језици породица из којих ови људи потичу. На пример, у Америци има преко пола милиона људи за које је *ASL*, а не енглески, матерњи језик. У овом контексту битно је појаснити да упркос уобичајеној перцепцији да су знаковни језици конструисани тако да су продужеци, односно репрезентације говорног језика из којег су поникли или да су својеврсна имитација свакодневног говорног језика, они то нису. *ASL* на пример има врло мало заједничког са говорним или писаним енглеским језиком, јер знаци *ASL* језика нису репрезентација речи у енглеском језику. Изненађујуће је да *ASL* има веће сличности са језиком Навахо Индијанаца или јапанским него са енглеским језиком, што је свакако резултат случајности, а не самог дизајна *ASL*-а. Иако је горе већ поменуто да су знаковни језици енглеског говорног подручја (*ASL*, *BSL* и *AUSLAN)* различити језици, није наведено да особа која добро познаје *ASL* неће разумети особу која комуницира на *BSL* на исти начин и истом мером, као што неће разумети ни знакове из руског знаковног језика – *RSL* (енгл. *Russian Sign Language*). Штавише, корисник *ASL* ће успешније комуницирати са познаваоцем француског знаковног језика - *LSF* (fr. *langue des signes française*), него са

особом која се користи *BSL*-ом, због тога што је стари француски знаковни језик корен *ASL*-а, док је *BSL* настао аутохтоно. Разлог за неразумевање између корисника *ASL*-а и *BSL*-а је то што су оба настали независно од енглеског језика, те као такви имају јединствену структуру знакова за изражавање речи и идеја без обзира што су настали у енглеском говорном подручју. Корен основних карактеристика које одређују значење знака у *ASL*-у је положај руке, оријентација палца, израз лица, положај тела итд. а не синтакса или синтагма енглеског језика. Штавише *ASL* (као и *BSL*) не користи ни граматику енглеског језика већ поседује сосптвену јединствену граматику која је различита од оне у енглеском језику.

У смислу лингвистичке структуре, знаковни језици су дакле различити од говорних језика и рефлектују визуелну и просторну природу језика. У овом контексту интересантно је указати на неке од лингвистичких разлика између енглеског и *ASL* да би се нагласила чињеница да се *de facto* ради о различитим језицима:

a) **Редослед речи**: „Ја једем пицу“ се на енглеском пише „*I eat pizza*“, док ће на *ASL* ово бити „*Pizza I Eat”*.

b) **Временски облик**: Док на српском и енглеском време прошле или будуће радње може бити неодређено, у *ASL* је то увек прецизно дефинисано. На пример реченица на српском „Јео сам“ се преводи на енглески као „*I ate*”, док на *ASL* то једино може бити „*Yesterday I ate*” или „*Two days ago I ate*“ итд.

c) **Изостављање чланова енглеског језика „*a*“, „*an*“, „*the*“:** „Идем до продавнице“ се на енглески пише „*I am going to the store*”, док ће на *ASL* то бити „*Store I go*”.

Чак и када се енглески језик директно употребљава у *ASL* он се искључиво користи да се шаком и длаком „спелује“ слово по слово енглеске речи. Иронично је да знакови којим се „спелују“ слова енглеске абецеде нису формалан део америчког знаковног језика (*ASL)*.

1.2 Машинско учење и знаковни језици

У овом раду коришћена је база података из руског знаковног језика (*RSL*) чији је корен као и код *ASL*-а стари француски знаковни језик и који се у одређеним формама у Русији почео користити од почетка 19-ог века. Треба приметити да су, упркос истом корену, *RSL* и *ASL* су у потпуности различити језици, тако да познаваоци једног језика не разумеју други језик. У односу на различитости знаковних језика унутар енглеског говорног подручја, где смо видели како су *ASL* и *BSL* потпуно различити језици, код *RSL* постоји далеко већа стандардизација, мада и унутар њега постоје регионалне разлике, тако да у

одређеним случајевима знакови за исти појам су различити у зависности од регије Русије из које извођач долази.

Као што је већ наведено, знаковни језик је визуелни језик, те је због тога већ неколико деценија предмет интереса у домену вештачке интелигенције. Рани покушаји коришћења вештачке интелигенције за распознавање знаковних језика су почели у 1980-им и 1990-им годинама фокусирајући се на препознавање знакова користећи основне алгоритме за препознавање образаца, као и тадашње технике обраде слика. Ови покушаји су углавном били безуспешни због ограничења у снази процесора тадашњих рачунара, ограничене величине узорака, што је резултирало у безначајној тачности препознавања знакова. Почетком 1990-их година почеле су се користити унапређене методе вештачке интелигенције, као што је скривени Марков модел (енгл. *hidden Markov model*) [2], које су коришћене за препознавање појединачних знакова, често тренирајући на малој бази података. Међутим успешност ових метода је и даље остала ограничена због тога што је било тешко ухватити репрезентацију динамичке природе знаковног језика. Први знатнији успех се десио тек са појавом дубоког учења (енгл. *deep learning)* након 2010. године, поготово са применом конволутивних неуронских мрежа – *CNN* (енгл. *convolutional neural networks*), рекурентних неуронских мрежа – *RNN* (енгл. *recurrent neural networks*), али и са доступношћу бољих база података првобитно базираним на *ASL*, а затим и на другим знаковним језицима. До 2016. године се појављују архитектуре као што су *VGG* и *ResNet*, а у последњих неколико година и трансформер модели који много боље препознавају гестикулације с обзиром да се успешније носе са просторно-временским димензијама извођења гестикулације.

Иако алтруистички аспект, због чега машинско учење посвећује доста пажње препознавању знаковног језика, није занемарљив, постоје и други дубоки разлози због чега је ова област интересантан предмет истраживања у машинском учењу. Знаковни језик је форма невербалне комуникације која укључује деликатне покрете руку, израз лица, положај и покрет тела, а што је у односу на вербалну комуникацију другачији и високо комплексан систем. За програмере ово представља јединствен изазов у смислу разумевања и „хватања“ динамичне вишеструке комуникације између људи, при томе користећи машине као посреднике у комуникацији. Дакле, програмерска заједница није искључиво посвећена креирању система препознавања знаковног језика у циљу помоћи људима са оштећеним слухом, већ она креирајући ове системе решава комплексне и широке изазове у области вештачке интелигенције, као и интеракције између човека и рачунара.

На пример, развој модела који са високом тачношћу могу интерпретирати знаковни језик у реалном времену је технолошки изазов који захтева ефикасне алгоритме као и сензорски вођен улаз информације уз минимално кашњење. Затим препознавање знаковног језика захтева решења како просторне тако и временске димензије, што је, из перспективе машинског учења, такође знатан изазов. Овакви типови изазова често воде ка технолошким иновацијама или проналаску нових решења у областима машинског учења и рачунарског вида које знатно унапређују целокупну област вештачке интелигенције. Осим тога многи модели машинског учења који се користе за препознавање знаковног језика, могу релативно лако бити модификовани за коришћење у другим применама, које се базирају на препознавању форми гестикулација, као што су интеракција између човека и рачунара (енгл. *human-computer interaction - HCI*), роботика итд. Дакле, решења пронађена у домену препознавања знаковног језика могу да имају широку употребу која далеко превазилази првобитну намену и која знатно може утицати на многе друге индустрије као што су роботика, аугментована реалност, здравство итд.

У овом раду намера је била представити три важне архитектуре у области рачунарског вида које су настале након првих примена техника дубоког учења, хронолошки пратећи њихову појаву: *VGG*, *ResNet* и *TimeSformer* архитектуре. Највећа пажња је посвећена варијантама и еволуцији трансформер модела, где је посебан труд уложен на детаљна објашњења, као и на математичко представљање.

2 Подаци

2.1 Опис података

Прикупљање података из домена знаковних језика у циљу њихове употребе у интерпретационим методама машинског учења је комплексан и тежак процес. Осим саме чињенице да релативно мали број људи са нормалним слухом уме да комуницира знаковним језицима на било којем нивоу, број оних који имају компетентност да са довољном прецизношћу демонстрирају знакове за њихову употребу у машинском учењу је још мањи. Ово ствара знатна ограничења и потешкоће приликом процеса креирања довољно добре знаковне базе података, од чијег квалитета и величине зависи успешност примене машинског учења. Осим тешкоћа везаних за формирање довољно велике базе података, проблем представља и чињеница да успешност интерпретације знаковног језика методама машинског учења зависи и од широке присутности разноразних варијација унутар узорка података. На пример, осим што постоји потреба за већим бројем демонстратора знакова, потребне су и варијације осветљења, боја, позадина, угла снимања итд. Додатна отежавајућа чињеница је и то што не постоји универзални знаковни језик за људе са оштећеним слухом, већ свака лингвистичка група често употребљава уникатне знакове за демонстрацију истог појма. Ово значи да аугментација (проширивање) узорка унутар једне лигнвистичке групе, придодавањем јој узорка из друге, није могућа, јер начин извођења симбола који представљају исти појам може бити различит. Ово свакако указује на то да један од основних проблема приликом креирања видео колекције знаковног језика (енгл. *SLR - sign language recognition*) за потребе машинског учења, су тешкоће у проналажењу довољно великог броја извођача знакова који поседују довољну компетентност и прецизност приликом извођења гестикулација, чак и у случајевима језика којим се као матерњим служе десетине милиона људи.

Модели машинског учења у овом раду, за препознавање гестикулација, користе податке из видео колекције „*SLOVO – Russian Sign Language*“, преузете из академског рада A. Kapitanova [3]. У циљу повећања броја извођача знакова, као и побољшања квалитета узорка, аутори су користили „crowdsourcing“ платформе („*Yandex toloka*” и „*ABC Elementary4*”). Свака појединачна демонстрација знака је подвргнута ригорозном верификационом тесту, који се састојао од тога да су коректност и прецизност извођења морали бити потврђени концензусом од стране троје сертификованих верификатора, независно један од другога. Прецизност извођења знакова је додатно осигурана тиме што је

захтевано да извођачи поседују одређени ниво компетентности и познавања стандарда знаковног језика, где се од њих захтевало да остваре резултат од најмање 80% на формално верификованом тесту служења руским знаковним језиком. Током верификационог процеса препознати су нерелевантни знакови, који су затим одстрањени из коначног узорка.

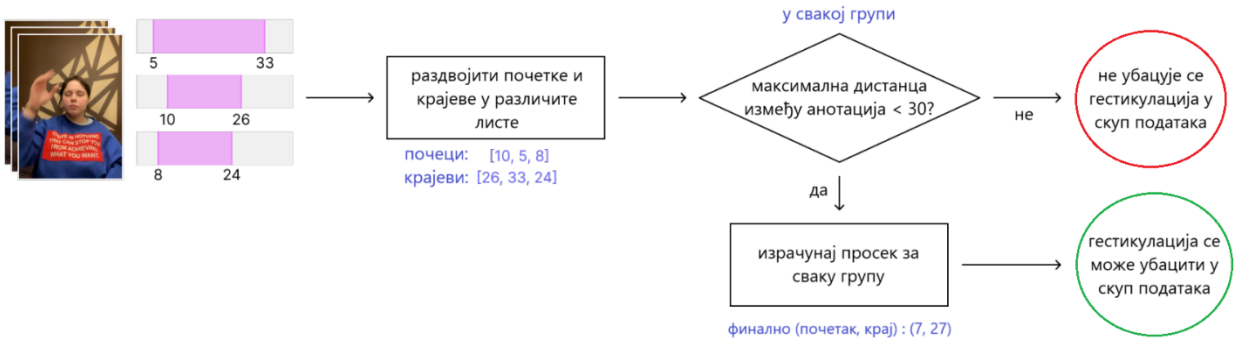
Извођачи знакова су видео клипове снимали најчешће у кућном амбијенту, испред лаптопа или камере паметног телефона, а била је потребна и техничка обрада клипова у смислу да се у свим релевантним видео клиповима означи почетак и крај извођења сваке појединачне гестикулације. Приликом једног снимања извођач би снимио, на пример, неких двадесетак гестикулација, што подразумева да је након снимања видео клип требало разделити на појединачне знакове. При томе се водило рачуна да делови видео клипова у којима извођач пали и гаси камеру буду исечени, као и делови између две гестикулације где се извођач припрема да изврши наредну гестикулацију. Дужине исечених клипова појединачних гестикулација варирају од 1 до 4 секунде. Овако је добијен обрађени узорак који садржи 20400 видео клипова, од 1000 различитих класа знакова, са по двадесет различитих клипова из сваке класе знакова. Преосталих 400 клипова, који се састоји од остатака исечених гестикулација, придружени су класи „no_event“, што значи да ови клипови не садрже извођења гестикулација знаковног језика.

Процес прикупљања и обраде података који су аутори применили се може сумирати констатацијом да се ради о уобичајеном стандарду приликом примене машинског учења на интерпретацију знаковног језика, што је подразумевало израду платформе која се састојала од три основна корака:

- **Прикупљање видео клипова**: Код одабира речи, обраћена је пажња да се изаберу речи које се фреквентно користе у свакодневном животу и које се односе на појмове као што су храна, животиње, емоције, боје, бројеви итд.
- **Валидација видео клипова**: С обзиром да интерпретација знаковног језика може изазивати недоумице приликом тумачења, аутори су се одлучили на горе објашњени верификациони процес (сагласност неколико познавалаца знаковног језика о исправно изведеној гестикулацији која се, тек потом, укључује у узорак).
- **Анотација временских интервала гестикулација**: Слика 2.1 додатно објашњава анотацију. Наиме, неколико анотатора би обележило почетак и крај гестикулација и уколико би се њихови крајеви (или почеци) међусобно знатно разликовали (за 30 слика, односно за једну секунду) гестикулација се не би додавала у видео колекцију. Иначе, гестикулација се додаје и за њен почетак и крај се узима просек свих

анотатора. Након сечења гестикулација из видеа, одстрањени делови су убачени у „no_event“ категорију и они су коришћени за тренирање модела у циљу препознавања видео снимка у коме се не извршава ниједна од 1000 гестикулација.

Потребно је нагласити и чињеницу да је ова база података као и њихова обрада добила оцену 10 на сајту „kaggle.com“ у погледу њеног квалитета, што је био мотив за њено коришћење у овом раду.

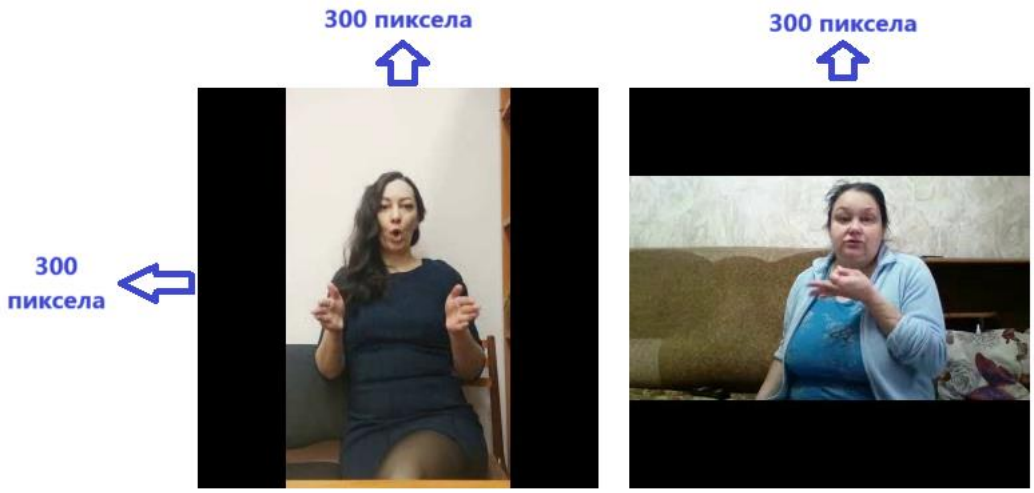


Слика 2.1 - Процес сечења гестикулација из видеа

2.2 Претпроцесирање података

Након преузимања података, био је потребан њихов даљи третман за примену у овом раду. Тренинг и тест скуп су већ били подељени у односу 15 : 5, а пошто је био потребан и валидациони скуп, он је креиран тако што су узета по два видео клипа од сваке класе из тренинг скупа и по један видео клип од сваке класе из тест скупа. Ово значи да је однос скупова тренинг : валидација : тест постао 13 : 3 : 4, односно 65% : 15% : 20%. Скуп података је балансиран по класама гестикулација, што олакшава тренирање модела.

Узимајући у обзир рачунарске ресурсе, било је потребно смањити резолуције видео клипова, да би модели брже обрађивали податке. С обиром да су видео клипови снимљени у хоризонталној или вертикалној перспективи, већа димензија је смањена на 300 пиксела, а на мањој димензији су додати црни пиксели (енгл. *black padding*), тако да свака слика видео клипа буде сачињен од 300x300 пиксела (Слика 2.2).



Слика 2.2 – Додавање црних пиксела

У приступу обраде података акценат је, што је више било могуће, стављен на примену принципа једноставности. У практичном смислу ово значи да је из сваког видео клипа узет исти број слика[1], шеснаест по видео клипу, које су расподељене на еквидистантним позицијама, то јест, оне су униформно расподељене кроз видео. У бази података „SLOVO” минимална дужина видео клипа је седамнаест слика, тако да је из сваког било могуће извући шеснаест слика, што је и иначе често примењивана пракса. Овим приступом је омогућено да су одабране слике из свих временских позиција, а не на пример само средишњи делови клипа и на тај начин је покривена цела временска димензија гестикулације. Од самог почетка неки од могућих недостатака ове методе су били очити, као на пример одсуство детекције краћих, финијих покрета код дужих гестикулација, али који су се могли превазићи одабиром већег броја слика. Одабир фиксног броја слика из сваког видео клипа је био мотивисан потребом како би се олакшало и убрзало тренирање. Величина подскупа за стохастички градијентни спуст (енгл. *batch*)[2] је била условљена капацитетом графичке картице тако да је подешена да буде што већа, како би меморија графичке картице била што боље искоришћена. Захтеви процеса тренирања за меморијом графичке картице се не мењају кроз време, пошто је улазна величина података фиксна, што помаже бржем тренирању. У оквиру једне класе, видео клипови могу имати различите дужине, али ако се једни посматрају као бржа, односно други као спорија извођења истог знака, овим униформним одабирањем слика теоријски се екстрахује иста информација, што је додатни разлог зашто је изабран овај метод. У случају када су коришћене дужине видеа

[1] У енглеском језику се за једну од слика које сачињавају видео клип користи реч „фрејм“ (енгл. *frame*)
[2] За појам „подскуп за стохастички градијентни спуст“ (енгл. *batch*), у остатку рада ће се користити само „(тренинг) подскуп“ као термин

од 24 слике, било је потребно додати црне слике (енгл. *black padding*) на почетак и крај краћих видеа како би оне попуниле места непостојећих слика. Ради потпунијег искоришћења ресурса графичке картице, пожељно је да видео клипови буду исте дужине због тренирања у подскуповима.

С обзиром да у тензор формату подаци заузимају пуно меморије, неке делове претпроцесирања је било потребно извршити у „ходу“, паралелно са тренирањем модела. Рађено је сечење слика на мање димензије од 224x224 пиксела. Ово је прихватљиво јер су гестикулације обично вршене у пределу средине слике. У оквиру нормализације података, пиксели су прво скалирани у интервал [0, 1]. Пошто су коришћени претренирани параметри конволутивног дела мреже модела који је истрениран на популарној „ImageNet“ бази података, а који је користио додатну стандардизацију података користећи средњу вредност и стандардну девијацију пиксела у тој бази података, исту методу је требало применити и у овом раду како би претренирани параметри били искоришћени у потпуности. На циљну променљиву, тј. припадност класи видео клипа, урађена је трансформација бинарног кодирања (енгл. *dummy coding*), такође у „ходу“, тако да се добије вектор дужине 1001 (1000 класа гестикулација + 1 класа неприпадности клипа ни једној гестикулацији).

3 Увод у машинско учење

Машинско учење је подобласт вештачке интелигенције која се бави индуктивним закључивањем на основу тога што користи мноштво података како би се пронашли извесни обрасци и законитости у њима. Користи се у сврхе предвиђања и одлучивања у оквиру разних рачунарских система. Дакле, уместо да се експлицитно пише алгоритам који на пример треба да препознаје лица људи са фотографија, што би у пракси био нереалан подухват, мета-алгоритам[3] машинског учења се служи великом количином фотографија која садрже лица људи. На тај начин се добија модел који на улазу прима фотографије, а на излаз прослеђује локацију лица човека у оквиру фотографије, на основу тога што је модел научио из података како да препозна лице. Машинско учење се користи и у друге сврхе, као што су аутономна вожња аутомобила, играње игара, предвиђање кретања цена на берзама, превођење са једног природног језика на други итд.

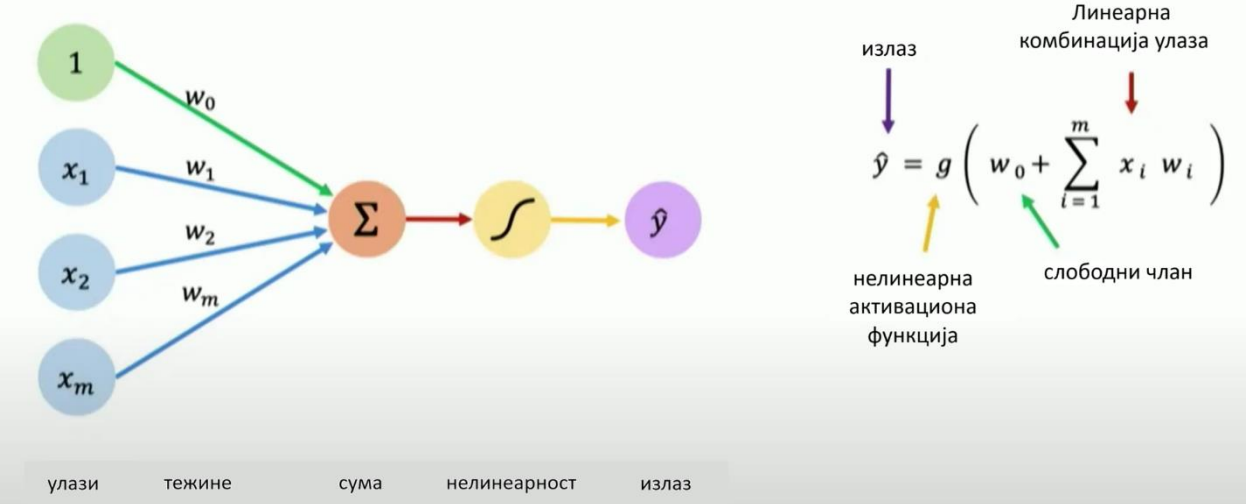
У машинском учењу се претпоставља да постоји добар модел који прати реалне законитости и обрасце у подацима и који ће самим тим моћи да се индуктивно примењује на нове податке. У машинском учењу се изабере нека фамилија (скуп) параметарских модела са претпоставком да јој жељени модел припада, а он се може пронаћи и издвојити из ње оптимизационим методама, то јест минимизацијом функције грешке над подацима које користи. При тренирању модела, доступни подаци се поделе на тренинг и тест подскуп. На тренинг подскупу података модел учи законитости и образце, а тест подскуп служи као потврда да модел врши добру генерализицију над подацима које није „видео“. Неретко се од тренинг подскупа додатно креира и валидациони подскуп који има сличну сврху као и тест подскуп, с тим што служи за одабир добрих хипер-параметара модела. У случају када је фамилија модела пребогата, односно када су модели из фамилије прекомплексни за дати проблем, може доћи до такозваног преприлагођавања модела подацима (енгл. *overfitting*). То значи да је модел са тренинг скупа изузетно научио податке из тренинг скупа, међутим да му је лоша генерализација на тест скупу. У овом случају се може прибећи регуларизацији функције грешке и у том случају модел ће имати велику грешку у једном пределу фамилије модела, па ће оптимизациони проблем минимизације функције грешке морати тражити оптимум у другом делу фамилије, то јест суженој фамилији у односу на првобитну фамилију, пре регуларизације. Циљ регуларизације је да се сужавањем фамилије смањи

[3] Мета-алгоритмом се креира неки други, жељени, алгоритам.

њена комплексност и да се у њој пронађе једноставнији модел који ће боље генерализовати решавање проблема.

3.1 Потпуно повезане неуронске мреже

Потпуно повезана неуронска мрежа (енгл. *fully connected neural network*) је сачињена од неурона који се називају перцептрони и они су међусобно повезани кроз слојеве. Перцептрон прима више улаза, од њих прави линеарну комбинацију помоћу параметара $\vec{w}$ и њу пропушта кроз такозвану нелинеарну активациону функцију (слика 3.1) која омогућава учење нелинеарних законитости и образаца међу подацима.



Слика 3.1 - Перцептрон

Код потпуно повезане неуронске мреже улазни сигнал, односно квантификовани подаци, се прослеђује унапред кроз неуроне и добија се излаз. Назива се потпуно повезана мрежа, јер је сваки неурон повезан са свим излазима из претходног слоја (Слика 3.5). Излаз из мреже се пореди са стварним вредностима података и рачуна се грешка односно одударање резултата од стварних података. Код неуронских мрежа, односно дубоког учења, оптимизациони методи се заснивају на концепту градијентног спуста. Како је вештачка неуронска мрежа композиција нелинеарних параметризованих функција, рачуна се градијент функције грешке по параметрима, а затим се параметри ажурирају померањем њихове вредности у смеру негативног градијента, пошто је то смер који из тренутне вредности параметара доноси највеће смањење грешке. Математички записано, задатак је минимизациони проблем функције грешке по параметрима модела:

$$\min_{\vec{w}} L(F(x; \vec{w}), y) \tag{3.1}$$

Где:

- L је функција грешке која мери одступање стварних података од оних које предвиђа модел
- $F(x; \vec{w})$ или $\hat{y}$ је резултат модела за дати улаз x
- y је стварни податак на излазу

Минимизација у оквиру градијетног спуста се одвија тако што се у итеративном процесу ажурирају параметри w. У сваком кораку се израчуна градијент функције грешке по w:

$$\overrightarrow{w_{new}} = \overrightarrow{w_{old}} - \alpha * (\frac{\partial L}{\partial w_1}, \frac{\partial L}{\partial w_2}, \ldots, \frac{\partial L}{\partial w_n}) \tag{3.2}$$

Где:

- α је корак учења којим се скалира градијент и тако подешава брзина учења
- n је број параметара

Теорема о универзалној апроксимацији [4] (енгл. *Universal Approximation Theorem*) каже да је потпуно повезана мрежа са једним скривеним слојем универзални апроксиматор, односно да се она може истренирати да произвољно близу апроксимира податке, ако јој се да довољан број неурона.

3.2 Конволутивне неуронске мреже

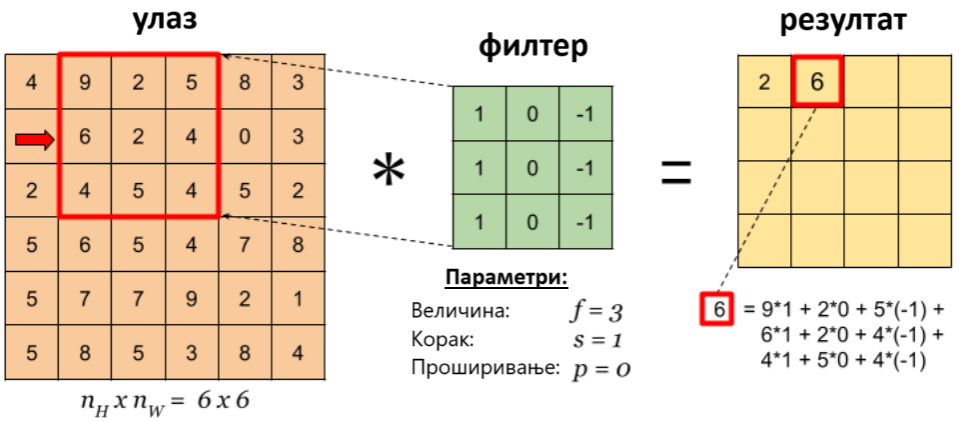
Конволутивна неуронска мрежа (енгл. *CNN – convolutional neural network*) је тип вештачких неуронских мрежа дизајнирана за обраду визуелних података, као што су слике и видео клипови. Принцип рада се своди на разлагање слика на мање делове и успутно анализирање сваког дела слике користећи једноставне неуронске јединице (неуроне) зване перцептрони. Перцептрон има подесиве параметре зване „тежине и слободни члан“ (енгл. *weights and bias*), који омугућавају мрежи да учи, односно да се прилагоди подацима. Како *CNN* обрађује различите делове слике, она учи да идентификује специфичне обрасце или атрибуте као што су ивице, текстуре или облици. Једна од предности *CNN* је то што ефикасно користи чињеницу да су суседни пиксели у слици у блиској вези, што се назива локална просторна кохеренција, због чега нема потребе да *CNN* има пуно различитих параметара да би учила обрасце о сваком појединачном пикселу, већ може да дели једне

исте параметре које користи да анализира целу слику. Ово смањује захтеве за меморијом која је потребна мрежи за цео просец учења [5]. Главни делови који чине класичну конволутивну мрежу су следећи:

а) **Конволутивни слој**: Конволутивни слој се састоји од параметризованих матрица званих кернели или филтери, које се користе за креирање мапа атрибута од примљеног сигнала које се пропуштају у следећи слој. Конволуција се извршава тако што филтер „клизи“ преко улазне слике и при томе рачуна производ елемената матрице филтера и подматрице слике, елемент по елемент, а затим се сви производи сумирају, слично скаларном производу вектора, и тако у сваком пределу слике (Слика 3.2). Ако је f матрица филтера димензије $N \times M$ чије су вредности у пољима параметри и g матрица која одговара делу сигнала неке слике истих димензија као f, онда се конволуција у том пределу врши по формули:

$$f \cdot g = \sum_{i=1}^{N} \sum_{j=1}^{M} f_{ij} * g_{ij} \qquad (3.3)$$

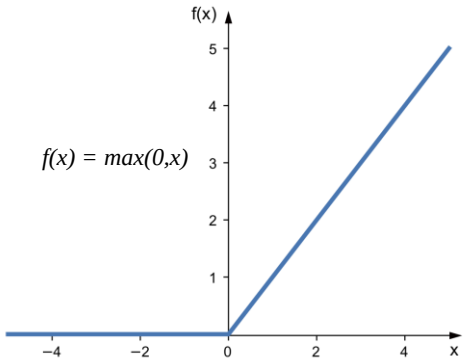
Дакле, нова мапа атрибута улазне слике се израчунава тако што филтер пређе преко свих делова улазне слике (Слика 3.2).



Слика 3.2 - Операција конволуције

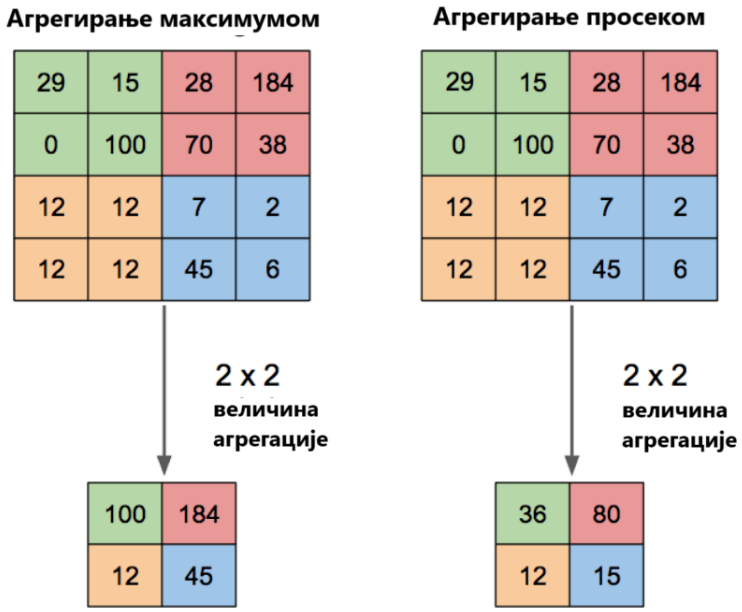
б) **Нелинеарна активациона функција:** Активациона функција се користи на крају конволутивног слоја и она је нелинарна трансформација коју примењујемо на улазни сигнал, након операције конволуције. У овом случају обично користимо такозвану

ReLU активациону функцију која враћа улазну вредност уколико је она већа од нуле, док у супротном враћа вредност нула (Слика 3.3).



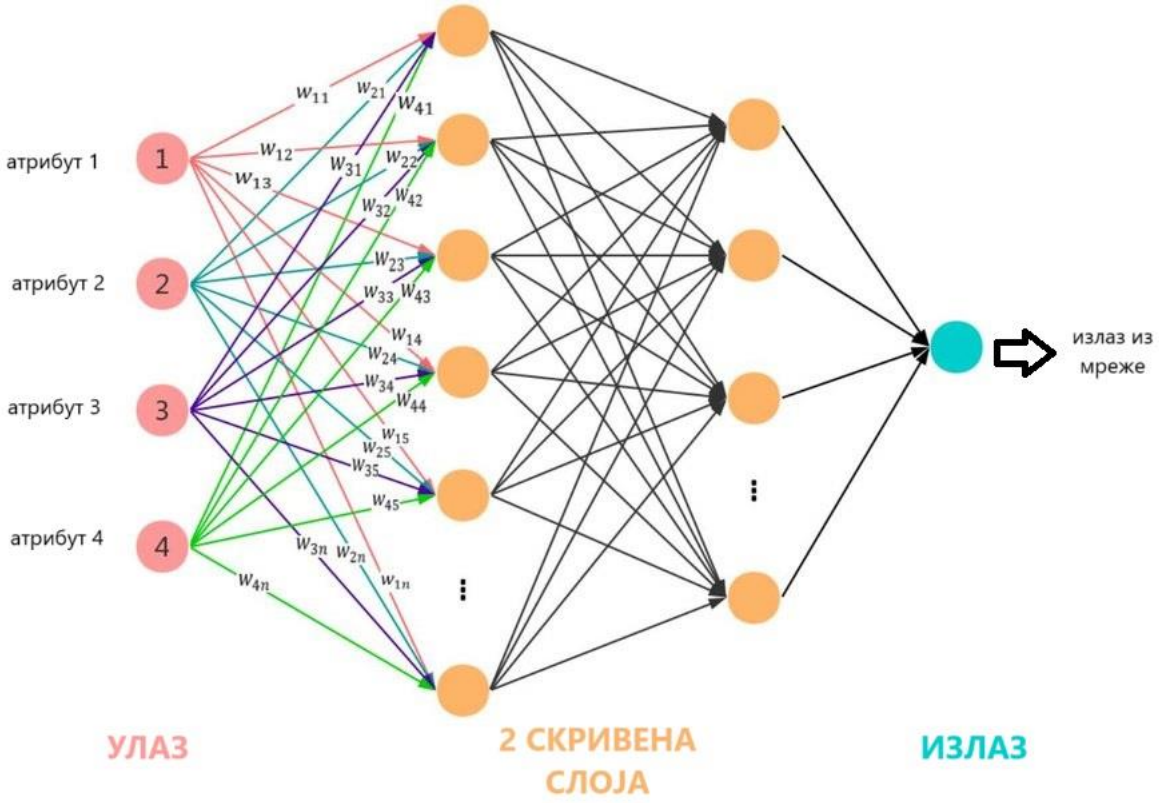
Слика 3.3 - *ReLU* активациона функција

в) **Непараметризовани слој агрегације** (енгл. *pooling layer*) у *CNN* се користи да се смање димензије мапе атрибута улазног сигнала, што помаже код компресовања информација, смањења комплексности рачунања и контроле над преприлагођавањем модела. Агрегација такође чини модел инваријантним на транслације и дисторзије у улазним сигналима. Најчешће се користе агрегација просеком (енгл. *average pooling layer)* и агрегација максимумом (енгл. *max pooling layer*) (Слика 3.4) [5, 6].



Слика 3.4 - Слојеви агрегације

г) **Потпуно повезан слој**: Излаз на крају конволутивне неуронске мреже се прослеђује као улаз у потпуно повезани слој и може бити један или више оваквих слојева.



Слика 3.5 - Потпуно повезана мрежа

4 Неуронске мреже за класификацију видео клипова

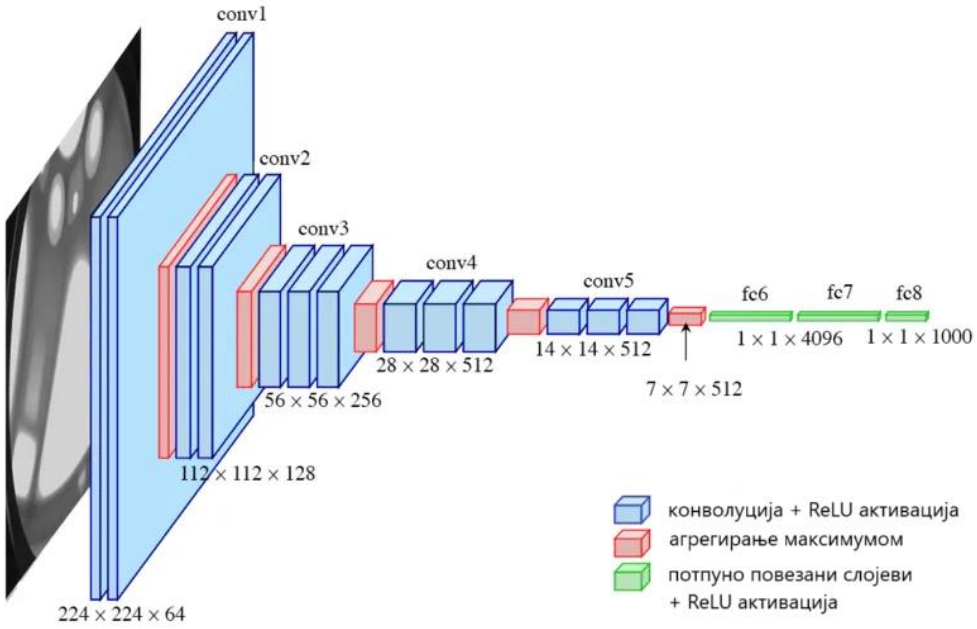
Основни проблем у примени машинског учења код препознавања знаковног језика је класификација видео клипова. Разлог због чега ово представља знатан проблем је због тога што видео клипови имају просторну и временску димензију. Као просторну димензију имамо слике са којих модел треба да уочава разне елементе. У случају овог рада углавном се ради о позицијама шака. Међутим, интерпретација неког знака не зависи само од различитих позиција шака, већ и од њиховог редоследа, што значи да временска димензија предодређује могућност тумачења.

Овај рад разматра три различита начина (архитектуре) за обраду и класификацију знаковних језика у форми видео клипова:

- *VGG* (енгл. *Visual Geometry Group*) архитектура
- *ResNet* (енгл. *Residual Network*) архитектура
- Временско-просторни трансформер модели (енгл. *TimeSformer – Time-Spatial Transformer*s)

4.1 *VGG* архитектура

VGG [5, 7] је једна од првих архитектура примењених у области рачунарског вида, која је позната због своје једноставности и дубине вештачке неуронске мреже. Осим тога, *VGG* архитектура комбинује дубоку неуронску мрежу са конволутивним филтерима мање димензије, тако да се на један начин може сматрати канонском архитектуром. Она се у данашње време углавном користи као базична архитектура, мада све ређе због њене неефикасности у погледу коришћења параметара у односу на новије и напредније архитектуре. Без обзира на ове недостатке, и баш због тога што се ради о канонској архитектури, одлучено је да се у овом раду отпочне од модела ове архитектуре, иако су њена ограничења унапред била позната. *VGG* архитектура се састоји од низа наизменично послаганих конволутивних и агрегационих слојева, а потом од потпуно повезане неуронске мреже, након чега се производи излаз из мреже (Слика 4.1).



Слика 4.1 - *VGG_16* архитектура

Генерална структура *VGG* архитектуре се може описати следећим компонентама:

а) **Улазни сигнал у мрежу** - Видео је репрезентован као низ слика, где је свака слика дводимензионална матрица вредности пиксела (висина *H*, ширина *W*) са три канала боја – црвене, зелене и плаве (енгл. *RGB – red, green & blue*). За видео клип који се састоји од *T* слика, улазни облик тензора *X* је облика:

$$X \in \mathbb{R}^{\mathrm{T}\times\mathrm{H}\times\mathrm{W}\times\mathrm{C}} \tag{4.1}$$

где је *C*=3, јер је то број канала боја.

б) **Конволутивни слојеви** - Конволутивни слојеви примењују скуп филтера (кернела) на улазне податке (сигнале). Сваки конволутивни слој трансформише улазни тензор[4] података примењујући операцију дводимензионалне конволуције на димензије простора, то јест висину и ширину (*H* и *W*), сваке слике. За филтер величине $K\times K$, излаз конволутивног слоја је дат следећом формулом:

$$Y_l = \sigma(W_l * X_l + b_l) \tag{4.2}$$

[4] Тензор је вишедимензионални квадар (слика се може приказати вредностима кроз тродимензионални квадар, док се видео клип може приказати кроз четвородимензионални квадар)

Где важи:

- l означава број слоја
- X је улаз у l-ти слој
- W_l су параметри филтера (који се уче) у l-том слоју
- * је оператор конволуције
- b_l је слободни члан (такође параметар који се учи)
- σ је нелинеарна активациона функција, обично *ReLU*
- Y_l је излаз l-ог слоја

в) **Агрегациони слојеви** – Агрегациони слојеви смањују просторне димензије (H и W) узимањем максимума или просечне вредности користећи агрегациони прозор величине $P \times P$ (Слика 3.4). Дакле, излаз из агрегационог слоја може се записати као:

$$X_{l+1} = MaxPool(X_l, P \times \mathrm{P}) \tag{4.3}$$

г) **Потпуно повезани слојеви** – Након низа конволутивних и агрегационих слојева, врши се изравнавање тензора у једнодимензионални вектор X који се прослеђује потпуно повезаном слоју:

$$Z_{l+1} = \sigma(W_l * X_l + b_l) \tag{4.4}$$

где су:

- $W_l \in \mathbb{R}^{\mathrm{d}_{l+1} \times d_l}$
- $b_l \in \mathbb{R}^{\mathrm{d}_l}$ слободни чланови уз неуроне
- $X \in \mathbb{R}^{\mathrm{d}_l}$ улаз
- d_l и d_{l+1} димензије l-ог $l+1$-ог слоја
- $Z_{l+1} \in \mathbb{R}^{\mathrm{d}_{l+1}}$

д) **Коначна класификација** – Последњи слој примењује познату софтмакс (енгл. *softmax*) активациону функцију да би на излаз пропустио вектор оцена вероватноћа припадности свакој класи:

$$\hat{y} = Softmax(Z) \tag{4.5}$$

где је:

$$Softmax(Z)_i = \frac{e^{z_i}}{\sum_j e^{z_j}} \tag{4.6}$$

ђ) **Функција грешке** – Током тренинга, модел користи унакрсну ентропију као функцију грешке (енгл. *cross-entropy*) да упореди оцене вероватноћа $\hat{y}$ са ознакама стварне припадности класама *y*:

$$\mathcal{L}(\hat{y}, y) = -\sum_i y_i \log(\hat{y_i}) \tag{4.7}$$

Где:

- $\hat{y_i}$ је оцена вероватноће да улаз припада класи *i*
- $y_i = 1$ ако улаз припада класи i, иначе $y_i = 0$

Пошто је намера употребити знање које је у циљу класификације слика *VGG* мрежа научила на већ поменутој *ImageNet* бази података, онда се користе њени претренирани параметри као почетне вредности који се затим дотрениравају (дообучавају) за класификацију гестикулација базе података „SLOVO“.

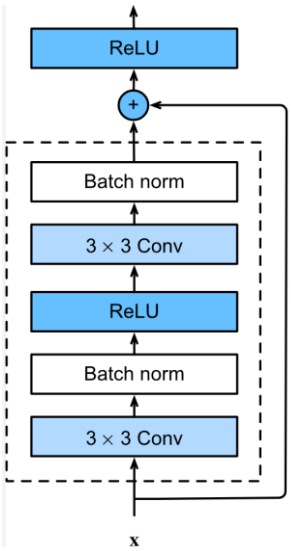
4.2 *ResNet* архитектура

Дубоке неуронске мреже омогућавају екстракцију комплекснијих атрибута из података и у смислу класификације слика остварују много боље резултате него плиће мреже. У овом контексту, поставља се питање: *„Да ли је за тренирање бољих неуронских мрежа једноставно довољно постојећу архитектуру мреже учинити дубљом, односно додати јој више слојева?“*

Познато је да приликом ажурирања параметара и поновне пропагације унапред, мала промена у параметрима доводи до велике промене сигнала у слојевима наслаганим после њих. То доводи до велике и честе промене расподеле сигнала у слојевима на дубоким нивоима током тренинга што веома отежава оптимизациони процес и остваривање конвергенције ка оптимуму. Такође, код традиционалних архитектура постоји и проблем нестајућих и експлодирајућих градијената који се појављују због дубине мреже. Потреба за превазилажењем ових проблема је мотивисала увођење нормализационих слојева, на пример нормализације тренинг подскупа (енгл. *batch normalization*). Њоме су остварене стабилније расподеле сигнала по слојевима, што је омогућило коришћење већих корака учења, бржу конвергенцију модела, те омогућило мрежама, које имају десетине слојева, да конвергирају ка оптимуму. Поменута нормализација сигнала над подскупом тренинг

података је и имплементирана као једна од компоненти *ResNet* архитектуре [8]. Ипак, како се додатно повећава дубина неуронских мрежа, јавља се деградациони проблем - додавање слојева више не помаже тренирању бољих модела, то јест мера тачности (енгл. accuracy) класификационог модела опада са додатним порастом дубине мреже. Аутори [9] *ResNet* (*residual network*) архитектуре сматрају да је проблем са деградацијом једноставно у томе што како мреже постају дубље, оне постају и комплексније, а тиме постаје теже оптимизовати их. Они су приступили решавању овог проблема додавањем прескачућих веза (енгл. *skip connections*).

Код *ResNet*-а слојеви мреже су подељени у такозване блокове. Улазни сигнал у блок се унапред пропагира двема путањама, једним путем кроз конволутивне и нормализационе слојеве, а другим путем везом која прескаче те конволутивне слојеве пропагирајући улазни сигнал такав какав јесте. Затим се врши сабирање та два тензора и њихов збир се пропушта кроз активациону функцију (Слика 4.2).



Слика 4.2 - *ResNet* блок

Дакле, код *VGG*-а су наслагани конволутивни слојеви, док су код *ResNet*-а наслагани блокови. Уколико посматрамо трансформацију сигнала у оквиру једног нивоа, пресликавање какво бисмо желели да овај ниво научи, можемо означити са *H(x)*, где је *x* улазни сигнал. Дакле, код *ResNet*-а за разлику од *VGG*-а, конволутивни аспект у једном нивоу не учи директно пресликавање *H(x)*, већ $F(x) = H(x) - x$, где са $F(x)$ означавамо конволутивну трансформацију која се учи. Аутори су приметили да, у случају када би жељено пресликавање била идентитетска функција, мрежа има могућност да параметре конволуције једноставно постави на нулу, с обзиром да је идентитет улаза већ прослеђен прескачућом везом. Они затим постављају хипотезу да је идентитетско пресликавање лакше

научити на овај начин, него да функцију идентитета учи користећи искључиво нелинеарне трансформације. Ова једноставна, али ефектна идеја, омогућава да се тренирају веома дубоке мреже са стотинама слојева дубине без појављивања проблема нестајућих или експлодирајућих градијената.

4.3 Трансформер архитектура

Иако је за овај рад релевантна тајмсформер варијанта трансформер архитектуре, за њено разумевање је првобитно било потребно разумети основну трансформер архитектуру која се примењује за потребе великих језичких модела (енгл. *large language models*). Међутим, за разумевање основне трансформер архитектуре, било је претходно потребно разумети и механизам пажње (енгл. *attention mechanism*), који је као главни концепт уграђен у трансформер моделе. Стога је у раду представљена хронолошка еволуција и математичка репрезентација од механизма пажње, преко трансформера, до тајмсформера.

4.3.1 Механизам пажње

Трансформер модели су еволуирали из потребе језичких модела за проналажењем бољих метода аутоматског превођења међу природним језицима где се овај процес побољшања одвијао постепено. На пример, аутори [10] примећују да код претходних енкодер-декодер модела за превођење, где је енкодер задужен за екстраховање вектора контекстних атрибута улаза (текста) на основу којих декодер потом генерише излаз (превод), изворна реченица се енкодира у вектор фиксне дужине. Ово представља „уско грло“, јер за потребе дужих реченица које се преводе и поседују више информација од краћих реченица, њихов садржај треба да се компресује у вектор фиксне дужине, што може бити недовољно у погледу очувања смисла дужих и сложенијих реченица. Они су предложили унапређење енкодер-декодер архитектуре, тиме што су дозволили да декодер аутоматски трага за релевантним деловима улазне реченице, уместо да врши декодирање само на основу вектора фиксне дужине који би енкодер самостално произвео. Дакле, у оквиру овог унапређења, енкодер од улазне секвенце производи нову секвенцу контекстуализованих вектора, а декодер при генерисању излаза „посматра” ову новокреирану секвенцу. На овај начин енкодер не покушава да улазну реченицу „спакује“ у вектор фиксне дужине, чиме је отклоњено идентификовано „уско грло“, а декодер у циљу адекватног превођења текста са једног језика на други, може да „посматра“ контекст улазне реченице који није ограничен унапред дефинисаном фиксном величином вектора.

Како се речи на другом језику током превођења генеришу секвенцијално, реч по реч, на процес генерисања нове речи у секвенци која се преводи утиче контекст улазне реченице, али и претходно генерисане речи, јер на пример, ако је нека именица из реченице већ преведена, она се вероватно неће опет преводити, односно неће се дуплирати реч у преводу. Ово гледање на свеукупан контекст улазне и излазне реченице током сваког корака генерисања превода се назива „механизам пажње“ (енгл. *attention mechanism*). Овај механизам је концеуптуално један од главних принципа који се примењују у трансформер моделима.

Ради бољег разумевања, корисно је представити и математички концепт који стоји иза еволуције трансформер модела. Архитектура коју су аутори [10] узели као полазну тачку се заснива на енкодер-декодер архитектури, где енкодер пресликава улазну реченицу, која је уствари секвенца вектора $\mathbf{x}=(x_1, \ldots, x_{Tx})$ (речи су представљене векторима), у контекст вектор **c**. Приступ који се овде најчешће користи се заснива на *RNN* (рекурентним неуронским мрежама, енгл. recurrent neural networks) тако да:

$$h_t = f(x_t, h_{t-1}) \tag{4.8}$$

$$c = q(\{h_1, \ldots, h_{T_x}\}) \tag{4.9}$$

Где су:

- $h_t \in \mathbb{R}^N$ је скривено стање (енгл. *hidden state*) улаза у времену *t*
- *c* је вектор контекста генерисан секвенцом скривених стања
- *f* и *q* су нелинеарне функције.

За функцију *f* се често употребљава форма *LSTM* (дуга краткорочна меморија, енгл. *long short term memmory*) функције, док се за *q* може узети рецимо $q(\{h_1,\ldots,h_T\})=h_T$.

Декодер се обично тренира да предвиди следећу реч $y_{t'}$, где је претходно дат контекст вектор *c*, као и до тада претходно предвиђене речи $\{y_1, \ldots, y_{t'-1}\}$. Ово такође можемо схватити на начин да декодер дефинише вероватноћу превода текста тако што изврши декомпозицију заједничке вероватноће (4.10) свих речи излазне реченице у производ појединачних условних вероватноћа сваке следеће речи:

$$p(\boldsymbol{y}) = \prod_{t=1}^{T} p(y_t|\{y_1, \ldots, y_{t-1}\}, c) \tag{4.10}$$

Где:

- $\boldsymbol{y}=(y_1, \ldots, y_{Tx})$ је излазна реченица.

RNN-ом свака од условних вероватноћа се моделира као:

$$p(y_t|y_1, \dots, y_{t-1}, c) = g(y_{t-1}, s_t, c) \quad (4.11)$$

Где:

- g је нелинеарна, потецијално вишеслојна неуронска мрежа, функција која одређује вероватноће од y_t
- s_t је скривено стање *RNN*-а.

Аутори [10] су модификовали ову базичну архитектуру на следећи начин. Суштина њихове модификације састоји се у томе да су користили двосмерни *RNN* као енкодер. Декодер обавља претрагу контекстне секвенце, то јест секвенце скривених стања коју је креирао енкодер од улазне реченице, током генерисања превода. Енкодер је имплементиран као двосмерни *RNN* да би скривена стања која одговарају речима реченице, тумачила контекст одговарајуће речи у односу на све речи у реченици, дакле у односу на речи које претходе и долазе након речи на које декодер обраћа пажњу. У својој модификацији аутори су дефинисали сваку од условних вероватноћа као:

$$p(y_i|y_1, \dots, y_{i-1}, \boldsymbol{x}) = g(y_{i-1}, s_i, c_i) \quad (4.12)$$

где:

- s_i је скривено стање *RNN*-а у времену *i*, које се рачуна као:

$$s_i = f(s_{i-1}, y_{i-1}, c_i) \quad (4.13)$$

Треба приметити да за разлику од претходних енкодер-декодер решења која користе јединствен контекст вектор *c*, овде условна вероватноћа зависи од различитих састава контекст вектора c_i за сваку реч која се генерише y_i. Контекст вектор c_i зависи од контекстне секвенце $(h_1, \dots, h_{Tx})$ које енкодер креира трансформацијом улазне реченице. Битно је приметити да свако h_i садржи информације о целој улазној секвенци, али да због „природе“ меморије скривених стања *RNN*-а поготово „памти“ информације о свом суседству, односно деловима који окружују *i*-ту реч, пре и после појављивања саме *i*-те речи.

Контекст вектор c_i се рачуна као тежински просек контекстне секвенце *h*:

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j \quad (4.14)$$

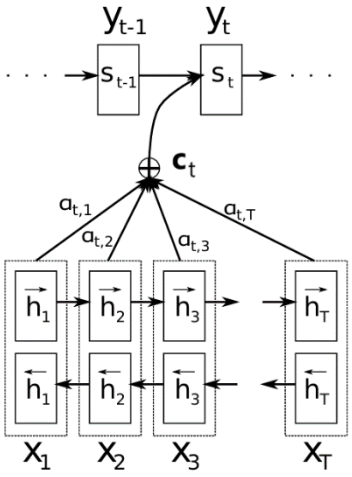
Тежина α_{ij} се добија као:

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})} \tag{4.15}$$

где:

$$e_{ij} = a(s_{i-1}, h_j) \tag{4.16}$$

Једначина 4.16 је подмодел сагласности који оцењује колико су улази у близини позиције *j* и излаз у позицији *i* у складу. Оцена је базирана на *RNN* скривеном стању s_{i-1} (односно непосредно пре генерисања y_i) и h_j елементу контекстне секвенце.

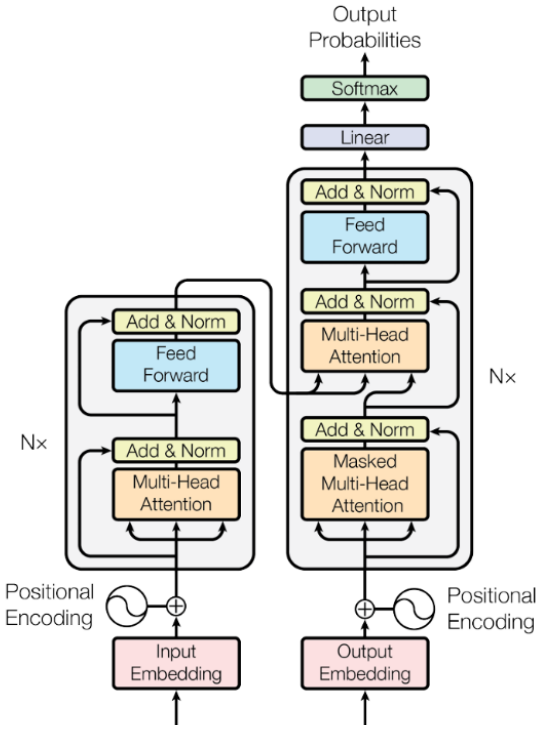


Слика 4.3 - Енкодер-декодер архитектура која користи „механизам пажње“

4.3.2 Трансформери

Академски рад [11] је прва употреба трансформер архитектуре, која је искључиво базирана на механизму (интроспективне) пажње (енгл. *self-attention mechanism*), дакле елиминишући потребу за коришћењем рекурентних или конволутивних компоненти у моделу. Код модела који као основни блок архитектуре користе конволуцију или рекуренцију појављује се проблем да број операција које је потребно извршити, да би се било која два сигнала узета из улаза или излаза довела у релацију, расте што је већа дистанца између ове две позиције. Ова чињеница чини компликованијим учење зависности између позиција које се налазе на већим дистанцама. Трансформер модел редукује ово на константан број операција, хватајући везе између токена[5] (елемената) у секвенци независно од њихове удаљености. Такође код модела који користе рекурентне мреже, због њиховог секвенцијалног тока израчунавања, није могуће паралелизовати тренирање модела. Трансформер омогућава бољу паралелизацију, јер не користи рекурентне операције, па стога захтева мање времена за тренирање модела.

Као и велики број других неуронских модела који оперишу секвенцијалним подацима, трансформер модел има енкодер-декодер архитектуру. Код њих енкодер пресликава улазну секвенцу репрезентација речи $(x_1, \ldots, x_n)$ у контекстуализовану секвенцу



Слика 4.4 - Трансформер архитектура

[5] Токени су основни елементи који граде секвенцу. Код превођења текста са једног језика на други, токени, то јест елементи секвенце, су речи и знакови интерпункције.

z=(z_1, …, z_n). Ако му је дат **z**, декодер онда генерише секвенцу речи y=(y_1, …, y_m), елемент по елемент. У сваком кораку модел је ауто-регресиван, што значи да користи претходно генерисане токене, као додатни улаз када генерише следећи нови токен. Трансформери примењују концепт ових архитектура, с тим што се састоје од слојева који примењују механизам пажње на читаву улазну/излазну секвенцу и од позиционих потпуно повезаних мрежа. Називају се позиционе јер се примењују на сваку репрезентацију токена посебно (Слика 4.4).

Функција интроспективне пажње (енгл. *self-attention function*) ради тако што прво од секвенце улазних токена линеарном пројекцијом креира упите (Q) и парове кључева и вредности (K и V). Вредности (V) у новом пресликаном простору представљају смисао тих токена, док кључеви (K) представљају адресу тих вредности преко којих ће упити (Q) моћи да приступају вредностима и да се функцијом сличности, која је уствари (скалирани) скаларни производ, „питају“ колико су новокреиране репрезентације токена у виду упита и кључева у међусобној сагласности. Након што се одреди у којој мери су од токена креирани упити (Q) у сагласности са креираним кључевима (K), за које смо рекли да имају свој пар у виду вредности (V), тада се софтмакс активационом функцијом, за сваки упит, добијају тежине које се користе за упросечавање вредности (V). Тако се креирају нове вредности које дају контекстуализованији смисао токенима, односно апстрахују се у репрезентацију међусобне повезаности. Ово се све обавља у форми алгебре матрица и математички се пише:

$$Q = XW^Q; K = XW^K; V = XW^V \tag{4.17}$$

Где:

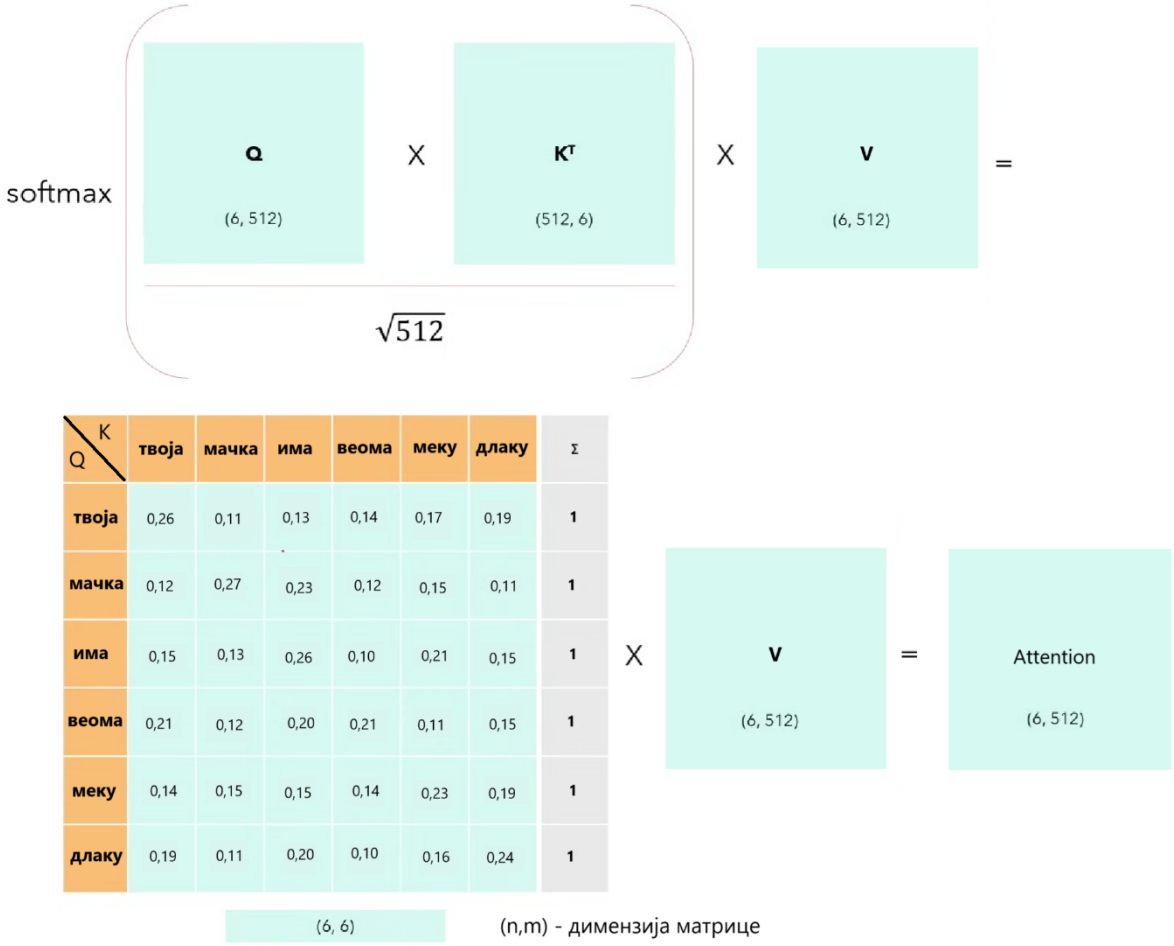
- X је матрица димензија $d_l \times d_{model}$ која представља секвенцу токена
- d_l је дужина секвенце
- d_{model} величина векторске репрезентације токена
- W^Q и W^K су матрице димензија $d_{model} \times d_k$
- W^Vје димензија $d_{model} \times d_v$

Дакле, помоћу матрица W^Q, W^K и W^V улазна секвенца токена X се линеарно пројектује на упите (Q), кључеве (K) и вредности (V). Креиране матрице Q и K су димензија $d_l \times d_k$, а V је димензије $d_l \times d_v$.

Фукнција пажње (енгл. *attention function*) се рачуна на следећи начин:

$$Attention(Q,K,V) = Softmax\left(\frac{QK^T}{\sqrt{d_k}}\right) * V \qquad (4.18)$$

Скаларни производ упита и кључева се скалира са $\frac{1}{\sqrt{d_k}}$ пре употребе софтмакса, пошто је у противном градијент софтмакс функције премали за велике вредности што онемогућава ефикасну конвергенцију приликом процеса учења. Приметићемо да је ова функција пажње слична фукнцији пажње из рада [10], јер (скалирани) скаларни производ и примена софтмакс активационе функције на њега одговарају једначинама из тог рада, које су у овом раду означене са 4.16 и 4.15, док множење тог резултата са матрицом V и прављење контекстуализованије репрезентације секвенце одговара једначини 4.14.



Слика 4.5 – Механизам пажње

Придодавање контекстуализованог смисла секвенци токена се може урадити на више начина, у зависности од перспективе посматрања, што би у циљу превођења текста са једног језика на други могле бити, на пример: перспективе синтаксне уређености у реченици, семантички карактер реченице, везе између субјеката, предиката, објеката или неке

префињеније везе међу речима. Због тога, да би се избегло креирање поједностављеног смисла кроз комбиновање вредности (V) само једним тежинским упросечавањем од којих се тежине добијају рачунајући сагласност упита (Q) и кључева (K), у оквиру једног слоја се користи механизам пажње са више глава (енгл. *multi-head attention*), односно механизам пажње се примењује више пута независно и тако се добија богатији контекстуални опис секвенце која се обрађује:

$$MultiHead_{Attention(Q,K,V)} = Concat(head_1, \dots, head_h)W^O \quad (4.19)$$

Где је:

$$head_i = Attention(Q_i, K_i, V_i) = Attention(XW_i^Q, XW_i^K, XW_i^V) \quad (4.20)$$

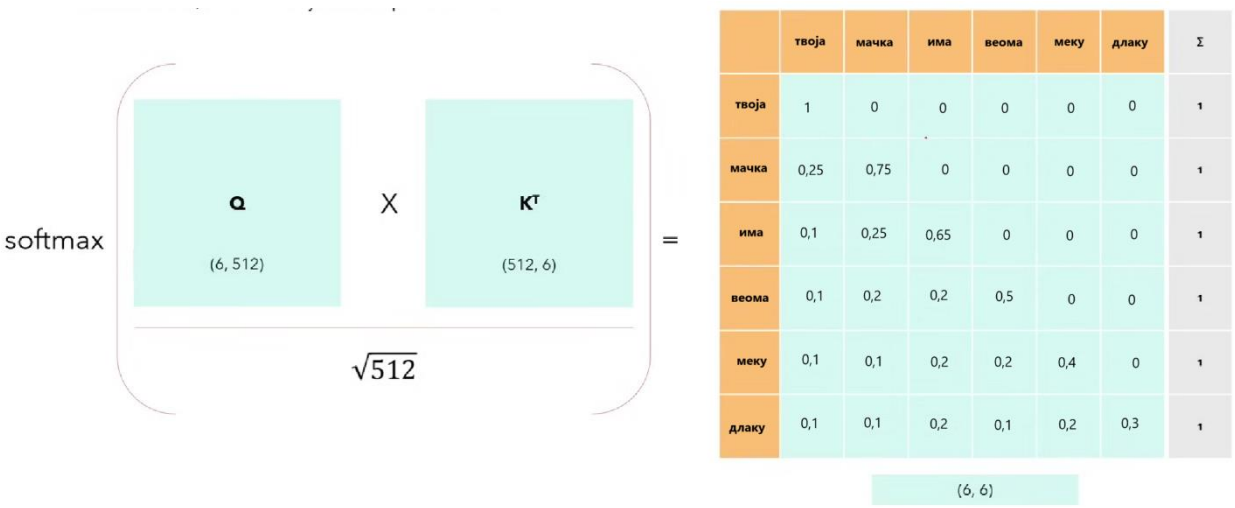
Дакле, за сваку главу се примени механизам пажње и тако направи нова контекстуализованија секвенца токена, а потом се надовежу резултати сваке главе једна на другу, те се опет линеарном пројекцијом комбинују резултати свих глава, а чува се и почетна димензија токена, с обзиром да је W^O матрица димензије $hd_v \times d_{model}$, где је h број глава.

Енкодер се код трансформера [11] састоји од шест слојева, где се сваки од њих састоји од два главна подслоја. Први од њих је механизам интроспективне пажње, док је други подслој потпуно повезана мрежа која пропагира, кроз себе, сваки токен посебно и њоме се уводи нелинеарност у неуронску мрежу (Слика 4.4).

Декодер се такође састоји од шест слојева, с тим што се сваки слој састоји од три подслоја. Први подслој, исто као код енкодера, је механизам интроспективне пажње. Други подслој је такође механизам пажње, али није интроспективни, јер упите (Q) креира од излаза из претходног подслоја декодера, док кључеве (K) и вредности (V) креира од излаза из енкодера и на тај начин комбинује секвенцу која треба да се преведе и оно што је већ преведено, тј. чува ауторегресивно својство. Трећи подслој је опет потпуно повезана мрежа као и код енкодера (Слика 4.4).

Излази из сваког слоја и подслоја унутар трансформера су истих димензија (d_{model}), што омогућава лако коришћење прескачућих веза (енгл. *residual connections*). После сваког подслоја користе се нормализациони слојеви над излазима из њих (Слика 4.4). Да би се корак учења у процесу тренирања модела могао извршити у једном пропагирању сигнала кроз мрежу, дакле без секвенцијалног пропагирања токена, што је случај код рекурентних неуронских мрежа, декодер при тренирању модела користи маскирање токена када се рачуна сагласност упита и кључева у оквиру првог подслоја сваког слоја, односно подслоја интроспективне пажње. Декодер је тако дизајниран да се при рачунању сагласности упита

и кључева, један токен од кога је креиран упит не би доводио у везу са другим токеном од кога је креиран кључ уколико други токен долази након првог у поретку секвенце, јер би ово на један начин представљало гледање у будућност, а тиме би се покварило ауто-регресивно својство модела. Дакле, посматрајући секвенцијално, када одговарајући токен кључа долази после одговарајућег токена упита, поменуто маскирање при тренирању се врши тако што се скаларни производ тих кључева и упита постави на $-\infty$. Стога ће приликом примене софтмакс функције тежина бити 0 и веза између упита и кључа тих токена неће учествовати у прављењу нове контекстуализованије секвенце токена. У овом случају, матрица сагласности упита и кључева ће имати све нуле изнад дијагонале (Слика 4.6). У тестирању или у примени модела, токени се ипак генеришу секвенцијално, па нема потребе за маскирањем токена, већ се најновије генерисани токен надовезује на до тада генерисану секвенцу и пропагација се понавља све док има потребе за генерисањем нових токена. Крај генерисања превода се прекида генерисањем „специјалне речи“, која служи за ту сврху.



	твоја	мачка	има	веома	меку	длаку	Σ
твоја	1	0	0	0	0	0	1
мачка	0,25	0,75	0	0	0	0	1
има	0,1	0,25	0,65	0	0	0	1
веома	0,1	0,2	0,2	0,5	0	0	1
меку	0,1	0,1	0,2	0,2	0,4	0	1
длаку	0,1	0,1	0,2	0,1	0,2	0,3	1

Слика 4.6 - Маскирана пажња

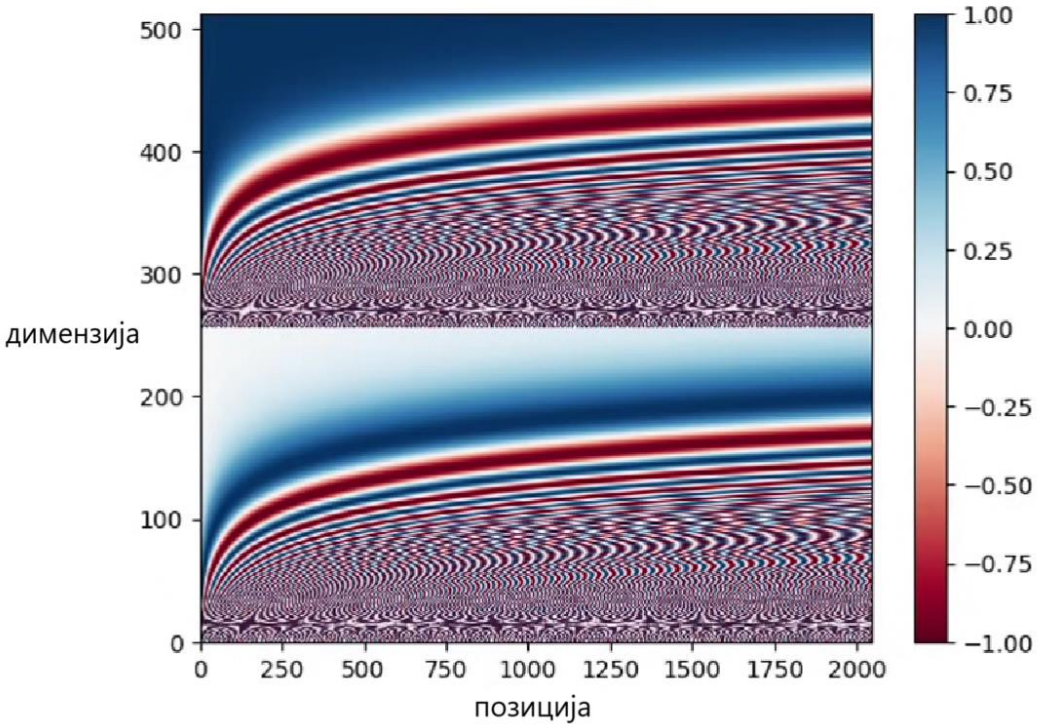
Важно је напоменути да се приликом прослеђивања токена у енкодер и декодер у њихову векторску репрезентацију утискује информација о њиховој позицији у секвенци, с обзиром да је то једна од суштински битних информација, јер позиције токена (речи) утичу на смисао секвенце (реченице). Ово се назива енкодирање позиције (енгл. *positional encoding* - *PE*) и ова информација се додаје на сваку димензију векторске репрезентације токена помоћу функција синуса и косинуса:

$$PE(pos, 2i) = sin\frac{pos}{10000^{\frac{2i}{d_{model}}}} \tag{4.21}$$

$$PE(pos, 2i+1) = cos\frac{pos}{10000^{\frac{2i}{d_{model}}}} \tag{4.22}$$

Где:

- *pos* је позиција токена у оквиру секвенце
- *2i/2i+1* је димензија (парна/непарна) векторске репрезентације у коју се утискује информација о позицији токена.



Слика 4.7 - Информације о позицијама токена које се сабирањем утискују на њихове векторске репрезентације

Дакле векторска репрезентација токена **x** се сабере са информацијом о позицији токена **PE** и збир се прослеђује на улаз трансформер модела. Дакле, кад је парна димензија у питању, утискује се информација користећи синусну функцију, док се у случају непарне димензије користи косинусна функција. Вредности које се утискују, односно додају на векторске репрезентације токена, у зависности од позиције токена у секвенци и од димензије векторске репрезентације токена могу се видети на слици 4.7. На основу овако утиснуте информације трансформер модел успева да разазна и самим тим користи информацију о позицијама токена у секвенци из којих долазе. Аутори [11] су могли да примене и друге методе за утискивање информације о позицијама токена на њихове векторске репрезентације, али су се одлучили на формулу која је у стању да екстраполира информацију о позицији на секвенце које су дуже од оних у тренинг скупу података.

4.3.3 Тајмсформери

Као што се код језичких модела текст посматра као секвенца речи и знакова интерпункције, тако се видео клипови могу посматрати као секвенца слика. Тајмсформери (енгл. *TimeSformers – Time-Space transformers*), представљени у раду [12], су варијанта трансфомер модела дизајнирана за рад са видео клиповима. Разумевање њиховог контекста се остварује посматрајући везе између елемената унутар појединачних слика, али и везе између елемената слика у различитим временским тренуцима. Код тајмсформера, да би се боље могао разумети контекст слика кроз време, уместо да један токен буде једна слика видео клипа, свака слика се дели на парчиће/токене и механизмом пажње се онда издваја контекстуални опис видео клипа како кроз време тако и унутар слике у једном тренутку. Као и оригиналан трансформер модел [11], тајмсформер је такође базиран искључиво на механизму пажње, односно не користи се конволутивним и рекурентним неуронским мрежама. Пошто циљ тајмсформера није да генерише нову секвенцу, већ само да класификује улазни видео, архитектура се базира само на енкодеру, односно декодер није потребан у овом случају.

Свака слика у видео клипу која је димензија $H \times W$ пиксела може да се подели на парчиће од по $P \times P$ пиксела и тако се добије $N = HW/P^2$ парчића у свакој слици. У примеру са базом података „SLOVO“, величине слика које се прослеђују тајмсформеру су 224×224 пиксела, а величине парчића/токена су 16×16 пиксела. Сваки парчић слика чини један токен за тајмсформер архитектуру и могу се означити са $x_{(p,t)}$, где је:

- p позиција токена/парчића у оквиру слике којој припада - $p \in 1 \dots N$
- t је ознака за слику - $t \in 1 \dots F$, а F је дужина видео клипа.

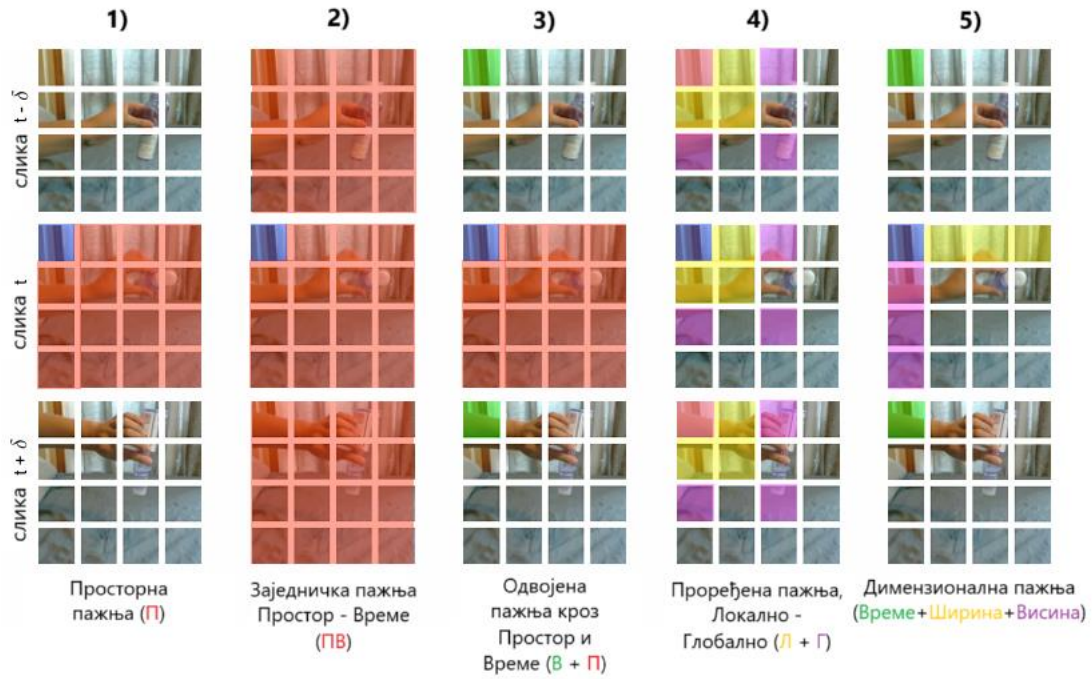
Дакле, сваки парчић се може сместити у вектор димензија $\mathbb{R}^{3P^2}$ (P^2 је број пиксела, а 3 је број канала – црвена, зелена и плава боја). Као и код оригиналног трансформера у [11], токени се пре прослеђивања у први слој преместе у векторски простор, такозвани „embedding” простор, који припада скупу $\mathbb{R}^D$. То се остварује линеарном пројекцијом користећи параметризовану матрицу (која се учи) $E \in \mathbb{R}^{D \times 3P^2}$, а затим се у њих утисне информација о временско-просторној позицији токена у видео клипу додавањем позиционо енкодираног вектора $e^{pos}_{(p,t)} \in \mathbb{R}^D$:

$$z^{(0)}_{(p,t)} = Ex_{(p,t)} + e^{pos}_{(p,t)} \tag{4.23}$$

Такође се креира један специјални, такозвани класификациони токен $z_{(0,0)}^{(0)}$, који се учи кроз тренирање модела и поставља се на нулту позицију секвенце. Из њега се на излазу из енкодера екстрахује информација о класи којој видео клип припада, те се на тај начин врши класификација видеа.

Аутори тајмсформера [12] су експериментисали са коришћењем механизма интроспективне пажње на више начина (Слика 4.8):

1. Коришћење само просторне пажње, што значи да су интроспективну пажњу користили само међу парчићима/токенима из исте слике
2. Коришћење заједничке пажње кроз простор и време, што значи да је сваки токен обраћао пажњу на све преостале токене без обзира на позицију у оквиру слике и на то да ли се налази у другој слици
3. Коришћење одвојене пажње кроз простор и време у два подслоја, што значи да се прво примени механизам интроспективне пажње међу токенима који се налазе на истим позицијама у оквиру слика, али у другим временским тренуцима, а затим у другом подслоју просторна пажња између токена из истих слика
4. Коришћење прорећене пажње кроз два подслоја – локалне и прорећене глобалне пажње. Користећи ознаке *H* и *W* за висину и ширину слика, а *F* за дужину видео клипа, у оквиру подслоја локалне пажње се за сваки токен примењује механизам пажње на *F·H/2·W/2* локално блиских токена (Слика 4.8 – под 4). Затим се у оквиру следећег подслоја (за сваки токен) рачуна глобална пажња у односу на сваки други токен уздуж временске и просторних димензија. Стога, ова метода је апроксимација методе број 2 – заједничке пажње кроз простор и време, која се брже извршава.
5. Коришћење пажње по три димензије – двапут по просторним димензијама - висини и ширини, а једанпут по временској димензији



Слика 4.8 - Примена механизма пажње на више начина

Аутори [12] су тестирали свих пет начина за коришћење механизма интроспективне пажње у циљу класификације видео клипова и закључили да најбоље резултате даје метода број 3 - „Одвојена пажња кроз простор и време“. Метод број 2 је показао боље резултате од метода број 1, с обзиром да користи пажњу и кроз временску димензију, али метод број 3 је показао бољи резултат од метода број 2, иако је једноставнији и троши знатно мање времена за израчунавање. Ако је, као што је већ наведено, N број парчића у једној слици, а F дужина видеа у сликама, користећи метод број 3, у оквиру механизма интроспективне пажње за сваки токен/парчић потребно је израчунати $N+F+2$ поређења са другим токенима, док је за метод број 2 то $N \cdot F+1$. Ово постаје још јасније ако се размотре формуле рачунања механизма интроспективне пажње. Број слоја о коме се ради се означава са „l“, а глава о којој се ради, пошто се опет користи вишеглави механизам пажње се означава са „a“:

$$\boldsymbol{q}_{(p,t)}^{(l,a)} = W_Q^{(l,a)} LN\left(\boldsymbol{z}_{(p,t)}^{(l-1)}\right) \in \mathbb{R}^{D_h} \tag{4.24}$$

$$\boldsymbol{k}_{(p,t)}^{(l,a)} = W_K^{(l,a)} LN\left(\boldsymbol{z}_{(p,t)}^{(l-1)}\right) \in \mathbb{R}^{D_h} \tag{4.25}$$

$$\boldsymbol{v}_{(p,t)}^{(l,a)} = W_V^{(l,a)} LN\left(\boldsymbol{z}_{(p,t)}^{(l-1)}\right) \in \mathbb{R}^{D_h} \tag{4.26}$$

Где:

- LN трансформација коју извршава нормализациони подслој

- $D_h = D/A$, а D је величина „*embedding*“ простора токена и A је број глава.

Када су од токена израчунати упити, кључеви и вредности, може се израчунати и „пажња“. Прво је потребно израчунати тежинске коефицијенте, које се у методу број 2 рачунају по формули:

$$\boldsymbol{\alpha}_{(p,t)}^{(l,a)} = Softmax(\frac{\boldsymbol{q}_{(p,t)}^{(l,a)^T}}{\sqrt{D_h}} \cdot [\boldsymbol{k}_{(0,0)}^{(l,a)} \{\boldsymbol{k}_{(p',t')}^{(l,a)}\}_{\substack{p'=1,\dots,N \\ t'=1,\dots,F}}]) \quad (4.27)$$

Код најуспешнијег метода број 3 – „механизам одвојене пажње кроз простор и време“, за примењивање механизма просторне пажње токен се пореди само са токенима из исте слике, па је формула за тежине:

$$\boldsymbol{\alpha}_{(p,t)}^{(l,a)space} = Softmax(\frac{\boldsymbol{q}_{(p,t)}^{(l,a)^T}}{\sqrt{D_h}} \cdot [\boldsymbol{k}_{(0,0)}^{(l,a)} \{\boldsymbol{k}_{(p',t)}^{(l,a)}\}_{p'=1,\dots,N}]) \quad (4.28),$$

док се код примењивања механизма временске пажње, токен пореди само са токенима из исте просторне позиције свих слика:

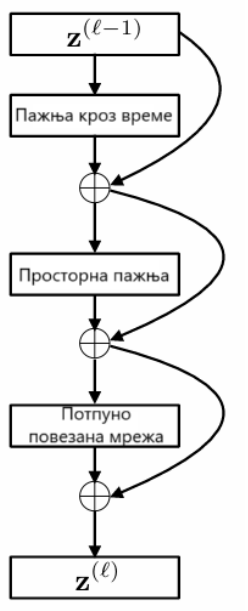
$$\boldsymbol{\alpha}_{(p,t)}^{(l,a)time} = Softmax(\frac{\boldsymbol{q}_{(p,t)}^{(l,a)^T}}{\sqrt{D_h}} \cdot [\boldsymbol{k}_{(0,0)}^{(l,a)} \{\boldsymbol{k}_{(p,t')}^{(l,a)}\}_{t'=1,\dots,F}]) \quad (4.29)$$

Дакле, може се видети да се у случају када се користи метод број 2, односно метода механизма заједничке пажње међу свим токенима, дати упит $q_{(p,t)}^{(l,a)^T}$, креиран од једног токена, се множи са *N·F+1* различитих кључева (4.27), а када се користи метод број 3, односно метода механизма подељене пажње кроз време и простор, дати упити $q'^{(l,a)^T}_{(p,t)}$ и $q''^{(l,a)^T}_{(p,t)}$, креирани од једног токена, се множе са *N+F+2* различитих кључева (прво *N+1* множења кроз временску, па *F+1* множења кроз просторну димензију – слика 4.9 и једначине 4.28 и 4.29), као што је наведено.

У методу број 3, једначине 4.28 и 4.29 се примењују за све токене, осим за специјални „класификациони токен“. Упити креирани од њега, израчунавају сагласност (тежине $\boldsymbol{\alpha}_{(0,0)}^{(l,a)ct}$)[6] са свим кључевима, па стога и пажњу, на исти начин као код метода број 2 (4.27).

[6] *ct* – Ознака за „класификациони токен“ (енгл. *classification token*)

Дакле, осим што се метод број 3 показао као најуспешнији у циљу класификације видео клипова, он користи мање операција за израчунавање од метода број 2. У овој методи основни блок неуронске мреже је састављен од три подслоја. У првом подслоју се користи механизам пажње међу токенима из исте просторне позиције кроз време. У другом подслоју се користи механизам пажње међу токенима који се налазе у истој слици, а затим се резултат прослеђује потпуно повезаној неуронској мрежи као трећем подслоју. Између свих подслојева постоје прескачуће везе. Основни неуронски блок у примени метода број 3 је приказан на слици 4.9:



Слика 4.9 - Метод број 3: „Одвојена пажња кроз простор и време“

Дакле, у овој методи, када су израчунати тежински коефицијенти пажње кроз време - $\alpha_{(p,t)}^{(l,a)time}$, након се израчуна и „пажња“, односно нове контекстуализоване вредности за сваку главу a:

$$\boldsymbol{s}_{(p,t)}^{(l,a)time} = \boldsymbol{\alpha}_{(p,t),(0)}^{(l,a)time} * \boldsymbol{v}_{(0,0)}^{(l,a)} + \sum_{t\prime=1}^{F} \boldsymbol{\alpha}_{(p,t),(t\prime)}^{(l,a)time} * \boldsymbol{v}_{(p,t\prime)}^{(l,a)} \qquad (4.30)$$

Пажња, односно контекстуализовани вектор који одговара специјалном (нултом) „класификационом токену“, ради подсећања, рачуна се као тежински просек вредности свих токена:

$$\boldsymbol{s}_{(0,0)}^{(l,a)time} = \boldsymbol{\alpha}_{(0,0),(0,0)}^{(l,a)ct} * \boldsymbol{v}_{(0,0)}^{(l,a)} + \sum_{p\prime=1}^{N} \sum_{t\prime=1}^{F} \boldsymbol{\alpha}_{(p,t),(p\prime,t\prime)}^{(l,a)ct} * \boldsymbol{v}_{(p\prime,t\prime)}^{(l,a)} \qquad (4.31)$$

Затим се резултати сваке главе надовежу, па се линеарном пројекцијом направи комбинација вредности резултата, а проследи се и прескачућа веза:

$$z'^{(l)}_{(p,t)} = W_O \begin{bmatrix} s^{(l,1)time}_{(p,t)} \\ \vdots \\ s^{(l,A)time}_{(p,t)} \end{bmatrix} + z^{(l-1)}_{(p,t)} \tag{4.32}$$

У следећем подслоју се аналогно, користећи посебне параметре, израчунају упити, кључеви и вредности за подслој механизма просторне пажње, потом њихови тежински коефицијенти $\boldsymbol{\alpha}^{(l,a)space}_{(p,t)}$ и на крају опет „пажња“, то јест нове контекстуализоване вредности, $\boldsymbol{s}^{(l,a)space}_{(p,t)}$. Тада се може израчунати и $z''^{(l)}_{(p,t)}$:

$$z''^{(l)}_{(p,t)} = W'_O \begin{bmatrix} s^{(l,1)space}_{(p,t)} \\ \vdots \\ s^{(l,A)space}_{(p,t)} \end{bmatrix} + z'^{(l)}_{(p,t)} \tag{4.33}$$

Коначни излаз из слоја се производи применом нормализационе трансформације (енгл. *layer normalization* - *LN*) на резултат претходног подслоја и прослеђивањем нормализованог сигнала кроз потпуно повезану неуронску мрежу (енгл. *fully connected network* - *FCN*), а такође се резултат претходног подслоја проследи и прескачућом везом:

$$z^{(l)}_{(p,t)} = FCN(LN(z''^{(l)}_{(p,t)})) + z''^{(l)}_{(p,t)} \tag{4.34}$$

Након дванаест овако наслаганих слојева, узима се специјални токен са нулте позиције и врши се класификација видеа:

$$\hat{y} = FCN(LN(z^{(12)}_{(0,0)})) \tag{4.35}$$

5 Експерименти и резултати

Као улазни подаци су коришћени видео клипови фиксних дужина од 16 слика, као што је описано у одељку 2.2 - „Претпроцесирање података“. У случају када је трениран тајмсформер модел, експериментисано је и са коришћењем дужих видео клипова, такође фиксне дужине, од 24 слике.

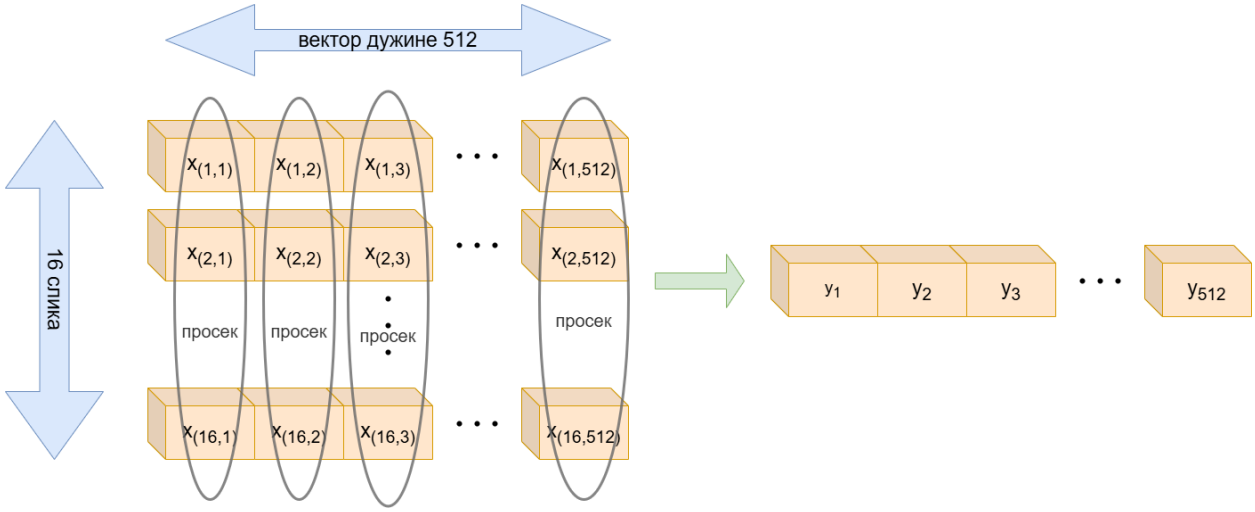
Тренирање модела и обрада података су испрограмирани у програмском језику *Python*, користећи доступне библиотеке као што су *pytorch*, *pandas*, *scikit-learn* и *numpy*. Модели су тренирани користећи „*NVIDIA 3070 RTX Laptop 8GB*“ графичку картицу.

5.1 VGG архитектура

У оквиру стандардне фамилије *VGG* архитектура, испробане су верзије дубина од 11 и 16 слојева. Могле су бити изабране и друге стандардне дубине од 13 или 19 слојева, али су коначно изабране најплића и једна дубља мрежа. Коришћен је претренирани конволутивни део неуронске мреже, који служи за екстракцију атрибута појединачних слика који се могу представити вектором реалних бројева и употребљавају се за потребе класификације слика. Уобичајена стручна терминологија која се користи за именовање конволутивног дела мреже је екстрактор атрибута (енгл. *feature extractor*). Након екстраховања атрибута сваке појединачне слике, израчунат је њихов просек, чиме је добијена векторска репрезентација видео клипа која је истих димензија као и векторска репрезентација појединачне слике (Слика 5.1). У циљу класификације видеа тај вектор се затим прослеђује потпуно повезаном слоју неуронске мреже, који се назива класификатор (енгл. *classifier*). С обзиром да конволутивни део мреже за једну слику величине 224x224 враћа тензор димензија 512x7x7, ради смањења броја параметара, додатно је урађена глобална агрегација максимумом (енгл. global max pooling) чиме је добијен тензор, односно вектор, дужине 512. Упросечавањем 16 таквих вектора опет се добије вектор дужине 512 (Слика 5.1).

Од хиперпараметара испробани су различити класификатори (потпуно повезане неуронске мреже) са једним или два скривена слоја и бројем неурона од 1024, 2048 и 4096 у њима. *Dropout* регуларизација је коришћена само унутар класификатора и за њу су коришћене вредности од 0.20, 0.35 и 0.50. Примењене су архитектуре *VGG11* и *VGG16*, за корак учења коришћене су вредности од 10^{-3}, 10^{-4}, 10^{-5}, 10^{-6} и коришћен је *AdamW* оптимизатор. Током експериментисања са разним варијантама модела, појавили су се

проблеми везани за брзину тренирања. Због тога, са коришћењем „*weight decay*“ хиперпараметра, односно *L2* регуларизацијом, није пуно експериментисано, јер је прилагођавање модела подацима ионако текло исувише споро, а то би само додатно успорило процес учења.

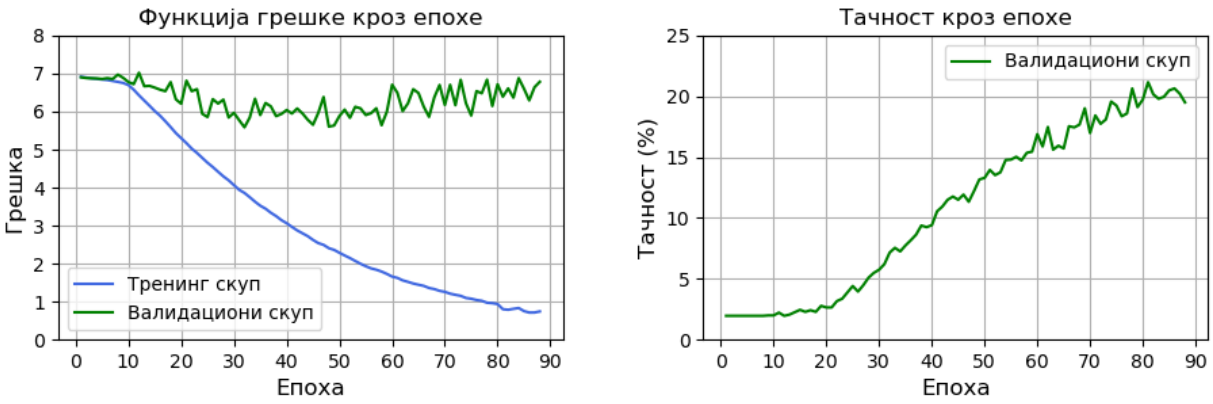


Слика 5.1 - Упросечавање екстрахованих атрибута

У случајевима када је корак учења био превелик или премали, функција грешке над тренинг скупом није опадала знатно, па је за оптималалан корак учења узета вредност од 10^{-5}. Чак и при примени овог оптималног корака, тренинг је и даље текао споро. За *VGG16*, да би се дошло до оптималног модела, било је потребно скоро 100 епоха, а извршавање сваке епохе је захтевало око 37 минута. Такође, у тренинг подскуп је стављен највећи могући број видео клипова у циљу максималног искоришћења меморије графичке картице и добијања стабилнијег градијента при стохастичком градијентном спусту. У зависности од дубине неуронске мреже то је било 2 или 4 видео клипа. Пошто је то доста мали број за симулирање градијента над целокупним скупом података (хиљаде пута мање од свеукупне количине видео клипова), примењено је акумулирање градијената, пре извршавања корака учења. Тако су за рачунање градијената коришћене величине тренинг подскупа до 32. Када је покушано додатно повећање величине тренинг подскупа, у циљу прецизнијих израчунавања градијената, тренинг би текао још спорије, јер се тада по епохи обави мање корака учења.

Најбољи резултат је остварила *VGG* архитектура дубине 16 слојева са потпуно повезаном мрежом од 2 слоја од по 2048 и 1024 неурона и *ReLU* активационим функцијама, са *DropOut* интезитетом од 0.50 у оба потпуно повезана слоја и кораком учења од 10^{-5}. Резултат од 21.18% тачности на валидационом скупу је остварен након 81 епохе (Слика 5.2), а на тест скупу је остварена тачност од 5.07%. Проблем код тренирања *VGG* архитектуре је

што тренинг иде доста споро. На пример за овај најуспешнији модел требало је 2, 3 дана тренирања. С обзиром да је ово базична архитектура овог рада, није постојала потреба за дужим задржавањем на њој.



Слика 5.2 - Функција грешке и тачност током тренирања најуспешнијег *VGG* модела

5.2 *ResNet* архитектура

Следећа архитектура коришћена у овом раду за препознавање знаковног језика је ResNet архитектура дубине 18 слојева. Као почетни параметри екстрактора атрибута, коришћени су параметри претренираног модела на *ImageNet* бази података, слично као код *VGG* архитектуре. Од хиперпараметара, испробане су агрегација просеком, као и агрегација максимумом тензора екстрахованих атрибута. Такође, као и код *VGG* архитектуре, опет је коришћено упросечавање вектора екстрахованих атрибута (Слика 5.1). Испробан је класификатор са једним скривеним слојем од 4096 неурона, као и класификатор без скривених слојева, који се није показао као добра опција. Скривени слој класификатора је коришћен у комбинацији са *DropOut* регуларизацијом која је имала интезитет од 0.35. У почетку корак учења је био већи да би се он временом, када дође до засићености функције грешке и мере тачности, смањивао како би се опрезније конвергирало ка оптимуму. И овде је коришћен *AdamW* оптимизатор, са подразумеваним вредностима хиперпараметара изузев корака учења.

Поред тренирања над клиповима у оригиналној форми, од аугментације података испробано је и обртање слика по хоризонталној оси (енгл. *random horizontal flip*), претпостављајући да неки од клипова нису сачувани у истој перспективи, већ да су неки од

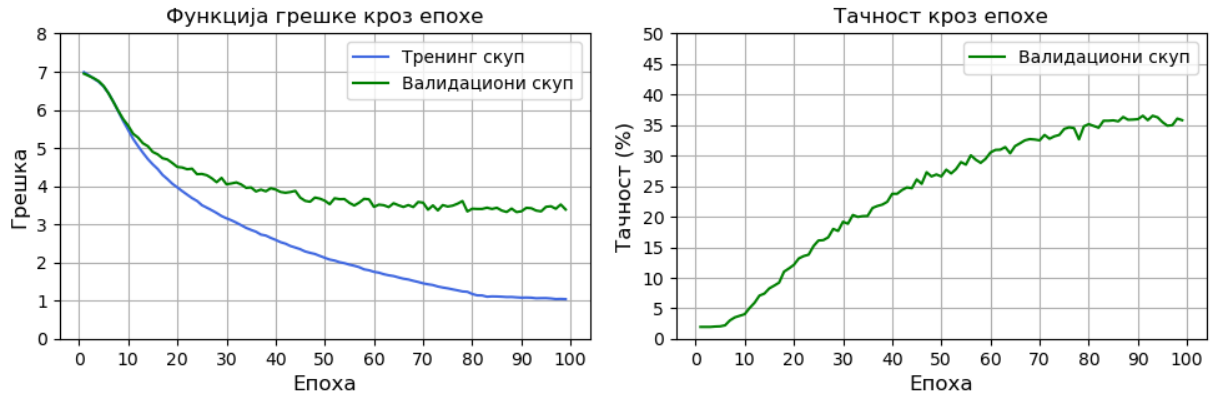
њих снимљени као у одразу огледала (Слика 5.3), те да ће то допринети тренирању бољег модела и бољим резултатима.

Слика 5.3 – Аугментација хоризонталним обртањем слике

Водећи се овим, експериментисано је са тренирањем нових модела који користе ову аугментацију података. Они су користили хиперпараметре који су се до тада показали као најбољи – класификатор са једним скривеним слојем од 4096 неурона и агрегацију екстрахованих атрибута просеком. Код једног од нових модела који је користио ову аугментацију, слике су обртане појединачно, свака са вероватноћом од 50%, тако да је за тренинг користио „хибридне“ видео клипове, односно видео клипове у којима су неке од слика обрнуте, док друге то нису. Други модел је користио ову аугментацију, али тако да се обрну или све слике или ниједна, такође са вероватноћом 50%.

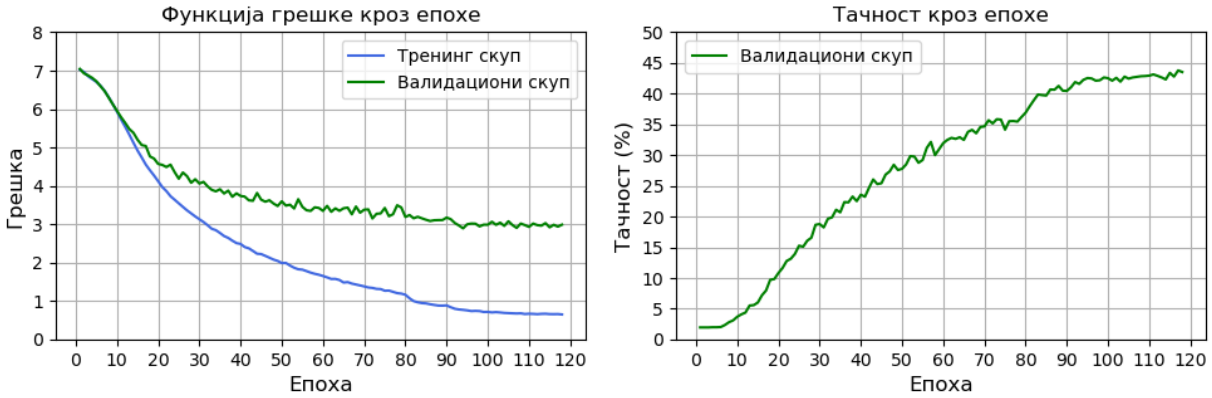
На овај начин су добијена три модела, у зависности да ли је и која врста аугментације била примењена:

1) Дакле, први модел није користио аугментацију обртања слика по хоризонталној оси. Остварио је тачност од 36.54% на валидационом скупу, након 93 епохе (Слика 5.4).



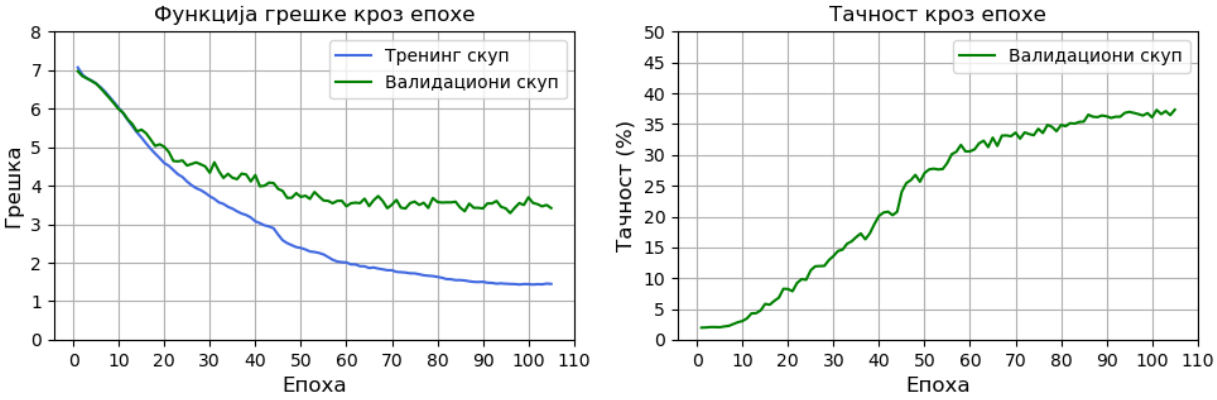
Слика 5.4 - Тренирање и метрике модела који није користио аугментацију хоризонталног обртања слика

2) Други модел је користио аугментацију обртања слика по хоризонталној оси са вероватноћом од 50% за сваку слику појединачно у оквиру видеа. Ово значи да је тренирањем сваке инстанце, модел паралелно учио њену оригиналну форму, као и њену перспективу у огледалу. Овај приступ је допринео најбољем резултату и најбољој конвергенцији, што се може видети из функције грешке (Слика 5.5). Овај модел је остварио резултат од 43.79% тачности на валидационом скупу након 117 епоха.



Слика 5.5 - Тренирање и метрике модела који јесте користио аугментацију хоризонталног обртања, али примењену посебно на појединачне слике

3) Трећи модел је током тренирања користио аугментацију обртања целог видеа одједном, са вероватноћом 0.50, међутим остварио је лошији резултат од модела број два. На валидационом скупу је остварио тачност од 37.35% након 105 епоха (Слика 5.6).



Слика 5.6 - Тренирање и метрике модела који јесте користио аугментацију хоризонталног обртања, али примењивану на све слике видео клипа одједном

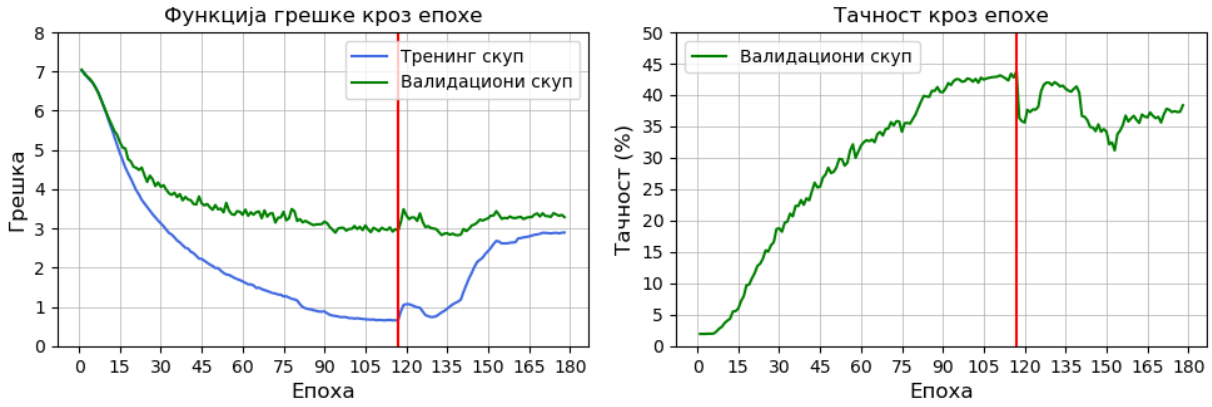
Од наведене три верзије *ResNet* модела, други модел се показао као најуспешнији, те су проверене његове перформансе на тест скупу, где је остварио 16.13% тачности. На тренинг скупу је остварио 92% тачности, а на валидационом скупу 43.79% као што је већ наведено. Међутим, поред знатне разлике између тачности на тренинг, валидационом и тест скупу, примећује се и знатна разлика у функцији грешке између тренинг и валидационог скупа. Ова чињеница је дала мотив за примену додатних мера регуларизације и аугментације у циљу смањења разлике између наведених функција грешке, а да при томе тачност валидационог скупа не буде оштећена и да се по могућству оствари бољи резултат на тест скупу.

Други модел је узет као почетна тачка имплементације нових аугментација и регуларизација. Од аугментација су примењене: *ColorJitter*, *RandomRotation*, *GausBlur*. Регуларизације које су примењене су *L2* регуларицица као и појачан интезитет *DropOut*-а. *ColorJitter* аугментација је употребљена, јер она модификује боје и осветљење слике, док *RandomRotation* ротира слику око њеног центра за неколико степени, а *GausBlur* замућује слику (Слика 5.7). Све ове аугментације на један начин имају ефекат повећања узорка што помаже код преприлагођавања модела. *L2* регуларизацијом се кажњавају велике вредности параметара, што практично значи да се смањује скуп могућих модела који се могу научити, што као ефекат има смањење могућности преприлагођавања подацима. Проблем приликом примене регуларизација је што не постоји јасно дефинисан приступ, већ је једино могуће емпиријски доћи до побољшања модела, односно испробавањем разних комбинација вредности хиперпараметара регуларизације. У овом раду потрошено је неколико дана док је пронађена оптимална комбинација хиперпараметара аугментације и регуларизације. Додатни проблем је што ова метода троши доста времена, јер је сваки пут потребно проћи

кроз одређен број епоха да би се схватио утицај одређеног избора хиперпараметара. До перцепције да ли се ради о великој или малој вредности хиперапараметра је једино могуће доћи експериментисањем и постепеним мењањем вредности параметара.

Слика 5.7 – Аугментације (проширивање) узорка

На слици 5.8 је приложен график дотренирања другог модела, при одабиру најуспешније комбинације хиперпараматера, а до којег се дошло њиховом постепеном модификацијом кроз епохе. Из графика је очито да се разлика између функција грешки тренинг и валидационог скупа знатно смањила, док се тачност на валидационом скупу благо смањила. Ипак модел се и даље показао само делимично успешан у препознавању знакова из тест скупа, где је тачност и даље била знатно мања у односу на валидациони скуп и износила је 13.87%, док је на валидационом скупу остварена тачност од 38.37%.



Слика 5.8 - Дотренирање другог модела уз додавање нових аугментација и регуларизације

Покушај побољшања модела применом додатне аугментације и регуларизације није био успешан, иако је њиме смањена разлика између функције грешке на тренинг и валидационом скупу, јер је остварени резултат на сва три скупа података био лошији.

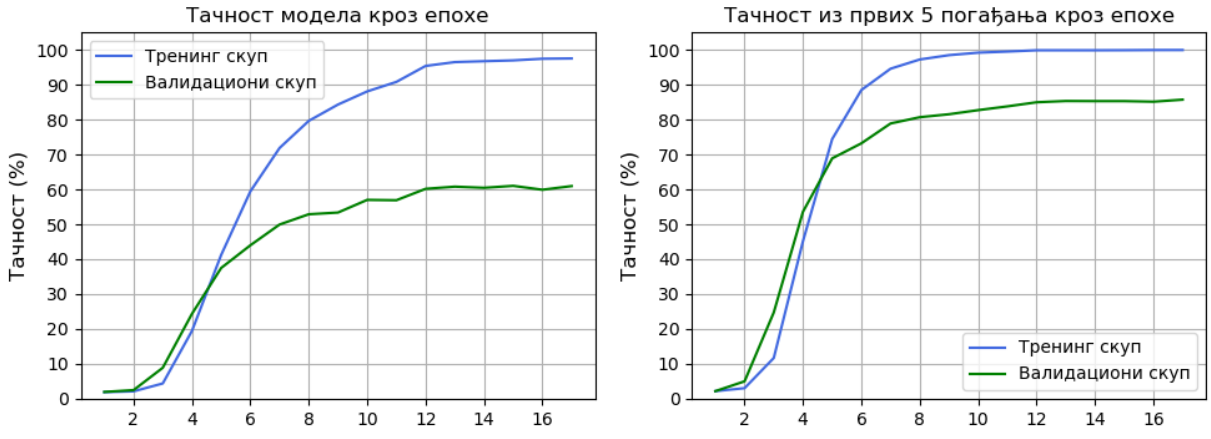
5.3 Тајмсформер архитектура

За тренирање тајмсформер модела, коришћен је програмски код који су аутори тајмсформера [12] поставили на *github* платформу, односно његову копију која је модификована да ради са новом верзијом *pytorch* библиотеке. [13].

Од хиперпараметара, током тренирања, једино је мењан интензитет *L2* регуларизације, док су за остале хиперпараметре коришћени они које су аутори пријавили као најбоље. За интезитет *L2* регуларизације су испробани 10^{-4}, 10^{-3}, $3{\cdot}10^{-3}$, $6{\cdot}10^{-3}$ и 10^{-2}. Интезитети регуларизације од $6{\cdot}10^{-3}$ и 10^{-2} су знатно покварили перформансе на тренинг и валидационом скупу, док су остали интезитети произвели приближно исти резултат на валидационом скупу - од око 60% тачности. Од аугментација је примењено хоризонтално обртање целог видео клипа са вероватноћом 50%, слично као што је урађено код *ResNet* модела број 3, као и „*Jitter Resize*“ аугментација, односно видео клипови се сваки пут скалирају на димензију одабрану по униформној расподели из интервала [256,320] пиксела (висина и ширина слика су исте). При коришћењу „*Jitter Resize*“ аугментације, модел учи и нешто увећане, односно нешто умањене видео клипове, што има за циљ да допринесе бољој генерализацији модела. Затим је из овако скалираног видео клипа, исечен подклип са димензијама слика 224x224 пискела, а његова просторна локација се такође бира случајно (униформно). Потом се овај подклип подели на токене/парчиће од по 16x16 пискела и тада је видео клип спреман за пропуштање кроз тајмсформер модел и његово обучавање.

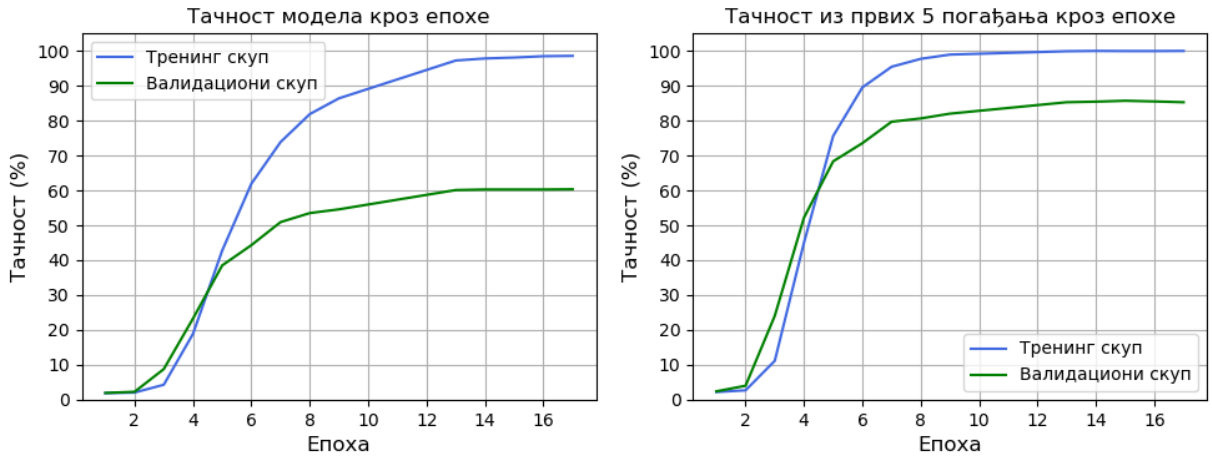
При тестирању, видео клип се увек скалира на 256x256 пискела, а затим се исече централни део видео клипа димензије 224x224 пиксела.

Најбољи модел је користио интезитет *L2* регуларизације од 10^{-3}. Остварио је тачност од 60.95% на валидационом скупу (Слика 5.9). На тест скупу је остварио тачност од 37.45%.



Слика 5.9 – Тачност трансформер модела током процеса тренирања

Истрениран је и модел који користи као улазне податке видео клипове од 24 слике, уместо 16, са интезитетом *L2* регуларизације од $5 \cdot 10^{-2}$. Иако се интуитивно очекивала већа тачност од овог модела, он није остварио бољи резултат. Остварио је 60.33% тачности на валидационом скупу, док је на тесту остварио 37.06% (Слика 5.10).



Слика 5.10 – Тачност трансформер модела који користи видео клипове дужине 24 слике током процеса тренирања

5.4 Поређење резултата

У следећој табели представљени су сумирани резултати свих архитектура:

	***VGG*_16 (16 слика)**		***ResNet*_18 (16 слика)**		***TimeSformer* (16 слика)**		***TimeSformer* (24 слике)**	
	Валид.	Тест	Валид.	Тест	Валид.	Тест	Валид.	Тест
Тачност из првог покушаја	21.18%	5.07%	43.79%	16.13%	60.95%	**37.45%**	60.33%	37.06%
Тачност из првих 5 погађања	47.55%	13.97%	70.39%	33.95%	85.72%	65.29%	85.26%	**65.59%**

Табела 5.1 – Резултати модела

Постоје два очита разлога за овакве разлике између резултата на тренинг и валидационом скупу. Први је релативно мали узорак, с обзиром да у свакој класи гестикулација имамо само 20 видео клипова. Други разлог је што је тест скуп креиран тако да укључи демонстраторе који су извели најмање гестикулација, тако да се већина њих уопште не налази у тренинг скупу. Аутори су овим желели да подигну кредибилитет оцене перформансе модела у реалном свету, а што је став који је подржан и прихваћен у овом раду. Да се претпоставити да би са порастом броја видео клипова, као и демонстратора, прецизност на тест скупу порасла.

За разлику од тајмсформера, *VGG* и *ResNet* нису учили компликоване везе између слика на различитим временским позицијама, с обзиром да су агрегирали информације о видео клиповима простим упросечавањем екстрахованих атрибута сваке слике. Ипак, можемо закључити да је коришћење прескачућих веза и нормализационих слојева код *ResNet*-а, за разлику од *VGG*-а, допринело бољим резултатима модела, узимајући у обзир да обе архитектуре користе конволуцију за екстракцију атрибута. С друге стране, тајмсформер се базира на механизму интроспективне пажње, која се примењује како у оквиру елемената исте слике, тако и на елементе у различитим временским тренуцима, што омогућава моделу да научи комплексније везе свих елемената видео клипова.

6 Закључак

Примена метода машинског учења у препознавању знаковних језика, осим алтруистичких мотива за помагањем људима којима је потребно превођење знаковног језика, је мотивисана потребом за проналажењем ефектних начина рачунарске обраде динамичких визуелних просторно-временских компоненти у видео клиповима. Ово доприноси бољем теоријско-практичном разумевању ове проблематике, стога и иновацијама у области интеракције и комуникације између човека и машине.

У тренутку када је отпочео рад на овом мастер раду, он је био друга, нама позната, практична примена метода машинског учења над подацима из базе података „*SLOVO*” из руског знаковног језика, након њених оригиналних аутора који су их применили у академском раду [3]. Међутим, треба приметити да упркос одличним подацима, методологија у њиховом раду је донекле спорна, јер при тренирању модела и њиховој евалуацији нису користили узорак података подељен на тренинг, валидациони и тест скуп, односно користили су само тренинг и тест скуп, што су оправдали малим узорком по класи. Стога, не може се искључити претпоставка да су њихови најуспешнији модели и резултати вероватно пристрасни према тест скупу због индиректног прилагођавања њему. Наиме, хиперпараметри модела су подешени тако да имају што боље резултате на тест скупу података. У поређењу са овим радом они су користили различите варијанте архитектура и на други начин су претпроцесирали податке. Резултати модела на валидационом скупу у овом раду су слични са резултатима модела које су пријавили аутори базе података „SLOVO”, што је повезано с тим што су тест скуп додатно користили као валидациони.

Остварени резултати у овом раду су у складу са почетним очекивањима, а то је да ће се тајмсформер архитектура показати боље од *ResNet* архитектуре, док ће *VGG* показати најлошије резултате. Ово је у складу и са самом хронологијом еволуције ових модела.

Треба напоменути и то да су резултати условљени самом количином доступних података. На пример, база података „*SLOVO*”, упркос томе што је оцењена високим оценама на сајту *Kaggle* и упркос томе што садржи преко 20 хиљада видео клипова, има тек по 20 видео клипова од сваког знака, што је релативно мали узорак по класи. Вероватно је било могуће направити модел са бољом тачношћу на овој бази података, уз додатне софистицираније методе обраде података и израде модела, међутим један од циљева овог

рада је била примена принципа једноставности који би био добра полазна основа за будуће комплексније надоградње модела.

У овом раду је доста времена посвећено објашњењима сваке од три архитектуре, а посебна пажња је посвећена опису трансформер/тајмсформер архитектуре с обзиром да се ради о напредној и новој методологији машинског учења која уједно представља револуционарни помак у целокупној области вештачке интелигенције.

Са појавом дубоког учења, употреба техника машинског учења у области рачунарског вида, у ужем, као и у области интеракције човека и машине у ширем смислу, се убрзано развијају. Извесно је да ће у будућности вештачка интелигенција и интеракција између човека и машине доживети велика унапређења, због даље еволуције и побољшања у области дубоког учења и обраде информација у реалном времену.

Библиографија

[1] "World Report on Hearing", https://www.who.int/publications/i/item/9789240020481, World Health Organization, United Nations 2021.

[2] Thad Eugene Starner; „Visual Recognition of American Sign Language Using Hidden Markov Models" - Massachusetts Institute of Technology, Cambridge MA, June 1991.

[3] Alexander Kapitanov, Karina Kvanchiani, Alexander Nagaev, Elizaveta Petrova; „SLOVO: Russian Sign Language Database" - SaluteDevices, Russia, 2023.

[4] G. Cybenko; „Approximation by Superpositions of a Sigmoidal Function" - Mathematics of Control, Signals, and Systems, 1989.

[5] Srikanth Tammina; „Transfer learning using VGG-16 with Deep Convolutional Neural Network for Classifying Images" - International Journal of Scientific and Research Publications · October 2019

[6] Урош Стегић; „Аутоматска жанровска класификација песама техникама дубоког учења" – Мастер рад, Математички факултет, Универзитет у Београду, 2023.

[7] Karen Simonyan & Andrew Zisserman; „Very deep convolutional networks for large-scale image recognition" - Visual Geometry Group, Department of Engineering Science, University of Oxford, 2015.

[8] Sergey Ioffe, Christian Szegedy; "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift" – Google, March 2015.

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun; "Deep Residual Learning for Image Recognition" - Microsoft Research, December 2015.

[10] Dzmitry Bahdanau, KyungHyun Cho, Yoshua Bengio; „Neural machine translation by jointly learning to align and translate" - Published as a conference paper at ICLR, 2015.

[11] Ashish Vaswani (et. al); „Attention Is All You Need" – Google, 2017.

[12] Gedas Bertasius, Heng Wang, Lorenzo Torresani; „Is Space-Time Attention All You Need for Video Understanding?" – ICML, 2021.

[13] Fork of „Official pytorch implementation of Timesformer architecture" that works with new version of pytorch - https://github.com/wanchichen/TimeSformer