



УНИВЕРЗИТЕТ У НОВОМ САДУ
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА У
НОВОМ САДУ



Анализа детекције емоција на лицима применом конволутивних неуронских мрежа

Завршни рад
Основне академске студије

кандидат

Срђан Протић ИН 51/2019

ментор

др Мирна Капетина, ванредни професор

Нови Сад, Октобар 2023



УНИВЕРЗИТЕТ У НОВОМ САДУ • ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА

21000 НОВИ САД, Трг Доситеја Обрадовића 6

КЉУЧНА ДОКУМЕНТАЦИЈСКА ИНФОРМАЦИЈА

Редни број, РБР:			
Идентификациони број, ИБР:			
Тип документације, ТД:	Монографска документација		
Тип записа, ТЗ:	Текстуални штампани материјал		
Врста рада, ВР:	Дипломски рад		
Аутор, АУ:	Срђан Протић		
Ментор, МН:	др Мирна Капетина, ванредни професор		
Наслов рада, НР:	Анализа детекције емоција на лицима применом конволутивних неуронских мрежа		
Језик публикације, ЈП:	Српски/ћирилица		
Језик извода, ЈИ:	Српски/енглески		
Земља публикавања, ЗП:	Република Србија		
Уже географско подручје, УГП:	АП Војводина		
Година, ГО:	2023.		
Издавач, ИЗ:	Ауторски репринт		
Место и адреса, МА:	Нови Сад, Трг Доситеја Обрадовића 6		
Физички опис рада, ФО: (поглавља/страна/цитата/табела/слика/графика/прилога)	5/43/0/7/17/0/0		
Научна област, НО:	Електротехника и рачунарство		
Научна дисциплина, НД:	Примењене рачунарске науке и информатика		
Предметна одредница/Кључне речи, ПО:	Машинско учење, емоција, неуронске мреже, класификација		
УДК			
Чува се, ЧУ:	У библиотеци Факултета техничких наука, Нови Сад		
Важна напомена, ВН:			
Извод, ИЗ:	У овом раду је описана и истражена примена конволутивних неуронских мрежа у комбинацији са оптимизационим алгоритмима за прецизну детекцију емоција на лицима људи на основу садржаја датог на сликама.		
Датум прихватања теме, ДП:	05.10.2023.		
Датум одбране, ДО:	16.10.2023.		
Чланови комисије, КО:	Председник:	др Зоран Јеличић, редовни професор	Потпис ментора
	Члан:	др Милан Рапаић, редовни професор	
	Члан, ментор:	др Мирна Капетина, ванредни професор	

Образац Q2.НА.04-05 - Издање 1




UNIVERSITY OF NOVI SAD • FACULTY OF TECHNICAL SCIENCES

21000 NOVI SAD, Trg Dositeja Obradovića 6

KEY WORDS DOCUMENTATION

Accession number, ANO :								
Identification number, INO :								
Document type, DT :	Monographic publication							
Type of record, TR :	Textual printed material							
Contents code, CC :	Bachelor Thesis							
Author, AU :	Srđan Protić							
Mentor, MN :	Mirna Kapetina, Associate Professor, Ph.D							
Title, TI :	Analysis of emotion detection on faces using convolutional neural networks							
Language of text, LT :	Serbian/Cyrillic							
Language of abstract, LA :	Serbian							
Country of publication, CP :	Republic of Serbia							
Locality of publication, LP :	AP Vojvodina							
Publication year, PY :	2023.							
Publisher, PB :	Author's reprint							
Publication place, PP :	Novi Sad, Dositeja Obradovića sq. 6							
Physical description, PD : (chapters/pages/ref./tables/pictures/graphs/appendixes)	5/43/0/7/17/0/0							
Scientific field, SF :	Electrical and computer engineering							
Scientific discipline, SD :	Applied computer science							
Subject/Key words, S/KW :	Machine learning, emotion, neural networks, classification							
UC								
Holding data, HD :	The Library of Faculty of Technical Sciences, Novi Sad							
Note, N :								
Abstract, AB :	This paper explores and describes the utilization of convolutional neural networks in combination with optimization algorithms for accurate emotion detection on human faces within image content.							
Accepted by the Scientific Board on, ASB :	05.10.2023.							
Defended on, DE :	16.10.2023.							
Defended Board, DB :	<table> <tr> <td>President:</td> <td>Zoran Jeličić, Full Professor, Ph.D</td> <td rowspan="3">Menthor's sign</td> </tr> <tr> <td>Member:</td> <td>Milan Rapać, Full Professor, Ph.D</td> </tr> <tr> <td>Member, Mentor:</td> <td>Mirna Kapetina, Associate Professor, Ph.D</td> </tr> </table>	President:	Zoran Jeličić, Full Professor, Ph.D	Menthor's sign	Member:	Milan Rapać, Full Professor, Ph.D	Member, Mentor:	Mirna Kapetina, Associate Professor, Ph.D
President:	Zoran Jeličić, Full Professor, Ph.D	Menthor's sign						
Member:	Milan Rapać, Full Professor, Ph.D							
Member, Mentor:	Mirna Kapetina, Associate Professor, Ph.D							

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

(Податке уноси предметни наставник - ментор)

Врста студија:	<input checked="" type="checkbox"/> Основне академске студије <input type="checkbox"/> Основне струковне студије
Студијски програм:	Информациони инжењеринг
Руководилац студијског програма:	др Славица Кордић, ванредни професор

Студент:	Срђан Протић	Број индекса:	IN51/2019
Област:	Електротехничко и рачунарско инжењерство		
Ментор:	др Мирна Капетина, ванредни професор		

НА ОСНОВУ ПОДНЕТЕ ПРИЈАВЕ, ПРИЛОЖЕНЕ ДОКУМЕНТАЦИЈЕ И ОДРЕДБИ СТАТУТА ФАКУЛТЕТА ИЗДАЈЕ СЕ ЗАДАТАК ЗА ДИПЛОМСКИ (Bachelor) РАД, СА СЛЕДЕЋИМ ЕЛЕМЕНТИМА:

- проблем – тема рада;
- начин решавања проблема и начин практичне провере резултата рада, ако је таква провера неопходна;
- литература

НАСЛОВ ДИПЛОМСКОГ (“BACHELOR”) РАДА:

Анализа детекције емоција на лицима применом конволутивних неуронских мрежа

ТЕКСТ ЗАДАТКА:

- Проучити емоције, начин израза на лицима.
- Проучити основне концепте конволутивних неуронских мрежа и функција активације.
- Проучити принципе оптимизационих алгоритама који се користе.
- Имплементирати модел над различитим архитектурама конволутивних неуронских мрежа и упоредити добијене резултате.
- Анализирати имплементирано решење.

Руководилац студијског програма:	Ментор рада:

Примерак за: ☐ - Студента; ☐ - Ментора

Садржај

Садржај	5
Списак скраћеница.....	6
Списак слика	7
Списак табела	8
Списак блок-кодова	9
1. Увод.....	10
2. Теоријске основе анализе емоције на лицима.....	11
2.1. Детаљна анализа емоционалних израза на лицима и њихов значај у разумевању емоција.....	11
2.2. Преглед досадашњих приступа и техника у анализи емоционалних израза.....	12
3. Методологија детекције емоција на лицима људи коришћењем CNN и оптимизационих алгоритама	13
3.1. CNN	13
3.1.1. Операција конволуције	13
3.1.2. Архитектура CNN	14
3.1.3. Имплементирана CNN	18
3.1.3. Прилагођавање хиперпараметара	27
2.2. Оптимизациони алгоритми	29
4. Резултати.....	32
4.1. Презентација резултата	32
4.1.1 Резултати пре подешавања хиперпараметара	32
4.1.2 Резултати након подешавања хиперпараметара (<i>engl. fine-tuning</i>)	35
4.2. Класификациони извештај	38
5. Закључак	39
Биографија	40
Референце	41

Списак скраћеница

Скраћеница	Значење
CNN	<i>Convolutional Neural Networks</i> – тип неуронских мрежа који користимо
AI	<i>Artificial Intelligence</i> - вјештачка интелигенција
ReLU	<i>Rectified Linear Unit</i> – тип функције активације који користимо
FER	<i>Facial Expression Recognition</i> – тип скупа података
SGD	<i>Stochastic Gradient Descent</i> - оптимизатор
SGDM	<i>Stochastic Gradient Descent with Momentum</i> – проширени SGD оптимизатор

Списак слика

Слика 1 - Приказ емоционалних израза на лицима људи	11
Слика 2 - Пример операције конволуције	14
Слика 3 - Први корак матричног множења вредности пиксела и кернела	14
Слика 4 - Други корак матричног множења вредности пиксела и кернела	14
Слика 5 - Трећи корак матричног множења вредности пиксела и кернела	15
Слика 6 - Четврти корак матричног множења вредности пиксела и кернела	15
Слика 7 - Врсте сажимања	15
Слика 8 - ReLU функција активације	16
Слика 9 - Пример CNN са фокусом на део класификације и улогу потпуно повезаног слоја.	16
Слика 10 - Расподела броја узорака по емоцијама	18
Слика 11 - Трострука подела модела података	19
Слика 12 - 'DenseNet-121' архитектура	21
Слика 13 - 'AlexNet' архитектура	25
Слика 14 - Резултат тачности 'DenseNet-121' пре подешавања хиперпараметара	33
Слика 15 - Резултат тачности 'AlexNet' пре подешавања хиперпараметара	34
Слика 16 - Резултат тачности 'DenseNet-121' након подешавања хиперпараметара	35
Слика 17 - Резултат тачности 'AlexNet' након подешавања хиперпараметара	37

Списак табела

Табела 1 - ' <i>DenseNet-121</i> ' архитектура.....	21
Табела 2 - Резултати ' <i>DenseNet-121</i> ' пре подешавања хиперпараметара	32
Табела 3 - Резултати ' <i>AlexNet</i> ' пре подешавања хиперпараметара.....	33
Табела 4 – Резултати ' <i>DenseNet-121</i> ' након подешавања хиперпараметара	35
Табела 5 - Резултати ' <i>AlexNet</i> ' након подешавања хиперпараметара	36
Табела 6 - Резултати класификационог извештаја <i>DenseNet-121</i> архитектуре	37
Табела 7 - Резултати класификационог извештаја ' <i>AlexNet</i> ' архитектуре	38

Списак блок-кодова

Блок-код 1 - Имплементација функције за генерисање нових слика	20
Блок-код 2 - Имплементација методе за обраду слика	20
Блок-код 3 - Имплементација функције 'feature_extractor'	22
Блок-код 4 - Имплементација функције 'classifier'	23
Блок-код 5 - Имплементација функције 'final_model'	23
Блок-код 6 - Имплементација функције 'define_compile_model'	24
Блок-код 7 - Имплементација функције 'earlyStoppingCallback'	24
Блок-код 8 - Имплементација функције за покретање обуке модела	24
Блок-код 9 - Имплементација конволуционих слојева у 'AlexNet' архитектури	26
Блок-код 10 - Имплементација потпуно повезаних слојева	26
Блок-код 11 - Имплементација функције 'compile'	27
Блок-код 12 - Имплементација функције за покретање обуке модела	27
Блок-код 13 - Имплементација методе 'compile_model' у 'DenseNet121' након подешавања параметара	28
Блок-код 14 - Имплементација функције за покретање обуке DenseNet121' након подешавања параметара	28
Блок-код 15 - Имплементација методе 'compile_model' у 'AlexNet' након подешавања параметара	28
Блок-код 16 - Имплементација функције за покретање обуке DenseNet121' након подешавања параметара	28

1. Увод

Осећајност, емоција и афективност су посебна димензија људске егзистенције која је донедавно пркосила задовољавајућој научној и стручној концептуализацији. Да би се схватила важност емоција довољно је замислити на шта би личио живот када људи не би имали осећања. Све док човек не разуме своја осећања он неће моћи разумети самога себе, као ни другог човека [39].

Емоције су кључни аспект људског искуства и комуникације. Разумевање емоција има дубок утицај на људски живот, то укључује социјалне интеракције, доношење одлука и психолошко благостање. У ери дигиталне технологије, способност рачунара да препознају и интерпретирају емоције постаје све важнија, отварајући врата за широк спектар примена. То обухвата разне области од персонализованих искустава корисника у е-трговини до унапређених система за аутоматско препознавање и тумачење људских емоција у реалном времену.

Један од најизазовнијих аспеката у анализи емоција је детекција емоција на лицима људи. Лица су изразито богати делови људског тела. Разумевање ових емоционалних израза на лицу има потенцијал да дубоко обогати различите области, укључујући рачунарску графику, медицинску дијагностику, едукацију и забаву.

Овај рад истражује примену конволутивних неуронских мрежа (CNN) у комбинацији са напредним оптимизационим алгоритмима за прецизну детекцију емоција на лицима људи у садржају датом на сликама. Ова истраживања покушавају да превазиђу изазове у препознавању и класификацији емоционалних израза лица, пружајући дубље разумевање како дубоко учење (*engl. Deep Learning*) и оптимизациони приступи могу побољшати тачност и ефикасност овог процеса.

У наредним поглављима ћемо да размотримо теоријски оквир за детекцију емоција на лицима, анализирати кључне компоненте модела CNN и оптимизационих алгоритама, а затим приказати експерименталне резултате и дискутовати о могућим будућим истраживањима у овој области.

2. Теоријске основе анализе емоције на лицима

2.1. Детаљна анализа емоционалних израза на лицима и њихов значај у разумевању емоција

Емоције су саставни део људског искуства, играјући кључну улогу у свакодневном животу. Представљају нам сложена и динамичка ментална стања која се изражавају на различите начине, укључујући вербалну комуникацију, невербалне сигнале и изразе лица. Истраживање емоционалних израза има дубок корен у психологији и социологији, пружајући нам могућност да спознамо на који начин се људи изражавају и како да тумачимо њихове изразе лица. Разумевање саме природе емоција захтева анализу физиолошких, изражајних и субјективних аспеката [39].

- **Физиолошка компонента:** Ова компонента припрема организам за адекватно реаговање на емоционалне стимулансе. Физиолошке промене, као што су убрзан рад срца, промене у дисању и промене у нивоу хормона, често прате емотивне реакције.
- **Изражајна компонента:** Емоције се изражавају кроз невербалну комуникацију. Фацијални изрази, телесни говор, тон гласа и гестикулације су начини на које људи изражавају своје емоције према другима.
- **Субјективна компонента:** Субјективна компонента се односи на наш лични доживљај емоција. Осећамо емоције кроз субјективно искуство, које може бити пријатно или непријатно.

Класификација емоционалних израза је процес препознавања и интерпретација одређених карактеристика на човјековом лицу које указују на присутност одређене емоције [39]. Поред основних емоција које ће се проучавати у овом задатку, постоје и сложенија емоционална стања као што су афект, расположење и сентимент. Афективни тон се односи на доживљај пријатности и непријатности и саставни је део сваког осећања. У овом задатку ради се о класификацији на седам различитих емоција, а то су емоције приказане на слици 1:

- **Срећа** (*eng. happy*)
- **Изненађење** (*eng. surprised*)
- **Равнодушност** (*eng. neutral*)
- **Страх** (*eng. fear*)
- **Туга** (*eng. sad*)
- **Љутња** (*eng. angry*)
- **Згроженост** (*eng. disgusted*)



Слика 1. Приказ емоционалних израза на лицима људи

Ово поглавље поставља основу за разумевање важности израза лица у контексту истраживања. Наредна поглавља бавиће се питањем како дубоко учење и оптимизациони алгоритми могу унапредити препознавање ових израза и допринети разумевању емоција.

2.2. Преглед досадашњих приступа и техника у анализи емоционалних израза

Анализа емоционалних израза је област која је привукла велику пажњу истраживача из различитих дисциплина, као што су психологија, рачунарство и инжењеринг. Досадашњи приступи и технике у овој области развијали су се током деценија како би боље разумели и квантификовали емоционалне изразе, омогућавајући увид у психолошко стање човека [8, 23].

- **Класификација емоционалних израза.** То је заправо препознавање карактеристика које одају црте на човековом лицу. Разумевање на лицима је попут дешифровања тајног језика. Истраживачи су се потрудили да развију посебне рачунарске програме који аутоматски могу препознати шта особа осећа, односно која је основна емоција краси. На овај начин развијена и унапређена технологија помаже нам на много начина, од бољег управљања стресом путем система препоруке до прављења апликација које служе за забаву како би побољшали тренутно стање ако је то потребно. Овај пасус покушава једноставније да објасни значај класификације емоционалних израза и како то утиче на технологију и наше свакодневне интеракције са рачунаром.
- **Универзалност и инструменти.** Пионир у истраживању емоционалних израза, Пол Екман (*engl. Paul Ekman*) је идентификовао основне емоције које су препознатљиве широм света. Његова истраживања су указала на чињеницу да се основне емоције изражавају на сличан начин у различитим културама, што је имало дубоке импликације за разумевање емоционалних сигнала на глобалном нивоу [8]. Инструменти су високо напредна технологија, компјутерска визија (*engl. Computer Vision*), сензори за праћење микро-фацијалних карактеристика, АИ и алгоритми дубоког учења. Развој мобилних апликација је омогућио експанзију доступности ових техника широм света.
- **Примена у различитим доменима.** Досадашњи приступи анализи нису само технички импресивни већ имају и дубок утицај на многе сфере нашег живота.
 - **Психологија.** У психологији нам анализа емоционалних израза игра кључну улогу у разумевању и третману емоционалних поремећеја. Истраживачи и терапеути користе ове анализе како би боље приступили сваком појединцу. То побољшава развој персонализованих терапија и стратегија за побољшање менталног здравља.
 - **Маркетинг.** У маркетингу, разумевање емоционалних реакција потрошача је од суштинског значаја за развој сваког бизниса (*engl. bussines*). Помаже стручњацима у овој области да разумеју како људи реагују на производе, рекламе, снижења. Ово омогућава прилагођавање маркетиншких стратегија како би се компанија боље повезала са циљном публиком и створила већи принос.
 - **Образовање.** У образовању, појава паметних учионица које могу користити камере за праћење израза лица студената током предавања, која у многome може значити за планирање саме наставе. Ако се детектује да већи број студената изражава тугу или конфузију, професор може брже реаговати, пружити додатна објашњења или понудити подршку. Крајњи циљ је боље разумевање градива и смањења стреса код студената.

У целини, досадашњи напредак је испреплетен са многим аспектима људског живота, пружајући нам боље разумевање и отварајући врата за разноврсне примене које доносе позитивне промене у свакодневном искуству.

3. Методологија детекције емоција на лицима људи коришћењем CNN и оптимизационих алгоритама

У овом кључном поглављу разматраћемо све релевентне аспекте методологије која стоји иза наше анализе детекције емоција на лицима људи. Ово укључује дубље разумевање CNN, детаљну имплементацију нашег модела, избор одговарајућег скупа података и оптимизационе алгоритме.

3.1. CNN

У овом делу се описује детаљно објашњење како CNN функционишу, изглед њихове архитектура и улога у анализи емоционалних израза на лицима. Представљена је операција конволуције и мотивација за креирање оваквог типа неуронских мрежа [1, 2]. Такође, извршено је истраживање како овај тип неуронских мрежа аутоматски екстрактује карактеристике из слика, укључујући обрасце који указују на емоције.

Конволуционе неуронске мреже (CNN) представљају посебну категорију дубоких неуронских мрежа и користе операцију конволуције да би обрадиле улазне податке.

3.1.1. Операција конволуције

Дефинисана и формули (1) је као интеграл производа две функције (f , g) при чему је једна функција обрнута у времену у односу на другу и померена за одређену вредност [12]:

$$h(t) = (f * g)(t) = \int_{-\infty}^{\infty} f(t - \tau)g(\tau) d\tau \quad (1)$$

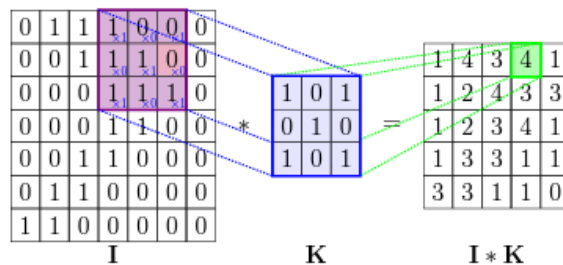
У случају конволуције два коначна дискретна сигнала, дефинишемо је на следећи начин у формули (2), узимајући у обзир да су нам дата два сигнала $f[k]$ и $g[k]$, $k \in \mathbb{Z}$:

$$h[k] = \sum_n f[k - n]g[n] \quad (2)$$

Конволуција нам говори о повезаности две функције. Другим речима, показује колико су сличне односно различите две функције на различитим релативним позицијама. Операција конволуције може бити проширена на вишедимензионалне сигнале [12, 28]. Конволуцију два коначна дискретна дводимензионална сигнала $I[i, j]$, $K[i, j]$ дефинишемо у формули (3):

$$H[i, j] = \sum_m \sum_n I[i - m, j - n]K[m, n] \quad (3)$$

Најчешће је први сигнал наш улаз у систем (слика) док је други сигнал матрица, чије су димензије 3×3 или 4×4 , такав сигнал се назива кернел (*engl. kernel*). На слици 2 можемо да видимо операцију конволуције за дводимензионални улазни сигнал и кернел димензија 3×3 .



Слика 2. Пример операције конволуције

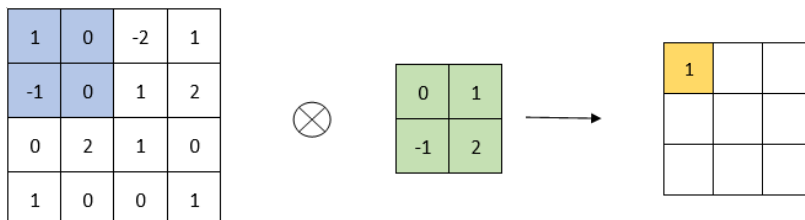
2.1.2. Архитектура CNN

Архитектура CNN је описана кроз слојеве[1]:

- **Конволуциони слој**

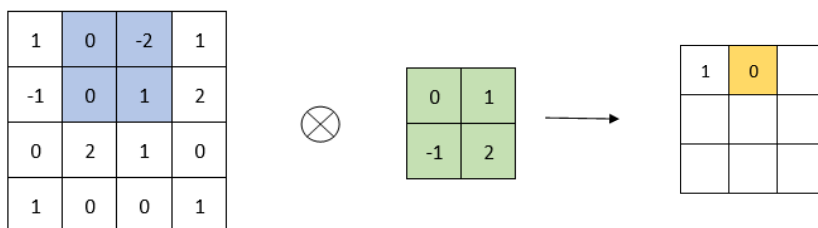
Најважнији слој је конволуциони слој (*engl. convolutional layer*). Он садржи скуп конволуционих филтара (тзв. кернела). Слика која се налази на улазу система, изражена је путем N -димензионалне метрике, конволуирана са овим филтерима даје тражени излаз [12]. Спроводи се итеративни процес тако што се филтар поставља на почетак слике, затим се путем матричног множења, множе одговарајуће вредности пиксела са одговарајућим вредностима у филтру које их преклапају. Тако помножене вредности се сабирају и уписују у излазну матрицу, а филтар се помера за одређени корак и тај процес се понавља све док не дође крај слике. На сликама 3, 4, 5 и 6 су приказана прва 4 корака овог процеса.

Корак 1:



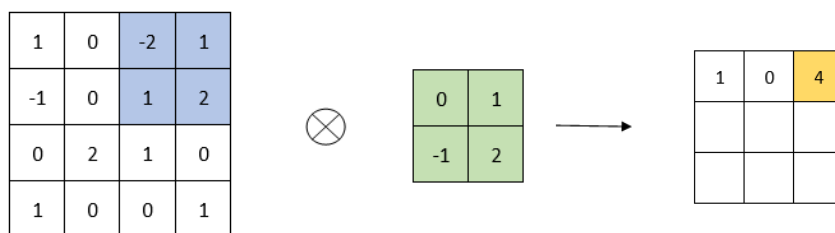
Слика 3. Први корак матричног множења вредности пиксела и кернела

Корак 2:



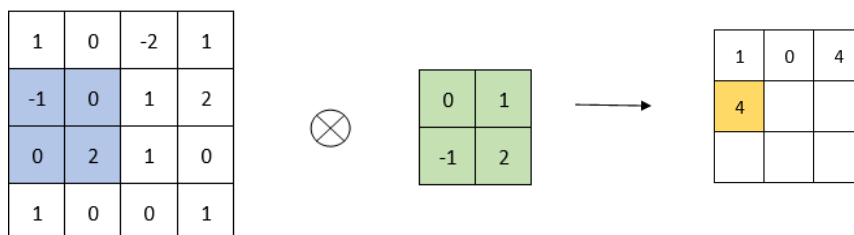
Слика 4. Други корак матричног множења вредности пиксела и кернела

Корак 3:



Слика 5. Трећи корак матричног множења вредности пиксела и кернела

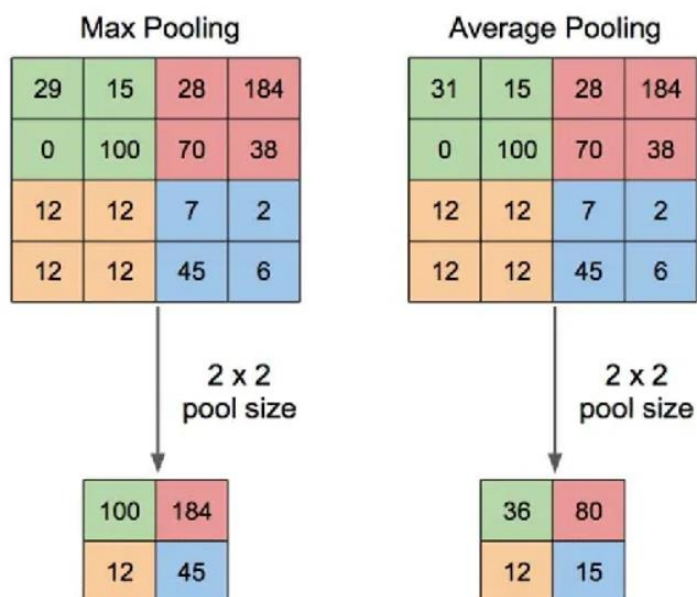
Корак 4:



Слика 6. Четврти корак матричног множења вредности пиксела и кернела

- Слој сажимања

Слој сажимања (*engl. Pooling Layer*) је слој који се јавља периодично [1, 12]. Главна улога је смањивање димензија слике, при чему се овим смањује број параметара и самим тим се олакшавају рачунске операције. Постоји више врста сажимања, а оне које се најчешће користе су сажимање максимумом (*engl. max-pooling*) и сажимање средњим вредностима (*engl. average-pooling*). На слици 7 су графички представљене врсте сажимања [14].



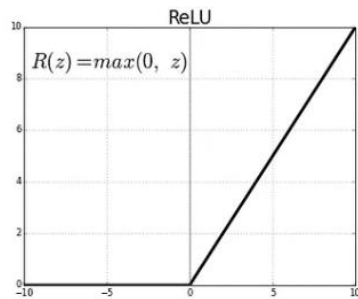
Слика 7. Врсте сажимања

- **Функција активације**

Мапирање улаза на излаз је главна функција свих типова функције активације за све неуронске мреже. Улазна вредност је одређена израчунавањем пондерисане (*engl. weighted*) суме заједно са пристрасношћу (*engl. bias*) [30]. Односно, функција активација доноси одлуку о томе да ли ће покренути неурон у односу на одређени улаз, креирањем коресподентног излаза [28].

Особина нелинеарности у активационим слојевима одговара томе да је мапирање улаза на излаз нелинеарно и оставља могућност решавања изузетно компликованих ствари. Функција активације такође мора имати могућност разликовања, што је изузетно значајна карактеристика, јер омогућава да се грешка пропагације уназад користи за обуку мреже. Тип активационе функције који се користи у овом истраживању је **ReLU** [12].

ReLU представља најчешће коришћену функцију активације у конволуционим неуронским мрежама. Конвертује комплетне вредности улаза у позитивне бројеве. Главна предност коришћења је мала рачунска сложеност и боље ширење градијента (*engl. gradient propagation*) са мањим проблемима око дивергирања градијента у поређењу са осталим функцијама. На слици 8 видимо изглед **ReLU** функције.



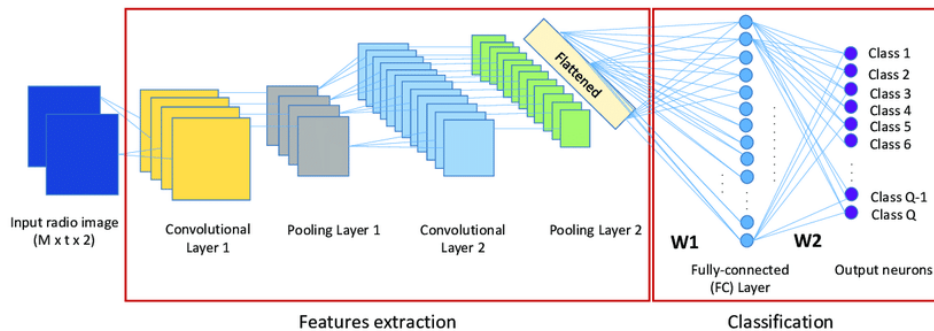
$$f(x)_{ReLU} = \max(0, x)$$

$$\frac{\partial g}{\partial x} = \begin{cases} 0, & \text{if } x \leq 0 \\ 1, & \text{if } x > 0 \end{cases}$$

Слика 8. ReLU функција активације

- **Потпуно повезани слој**

Потпуно повезани слој (*engl. Fully connected layer*) је лоциран на самом крају сваке CNN. Унутар овог слоја сваки неурон је повезан са свим неуронима претходног слоја, зато се и назива потпуно повезан (*engl. fully connected*) **FC** приступ [2]. Користи се као класификатор CNN. Следи основну методу конвенционалне вишеслојне перцептронске неуронске мреже. Улаз **FC** слоја долази из последњег слоја за сажимање или неког конволуционог слоја. Овај улаз је у облику вектора, који се креира од мапа обележја (*engl. feature maps*) након изравњавања (*engl. flattening*). На слици 9 је представљен пример CNN и како изгледа потпуно повезани слој:



Слика 9. Пример CNN са фокусом на део класификације и улогу потпуно повезаног слоја

- **Функција цене**

Функција цене (*engl. Loss function*) користе се за израчунавање предвиђене грешке створене у узорцима за обуку у нашем CNN моделу [11]. Ова грешка открива разлику између стварног и предвиђеног резултата. Функција цене има два параметра, први је процењени излаз CNN (представља предвиђање), а други параметар је стварни излаз. Постоји неколико типова:

- 1) **Унакрсна ентропија** (*engl. Cross-entropy*) се обично користи за мерење перформанси CNN модела [7]. Такође се назива и логаритамска функција цене. Излаз је вероватноћа $p \in \{0,1\}$. Додатно, може да се користи као замена функције цене квадратне грешке у проблемима класификације више класа. Дефинисан је у формули (4):

$$p_i = \frac{e^{a_i}}{\sum_{k=1}^N e_k^a} \quad (4)$$

У формули изнад, израз представља ненормализовани излаз из претходног слоја, док N представља број неурона у излазном слоју. Коначно, математички приказ функције цене унакрсне ентропије је приказан у формули (5):

$$H(p, y) = - \sum_{i=1}^N y_i \log(p_i) \quad (5)$$

- 2) **Еуклидска** (*engl. Euclidean*) је функција цене која је широко распрострањена у регресионим проблемима [5]. Познатија је и као функција средње квадратне грешке. Математички израз дефинисан у формули (6):

$$H(p, y) = \frac{1}{2N} \sum_{i=1}^N (p_i - y_i)^2 \quad (6)$$

- 3) **Хингеова** се често користи за проблеме повезане са бинарном класификацијом [3]. Овај проблем је уско везан са класификатором максималне маргине који представља основу машина на бази вектора носача (*engl. Support Vector Machines*). Маchine на бази вектора носача користе хингеову функцију цене док оптимизатор покушава да максимизује маргину и појача линеарну раздвојивост класа. Математички израз дефинисан у формули (7):

$$H(p, y) = \sum_{i=1}^N \max(0, m - (2y_i - 1)p_i) \quad (7)$$

- **Регуларизација**

За решавање овог проблема користили смо два типа регуларизације:

- 1) **Иступање** (*engl. dropout*) је широко коришћена техника за генерализацију [12, 33]. Током сваке епохе тренинга, неурони се насумичко испуштају. При томе, моћ селекције обележја се подједнако распоређује на целу групу неурона, тако што приморава модел да научи различита независна обележја. Током процеса обуке, испуштени неурон неће бити део алгорита прогације уназад или напред.

- 2) **Нормализација серије** (*engl. batch normalization*) осигурава добре перформансе активације излаза [12, 29]. Овај начин прати јединичну Гаусову расподелу [19] Одузимање средње вредности и дељење стандардном девијацијом нормализоваће излаз на сваком слоју. Иако се ово сматра задатком пред обраде података на сваком слоју у мрежи, могуће га је разликовати и интегрисати са другим мрежама. Поред тога користи се за смањење унутрашњег померања коваријансе у слојевима активације.

Предности коришћења овог типа нормализације су следеће:

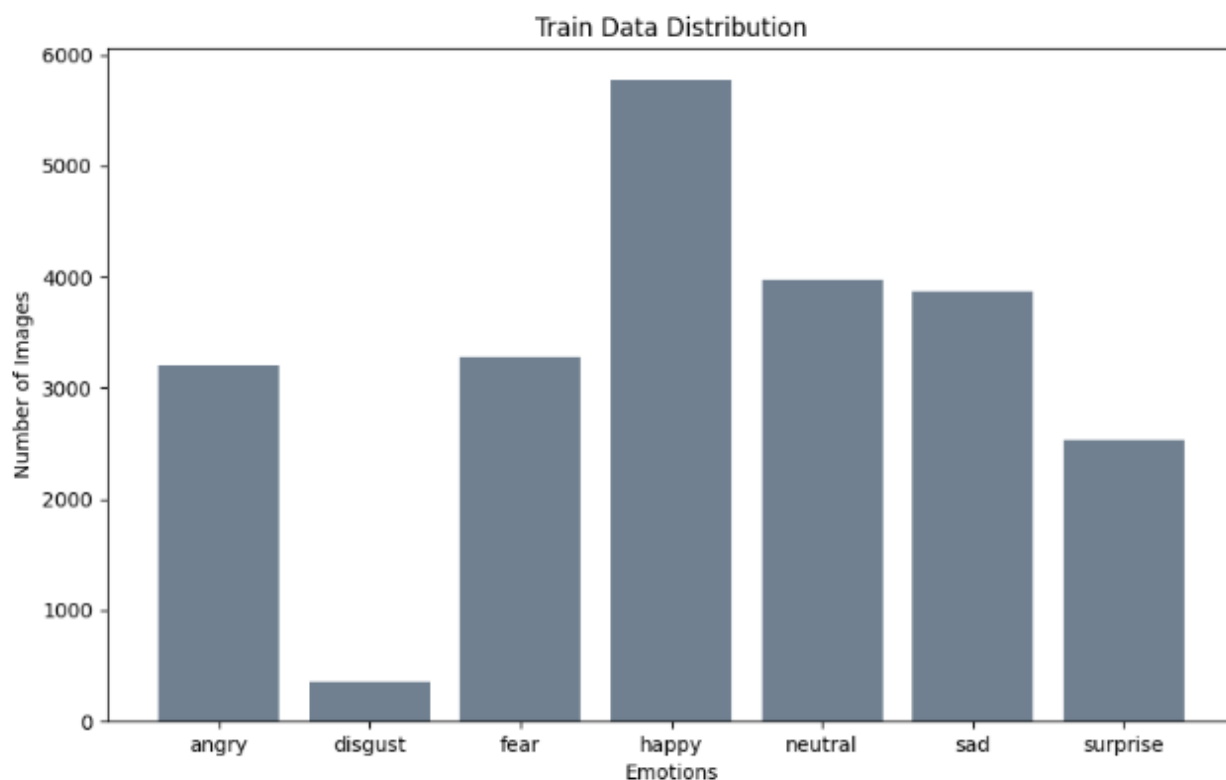
- Може ефикасно контролисати лошу иницијализацију тежина
- Значајно смањује време потребно за конвергенцију мреже што ће бити изузетно корисно за велике скупове података
- Бори се да смањи зависност између тренинг скупа и хиперпараметара

2.1.3. Имплементирана CNN

- **Скуп података**

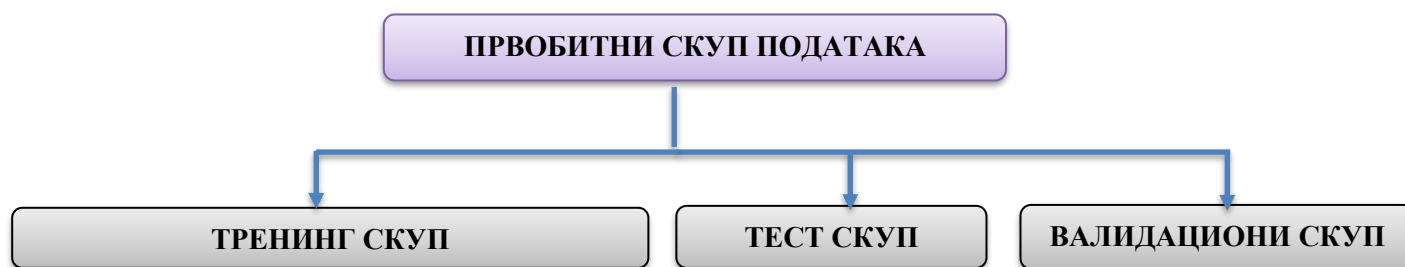
Скуп података за детекцију емоција на лицима, често означавањем као FER, представља колекцију слика лица људи са различитим емоционалним изразима. Овај скуп има значајну примену у области компјутерске визије (*engl. Computer Vision*) и дубоког учења (*engl. Deep Learning*), јер омогућава развој и обуку модела за аутоматско препознавање емоција на лицима појединаца [40].

Скуп података садржи 35685 примера слика лица димензија 48x48 пиксела, подељених на скуп за обуку и тест скуп. Сlike су категорисане на основу емоција приказаних у изразима лица (срећа, равнодушност, туга, љутња, изненађење, гађење, страх) [40]. Свака слика је означена одговарајућом емоцијом, што је кључно за развој модела за препознавање емоција. На слици 10 је приказан број узорака за све емоције појединачно.



Слика 10. Расподела броја узорака по емоцијама

За поделу скупа података користимо **модел троструке поделе** односно раздвојили смо скуп података на тренинг скуп, тест скуп и валидациони скуп [12, 13]. На слици 11 се налази пример поделе скупа података.



Слика 11. Трострука подела модела података

Ова пракса је коришћена из неколико следећих разлога:

1. **Оцена модела:** Како бисмо измерили перформансе свог модела за детекцију емоција или било којег другог задатка потребно је имати независни скуп података за тестирање. Тест скуп користимо за оцену тачности модела и процену његове способности да генерализује научено на тренинг скупу и валидационом скупу на нове, до сад некоришћене податке.
 2. **Избегавање преобучавања (*engl. overfitting*):** Како бисмо избегли претерану обученост мреже, процес учења модела на тренинг скупу и праћење његове тачности на валидационом скупу, омогућава нам да анализирамо како се модел понаша на подацима које није видео током тренинга. Ако модел има добре резултате на тренинг и валидационом скупу, али лоше на тест скупу, то може указивати на претерану обученост нашег модела.
 3. **Подешавање хиперпараметра:** Приликом развоја модела, експериментисали смо са различитим хиперпараметрима (стопа учења, број слојева и неуронских јединица у мрежи, одабир оптимизатора). Валидациони скуп смо користили за процену перформанси различитих конфигурација модела и одабир најбољег сета хиперпараметара.
 4. **Контрола:** Раздвајањем скупа података на тренинг, валидациони и тест скуп, осигурали смо да ниједна информација из тест скупа не цури у процес обуке над тренинг скупом. То значи да модел неће бити у могућности да „запамти“ резултате из тест скупа и лажно повећа тачност.
- **Тренинг скуп** се користи за тренирање модела. Модел се прилагођава тренинг подацима како би научио како да препозна емоција на лицима у складу са постављеним циљем.
 - **Валидациони скуп** се користи током процеса обуке да би се пратиле перформансе модела на независном скупу података. То омогућава експериментисање са хиперпараметрима и рану детекцију проблема преобучености мреже.
 - **Тест скуп** се користи као коначна евалуација модела након што је обука завршена. Модел се тестира на овом скупу да би се проценила његова тачност у реалним условима.

Кораци које користимо приликом обуке:

1. Подела расположивог скупа на тест, тренинг и валидациони скуп
2. Селекција модела
3. Процена грешке на тест скупу

Помоћу класе ‘*ImageDataGenerator*’ [29, 36] омогућавамо генерисање нових слика применом различитих трансформација на оригиналне слике односно користимо аугментацију што је приказано у блок-коду 1. Аугментација је техника која се користи у дубоком учењу како би се обогатили скупови података за обуку тако да модел има бољу генерализацију и има способност да ради са различитим варијацијама слика [1, 12]. Ове методе су примењене над сва три скупа за обуку [15].

- *rescale* = *1./255* – Нормализација пиксела тако да буду у опсегу [], доста помаже у конвергенцији модела
- *rotation_range* = *40* – Насумична ротација слика у опсегу -40 до +40
- *width_shift_range* = *0.1* - Насумично хоризонтално померање слика за највише 10% ширине слике
- *height_shift_range* = *0.1* - Насумично вертикално померање слика за највише 10% ширине слике
- *zoom_range* = *0.2* – Насумично померање слике за највише 20%
- *validation_split* = *0.2* – Насумично узимање 20% података који ће чинити валидациони скуп

```
train_datagen = ImageDataGenerator(  
    rescale=1./255,  
    rotation_range=40,  
    width_shift_range=0.1,  
    height_shift_range=0.1,  
    zoom_range=0.2,  
    fill_mode='nearest',  
    validation_split= 0.2  
)
```

Блок-код 1. Имплементација функције за генерисање нових слика

‘*train_datagen.flow_from_directory()*’ блок-коду 2 је метода која користи ‘*ImageDataGenerator*’ за обраду слика из одређеног директоријума [29, 36], конкретно у овом случају из директоријума са тест подацима ‘*train_dir*’ и генерисање минијатурних скупова (*engl. mini-batch*) које се могу користити за обуку модела [15].

- *directory* – Путања до директоријума са сликама за обуку
- *target_size* – Величина циљних слика у овом случају 48x48
- *batch_size* – Величина сваког мини скупа који ће бити генерисан
- *color_mode* – Тип боје слика у овом случају ‘*rgb*’
- *class_mode* – Тип задатка, у нашем случају значи да се користи вишекласна класификација
- *subset* – Подскуп података који желимо да користимо
- *seed* – Семе за генерисање насумичних трансформација, за репродукцију исте трансформације

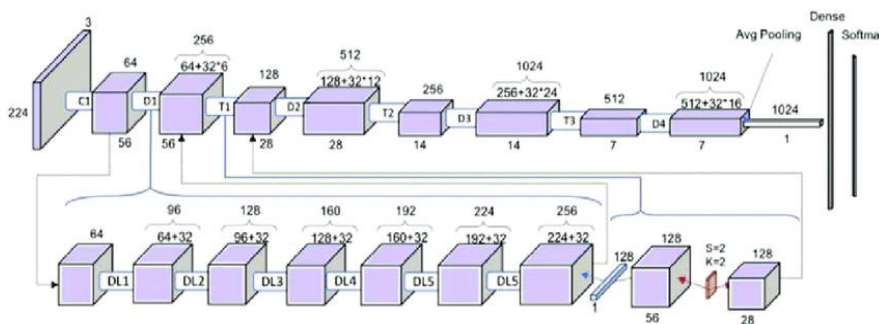
```
train_generator = train_datagen.flow_from_directory(  
    directory=train_dir,  
    target_size=(48, 48),  
    batch_size=128,  
    color_mode="rgb",  
    class_mode="categorical",  
    subset="training",  
    seed=12  
)
```

Блок-код 2.. Имплементација методе за обраду слика

- 'DenseNet121' архитектура

'DenseNet-121' је архитектура неуронске мреже коју користимо за решавање нашег проблема. Ова архитектура је део породице 'DenseNet' (engl. *Densely Connected Convolutional Networks*) која се одликује густо повезаним слојевима и то је заправо главна карактеристика ове архитектуре [10, 17]. Пример архитектуре је приказан на слици 12 [10].

Уместо повезивања улаза и излаза слојева, као што је случај у традиционалним приступима CNN, овај тип повезује сваки слој са сваким претходним слојем уназад. Ово значи да излаз сваког слоја служи као улаз за све будуће слојеве, чинећи мрежу густо повезаном. Такав приступ нам је омогућио боље искоришћавање информација из различитих нивоа слојева и помогао у превазилажењу проблема умирућег градијента [19].



Слика 12. 'DenseNet-121' архитектура

Архитектура приказана табеларно се налази у табели 1:

Layers	Output size	Operations
Convolution	112x112	7x7 convolution, stride 2
Pooling	56x56	3x3, max pooling, stride 2
Dense block (1)	56x56	$\begin{pmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{pmatrix} \times 6$
Transition layer (1)	56x56 28x28	1x1 convolution 2x2 avg. pooling, stride 2
Dense block (2)	28x28	$\begin{pmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{pmatrix} \times 12$
Transition layer (2)	28x28 14x14	1x1 convolution 2x2 avg. pooling, stride 2
Dense block (3)	14x14	$\begin{pmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{pmatrix} \times 24$
Transition layer (3)	14x14 7x7	1x1 convolution 2x2 avg. pooling, stride 2
Dense block (4)	7x7	$\begin{pmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{pmatrix} \times 16$
Classifier layer	1x1	7x7 global avg. pooling Full connection (1000) Softmax activation

Табела 1. 'DenseNet-121' архитектура

‘*feature_extractor*’ функција приказана у блок-коду 3 [11, 16], гради модел за издвајање карактеристика користећи архитектуру. Користи се ‘*tf.keras.applications.DenseNet121*’ ‘*DenseNet-121*’ модела. Параметри који описују функцију [29, 32]:

- ‘*input_shape = (48, 48, 3)*’ – Поставља очекивани облик улазних података. Улазни подаци представљају тродимензионални тензор 48x48x3 што одговара сликама димензија 48x48 пиксела висине и ширине и 3 канала боја RGB (црвена, зелена и плава).
- ‘*include_top = False*’ – Означава да нећемо користити последњи потпуно повезани слој из разлога што би наши подаци требали тада да буду димензија (224, 224, 3) а због ограничавања ресурса то онемогућујемо.
- ‘*weights = “imagenet”*’ – Користи се тежина претходно обучене ‘*DenseNet-121*’ мреже са ‘*ImageNet*’ скупом података. Ово је корисно из разлога што претходно обучени модел већ садржи научене карактеристике које можемо даље користити.

Као резултат, функција ‘*feature_extractor*’ враћа излаз из ‘*DenseNet-121*’ модела, који ће бити тензор са издвојеним карактеристикама улазних слика. Ове карактеристике даље користимо за решавање проблема класификације.

```
def feature_extractor(inputs):
    feature_extractor = tf.keras.applications.DenseNet121(input_shape=(48, 48, 3),
                                                           include_top=False,
                                                           weights="imagenet")(inputs)

    return feature_extractor
```

Блок-код 3.. Имплементација функције ‘*feature_extractor*’

Функција ‘*classifier*’ је дио модела за класификацију слика приказана у блок-коду 4.

- ‘*x = tf.keras.layers.GlobalAveragePooling2D()(inputs)*’ – Ова линија користи глобално просјечно груписање на улазном слоју и то је техника која смањује димензионалност излаза из претходног слоја тако да се добије један вектор за свако обележје [34]. У нашем случају, то значи да ће за свако обележје у излазу добити просечна вредност свих пиксела тог обележја по целој слици. Помаже нам у смањивању сложености и смањује потребу за великим бројем параметара.
- ‘*x = tf.keras.layers.Dense(batch_size, activation, kernel_regularizer)(x)*’ – Представља нам потпуно повезани (*engl. Dense*) слој гдје параметар ‘*batch_size*’ означава број неурона. Активациона функција коју ћемо користити у свим слојевима осим последњег је ‘*ReLU*’, а у последњем слоју користимо активациону функцију ‘*softmax*’ јер је она уобичајена приликом вишекласне класификације. Користимо L2 – *Ridge* регуларизацију како би се смањила претерана обученост модела и побољшала генерализација.
 - *Ridge* регуларизација [21] функционише тако што се додају казни чланови у функцију губитка (*engl. loss function*) који кажњавају велике апсолутне вредности тежина (*engl. weights*) неуронских веза. Овај додатни члан у функцији губитка представљен је у формули 8 [37]:

$$L2_{loss} = \frac{\lambda}{2} \sum_i \sum_j \theta_{i,j}^2 \quad (8)$$

- `'x = tf.keras.layers.Dropout(rate)(x)'` – Ово је функција у којој искључујемо онолики проценат неурона који је задат за параметар `'rate'` и такође је једна од мера спречавања превелике обучености модела [29, 33].

Коначни излаз овог модела виће вектор вероватноћа за сваку класу, а модел ће се тренирати у циљу минимизације губитка (нпр. коришћење категоријалне унакрсне ентропије) како би научио како правилно класификовати улазне примере у одговарајуће класе. У суштини, у овом делу постоје слојеви за смањење димензионалности, регуларизацију и `'dropout'` метода како би се побољшала регуларизација модела и смањила преобученост током обуке над тренинг скупом.

```
def classifier(inputs):
    x = tf.keras.layers.GlobalAveragePooling2D()(inputs)
    x = tf.keras.layers.Dense(256, activation="relu", kernel_regularizer=tf.keras.regularizers.l2(0.01))(x)
    x = tf.keras.layers.Dropout(0.3)(x)
    x = tf.keras.layers.Dense(1024, activation="relu", kernel_regularizer=tf.keras.regularizers.l2(0.01))(x)
    x = tf.keras.layers.Dropout(0.5)(x)
    x = tf.keras.layers.Dense(512, activation="relu", kernel_regularizer=tf.keras.regularizers.l2(0.01))(x)
    x = tf.keras.layers.Dropout(0.5)(x)
    x = tf.keras.layers.Dense(NUM_CLASSES, activation="softmax", name="classification")(x)

    return x
```

Блок-код 4. Имплементација функције `'classifier'`

Функција `'final_model'` у блок-коду 5 је функција која користи претходно две дефинисане функције `'feature_extractor'` и `'classifier'` како би се креирао коначни модел за класификацију. У суштини она комбинује извлачење обележја помоћу функције `'feature_extractor'` и класификацију тих добијених обележја помоћу функције `'classifier'`.

```
def final_model(inputs):
    densenet_feature_extractor = feature_extractor(inputs)
    classification_output = classifier(densenet_feature_extractor)

    return classification_output
```

Блок-код 5. Имплементација функције `'final_model'`

Функција `'define_compile_model'` у блок-коду 6 укључује дефинисање архитектуре модела, постављање параметара за тренирање и компилацију модела како би се добио модел који је спреман за тестирање и евалуацију [29, 35].

- `'inputs = tf.keras.layers.Input(shape = (48, 48, 3))'` – Прво се дефинише улазни слој модела и поставља очекивани облик улазних података (48, 48, 3) што сугерише на очекивање слике димензија 48x48 пиксела са три канала боја.
- `'classification_output = final_model(inputs)'` – Користи претходно дефинисану функцију `'final_model'` како би се добио излаз модела за класификацију који је у облику вектора са вероватноћама класификације.
- `'model = tf.keras.Model(inputs = inputs, outputs = classification_output)'` – Спецификација улаза и излаза, чиме се дефинише цели модел који укључује све слојеве од улаза до излаза.
- `'model.compile(...)'` – Функција у којој се врши покретање модела, што значи да се постављају параметри за тренинг модела. Кориштен је оптимизациони параметар SGD са брзином учења 0.1. Као функција губитка се користи `'categorical_crossentropy'` уобичајена за проблем вишекласне класификације. Такође дефинисали смо метрику коју ћемо пратити током обуке, а то је прецизност - `'accuracy'`.


```
def define_compile_model():
    inputs = tf.keras.layers.Input(shape=(48, 48, 3))
    classification_output = final_model(inputs)
    model = tf.keras.Model(inputs=inputs, outputs=classification_output)

    model.compile(optimizer=tf.keras.optimizers.SGD(0.1),
                  loss='categorical_crossentropy',
                  metrics=['accuracy'])

    return model
```

Блок-код 6. Имплементација функције 'define_compile_model'

Главни задатак тренирање модела за класификацију слика, приказано у блок-коду 8, укључује и примену раног заустављања (*engl. early stopping*) као позивајуће функције током обуке над тренинг скупом [31, 38].

- Функција 'earlyStoppingCallback' у блок-коду 7 је техника раног заустављања која се користи како би се спречила претерана обученост мреже и побољшала ефикасност тренирања [29, 31]. Аргументи ове функције су:
 - 'monitor='val_loss'' – Метрика која се прати током тренинга како би се успешно одлучило када засуставити обуку, у нашем случају се прати губитак на валидационом скупу
 - 'patience = EARLY_STOPPING_CRITERIA' – Означава број епоха без побољшања односно без смањења губитка на валидационом скупу прије него што се обука модела заустави. Овај хиперпараметар нам заправо одређује колико епоха ће се чекати без побољшања пре заустављања.
 - 'verbose' = 1 – Овај аргумент контролише колико ће информација бити исписано током обуке и вредност 1 говори да ће поруке исписивати током обуке.
 - 'restore_best_weights = True' – Након раног засунављања вратиће се тежине које су постигле најбољи резултат на валидационом скупу. Ово помаже да се модел пребрзо заустави односно пребрзо конвергира и да се сачува најбоља верзија модела.

```
earlyStoppingCallback = tf.keras.callbacks.EarlyStopping(
    monitor='val_loss',
    patience=EARLY_STOPPING_CRITERIA,
    verbose=1,
    restore_best_weights=True
)
```

Блок-код 7. Имплементација функције 'earlyStoppingCallback'

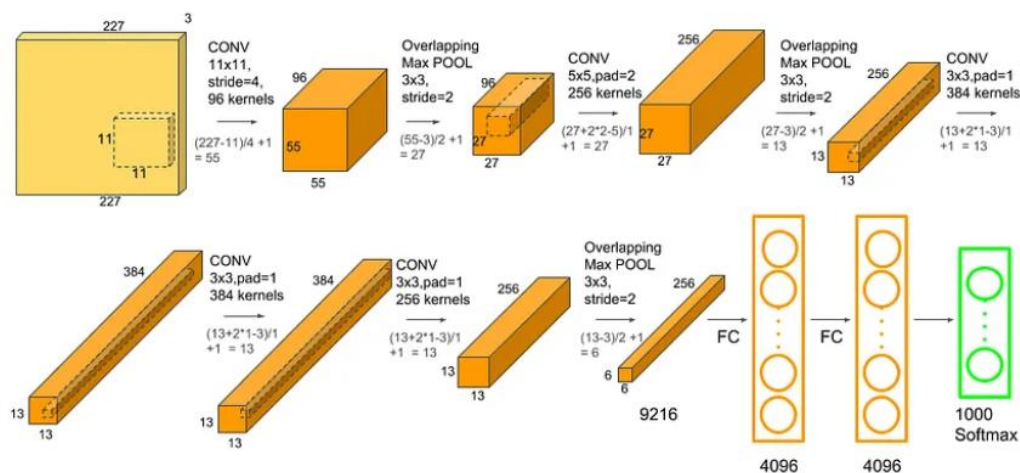
```
history = model.fit(
    x=train_generator,
    epochs=EPOCHS,
    validation_data=validation_generator,
    callbacks=[earlyStoppingCallback]
)
```

Блок-код 8. Имплементација функције за покретање обуке модела

- ‘AlexNet’ архитектура

‘AlexNet’ је архитектура која је стекла велику популарност након што је постигла изванредне резултате на такмичењу ‘ImageNet Large Scale Visual Recognition Challenge’ које је одржано 2012. године. Такви резултати су постигнути коришћењем ‘ReLU’ активационе функције, а уз то је смањена могућност појаве проблема умирућег градијента.

На слици 13 можемо видети да се модел састоји од укупно 8 слојева: 5 слојева са комбинацијом максималног издвајања обележја (*engl. Max Pooling*), праћених са 3 потпуно повезана слоја [18]. Први је тип архитектуре у коју су уведени узастопни конволутивни слојеви.



Слика 13. ‘AlexNet’ архитектура

Структура конволуционих слојева:

- ‘Conv2D’ – Функција која користи мале филтере (матрице) које се померају преко слике и врше математичку операцију звану конволуција. Ова операција омогућава мрежи да научи различите карактеристике слика, као што су ивице, облици и текстуре.
- ‘MaxPooling’ – максимално издвајање обележја је слој у CNN који се користи за смањење димензионалности и броја параметара. Ради тако што дели слику на мање области и за сваку област враћа максималну вредност. Ово помаже у очувању најважнијих особина слике и смањењу рачунарског оптерећења.
- ‘BatchNormalization’ – Нормализација серије (*engl. Batch*) је техника која се користи за нормализацију излаза слојева у неуронској мрежи. Ово помаже у убрзању тренинга и побољшању стабилности мреже. Нормализацијом се скалирају излазни слојеви тако да имају средњу вредност близу нуле и стандардну девијацију близу јединице.
- ‘Dropout’ – Техника регуларизације која се користи током обуке. Идеја је да се насумично деактивирају одређени неурони током сваког пролаза кроз мрежу. Ово помаже у спречавању претеране обучености тако што присиљава мрежу да не зависи превише од одређених неурона и тако постиже бољу генерализацију.

У блок-коду 9 можемо видети имплементацију конволуционих слојева.

```

# Prvi konvolucioni sloj
x = Conv2D(96, (11, 11), strides=(4, 4), activation='relu')(input_layer)
x = MaxPooling2D((3, 3), strides=(2, 2))(x)
x = BatchNormalization()(x)

# Drugi konvolucioni sloj
x = Conv2D(256, (5, 5), activation='relu', padding='same')(x)
x = MaxPooling2D((2, 2), strides=(2, 2))(x)
x = BatchNormalization()(x)

# Treci konvolucioni sloj
x = Conv2D(384, (3, 3), activation='relu', padding='same')(x)
x = Conv2D(384, (3, 3), activation='relu', padding='same')(x)
x = Conv2D(256, (3, 3), activation='relu', padding='same')(x)
x = MaxPooling2D((3, 3), strides=(2, 2))(x)
x = BatchNormalization()(x)

# Cetvrti konvolucioni sloj
x = Conv2D(256, (3, 3), activation='relu', padding='same')(x)
x = MaxPooling2D((3, 3), strides=(2, 2))(x)
x = BatchNormalization()(x)

# Peti konvolucioni sloj
x = Conv2D(256, (3, 3), activation='relu', padding='same')(x)
x = MaxPooling2D((3, 3), strides=(2, 2))(x)
x = BatchNormalization()(x)

```

Блок-код 9. Имплементација конволуционих слојева у 'AlexNet' архитектури

Структура потпуно повезаних слојева:

- *'Flatten'* – Слој за развлачење се користи за претварање вишедимензионалних података у једнодимензионални формат. Овај слој се обично користи као прелаз између конволутивних слојева и потпуно повезаних слојева. На пример, ако су дати вишедимензионални излази из конволутивних слојева, слој за развлачење ће те излазе претворити у једноставан низ података који се може користити као улаз за потпуно повезане слојева
- *'Dense'* – Потпуно повезани слој је тип слоја у неуронској мрежи у којем су сви неурони у слоју повезани са свим неуронима из претходног слоја. Ови слојеви се често користе за класификацију, и у ту сврху их и ми користимо. Они успешно доносе одлуке на основу карактеристика које су научене у претходним слојевима.

Ови слојеви, приказани у блок-коду 10, представљају основе градивне блокове кориштене архитектуре и закључак је да слој за развлачење (*engl. flatten*) омогућава прелазак из вишедимензионалних података у равни низ, док потпуно повезани слојеви (*engl. dense*) обаваљају класификацију и доношење одлука на основу тих података.

```

# Potpuno povezani slojevi
x = Flatten()(x)
x = Dense(4096, activation='relu')(x)
x = Dropout(0.5)(x)
x = Dense(4096, activation='relu')(x)
x = Dropout(0.5)(x)
output_layer = Dense(7, activation='softmax')(x)

```

Блок-код 10. Имплементација потпуно повезаних слојева

- `'alexnet_model.compile(...)` – Функција у блок-коду 11, у којој се врши покретање модела, што значи да се постављају параметри за тренинг модела. Кориштен је оптимизациони параметар ADAM са брзином учења 0.001. Као функција губитка се користи `'categorical_crossentropy'` уобичајена за проблем вишекласне класификације. Такође дефинисали смо метрику коју ћемо пратити током обуке, а то је прецизност - `'accuracy'`. Приказано блок коду ...

```
alexnet_model = Model(inputs=input_layer, outputs=output_layer)

alexnet_model.compile(optimizer=tf.keras.optimizers.Adam(lr=0.001),
                      loss='categorical_crossentropy',
                      metrics=['accuracy'])
```

Блок-код 11. Имплементација функције `'compile'`

Главни задатак тренирање модела за класификацију слика, приказано у блок-коду 12, укључује и примену раног заустављања (*engl. early stopping*) као позивајуће функције током обуке над тренинг скупом [31, 38].

- Функција `'earlyStoppingCallback'` у блок-коду 12 је техника раног заустављања која се користи како би се спречила претерана обученост мреже и побољшала ефикасност тренирања [29, 31]. Описана је у претходној архитектури на страници 24.

```
earlyStoppingCallback = tf.keras.callbacks.EarlyStopping(
    monitor='val_loss',
    patience=EARLY_STOPPING_CRITERIA,
    verbose=1,
    restore_best_weights=True
)

history = alexnet_model.fit(
    x=train_generator,
    epochs=20,
    validation_data=validation_generator,
    callbacks=[earlyStoppingCallback]
)
```

Блок-код 12. Имплементација функције за покретање обуке модела

3.1.3. Прилагођавање хиперпараметара

• **'DenseNet121' и 'AlexNet' архитектура**

Прилагођавање (*engl. fine-tuning*) хиперпараметара је важан корак у тренирању модела дубоког учења како би се постигле што боље перформансе, приказано у блок-кодовима 9, 10, 11 и 12 [25, 26]. Осим брзине учења коју мењамо, функција губитка остаје иста, а други хиперпараметри који су прилагођени за побољшање перформанси:

- **Број епоха (*engl. epochs*):** Број епоха одређује колико пута ће се модел тренирати на целом скупу података. Превише епохе може довести до преобучености мреже. На основу перформанси на валидационом скупу наш број епоха је смањен са 30 на 20.
- **Величина скупа (*engl. batch size*):** Величина групе одређује колико примера се обрађује у свакој итерацији током обуке. Величину групе смо повећали са 64 на 128.

- **trainable = True** : Сваки слој има атрибут који контролише хоће ли се параметри тог слоја оптимизовати током тренинга или ће остати фиксни. У овом случају, параметри тог слоја ће се оптимизовати током тренинга.

```
model.layers[1].trainable = True

model.compile(
    optimizer=tf.keras.optimizers.SGD(0.001, 0.9),
    loss='categorical_crossentropy',
    metrics=['accuracy']
)
```

Блок-код 13. Имплементација методе 'compile_model' у 'DenseNet121' након подешавања параметара

```
history_ = model.fit(
    x=train_generator,
    epochs=FINE_TUNING_EPOCHS,
    validation_data=validation_generator,
    callbacks=[earlyStoppingCallback],
    use_multiprocessing=True
)
```

Блок-код 14. Имплементација функције за покретање обуке DenseNet121' након подешавања параметара

```
alexnet_model.compile(
    optimizer=tf.keras.optimizers.Adam(0.01),
    loss='categorical_crossentropy',
    metrics=['accuracy']
)
```

Блок-код 15. Имплементација методе 'compile_model' у 'AlexNet' након подешавања параметара

```
history_ = alexnet_model.fit(
    x=train_generator,
    epochs=FINE_TUNING_EPOCHS,
    validation_data=validation_generator,
    callbacks=[earlyStoppingCallback],
)
```

Блок-код 16. Имплементација функције за покретање обуке DenseNet121' након подешавања параметара

2.2. Оптимизациони алгоритми

За оптимизацију неуронске мреже, кориштени су оптимизациони алгоритми Стохастички градијентни спуст и ADAM (*Adaptive Moment Estimation*) [25].

- **Стохастички градијентни спуст - SGD**

Ради на основу идеје оптимизације грешке или функционала кроз мале подскупове података, уместо претраге целог скупа података [27]. Широко је распрострањена примена у машинском учењу. Циљ је минимизација ризика, а представља се математичким изразом у формули 9:

$$\min_{x \in \mathbb{R}^d} f(x) = \frac{1}{n} \sum_{i=1}^n \ell(x, q_i) \quad (9)$$

- ℓ - представља функцију цене
- $\{q_i\}, i = 1..n$ – представља тренинг податке
- x – представља тренирајуће параметре нашег модела (нпр. матрица тежина у неуронској мрежи)

У општем случају, SGD може бити описана изразима представљеним у формули 10 и 11:

$$m^k = \beta m^{k-1} + (1 - \beta) g^{-k}, \quad (10)$$

$$x^{k+1} = x^k - \alpha m^k. \quad (11)$$

- $\alpha > 0$ – представља број корака за који се ажурирају параметри модела током сваке итерације оптимизационог алгоритма
- $\beta \in [0,1)$ – коефицијент моментума, контролише количину инерције која се додаје промени параметара у сваком кораку оптимизације
- $m^0 = 0$

Класични SGD оптимизациони алгоритам користи $\beta = 0$ и $m^k = g^{-k}$, где је g^{-k} стохастички градијент функције $f(x)$ за x^k . За побољшање перформанси у пракси, често се користи коефицијент моментума $\beta > 0$ и такав оптимизациони је често назван Стохастички градијент са моментумом SGDM. SGDM је веома популаран за обучавање неуронских мрежа са изузетним успехом и имплементиран је као подразумевани оптимизатор у библиотеци *TensorFlow* [29] коју користимо.

Идеја која стоји иза SGDM потиче од Пољакове (*engl. Polyak*) методе тежине-лопте (*engl. heavy-ball method*) [24] за детерминистичку оптимизацију. Ова метода има доста бржу линеарну конвергентну стопу у односу на градијентни спуст [9, 24].

У дубоком учењу, SGDM се често примењује уз неколико варијација подешавања параметара да би се постигла ефикасност приликом обуке. Најраширенија метода, уједно она која се и користи је правило названо “константа и смањење” (*engl. constant and drop*), где се константна величина корака примењује током дужег периода, а потом се смањује неким константним фактором како бисмо омогућили подешавање параметара (*engl. fine-tuning*) током обуке, док се коефицијент моментума одржава непромењним (у овом случају 0.9) или постепено повећава.

Ову стратегију често називамо SGDM са више етапа (*engl. Multistage SGDM*) и сумирамо је у Алгоритму 1. практично гледано, веома је успешан за обуку великих неуронских мрежа [18, 26] и утврђено је да одговарајуће подешавање параметара доводи до извандредних резултата [18]. Од тада, SGDM постаје све популарнији.

Алгоритам 1

Улаз: скуп података $f(x)$, број етапа n , коефицијент моментума $\{\beta_i\}, i = 1..n \subseteq [0, 1)$, величина корака $\{\alpha_i\}, i = 1..n$, дужина етапе $\{T_i\}, i = 1..n$ за n етапа, иницијализација $x^1 \in \mathbb{R}^d, m^0 = 0$, бројач итерације $k = 1$.

```
1: for  $i = 1, 2 .. n$  do
2:    $\alpha \leftarrow \alpha_i, \beta \leftarrow \beta_i$ 
3:   for  $j = 1, 2 .. T_i$  do
4:     случајно изабери мини скуп (engl. mini batch)  $\zeta^k$  из скупа података униформно
5:      $g^{-k} \leftarrow \nabla_x l(x^k, \zeta^k)$ 
6:      $m^k \leftarrow \beta m^{k-1} + (1 - \beta)g^{-k}$ 
7:      $x^{k+1} \leftarrow x^k - \alpha m^k$ 
8:      $k \leftarrow k + 1$ 
9:   end for
10: end for
11: return  $\tilde{x}$ , које се генерише тако што се прво изабере етапа из  $l \in \{1, 2 .. n\}$ , а затим се насумично изабере  $\tilde{x} \in \{x^{T_1+...+T_{l-1}+1}, x^{T_1+...+T_{l-1}+2}, \dots, x^{T_1+...+T_l}\}$ .
```

• ADAM

‘ADAM’ је популаран оптимизациони алгоритам, за обуку неуронских мрежа. Комбинује предности других оптимизационих алгоритама како би ефикасно конвергирао према минимуму функције губитка. Основна идеја се заснива на комбинација експоненцијалног просека првог и другог момента градијента (средња вредност и варијанса) [24, 27].

Експоненцијални покретни просек првог момента, приказан у формули 12. Први момент се рачуна као покретни просек градијената и представља процену средње вредности градијента:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1)g_t \quad (12)$$

- m_t – процена првог момента за тренутни корак t
- g_t – градијент за тренутни корак t
- β_1 – параметар за експоненцијални покретни просек првог момента (близу 1)

Експоненцијални покретни просек другог момента, приказан у формули 13. Други момент се рачуна као покретни просек квадрата градијената и представља процену варијансе градијената:

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2)g_t^2 \quad (13)$$

- v_t – процена другог момента за тренутни корак t
- g_t – градијент за тренутни корак t
- β_2 – параметар за експоненцијални покретни просек првог момента (близу 1)

Корекција исправке за биас. Почетни кораци алгоритма имају пристрасност према нули због иницијалних вредности m и v . Како би се то исправило користи се ова корекција представљена формулама 14 и 15:

$$\hat{m} = \frac{m_t}{1 - \beta_1^t} \quad (14), \quad \hat{v} = \frac{v_t}{1 - \beta_2^t} \quad (15)$$

- \hat{m} и \hat{v} су кориговане вредности првог и другог момента, респективно
- t је тренутни корак

Корекција корака приказана у формули 16. Корак учења за сваки параметар се прилагођава на основу коригованих вредности првог и другог момента:

$$\Delta\theta_t = \theta_{t-1} - \frac{\alpha \cdot \hat{m}}{\sqrt{\hat{v}} + \epsilon} \quad (16)$$

- $\Delta\theta_t$ – корекција корака за тренутни корак t
- α – стопа учења
- ϵ – мала константа која се додаје како би се избегло дељење са нулом

Алгоритам 2

Улаз: α величина корака, $\beta_1, \beta_2 \in [0, 1)$ параметри за експоненцијални покретни просек, $f(\theta)$ стохастичка циљна функција са параметрима θ ;

Иницијалне вредности: $\theta_0, m_0 \leftarrow 0, v_0 \leftarrow 0, t \leftarrow 0$

```

1: while  $\theta_t$  не конвергира do
2:    $t \leftarrow t + 1$ 
3:    $g_t \leftarrow \nabla_{\theta} f_t(\theta_{t-1})$ 
4:    $m_t \leftarrow \beta_1 m_{t-1} + (1 - \beta_1) g_t$ 
5:    $v_t \leftarrow \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$ 
6:    $\hat{m} \leftarrow m_t / (1 - \beta_1^t)$ 
7:    $\hat{v} \leftarrow v_t / (1 - \beta_2^t)$ 
8:    $\Delta\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \hat{m} / (\sqrt{\hat{v}} + \epsilon)$ 
9: end while
10: return  $\theta_t$ 

```

4. Резултати

4.1. Презентација резултата

4.1.1 Резултати пре подешавања хиперпараметара

Резултати су представљани метрикама перформанси током обуке *DenseNet-121* и *AlexNet* неуронских мрежа [10] над скупом података за класификацију емоција на лицима људи. Резултати су приказани у табелама 2 и 3, а анализа ових резултата је следећа:

- **Редни број епохе:** Ово представља број пута током обуке када су забележени резултати и статистика. На пример број 1 у колони значи да су то статистике и метрика након прве епохе обуке.
- **Време извршавања:** Време потребно за извршавање једне епохе, у случају коришћења *DenseNet-121* архитектуре потребно је 331 секунде до 374 секунди по епохи, а у случају коришћења *AlexNet* потребно је до 134 секунде до 161 секунду по епохи.
- **Функција губитка на тренинг скупу:** Функција губитка је метрика која мери колико добро модел ради на тренинг скупу. Приметно је да се у оба случаја губитак смањује с повећањем броја епоха, што указује на то да модел постаје бољи у прилагођавању тренинг подацима.
- **Тачност на тренинг скупу:** Ова метрика мери колико модел тачно класификује податке на тренинг скупу. Тачност се повећава током обуке у случају коришћења обе архитектуре, што је очекивано јер модел постаје бољи у класификацији тренинг података.
- **Функција губитка на валидационом скупу:** Функција губитка на валидационом скупу мери колико добро модел генерализује податке које није видео током обуке. Приметно је да се у оба случаја коришћења губитак на валидационом скупу смањује са напретком обуке, што указује на побољшану генерализацију.
- **Тачност на валидационом скупу:** Ова метрика мери колико тачно модел класификује податке на валидационом скупу. Слично као и функција губитка, тачност на валидационом скупу се повећава током обуке у случају.

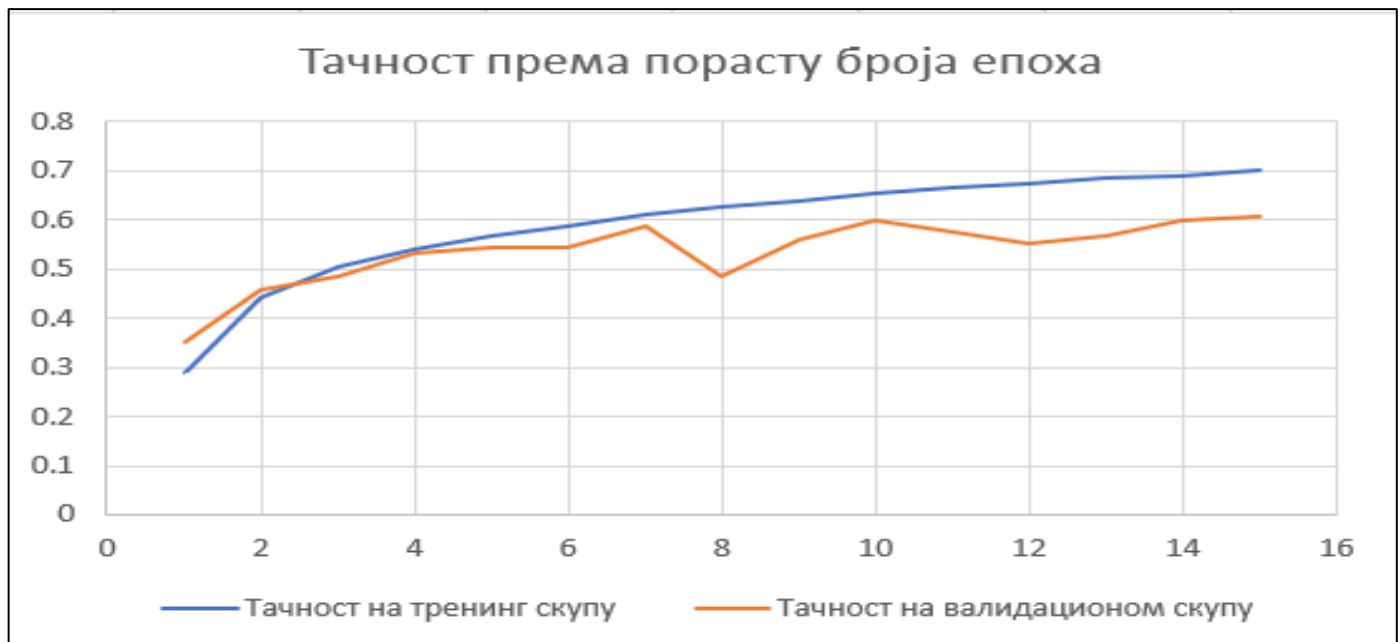
- ***DenseNet-121***

У општем смислу, резултати на слици 13 сугеришу да је модел '*DenseNet-121*' добро трениран на задатом скупу података. Искориштена је метода раног заустављања (*engl. earlyStoppingCallback*) како би се спречила претерана обученост мреже. У овом случају, обука се зауставила након 15 епоха, а тежине модела су сачуване у тачки где је постигнута најбоља тачност на валидационом скупу. Ово нам је помогло у очувању модела који најбоље генерализује, чак и ако би се могао десити раст губитка на валидационом скупу у каснијим епохама.

Функција губитка се смањује, а тачност се повећава током обуке, како на тренингу тако и на валидационом скупу. Међутим, резултати су нам потенцијално показало на претереану обученост мреже, јер се тачност на тренинг скупу повећава брже него на валидационом скупу након епохе 7. Ово је указало на потребу за коришћењем додатне регуларизације и прилагођавањем хиперпараметара како би се добили бољи резултати [22].

Редни број епохе	Време извршавања	Ф-ја губитка на тренинг скупу (engl. loss)	Тачност на тренинг скупу (engl. accuracy)	Ф-ја губитка на валидационом скупу (engl. loss)	Тачност на валидационом скупу (engl. accuracy)
1	362s 2s/step	12.4799	0.2902	8.8731	0.3527
2	337s 2s/step	6.6651	0.4418	4.9156	0.4581
3	346s 2s/step	3.8724	0.5040	3.2194	0.4867
4	355s 2s/step	2.4990	0.5403	2.0649	0.5328
5	356s 2s/step	1.8076	0.5697	1.6569	0.5454
6	356s 2s/step	1.4576	0.5897	2.7518	0.5430
7	357 2s/step	1.2642	0.6108	1.4672	0.5870
8	340 2s/step	1.1550	0.6283	1.6421	0.4874
9	361 2s/step	1.0875	0.6403	1.3304	0.5603
10	342 2s/step	1.0409	0.6541	1.3887	0.5993
11	374 2s/step	1.0015	0.6671	1.3093	0.5749
12	343 2s/step	0.9723	0.6751	1.1725	0.5513
13	331 2s/step	0.9472	0.6869	1.2413	0.5668
14	334 2s/step	0.9262	0.6911	1.3874	0.5996
15	367 2s/step	0.8975	0.7018	1.1788	0.6089

Табела 2. Резултати DenseNet-121 пре подешавања хиперпараметара



Слика 14. Резултати тачности DenseNet-121 пре подешавања хиперпараметара

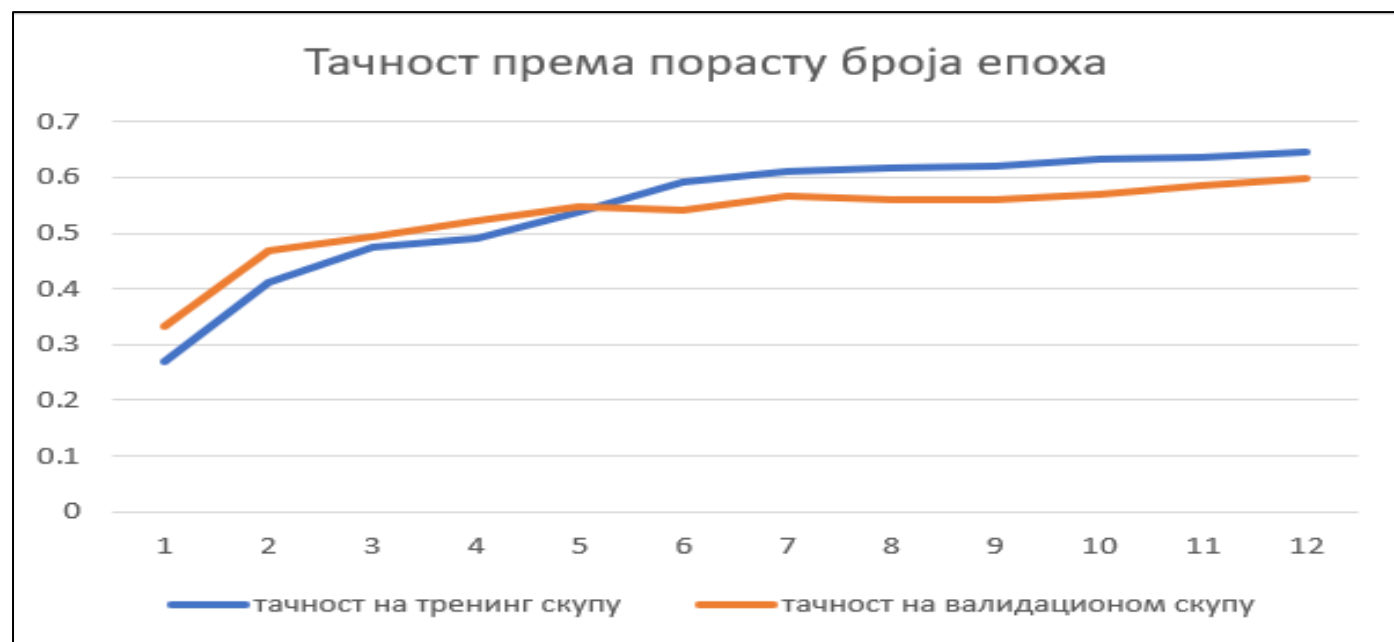
- AlexNet

У општем смислу, резултати на слици 13 сугеришу да је модел 'AlexNet' добро трениран на задатом скупу података током 12 епоха. Искориштена је метода раног заустављања (*engl. earlyStoppingCallback*) како би се спречила претерана обученост мреже. У овом случају, обука се зауставила након 12 епоха, а тежине модела су сачуване у тачки где је постигнута најбоља тачност на валидационом скупу.

Функција губитка конитнуирано опада, што указује на то да се модел прилагођава тренинг подацима а тачност се повећава током обуке, како на тренингу тако и на валидационом скупу. Ови позитивни трендови су обећавајући, али треба бити опрезан код појаве претеране обучености мреже.

Редни број епохе	Време извршавања	Ф-ја губитка на тренинг скупу (<i>engl. loss</i>)	Тачност на тренинг скупу (<i>engl. accuracy</i>)	Ф-ја губитка на валидационом скупу (<i>engl. loss</i>)	Тачност на валидационом скупу (<i>engl. accuracy</i>)
1	138s 1s/step	13.1254	0.2705	12.8766	0.3329
2	137s 1s/step	8.4311	0.4118	7.9101	0.4691
3	156s 1s/step	3.9714	0.4743	3.2384	0.4932
4	155s 1s/step	3.1804	0.4913	2.7658	0.5219
5	152s 1s/step	2.0172	0.5368	1.8563	0.5466
6	159s 1s/step	1.7576	0.5908	1.6512	0.5410
7	143 2s/step	1.6612	0.6108	1.6023	0.5674
8	140 1s/step	1.4559	0.6187	1.4023	0.5591
9	161 1s/step	1.2855	0.6204	1.2677	0.5607
10	142 1s/step	1.2419	0.6341	1.2187	0.5711
11	154 1s/step	1.2015	0.6371	1.2383	0.5849
12	143 1s/step	1.1983	0.6451	1.1728	0.5977

Табела 3. Резултати AlexNet пре подешавања хиперпараметара



Слика 15. Резултати тачности AlexNet пре подешавања хиперпараметара

4.1.2 Резултати након подешавања хиперпараметара (*engl. fine-tuning*)

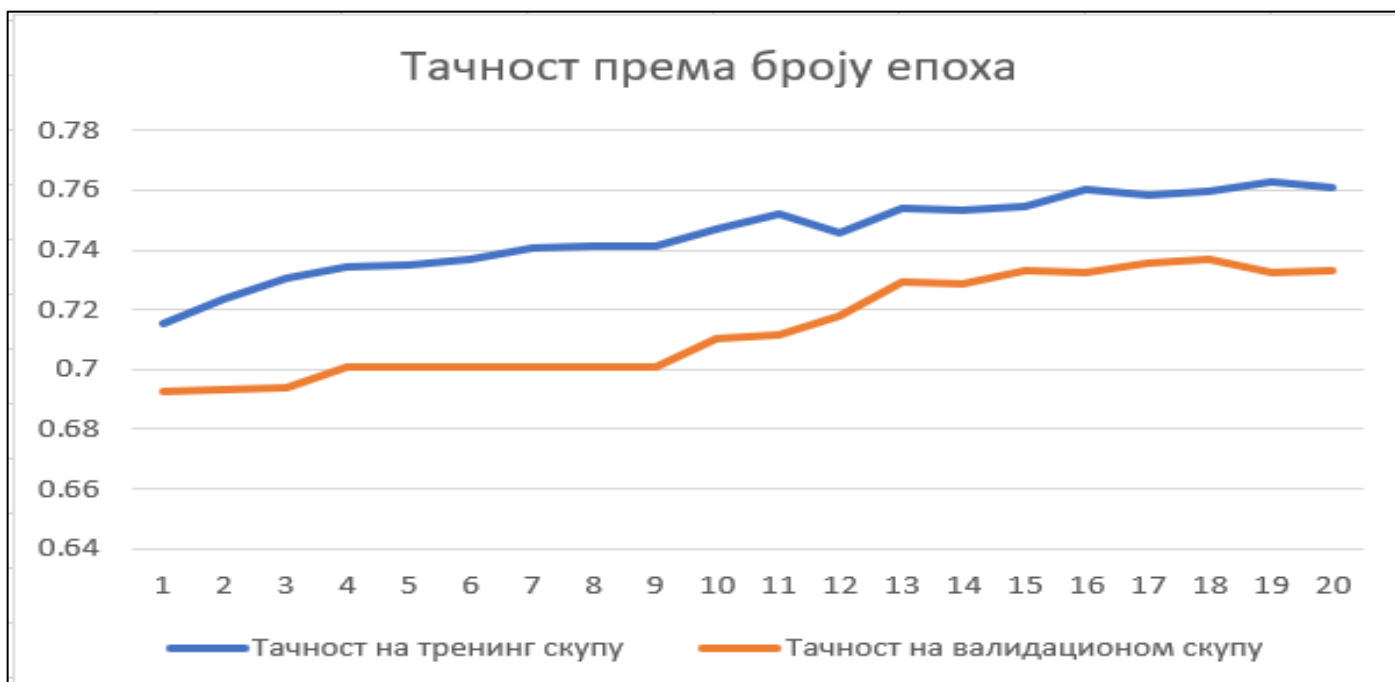
• *DenseNet-121*

Уз промену неколико хиперпараметара, као што је смањене брзине учења оптимизационог алгоритма SGD са 0.1 на 0.001, уз коришћење SGDM са коефицијентом моментума 0.9, смањење броја епоха са 30 на 20 и повећање величине скупова (*engl. batch size*) за обуку, добила се стабилнија обука и боља генерализација модела. Такође, сада не можемо да уочимо назнаке за претерану обученост мреже што нам је велики напредак у доносу на прошли модел. Циљ нам је довести стабилност тачности на валидационом скупу. Резултати су приказани у табели 3 и на слици 14.

- **Време извршавања:** Време потребно за извршавање једне епохе се значајно смањило и сада се оно креће између 141 секунде до 215 секунди по епохи.
- **Функција губитка на тренинг скупу:** Функција губитка на тренинг скупу опада током епоха, што указује на то да се модел боље прилагођава тренинг подацима како време пролази.
- **Тачност на тренинг скупу:** Тачност на тренинг скупу такође се повећава током обуке и то је добар знак да модел постаје бољи у класификацији над тренинг подацима.
- **Функција губитка на валидационом скупу:** Функција губитка на валидационом скупу има тенденцију да остане релативно стабилна током већег дела обуке. У каснијим епохама, губитак на валидационом скупу благо расте, али није драматично.
- **Тачност на валидационом скупу:** Тачност на валидационом скупу остаје доста стабилнија у односу на претходни модел што је и био циљ.

Редни број епохе	Време извршавања	Ф-ја губитка на тренинг скупу (<i>engl. loss</i>)	Тачност на тренинг скупу (<i>engl. accuracy</i>)	Ф-ја губитка на валидационом скупу (<i>engl. loss</i>)	Тачност на валидационом скупу (<i>engl. accuracy</i>)
1	149s 2s/step	0.8708	0.7154	1.0625	0.6927
2	187s 2s/step	0.8490	0.7238	1.0525	0.6934
3	150s 2s/step	0.8305	0.7306	1.0511	0.6941
4	179s 2s/step	0.8159	0.7345	1.0526	0.7011
5	176s 2s/step	0.8073	0.7353	1.0524	0.7011
6	196s 2s/step	0.7997	0.7369	1.0533	0.7011
7	208s 2s/step	0.7968	0.7406	1.0553	0.7011
8	179s 2s/step	0.7936	0.7415	1.0540	0.7011
9	141s 2s/step	0.7783	0.7415	1.0575	0.7011
10	215s 2s/step	0.7723	0.7472	1.0591	0.7104
11	174s 2s/step	0.7786	0.7522	1.0598	0.7118
12	178s 2s/step	0.7682	0.7460	1.0588	0.7178
13	177s 2s/step	0.7609	0.7538	1.0410	0.7290
14	207s 2s/step	0.7611	0.7531	1.0620	0.7285
15	187s 2s/step	0.7546	0.7547	1.0626	0.7332
16	175s 2s/step	0.7506	0.7600	1.0634	0.7325
17	178s 2s/step	0.7470	0.7584	1.0606	0.7359
18	173s 2s/step	0.7482	0.7596	1.0650	0.7366
19	177s 2s/step	0.7406	0.7631	1.0654	0.7325
20	191s 2s/step	0.7435	0.7609	1.0649	0.7330

Табела 4. Резултати *DenseNet-121* након подешавања хиперпараметара



Слика 16. Резултати тачности DenseNet-121 након подешавања хиперпараметара

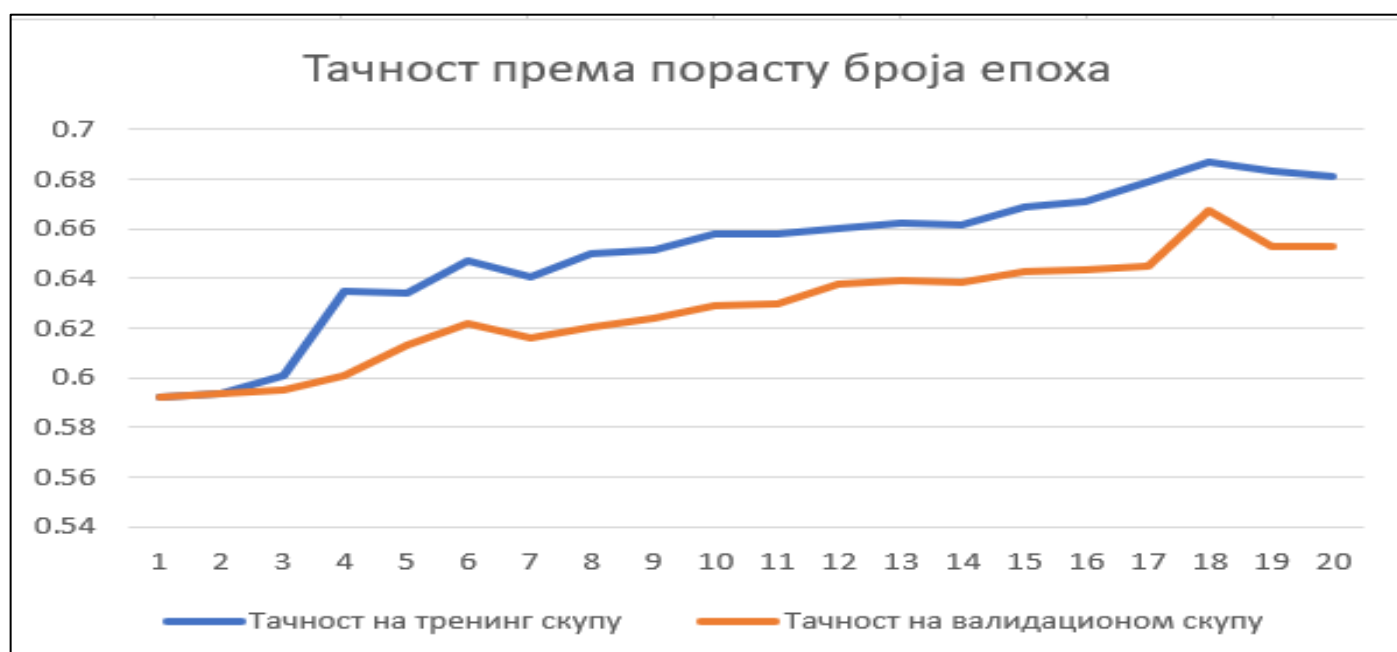
- **AlexNet**

Уз промену неколико хиперпараметара, као што је повећање брзине учења оптимизационог алгорита ADAM са 0.001 на 0.01, смањење броја епоха са 30 на 20 и повећање величине скупова (*engl. batch size*) за обуку, добила се стабилнија обука и боља генерализација модела. Циљ нам је довести већу стабилност тачности на валидационом скупу. Резултати су приказани у табели 4 и на слици 16.

- **Време извршавања:** Време потребно за извршавање једне епохе се значајно смањило и сада се оно креће између 68 секунди до 112 секунди по епохи.
- **Функција губитка на тренинг скупу:** Функција губитка на тренинг скупу опада током епоха, што указује на то да се модел боље прилагођава тренинг подацима како време пролази.
- **Тачност на тренинг скупу:** Тачност на тренинг скупу такође се повећава током обуке и то је добар знак да модел постаје бољи у класификацији над тренинг подацима.
- **Функција губитка на валидационом скупу:** Функција губитка на валидационом скупу има тенденцију континуираног опадања током раста броја епоха.
- **Тачност на валидационом скупу:** Тачност на валидационом скупу остаје доста стабилнија у односу на претходни модел.

Редни број епохе	Време извршавања	Ф-ја губитка на тренинг скупу (engl. loss)	Тачност на тренинг скупу (engl. accuracy)	Ф-ја губитка на валидационом скупу (engl. loss)	Тачност на валидационом скупу (engl. accuracy)
1	79s 1s/step	1.0709	0.5924	1.0655	0.5923
2	68s 1s/step	1.0412	0.5938	1.0551	0.5936
3	71s 1s/step	1.0325	0.6011	1.0403	0.5949
4	87s 1s/step	0.9852	0.6349	1.0378	0.6011
5	112s 1s/step	0.9873	0.6342	1.0312	0.6133
6	95s 1s/step	0.9712	0.6469	1.0267	0.6215
7	85s 1s/step	0.9636	0.6406	1.0253	0.6157
8	89s 1s/step	0.9636	0.6499	1.0212	0.6201
9	77s 1s/step	0.9683	0.6515	1.0199	0.6241
10	69s 1s/step	0.9523	0.6577	1.0123	0.6292
11	101s 1s/step	0.9586	0.6581	0.9934	0.6299
12	95s 1s/step	0.9582	0.6603	0.9912	0.6378
13	98s 1s/step	0.9409	0.6625	0.9902	0.6391
14	102s 1s/step	0.9411	0.6618	0.9900	0.6385
15	84s 1s/step	0.9412	0.6687	0.9832	0.6427
16	73s 1s/step	0.9466	0.6711	0.9814	0.6433
17	76s 1s/step	0.9373	0.6789	0.9761	0.6450
18	81s 1s/step	0.9421	0.6866	0.9710	0.6676
19	82s 1s/step	0.9369	0.6831	0.9704	0.6527
20	89s 1s/step	0.9354	0.6809	0.9694	0.6530

Табела 5. Резултати AlexNet након подешавања хиперпараметара



Слика 17. Резултати тачности AlexNet након подешавања хиперпараметара

4.2. Класификациони извештај

У класификационом извештају приказаном у табели 5 и 6 користимо следеће метрике за оцену квалитета класификационог модела:

- **Прецизност (engl. *precision*):** Прецизност мери колико од укупно предвиђених позитивних инстанци (TP+FP) заиста припадају класи позитивних инстанци (TP). Прецизност одмерава колико је модел прецизан у предвиђању позитивних инстанци. Више прецизности значи мање лажних позитивних предвиђања. Формула за прецизност ' $precision = TP / (TP + FP)$ '.
- **Одзив (engl. *recall*):** Одзив мери колико од укупно позитивних инстанци (TP + FN) модел успешно предвиђа као истинитно позитивне (TP). Формула за одзив је ' $recall = TP / (TP + FN)$ '.
- **Ф1-мера (engl. *F1-score*):** Хармонијска средња вредност између прецизности и одзива. Користи се као балансирана метрика између прецизности и одзива. Формула за Ф1-меру је ' $f1-score = 2 / (1/precision + 1/recall)$ '.
- **Подршка (engl. *support*):** Подршка представља број инстанци у одређеној класи на тест скупу.

	Precision	Recall	F1-score	Support
Angry	0.62	0.71	0.66	958
Disgust	0.00	0.00	0.00	11
Fear	0.67	0.54	0.66	1024
Happy	0.87	0.86	0.87	1774
Neutral	0.55	0.69	0.61	1233
Sad	0.51	0.48	0.49	1247
Surprise	0.75	0.75	0.75	831
Accuracy			0.67	7178

Табела 6. Резултати класификационог извештаја DenseNet-121 архитектуре

	Precision	Recall	F1-score	Support
Angry	0.56	0.61	0.59	958
Disgust	0.00	0.00	0.00	11
Fear	0.59	0.55	0.56	1024
Happy	0.79	0.81	0.80	1774
Neutral	0.48	0.52	0.50	1233
Sad	0.51	0.50	0.51	1247
Surprise	0.70	0.72	0.71	831
Accuracy			0.62	7178

Табела 7. Резултати класификационог извештаја AlexNet архитектуре

На основу резултата истраживања, могу се извући следећи закључци:

1. Ефикасност модела '*DenseNet-121*' – Модел трениран овом архитектуром је постигао високу тачност у класификацији емоција након подешавања хиперпараметара. Показао се изузетно ефикасан у обуци и генерализацији, чиме је демонстрирана способност успешног препознавања емоција на лицима.
2. Стабилност модела '*AlexNet*' – Након подешавања хиперпараметара, овај модел је такође истакао побољшање тачности. Важно је истаћи особину стабилности која се добила након подешавања хиперпараметара.

5. Закључак

Закључак овог истраживања о примени CNN за класификацију емоција на лицима људи доноси неколико важних увида. Прво, ово истраживање показује да је могуће постићи доста високу тачност у класификацији емоција на сликама лица уз помоћ дубоких неуронских мрежа. Коришћење модела *'AlexNet'* и *'DenseNet-121'*, након одговарајућег подешавања хиперпараметара, омогућило је обезбеђивања прецизних резултата.

'DenseNet-121', са својом дубоком структуром и способношћу добре генерализације, може бити одличан избор за задатке где постоји обиман скуп података и када је тачност кључна. Дубина омогућава овом моделу да научи комплексне обрасце у подацима. Међутим, *'AlexNet'* је ефикаснији по питању ресурса и брзине извршавања, што га чини пожељнијим избором за примену са ограниченим ресурсима, као што су уређаји са мање процесорске снаге или са ограниченом радном меморијом. Иако је јако брз у извршавању анализе, може бити мање способан за генерализацију у сложеним задацима, због своје мање дубине.

Основна сврха овог истраживања јесте да се утврди који од ова два модела може бити бољи избор за конкретну примену. На основу приказаних резултата и анализе, закључујемо да је бољи избор архитектура *'DenseNet-121'* јер висока прецизност је кључна у нашем задатку, иако су оба модела приказала своје предности и мане.

Важно је напоменути да и даље постоји простор за побољшање и истраживање. Додатни експерименти и анализе могу омогућити даље оптимизације и евентуално прилагођавање оба модела за специфичне примене. Потенцијал и будуће истраживање ће бити усмерени ка имплементацији апликације која ће успешно детектовати емоције у реалном времену. Овај корак може бити посебно користан за широк спектар примена, као што су праћење емоција током видео конференција, анализа реакција публике на догађајима или интеракција корисника у апликацијама виртуелне стварности.

Биографија

Срђан Протић је рођен 14. априла 2000. године у Мркоњић Граду, Босни и Херцеговини. Завршио је општи смер Гимназије у Мркоњић Граду. Школске године 2019/2020. уписује Факултет техничких наука у Новом Саду, смер Информациони инжењеринг. Основне академске студије је завршио школске 2022/2023. године.

- [1] Aggarwal, C. C. (2018). Chapter 9: Convolutional Neural Networks. У C. C. Aggarwal, *Neural Networks and Deep Learning: A Textbook*. Springer.
- [2] Alex Krizhevsky, I. S. (2012). Advances in neural information processing systems. У I. S. Alex Krizhevsky, *Imagenet classification with deep convolutional neural networks*. (стр. 1097–1105).
- [3] Aliyev, V. (2020). A definitive explanation to the Hinge Loss for Support Vector Machines. *Towards Data Science*. Преузето са <https://towardsdatascience.com/a-definitive-explanation-to-hinge-loss-for-support-vector-machines-ab6d8d3178f1>
- [4] Behnam Neyshabur, R. S. (2019). *Path-SGD: Path-Normalized Optimization in Deep Neural Networks*. University of Toronto.
- [5] Bishop, C. M. (2006). Linear Regression. У C. M. Bishop, *"Pattern Recognition and Machine Learning* (стр. Chapter 5).
- [6] Bishop, C. M. (2006). Logistic Regression. У C. M. Bishop, *Pattern Recognition and Machine Learning* (стр. Chapter 3).
- [7] Brownlee, J. (2020). *A Gentle Introduction to Cross-Entropy for Machine Learning*. Преузето са <https://machinelearningmastery.com/cross-entropy-for-machine-learning/>
- [8] Ekman, P. (2003). *Emotions Revealed: Recognizing Faces and Feelings to Improve Communication and Emotional Life*.
- [9] Euhanna Ghadimi, H. R. (2015). Global convergence of the heavy-ball method for convex optimization. *European Control Conference (ECC)*, (стр. 310–315).
- [10] Gao Huang, Z. L. (2017). *Densely Connected Convolutional Networks*. arXiv.
- [11] Géron, A. (2017). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*. O'Reilly Media.
- [12] Goodfellow, I. B. (2016). *Deep Learning*. MIT Press. Преузето са <https://www.deeplearningbook.org/>
- [13] Ilya Sutskever, J. M. (2013). On the importance of initialization and momentum in deep learning. *International conference on machine learning*, (стр. 1139–1147).
- [14] Ivan Vasilev, D. S. (2017). *Python Deep Learning*. Packt Publishing.

- [15] J., V. (2018). Tutorial on using Keras flow_from_directory and generators. *Medium*.
 Пpeызeтo ca <https://vijayabhaskar96.medium.com/tutorial-image-classification-with-keras-flow-from-directory-and-generators-95f75ebe5720>
- [16] Manav, D. *DenseNet 121 feature extraction*. Пpeызeтo 9 26, 2023 ca Kaggle:
<https://www.kaggle.com/code/darthmanav/densenet-121-feature-extraction>
- [17] Manjusha Pandey, G. J. (2022). Efficacy Determination of Various Base Networks in Single Shot Detector for Automatic Mask Localisation in a Post COVID Setup. *Journal of Experimental & Theoretical Artificial Intelligence* 35(3), 9-12.
- [18] Martín Abadi, P. B. (2016). et al. Tensorflow: A system for large-scale machine learning. *12th {USENIX} symposium on operating systems design and implementation ({OSDI}16)*, (стр. 265–283).
- [19] Monro, H. R. (1951). A stochastic approximation method. Y H. R. Monro, *The annals of mathematical statistics* (стр. 400-407).
- [20] Owen, L. (2022). *Hyperparameter Tuning with Python: Boost your machine learning model's performance via hyperparameter tuning*. Packt Publishing.
- [21] Padmanabha, A. *Ridge Regression*. Пpeызeтo 9 28, 2023 ca Brilliant:
<https://brilliant.org/wiki/ridge-regression/>
- [22] Pandian, S. (2023, 9 28). *A Comprehensive Guide on Hyperparameter Tuning and its Techniques*. Пpeызeтo ca Analytics Vidhya: <https://www.analyticsvidhya.com/blog/2022/02/a-comprehensive-guide-on-hyperparameter-tuning-and-its-techniques/>
- [23] Paul Ekman, W. V. (2006). Facial Action Coding System. *PubMed*.
- [24] Polyak, B. T. (1964). Some methods of speeding up the convergence of iteration methods. Y B. T. Polyak, *USSR Computational Mathematics and Mathematical Physics* (стр. 4(5):1–17).
- [25] Quoc Tran-Dinh, M. v. Gradient Descent-Type Methods: Background and Simple Unified Convergence Analysis. *arXiv*. Пpeызeтo ca <https://doi.org/10.48550/arXiv.2212.09413>
- [26] Rahul Kidambi, P. N. (2018). On the insufficiency of existing momentum schemes for stochastic optimization. *Information Theory and Applications Workshop (ITA)*, 1–9.
- [27] Sun, R. (2019). *Optimization for deep learning : theory and algorithms*. arXiv preprint arXiv.
- [28] Szeliski, R. (2010). *Computer Vision: Algorithms and Applications*. Springer.
- [29] *TensorFlow*. Пpeызeтo 10 1, 2023 ca TensorFlow: <https://www.tensorflow.org/>

- [30] *tf.keras.activations.relu*. Преузето 10 2, 2023 са TensorFlow:
https://www.tensorflow.org/api_docs/python/tf/keras/activations/relu
- [31] *tf.keras.callbacks.EarlyStopping*. Преузето 9 28, 2023 са TensorFlow:
https://www.tensorflow.org/api_docs/python/tf/keras/callbacks/EarlyStopping
- [32] *tf.keras.layers.Dense*. Преузето 9 28, 2023 са TensorFlow:
https://www.tensorflow.org/api_docs/python/tf/keras/layers/Dense
- [33] *tf.keras.layers.Dropout*. Преузето 9 28, 2023 са TensorFlow:
https://www.tensorflow.org/api_docs/python/tf/keras/layers/Dropout
- [34] *tf.keras.layers.GlobalAveragePooling2D*. Преузето 9 29, 2023 са TensorFlow:
https://www.tensorflow.org/api_docs/python/tf/keras/layers/GlobalAveragePooling2D
- [35] *tf.keras.Model*. Преузето 10 1, 2023 са TensorFlow:
https://www.tensorflow.org/api_docs/python/tf/keras/Model
- [36] *tf.keras.preprocessing.image.ImageDataGenerator*. (.). Преузето 9 30, 2023 са TensorFlow:
https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/image/ImageDataGenerator
- [37] Trevor Hastie, R. T. (2009). Linear Methods for Regression, Ridge Regression. У R. T. Trevor Hastie, *The Elements of Statistical Learning* (стр. Chapter 3 (4)). Springer.
- [38] Yuke Wang, B. F. (2021). An Efficient Quantitative Approach for Optimizing Convolutional Neural Networks. *CIKM: Conference on Information and Knowledge Management*, (стр. 2050-2059). Преузето са <https://doi.org/10.1145/3459637.3482230>
- [39] Миливојевић, Д. З. (2000). *Психотерапија и разумевање емоција*.
- [40] Ares. *Emotion Detection*. Преузето са Kaggle:
<https://www.kaggle.com/datasets/ananthu017/emotion-detection-fer>