SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

Marko Marijić

Zagreb, 2022.

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

Mentor:

Dr. sc. Tomislav Stipančić, dipl. ing.

Student:

Marko Marijić

Zagreb, 2022.

Izjavljujem da sam ovaj rad izradio samostalno koristeći znanja stečena tijekom studija i navedenu literaturu.

Zahvaljujem se mentoru dr. sc. Tomislavu Stipančiću na pomoći i sugestijama prilikom izrade diplomskog rada.

Jedno veliko hvala mojoj obitelji, djevojci i prijateljima na podršci i razumijevanju pruženom tijekom trajanja studija.

Marko Marijić

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE
Središnje povjerenstvo za završne i diplomske ispite
Povjerenstvo za diplomske radove studija strojarstva za smjerove:
proizvodno inženjerstvo, računalno inženjerstvo, industrijsko inženjerstvo i menadžment,
inženjerstvo materijala te mehatronika i robotika

Sveučilište u Zagrebu Fakultet strojarstva i brodogradnje	
Datum:	Prilog:
Klasa: 602-14/22-6/1	
Ur. broj: 15-1703-22-	

DIPLOMSKI ZADATAK

Student: **MARKO MARIJIĆ** Mat. br.: 0035209859

Naslov rada na hrvatskom jeziku: **Usporedba i evaluacija baza podataka emocija koristeći konvolucijsku neuronsku mrežu**

Naslov rada na engleskom jeziku: **Comparison and evaluation of emotion databases using a convolutional neural network**

Opis zadatka:

Tehnike dubokog učenja koriste velike količine podataka da bi kreirale računalne modele koji donose hipoteze koje aproksimiraju ono što sadrže podaci. Kvaliteta dostupnih baza podataka odstupa u ovisnosti o veličini baza te o kvaliteti podataka koje baza sadrži.

U radu je potrebno:

- Odabrati tri proizvoljne baze podataka koje sadrže slike osoba u različitim emocionalnim raspoloženjima.
- Odrediti parametre te trenirati konvolucijsku neuronsku mrežu VGG16 arhitekture da prepozna emocije osoba sa slika.
- Koristiti standardne evaluacijske tehnika za analizu rada konvolucijske neuronske mreže i na temelju tih podataka donijeti zaključke o kvaliteti odabranih baza slika (K-FOLD metoda).
- Usporediti te ocijeniti kvalitetu korištenih baza podataka temeljem rezultata primijenjenih evaluacijskih tehnika.

Razvijenu i korištenu metodologiju potrebno je evaluirati u sklopu Laboratorija za projektiranje izradbenih i montažnih sustava.
U radu je potrebno navesti korištenu literaturu te eventualno dobivenu pomoć.

Zadatak zadan: 29. rujna 2022.

Rok predaje rada: 1. prosinca 2022.

Predviđeni datum obrane: 12. prosinca do 16. prosinca 2022.

Zadatak zadao: doc. dr. sc. Tomislav Stipančić

Predsjednica Povjerenstva: prof. dr. sc. Biserka Runje

SADRŽAJ

POPIS SLIKA

POPIS TABLICA

POPIS PROGRAMSKIH KODOVA

SAŽETAK

Unutar ovog rada uspoređuju se dostupne baze podataka za automatsko prepoznavanje emocija s lica osobe. Zadatak je upotrebom VGG16 konvolucijske neuronske mreže provesti trening s tri različite baze podataka te evaluirati njihov rad. Koristeći isti model mreže, mogu se donijeti konkretni zaključci o kvaliteti podataka unutar svake baze. Uvodni dio rada pojašnjava što je to duboko učenje te predstavlja dosadašnja postignuća u području automatskog prepoznavanja emocija. Počevši od modela umjetnog neurona, objasniti će se cjelokupna arhitektura mreže. Definirati će se slojevi konvolucijske neuronske mreže, a zatim i objasniti svaki od njih. Nadalje, predstaviti će se baze podataka te opisati proces treninga. Na kraju će se prikazati dobiveni rezultati te će se komparativnom analizom donijeti zaključak o kvaliteti rada svake mreže. Za potrebe istraživanja u ovom radu korištene su brojne znanstvene metode od kojih se izdvajaju znanstvene metode analize i sinteze te metode indukcije i dedukcije.

Ključne riječi: automatsko prepoznavanje emocija, prepoznavanje emocija s lica osobe, duboko učenje, VGG16, konvolucijska neuronska mreža

SUMMARY

Within this paper the available databases for automatic emotion recognition from facial features will be compared. The task is by using same VGG16 convolution neural network conduct training with three different databases and evaluate their work. Using the same network model, specific conclusions can be made about the quality of the date within each database. The introductory part of the paper explains what deep learning is and presents the achievements so far in the field of automatic emotion recognition. Starting with an artificial neuron model, the overall architecture of the network will be explained. The layers of convolutional neural networks will be defined and then each of them will be explained. Furthermore, databases will be presented, and the training process will be described. At the end, the obtained results will be presented, and a conclusion will be drawn about the quality of work of each network through a comparative analysis. For the purposes of research in this work, numerous scientific methods are used, of which the scientific methods of analysis and synthesis and the methods of induction and deduction stand out.

Key words: automatic emotion recognition, facial emotion recognition, FER , deep learning, VGG16, convolutional neural network, CNN

1. Uvod

Ljudski osjećaji su ključni faktor za razumijevanje ponašanja i stanja uma pojedinca. Automatsko prepoznavanje ljudskih emocija veliko je i važno istraživačko područje koje se bavi psihološkim prepoznavanjem emocija i umjetnom inteligencijom. Emocionalno stanje ljudi može se očitati iz verbalnih i neverbalnih informacija koje prikupljaju različite vrste senzora. Prema [1] čak 55% emocionalnih informacija je vizualno, 38% vokalno i 7% verbalno. Promjene lica tijekom komunikacije su prvi znakovi koji prenose emocionalno stanje, zbog čega je većina istraživača vrlo zainteresirana za ovo područje. Iako su vizualne informacije o emocionalnom stanju osobe najzastupljenije, izdvajanje tih značajki od čovjeka do čovjeka tj. od jednog lica do drugog je vrlo osjetljiv i težak zadatak. Godine 1978. Ekman i Freisen su među prvim znanstvenicima zainteresiranim za ekspresiju lica razvili FACS u kojem opisuju 46 različitih akcijskih stanja kod kojih je svako stanje povezano s kontrakcijom jednog ili više mišića lica [1]. Prepoznavanje emocija temeljeno na FACS sustavu dijeli se na dva tipa: jednofazno gdje se emocije prepoznaju direktno i dvofazno, gdje se anotiraju akcijske jedinice lica, a zatim se izlaz emocija zaključuje iz otkrivenih akcijskih jedinica. Iako je to bio velik korak ka uspješnijem automatskom prepoznavanju emocija, on je otvorio još novih pitanja i problema. Varijacije u dobi čovjeka, položaju glave, osvjetljenju, spolu i pozadini na kojoj se on nalazi kao i kožni problemi ili ožiljci koje osoba može imati su sve dodatni parametri koji itekako utječu na zahtjevnost ovog problema. Sustav na ulazu dobiva slike koje sadrže lica osoba te nakon procesiranja na izlazu se očekuje prepoznata emocija. Iako se na prvi pogled čini kao vrlo jednostavan zadatak posebice za ljudski rod, to je zapravo vrlo kompliciran proces za strojeve da prepoznaju emocije bez pomoći čovjeka. Detekcija lica prvi je korak ka automatskom prepoznavanju emocija, jer se slike ne sastoje samo od lica, nego često sadrže složenu pozadinu iz koje je potrebno izdvojiti lice. Zatim slijedi korak prepoznavanja izraza lica te izdvajanje značajki iza kojih slijedi klasifikacija emocionalnog stanja (slika 1.). Strojna analiza ljudskog afektivnog ponašanja ima potencijalno širok opseg upotrebe kao što je interakcija čovjeka i računala, zdravstvena skrb, učenje uz pomoć računala, otkrivanje anomalija u ljudskom ponašanju te povećanje doživljaja prilikom igranja interaktivnih računalnih igrica [2]. Posljednjih godina duboko učenje, zahvaljujući svojoj arhitekturi koje omogućuje automatsko izdvajanje značajki i klasifikaciju konvolucijskih i rekurentnih

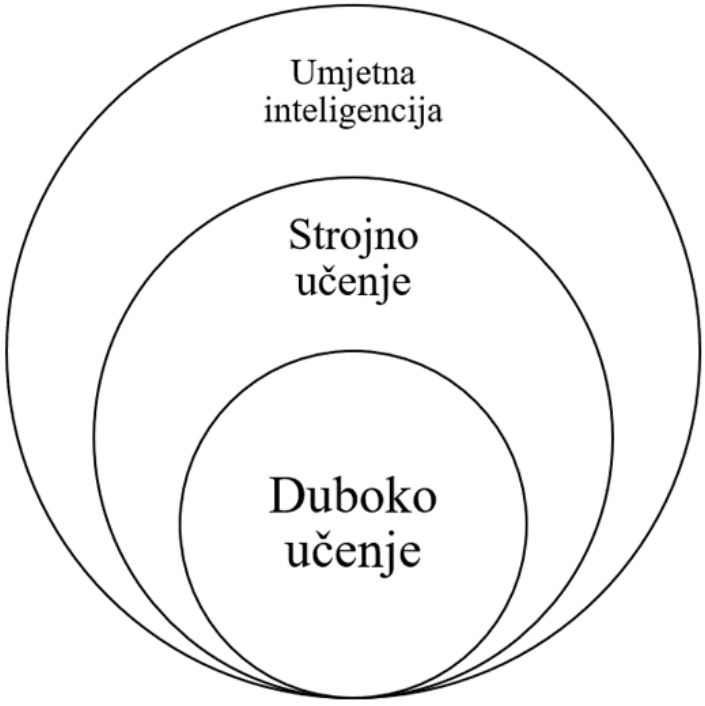
neuronskih mreža se pokazalo kao uspješan i učinkovit pristup te je potaknulo istraživače na daljnji razvoj.

Slika 1. Koraci automatskog prepoznavanja emocija [2]

U sklopu ovog rada provedena je klasifikacija prema sedam osnovnih emocija: sreća, tuga, strah, ljutnja, iznenađenje, gađenje i neutralno stanje. Korišten je model VGG16 konvolucijske neuronske mreže koji je treniran na tri odabrane baze podataka. Usporedbom rada istovrsnih neuronskih mreža koje su trenirane na različitim bazama podataka može se donijeti zaključak o kvaliteti pojedine baze podataka.

2. Duboko učenje

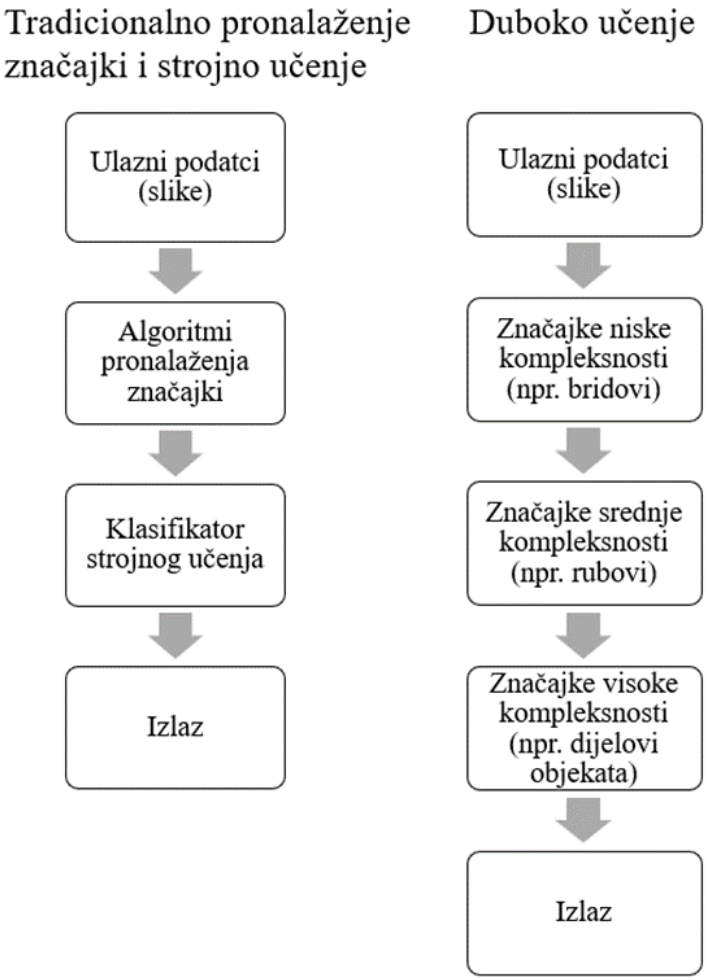
Duboko učenje definira se kao metoda strojnog učenja na više razina apstrakcije korištenjem umjetnih neuronskih mreža. Ulazni podatci se sastoje od više slojeva koje je potrebno obraditi iz čega je proizašla potreba za dubokim učenjem. Metode dubokog učenja su uvelike unaprijedile sposobnost strojnog vida, prepoznavanja govora i mnoga druga područja poput medicine gdje se može koristiti za razvijanje novih lijekova. Duboko učenje je dio strojnog učenja, a sve skupa dio je umjetne inteligencije (slika 2.). Glavni cilj umjetne inteligencije je pronaći algoritam koji može rješavati zadatke koje čovjek rješava intuitivno i gotovo automatski, a koji su gotovo redovito vrlo zahtjevni za računalo. Dok se umjetna inteligencija temelji na sposobnosti racionalizacije i poduzimanju radnji kako bi se postigla najveća šansa za izvršavanje zadanog cilja, strojno učenje se fokusira na prepoznavanje uzoraka i učenje iz seta podataka. Duboko učenje kao grana strojnog učenja ističe se po tome što koristi umjetne neuronske mreže.



Slika 2. Odnos dubokog učenja, strojnog učenja i umjetne inteligencije [3]

Duboko učenje se odnosi na korištenje kompleksnih mreža s više međusobno povezanih slojeva. Još u početcima strojnog učenja bili su vidljivi nedostatci. Strojno učenje upotrebljava

linearni perceptron koji je najjednostavnija mreža za klasifikaciju linearno separabilnih uzoraka. No, brzo se došlo do zaključka da linearni perceptron ne može biti univerzalni klasifikator, što je potaklo na razmišljanje o drugim aktivacijskim funkcijama koje nisu polinomi. Arhitekture dubokog učenja poput dubokih neuronskih mreža, pojačanog dubokog učenja, ponavljajuće neuronske mreže i konvolucijske neuronske mreže mogu u nekim slučajevima pružiti rezultate usporedive s ljudskim rezultatima u istim područjima. Na slici 3 vidljiva je razlika u slijedu izvođenja algoritma dubokog učenja u odnosu na strojno učenje.



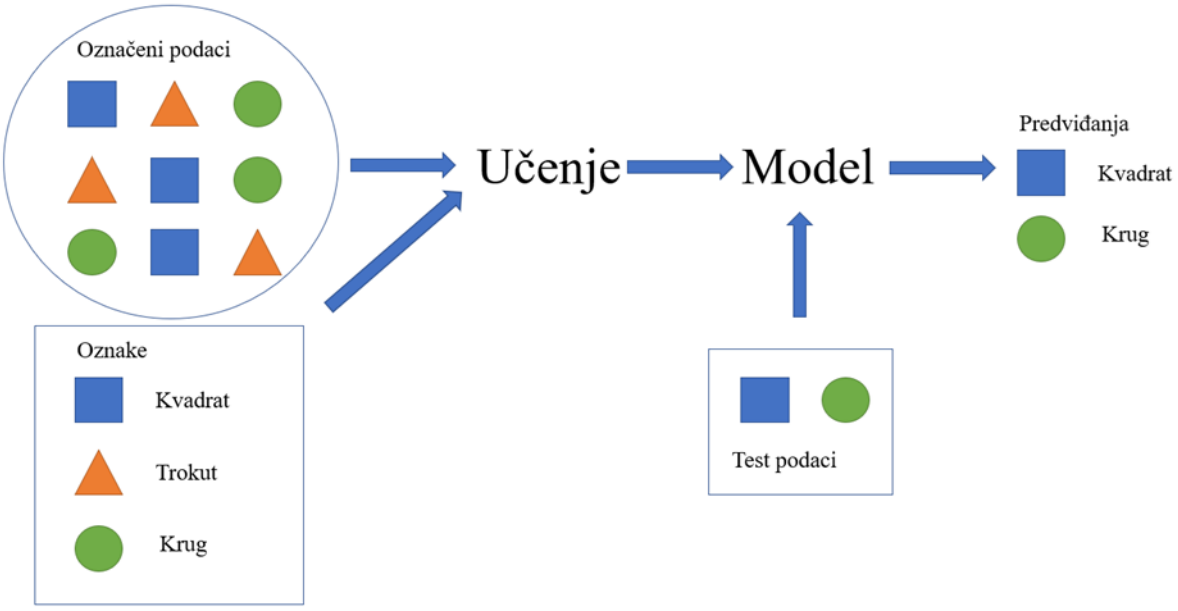
Slika 3. Usporedba algoritama tradicionalnog strojnog učenja i dubokog učenja [3]

U tradicionalnom strojnom učenju koriste se algoritmi za „ručno“ pronalaženje značajki za kvantificiranje slike jer ih osoba koja izrađuje algoritam mora ručno definirati, a duboko učenje koristi neobrađene intenzitete piksela kao ulazne podatke. Za svaku sliku u skupu podataka kod tradicionalnog strojnog učenja potrebno je izvršiti izdvajanje značajki na način da se

kvantificira ulazna slika prema nekom algoritmu koji se naziva ekstraktor značajki i kao rezultat prikaže vektor brojeva koji se koristi za daljnju obradu. Obrađuje se samo određena značajka slike, a pikseli kao takvi nisu imali veliku svrhu osim da budu ulaz za ekstraktor značajki slike. Duboko učenje, a posebice konvolucijske neuronske mreže imaju skroz drugačiji pristup. Umjesto definiranja skupa pravila i algoritama za izvlačenje značajki, te su značajke naučene automatski u fazi treninga. Kada se na ulaz dovede slika, pikseli te slike su ulazi u konvolucijsku neuronsku mrežu. Skriveni slojevi te mreže se koriste za izvlačenje značajki iz ulazne slike na način da se grade jedan na drugi hijerarhijski. Prvo se izvlače značajke poput bridova u nižim slojevima mreže, a zatim kutovi i rubovi koji se onda kasnije koriste u višim slojevima za izvlačenje drugih značajki. Svaki sloj mreže koristi izlaz prethodnog sloja kao građevni materijal za gradnju sve više apstraktnih koncepata [4].

2.1. Nadzirano učenje

Najčešći oblik strojnog učenja, bilo ono duboko ili ne, je nadzirano učenje. To je vrsta strojnog učenja kod kojeg se mreža trenira pomoću označenih podataka te se na temelju tih podataka predviđa izlaz. Nadzirano učenje podrazumijeva preslikavanje ulazne varijable (x) na pripadajuće izlazne varijable (Y). Cilj je tako dobro aproksimirati funkciju Y=f(x), da kada se dovede novi set ulaznih podataka funkcija i dalje predviđa dobra rješenja [5]. Naziva se nadziranim jer je proces učenja algoritma iz skupa podataka takav da se nakon svake iteracije predviđanje nadzire, te se ispravlja. Na slici 4 prikazan je koncept nadziranog učenja.



Slika 4. Koncept nadziranog učenja [5]

2.2. Prepoznavanje emocija s lica osoba na slici upotrebom dubokog učenja

Prepoznavanje izraza lica jedan je od ključnih faktora u prenošenju ljudskih emocija i osjećaja. Automatizirani sustavi za prepoznavanje izraza lica omogućuju različitim strojevima da prepoznaju emocije bez pomoći ljudi, što se smatra vrlo izazovnim problemom u strojnom učenju. Tradicionalne metode strojnog učenja su imale ograničene sposobnosti procesiranja podataka koji nisu bili prethodno obrađeni. Desetljećima je bila potrebna izgradnja cjelokupnog sustava za prepoznavanje uzoraka, što je zahtijevalo pažljivo projektiranje i stručnost. Bilo je potrebno izraditi algoritam za ekstrakciju značajki koji transformira neobrađene podatke poput vrijednosti piksela u odgovarajući prikaz ili vektor obilježja iz kojeg bi sustav mogao klasificirati uzorke. Unatoč značajnom napretku tradicionalnih metoda prepoznavanja lica izdvajanjem ručno definiranih značajki, tijekom prošlog desetljeća istraživači su se usmjerili na pristup dubokog učenja zbog velikih mogućnosti koje ono nudi. Važan dio nedavnog napretka u ovom polju postignut je zahvaljujući pojavi metoda dubokog učenja i posebno konvolucijskim neuronskim mrežama. Još od ranih godina 20. stoljeća konvolucijske neuronske mreže su se primjenjivale za detekciju, segmentaciju i prepoznavanje objekata na slici. Sve su to bili zadatci u kojima je bio dostupan veliki broj podataka poput prepoznavanja prometnih znakova, prepoznavanja teksta, izdvajanja ljudi iz slike i prepoznavanja lica osobe. Unatoč svim uspjesima konvolucijske neuronske mreže nisu bile u fokusu glavne zajednice za računalni vid i strojno učenje sve do natjecanja ImageNet 2012. godine. Tada je konvolucijska neuronska mreža primijenjena na skup podataka od oko milijun slika s oko 1000 različitih klasa te su postignuti izvanredni rezultati koji su prepolovili stope pogrešaka konkurentskih pristupa. Jedan od razloga je efikasna upotreba grafičke procesorske jedinice koja je omogućila treniranje više stotina milijuna težina i čvorova u svega nekoliko sati. Taj uspjeh je doveo do revolucije u računalnom vidu te od tada konvolucijske neuronske mreže dominiraju u ovom području. U nastavku su predstavljena neka od najvažnijih istraživanja u ovom području [1].

Mollahosseini i sur. [6] predložili su duboku konvolucijsku neuronsku mrežu za automatsko prepoznavanje emocija osoba sa slike koja koristi više različitih baza podataka. Nakon što su izdvojili značajke lica sa slika dimenzija 48x48 piksela, primijenili su metodu povećanja podataka. Arhitektura neuronske mreže koja je korištena u sklopu ovog istraživanja sastojala se od dva konvolucijska sloja kombinirana sa slojevima sažimanja, te kombinaciju slojeva koji su sadržavali konvolucijske slojeve s filterima veličine 1x1, 3x3 i 5x5. Ovim istraživanjem prezentirali su sposobnost korištenja tehnike mreže u mreži, koja omogućuje povećanje

lokalnih performansi zbog konvolucijskih slojeva primijenjenih lokalno što je za rezultat imalo smanjenje problema pretreniranosti.

S druge strane u istraživanju Lopesa i sur. [7] proučavan je utjecaj prethodne obrade podataka odnosno slika prije faze treniranja kako bi osigurali bolju klasifikaciju emocija. Povećanje broja podataka, korekcija rotacije slike, rezanje slike, smanjivanje uzorkovanja s 32x32 piksela i normalizacija intenziteta su koraci koji su primijenjeni prije ulaza u konvolucijsku neuronsku mrežu. Njihova mreža sastojala se također od dva konvolucijska sloja kombinirana sa slojevima sažimanja te je završavala s dva potpuno povezana sloja s 256 i 7 neurona. Najbolje težine dobivene tijekom faze treninga korištene su u fazi testiranja. Istraživanje je provedeno na sljedećim bazama podataka: CK+, JAFFE, BU-3DFE. Istraživanje je pokazalo da se kombinacijom svih ovih koraka pred procesiranja postižu bolji rezultati, u odnosu ako se ovi postupci primjenjuju samostalno.

Ove tehnike pred procesiranja također su implementirali Mohammadpour i sur. [8] Oni su predložili novi model konvolucijske neuronske mreže za otkrivanje aktivacijskih jedinica lica. Njihov model sastoji se od dva konvolucijska sloja, nakon svakog slijedi sloj sažimanja te završava s dva potpuno povezana sloja koji pokazuju broj aktiviranih akcijskih jedinica lica.

Detekciju važnih dijelova lica predložili su Yolcu i sur. [9] koristili su tri konvolucijske neuronske mreže s istim arhitekturama, a svaka od njih otkriva dio lica kao što su obrve, oči i usta. Prije uvođenja u konvolucijske neuronske mreže, slike prolaze kroz fazu rezanja i otkrivanja ključnih točaka lica. Tako složeni oblici spajaju se u lice koje se sastoji samo od obrva, očiju i usta te u kombinaciji s neobrađenom slikom uvode se u drugu vrstu konvolucijske neuronske mreže kako bi se otkrio izraz lica. Istraživači pokazuju da ova metoda pruža bolju točnost od upotrebe neobrađenih slika ili samo slaganja lica od ključnih značajki.

Jedan od kritičnih faktora koji utječe na točnost prepoznavanja su smetnje na licu poput naočala, maski ili većih ožiljaka. Stoga su Li i sur. [10] predstavili novu metodu konvolucijske neuronske mreže koja prvo podatke provlači kroz VGGNet mrežu, a zatim primjenjuje tehnike konvolucijske neuronske mreže s korištenjem dodatnih mehanizama. Za svoj model koristili su tri velike baze podataka: FED-RO, RAF-DB i AffectNet, od kojih će se zadnje dvije kasnije predstaviti detaljno te koristi unutar ovog rada.

Tablica 1. Usporedba točnosti prepoznavanja emocija prema bazama podataka koristeći raznovrsne konvolucijske neuronske mreže [1]

Autori	Baze podataka	Model	Točnost prepoznavanja
Mollahosseini i sur.	MultiPie[10] MMI[11] DISFA[34] FERA[12] SFEW[13] CK+ [14] FER2013[15]	CNN	94,7% 77,9% 55,0% 76,7% 47,7% 93,2% 61,1%
Lopes i sur.	CK+[14] JAFFE[16] BU-3DFE[17]	CNN	96,76% for CK+
Mohammadpour i sur	CK+[14]	CNN	97,01%
Yolcu i sur.	RafD[22]	CNN	94.44%
Li i sur.	RAF-DB[21] AffectNet[20]	ACNN	80.54% 54.84%

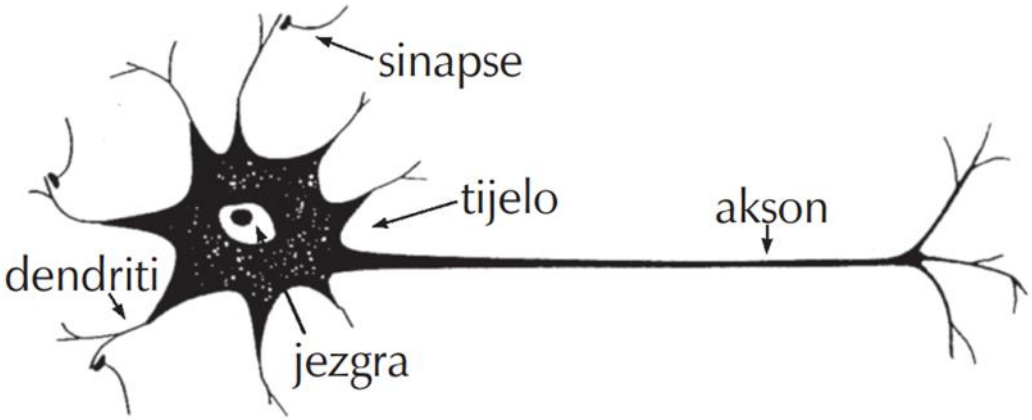
Iako se počeo pojavljivati sve veći broj dostupnih baza podataka za prepoznavanje emocija s lica osobe, često su sve ograničene na samo šest osnovnih emocija plus neutralna. Taj broj od sedam različitih stanja nije u skladu s onim što je prisutno u svakodnevnom životu. To je također jedna od motivacija za istraživače u budućnosti da izgrade veće baze podataka i motivacija stvaranju moćnijih arhitektura dubokog učenja za prepoznavanje svih osnovnih i sekundarnih emocija.

3. Umjetne neuronske mreže

Izraz umjetna neuronska mreža odnosi se na područje umjetne inteligencije zasnovano na biološkim strukturama mozga. Umjetna neuronska mreža je računalna mreža koja je građom slična ljudskom mozgu. Slično kao što ljudski mozak ima međusobno povezane neurone, umjetne neuronske mreže imaju neurone koji su međusobno povezani u različitim slojevima mreže, koji se još nazivaju čvorovi mreže.

3.1. Usporedba biološkog i umjetnog neurona

Ljudski mozak sastoji se od oko 1011 neurona podijeljenih na više od 1000 različitih vrsta. U prosjeku je svaki neuron povezan sa 104 susjednih neurona. Biološki neuron se sastoji od tri odnosno četiri osnovna dijela: dendrita, tijela stanice i aksona te sinapsi koje predstavljaju završetke aksona.



Slika 5. Biološki neuron [11]

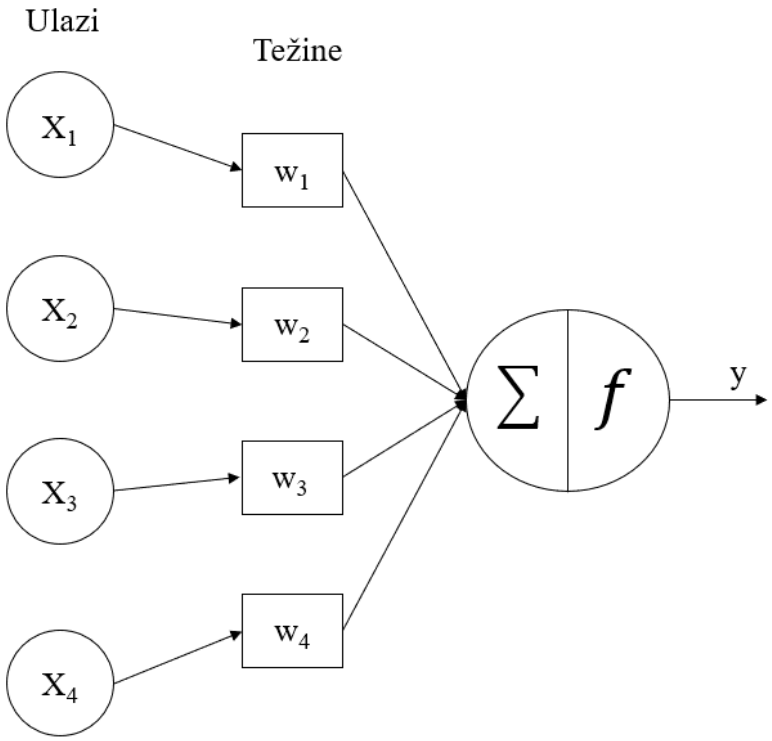
Dendriti su kratke niti oko tijela stanice koje prenose signale s drugih neurona i predstavljaju ulaze u mrežu. Tijelo stanice s jezgrom sadrži informaciju koja se može opisat kao električni potencijal između unutrašnjeg i vanjskog dijela stanice, a u umjetnoj neuronskoj mreži se oni opisuju kao čvorovi mreže. Sinapse su funkcionalne jedinice koje spajaju završetak jednog neurona s početkom sljedećeg, s ciljem prijenosa informacije u vidu post-sinaptičkog potencijala koji utječe na potencijal cjelokupne stanice na način da ga poveća ili smanji. Aksoni predstavljaju izlaze iz mreže, odnosno njihova zadaća je prijenos signala do drugih neurona pri čemu se grana u tanke niti.

Tablica 2. Usporedba elemenata biološke i umjetne neuronske mreže [12]

Biološka neuronska mreža	Umjetna neuronska mreža
Dendriti	Ulazi mreže
Jezgra	Čvor
Sinapse	Težine
Akson	Izlaz Mreže

3.2. Model umjetnog neurona

Umjetna neuronska mreža je učinkovit alat za modeliranje temeljen na podacima koji se koriste za dinamičko modeliranje i identifikaciju nelinearnih sustava, zbog svoje sposobnosti univerzalne aproksimacije i fleksibilne strukture koja omogućuje prepoznavanje složenih nelinearnih podataka. Općeniti model umjetnog neurona prikazan je na slici. Krugovi označeni s X_1, X_2 i X_3 predstavljaju ulaze u mrežu, a pravokutnici označavaju težine. Sumiranjem umnožaka svih ulaza i težina dobiva se vrijednost koja predstavlja ulaz u aktivacijsku funkciju. Slovom f označena je aktivacijska funkcija koja rezultira izlazom y.

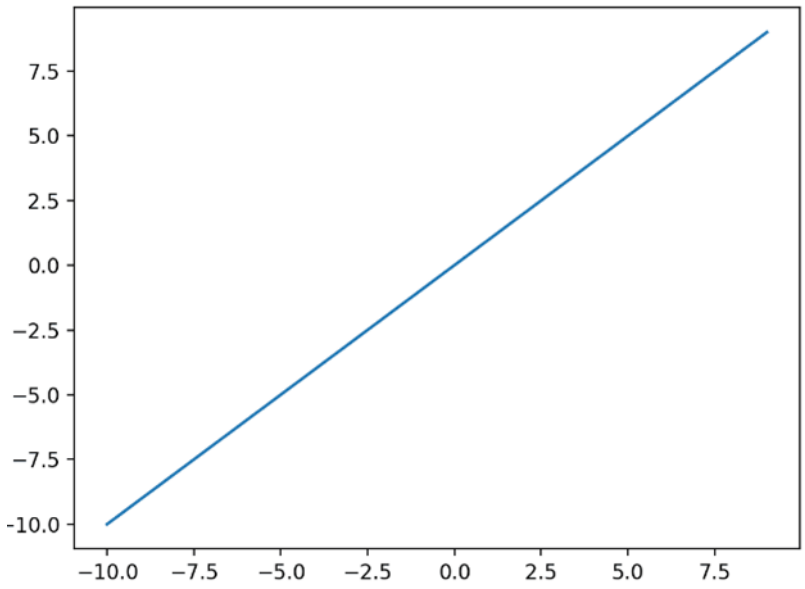


Slika 6. Arhitektura jednostavnog umjetnog neurona [12]

3.2.1. Aktivacijske funkcije

S ciljem postizanja nelinearnosti između slojeva u neuronskoj mreži, koriste se različite aktivacijske funkcije. S obzirom da aktivacijska funkcija direktno određuje izlaz iz neuronske mreže, od izuzetnog je značaja izabrati prikladnu aktivacijsku funkciju prilikom izrade modela neuronske mreže. Taj odabir aktivacijske funkcije će definirati sposobnosti modela neuronske mreže odnosno definirati njegova ograničenja. Neke od najčešćih aktivacijskih funkcija su linearna, softmax, sigmoidna i ReLU. Linearna aktivacijska funkcija ne ograničava vrijednosti u nekim granicama, stoga ne postiže nelinearnost. Definira se formulom:

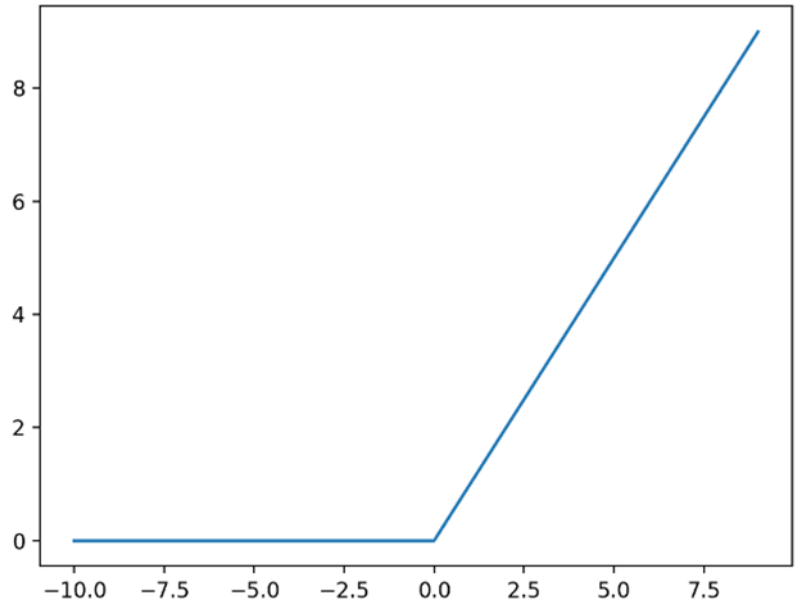
$$f(x) = x \tag{3.1}$$



Slika 7. Graf linearne aktivacijske funkcije[13]

Vjerojatno najčešće korištena aktivacijska funkcija je ReLU zbog lakoće implementacije i efikasnosti u rješavanju problema gradijenta funkcije gubitka koji se približava vrijednosti 0. Najveća mana ReLU aktivacijske funkcije je mogućnost izoliranja neželjenih neurona. Definira se formulom:

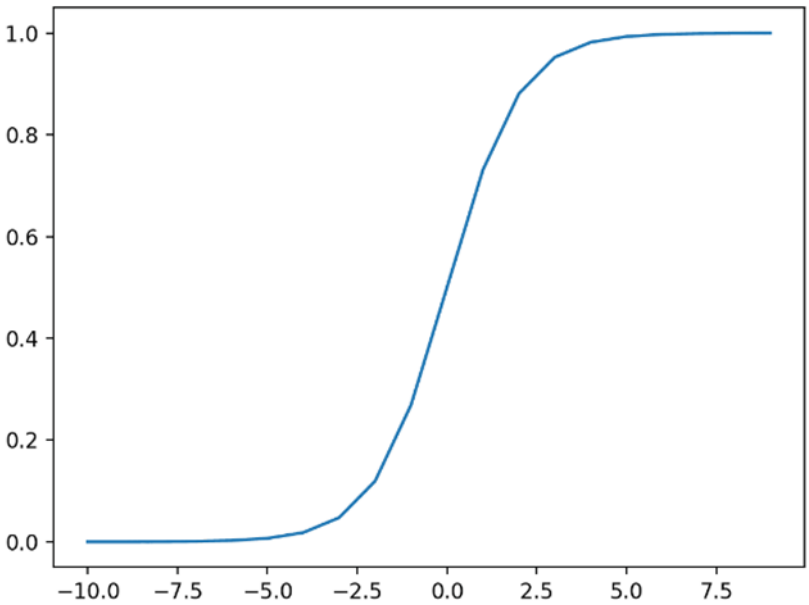
$$f(x) = max\,(0, x) \tag{3.2}$$



Slika 8. Graf ReLU aktivacijske funkcije [13]

U slučaju kada se javlja potreba da se ograniči vrijednost izlaza između 0 i 1, koristi se sigmoidna funkcija, često zvana funkcija logistike. Najčešće se koristi u modelima gdje je potrebno predvidjeti vjerojatnosti s obzirom da kao izlaz daje raspon između 0 i 1. Sigmoidna funkcija definirana je formulom:

$$f(x) = \frac{1}{1 + e^{-x}} \tag{3.3}$$



Slika 9. Graf sigmoidne aktivacijske funkcije [13]

Neželjeno svojstvo sigmoidne funkcije je problem gradijenta funkcije gubitka koji se približava vrijednosti 0 koji nije riješen kao u slučaju ReLU aktivacijske funkcije.

Naposlijetku softmax aktivacijska funkcija kao izlaz daje vektor čiji zbroj svih vrijednosti iznosi 1 i može se interpretirati kao klasa vjerojatnosti. Softmax aktivacijska funkcija je definirana formulom:

$$f(x) = \frac{e^{x_i}}{\sum_{j=1}^{K} e^{x_j}} \tag{3.4}$$

Također treba istaknuti da postoji još puno drugih funkcija koje se mogu koristiti kao aktivacijske funkcije u modelu neuronske mreže. Tablica 3 daje pregled najčešće korištenih funkcija.

Tablica 3. Matematički zapis osnovnih aktivacijskih funkcija

Naziv funkcije	$f(x)$
Linearna	$f(x)=x$
ReLU	$f(x)=\max(0,x)$
Sigmoidna	$f(x)=\frac{1}{1+e^{-x}}$
Softmax	$f(x)=\frac{e^{x_i}}{\sum_{j=1}^{K} e^{x_j}}$

3.2.2. Funkcija gubitka

Funkcija gubitka izračunava razliku između trenutnog izlaza iz neuronske mreže i očekivanog izlaza. Cilj je smanjiti tu razliku odnosno optimizirati naše izlaze iz neuronske mreže kako bi vrijednost funkcije gubitka za zadani ulaz bila što bliža 0. U zadacima klasifikacije, cilj je predvidjeti određenu klasu, dok je u zadacima regresije cilj predvidjeti konstantnu kontinuiranu vrijednost. Postoji puno različitih funkcija gubitaka te je potrebno izabrati odgovarajuću funkciju za problem koji se nastoji riješiti. Jedna od najčešće upotrebljavanih funkcija gubitka u modelima klasifikacije je funkcija unakrsne entropije često nazivana još i logaritamska funkcija gubitka. Za svaku klasu vrijednost izlaza koji predstavljaju vjerojatnosti u rasponu 0 do 1 se uspoređuju s traženim izlazom te se računa gubitak koji je razlika predikcije i traženog rezultata. S obzirom da je riječ o logaritmu, funkcija pokazuje veće gubitke blizu jedinice i

manje koje teže nuli. Također treba spomenuti da pogreška unakrsne entropije ima svojstvo da izbjegne problem usporavanja učenja.

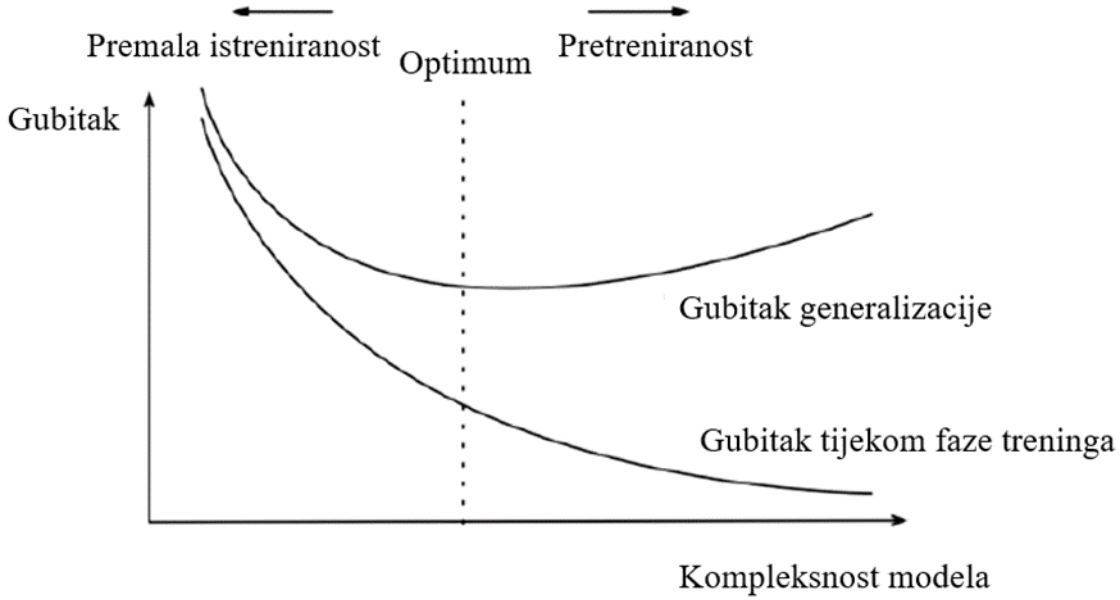
Unakrsna entropija definira se kao:

$$L_{CE} = -\sum_{i=1}^{n} t_i \cdot \log(p_i) \ , \tag{2.5}$$

za n klasa, gdje t_i označava traženu vrijednost, a p_i označava softmax vjerojatnost za n-tu klasu.

3.2.3. Pretreniranost neuronske mreže

Jedan od velikih izazova tijekom faze treninga neuronske mreže je izbjeći pretreniranost mreže. U situaciji kad se model dobro istrenira na podacima za trening i pokazuje izvrsne rezultate na njima, ali prilikom izvođenja na testnim podacima prikazuje loše rezultate, naziva se pretreniranost neuronske mreže. Ona se događa kada je model previše kompleksan te se prilikom obrade prikupi previše šumova u podacima za trening, stoga model ne može dobro generalizirati na novim podacima. Jedan od mogućih uzroka može biti premali skup podataka ili na podacima koji su previše slični.



Slika 10. Prikaz učenja neuronske mreže s optimumom i dijelovima premale i prevelike istreniranosti [14]

Regularizacijom neuronske mreže nastoji se riješiti problem pretreniranosti dok se istovremeno nastoje smanjiti pogreške prilikom treniranja. Može se reći da regularizacija čini model jednostavnijim. Postoje različite metode regularizacije, no jedna od najčešće korištenih u

dubokom učenju je metoda ranijeg zaustavljanja (Early stopping). Pogreška tijekom faze treninga se smanjuje svakom sljedećom epohom, ali u jednom trenutku započne rasti. Metoda ranijeg zaustavljanja nadzire više vrijednosti na testnim podatcima poput preciznosti te u trenutku kada iznos preciznosti prestane rasti, trening se obustavlja. To se radi zbog toga što se nakon te točke dvije krivulje razdvajaju, što je znak pretreniranosti. Ta se točka često naziva točka optimuma jer je to mjesto gdje je najbolji omjer koraka treninga i preciznosti neuronske mreže [14].

3.3. Konvolucijske neuronske mreže

Konvolucijska neuronska mreža je vrsta neuronske mreže koja se pokazala uspješnom u obradi podataka matričnih struktura. Iako je prva pomisao koja se veže uz konvolucijske neuronske mreže obrada slike, one također pokazuju odlične rezultate pri obradi zvuka, odnosno procesiranju ljudskog govora. Glavni gradivni elementi konvolucijske neuronske mreže su konvolucijski slojevi, slojevi sažimanja i softmax slojevi. Konvolucijski sloj sadrži skup filtera i parametara koji se uče u fazi treninga. Filterima se prolazi kroz sliku i provode matematičke operacije. U sloju sažimanja se mijenja dimenzija tenzora kako bi se reprezentacija učinila manjom i boljom za upravljanje. Uloga softmax sloja ili često još nazivanog potpuno povezanog sloja je da transformira sliku u vektor stupac koji predstavlja ulaz u unaprijednu neuronsku mrežu.

3.3.1. Konvolucijski sloj

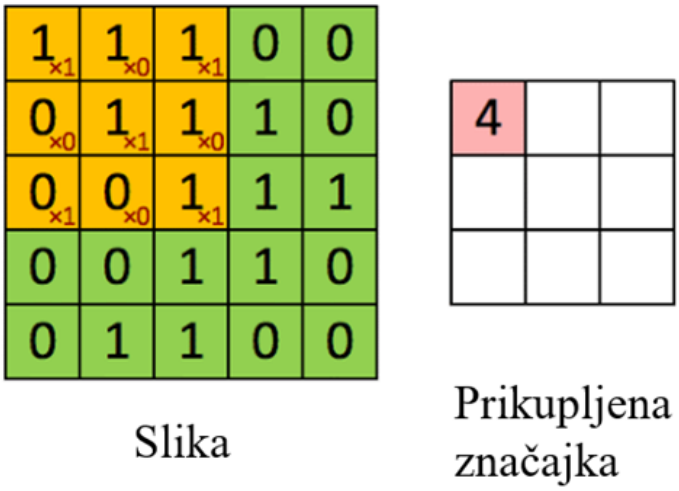
Centralni dio konvolucijske neuronske mreže je konvolucijski sloj po kojem je i cijela mreža dobila ime. Operacija konvolucije označava jednostavnu primjenu filtera na ulazni set podataka čije parametre treba učiti tijekom faze treniranja. Ponovljena primjena istovrsnog filtera na ulaz rezultira mapom aktivacija koja se još naziva mapa značajki. Mapa značajki sadrži informacije o lokaciji i intenzitetu detektirane značajke na ulazu kao što je slika.

Konvolucijski slojevi sastoje se od filtera odnosno težinskih matrica. Tipične veličine filtera su 3x3, 5x5 i 7x7. Treća dimenzija filtera odgovara broju kanala ulaza. Dubina slike u sivim tonovima je 1, a slika u boji ima 3 kanala boja (RBG). Tijekom unaprijedne faze svaki filter izvodi operaciju konvolucije po cijelom volumenu ulaznog podatka, odnosno izračunava umnožak filtera s dijelom ulaznih podataka i ponavlja se po širini i visini. Filtri će se aktivirati kada množenje po elementima rezultira visokim, pozitivnim vrijednostima. Ovo informira mrežu o prisutnosti nečega na slici, kao što je rub ili mrlja boje. Nakon ove operacije slijedi

nelinearna aktivacijska funkcija, najčešće sigmoidna, tangens hiperbolna ili ReLU. Vrijednosti izlazne mape značajki mogu se izračunati sljedećom formulom za konvoluciju [15]:

$$G[m,n] = (f * h)[m,n] = \sum_j \sum_k h[j,k] \cdot f[m-j, n-k] \tag{2.6}$$

gdje je ulazna slika označena s f, filter s h, a m i n predstavljaju indekse redaka i stupaca izlazne matrice. Dok filter prelazi preko piksela slike, posebna vrsta matričnog množenja označena s „*“ u svakoj pod-regiji ulaznog volumena spaja te značajke u mapu značajki.



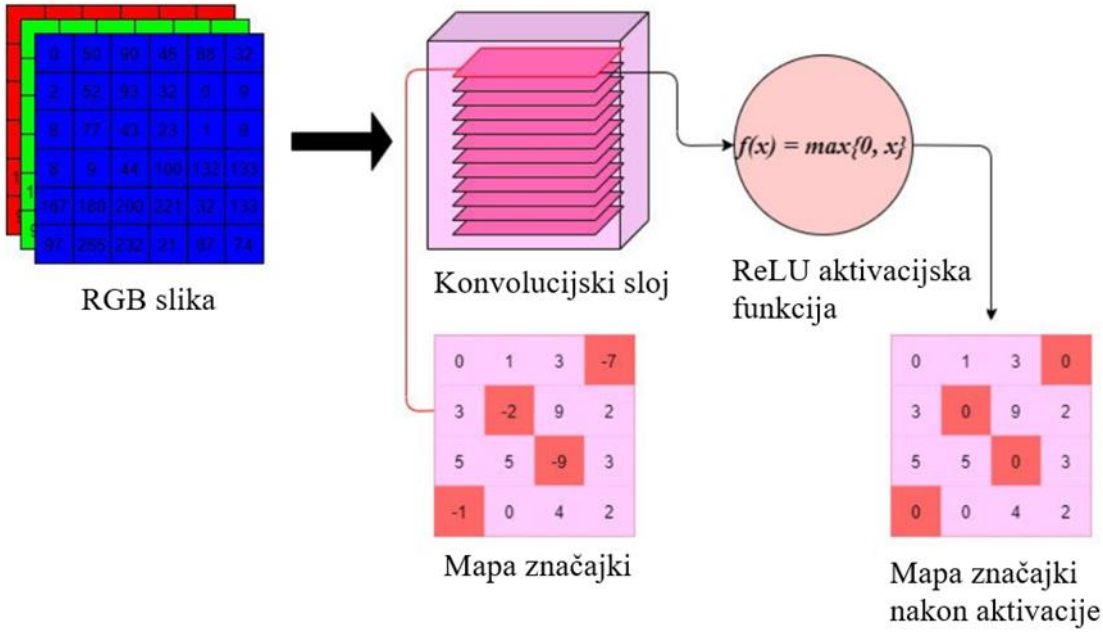
Slika 11. Prikaz operacije konvolucije [15]

Filteri sadrže parametre koji će direktno utjecati na veličinu izlaznog volumena. Prvi od njih je korak. Korak je udaljenost koju filter prijeđe između dvije operacije konvolucije. Filter s korakom od 1 pomicat će se preko ulazne slike 1 piksel između operacija. Padding je pojam koji označava količinu piksela dodanih slici kada je obrađuje filter. Najčešće je postavljen na vrijednost 0 jer će tada vrijednost piksela koji se dodaju biti 0 i neće utjecati na razumijevanje slike.

Nakon provedene operacije konvolucije, ulazna slika je pretvorena u mape značajki. Svaka mapa značajki odgovara vizualnoj značajki koja se vidi na određenim lokacijama unutar ulazne slike. Veličina ovog skupa značajki jednaka je broju čvorova (tj. filtera) u konvolucijskom sloju. Aktivacijske funkcije zatim određuju relevantnost određenog čvora u neuronskoj mreži. Čvor relevantan za predviđanje modela će se aktivirati nakon prolaska kroz aktivacijsku funkciju. Konvolucijska neuronska mreža kao aktivacijsku funkciju često koristi prethodno spomenutu ReLU funkciju. Aktivacijsku funkciju potrebno je primijeniti na tisuće čvorova istovremeno tijekom faze treninga. Kao primjer ReLU funkcija može biti definirana kao [15]:

$$ReLU'(x) = \{0, x < 0 | 1, x \geq 0\} \quad (2.7)$$

Iz jednadžbe 2.7 je vidljivo da će vrijednost čvorova koji imaju negativne vrijednosti biti ispisane kao 0, što čini irelevantnim za predviđanje. Ti se čvorovi neće aktivirati. Čvorovi čije su vrijednosti jednake ili veće od 0 će se aktivirati, jer se smatraju relevantnim za predviđanje. Može se zaključiti da ReLU aktivacijski slojevi ne mijenjaju veličinu skupa značajki.



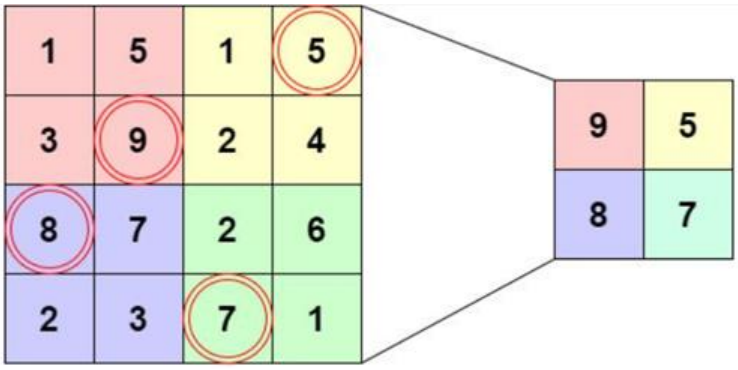
Slika 12. Struktura konvolucijskog sloja [15]

3.3.2. Sloj sažimanja (eng. Pooling Layer)

Izlaz konvolucijskog sloja je skup mapa vizualnih značajki. Kako se složenost skupa podataka povećava, tako mora rasti i broj filtera u konvolucijskim slojevima. Svaki filter ima zadatak identificirati različite vizualne značajke na slici. S povećanjem broja mapa značajki, povećava se dimenzionalnost, broj parametara i mogućnost pretreniranosti. U cilju da se riješe ti problemi, konvolucijske neuronske mreže imaju slojeve sažimanja (eng. Pooling layer) nakon konvolucijskih slojeva. Slojevi sažimanja uzimaju cijeli skup mapa značajki kao ulaz i provode operacije s ciljem smanjivanja uzorkovanja. Postoji više načina kako bi se provelo sažimanje, ali VGG16 koristi sloj sažimanja.

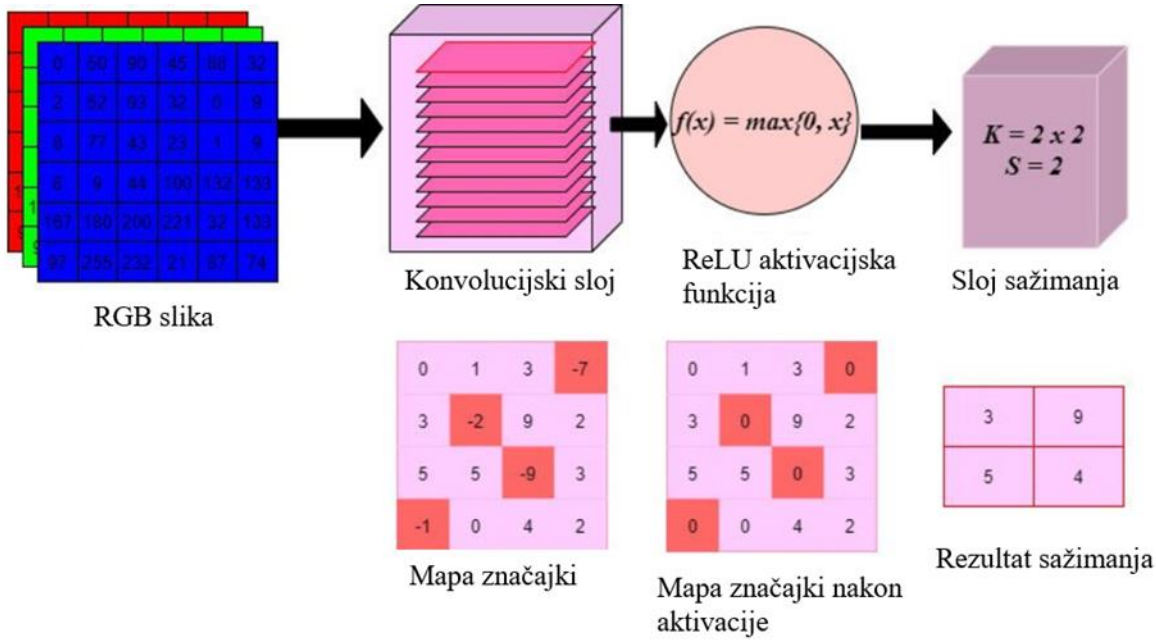
Slojevi sažimanja uzimaju širinu i visinu filtera te korak kao ulazne argumente. Filter započinje u gornjem lijevom kutu mape značajki, a zatim prelazi preko piksela udesno s određenim

korakom. Piksel s najvećom vrijednošću koji se nalazi u prozoru filtera će se koristit kao vrijednost za pripadajući čvor u sloju sažimanja [15].



Slika 13. Sažimanje mape značajki s filterom dimenzije 2x2 i korakom 2 [15]

Ako se uzme filter dimenzije 2x2 s korakom 2, prolaskom kroz mapu značajki dimenzija 4x4 reducira se njena dimenzija na pola. Sloj sažimanja se najčešće koristi za klasifikaciju slika. Iza toga stoji pretpostavka da će vizualne značajke imati veće vrijednosti piksela, stoga će sloj sažimanja identificirati te značajke pritom zadržavajući prostorni raspored značajki.

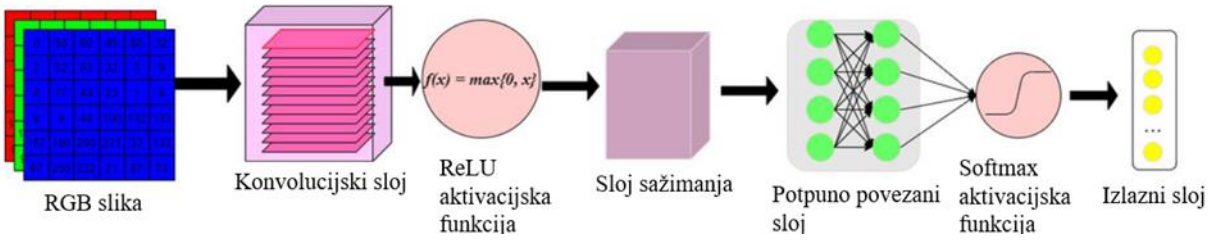


Slika 14. Struktura sloja sažimanja [15]

3.3.3. Potpuno povezani sloj (eng. Fully-Connected Layer)

Potpuno povezane neuronske mreže su izgrađene od slojeva pri čemu je svaki čvor povezan sa svim drugim čvorovima u prethodnom sloju. Takav sloj nalazi se na kraju arhitekture konvolucijske neuronske mreže kako bi se napravilo predviđanje s obzirom na naučene značajke. Cilj potpuno povezanog sloja je napraviti predviđanje klase. Na ulaz ovog sloja dolazi

vektor čvorova koji je prethodno prošao sloj konvolucije i sažimanja te su određeni čvorovi aktivirani. Vektorski ulaz prolazi kroz dva do tri potpuno povezanih slojeva te kroz konačnu aktivacijsku funkciju prije nego što bude proslijeđen izlaznom sloju. Aktivacijska funkcija koja se koristi ne mora biti ReLU, a dvije najčešće korištene funkcije su sigmoidna i softmax. Sigmoidna funkcija se najčešće koristi za binarni klasifikacijski problem jer je to zapravo funkcija logistike. Softmax funkcija osigurava da suma vrijednosti u izlaznom sloju iznosi 1 te se ova funkcija stoga može koristiti i za binarnu klasifikaciju i za klasifikaciju sustava s više klasa [15].



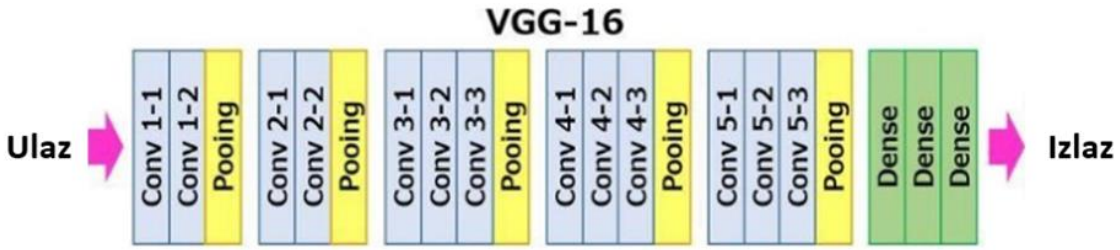
Slika 15. Koraci konvolucijske neuronske mreže [15]

3.4. VGG16

VGG16 je konvolucijska neuronska mreža koja potječe od Karen Simonyan i Andrew Zisserman sa Sveučilišta u Oxfordu. Također je poznata kao ConvNet, a sastoji se od ulaznog, izlaznog i 16 težinskih slojeva. Ovaj model pokazao se kao značajna prekretnica u računalnom vidu jer je počeo upotrebljavati 3x3 filtere s korakom od 1 piksela kroz sve slojeve neuronske mreže, za razliku od AlexNeta koji upotrebljava filter 11x11 s korakom 4 i ZFNeta koji upotrebljava filter 7x7 s korakom 2. Ideja korištenja 3x3 filtera je ta što on može lako replicirati 5x5 ili 7x7 filter kombinacijom više uzastopnih 3x3 filtera.[4] Iako bi se moglo pomisliti da se smanjenjem dimenzije filtera povećava broj slojeva i tako nepotrebno povećava kompleksnost mreže, korištenjem više 3x3 filtera umjesto jednog 7x7 bi se povećao broj konvolucijskih slojeva, ali isto tako i broj nelinearnih aktivacijskih slojeva, što za rezultat ima lakše odlučivanje i brže konvergiranje mreže [16].

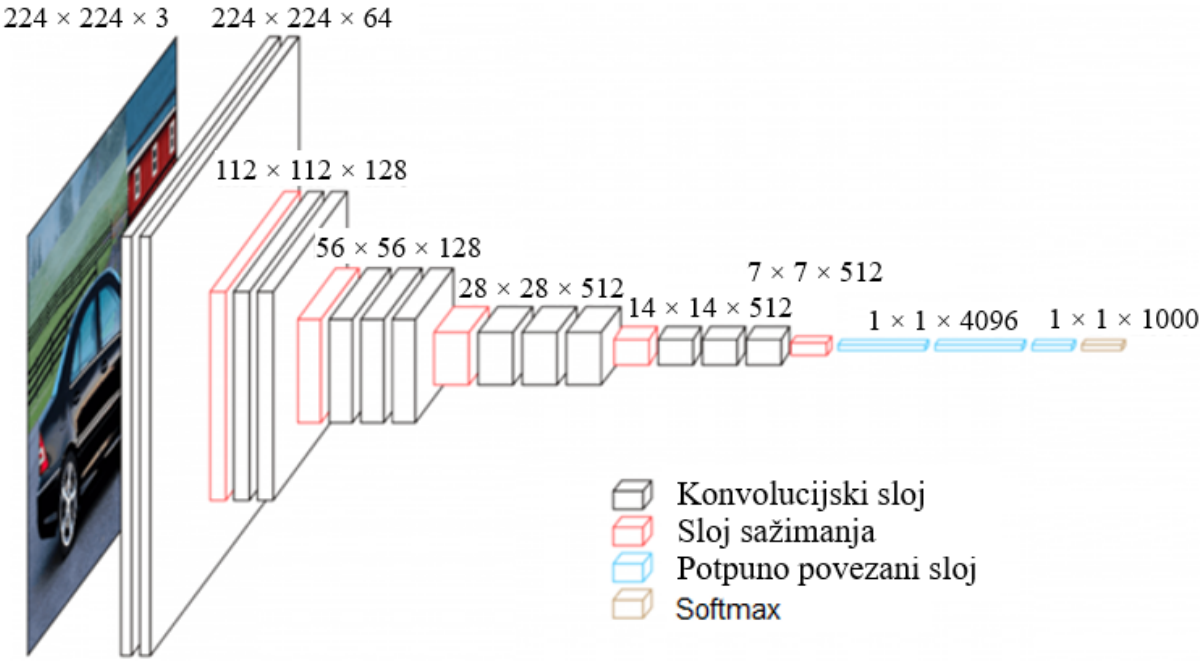
3.4.1. Arhitektura VGG16 konvolucijske neuronske mreže

VGG16 mreža se sastoji od 21 sloja od kojih 16 njih ima težine odnosno parametre koje je potrebno naučiti (slika 4.). U VGG16 mreži postoji 13 konvolucijskih slojeva, 5 slojeva sažimanja i 3 softmax sloja. [16]



Slika 16. Prikaz slojeva VGG16 konvolucijske neuronske mreže [17]

Prema slici VGG16 na ulazu zahtjeva tenzor veličine 224x224x3 što označava sliku veličine 224x224 piksela s 3 RGB kanala. Prije samog ulaza u mrežu potrebno je izvršiti normalizaciju RGB vrijednosti za svaki piksel slike i to na način da se oduzme srednja vrijednost od svakog piksela. Slika zatim prolazi kroz prva dva konvolucijska sloja od koji svaki od njih sadrži 64 filtera dimenzija 3x3. Zatim se u sloju sažimanja mijenja dimenzija slike kako bi bila pogodna za sljedeća dva konvolucijska sloja koji sadrže po 128 filtera. Izmjena konvolucijskih i slojeva sažimanja se ponavlja 5 puta nakon čega slijede 3 softmax sloja nakon čega se dobije vektor dimenzije 1x1000 što predstavlja 1000 ulaznih neurona za unaprijednu neuronsku mrežu [17].



Slika 17. Dimenzije na ulazu pojedinog sloja VGG16 [17]

4. Baze podataka

Postoje različite baze podataka koje se koriste u istraživačke svrhe prilikom razvijanja modela dubokog učenja. Mnoge od njih su besplatne u svrhu istraživanja u edukacijske svrhe te su dostupne za preuzimanje s web stranica. Najčešće se baza podataka direktno skida s interneta, no ponekad postoji samo tekstualna datoteka koja sadrži popis poveznica na slike. Svaki model dubokog učenja treba podatke za obuku, a kasnije i za testiranje odnosno za procjenu izvedbe na neviđenim instancama. Označeni skupovi podataka za prepoznavanje emocija s lica osoba trebaju biti što raznovrsniji pokrivajući mnoge varijacije unutar klase kao što su načini izražavanja pojedine emocije, poza osobe, osvjetljenje i ostalo. U tablici su vidljive najpoznatije baze podataka, a unutar ovog rada korištene su AffectNet, EmotioNet i RAF-DB baze podataka.

Tablica 4. Najpoznatije baze podataka za prepoznavanje emocija s lica osoba na slici [1]

Baza podataka	Opis	Klase (emocije)
MultiPie	Više od 750.000 slika snimljenih s 15 pogleda i 19 uvjeta osvjetljenja	Ljutnja, gađenje, sreća, vrisak, škiljenje, iznenađenje, neutralno
FER2013	35.887 slika u svim tonovima prikupljenih s Google pretraživača	6 osnovnih emocija i neutralno stanje
AffectNet	Više od 440.000 slika prikupljenih s interneta	6 osnovnih emocija i neutralno stanje
RAF-DB	30.000 slika iz stvarnog svijeta	6 osnovnih emocija i neutralno stanje
CK+	593 video uratka s osobama koje poziraju ili su spontane	6 osnovnih emocija, neutralno stanje i prijezir
BU-3DFE	2500 3D slika lica fotografiranih iz dva smjera -45°, +45°	6 osnovnih emocija i neutralno stanje

EmotioNet	Preko 1.000.000 slika izraza lica preuzetih s WordNeta s povezanim ključnim riječima emocija s interneta	16 osnovnih emocija i neutralno stanje

4.1. AffectNet

AffectNet je najveća baza podataka modela ljudskih lica u neuređenom okruženju. Sadrži oko 1 milijun slika lica prikupljenih s interneta upitima na tri glavne tražilice pomoću 1250 ključnih riječi povezanih s emocijama na šest različitih jezika. Otprilike polovica dohvaćenih slika ručno je označena za prisutnost sedam diskretnih izraza lica (šest osnovnih emocija i neutralno stanje) te za intenzitet za jačinu ugodnosti i uzbuđenja. Druga polovica slika automatski je označena koristeći ResNext neuronsku mrežu obučenu na svim ručno označenim uzorcima seta za obuku s prosječnom točnošću od 65% [18]. AffectNet je daleko najveća postojeća baza podataka s izrazima lica, ugodnosti i uzbuđenja u neuređenom okruženju koja omogućuje istraživanje automatiziranog prepoznavanja izraza lica prema dva različita modela emocija.

Slika 18. Sadržaj AffectNet baze podataka [18]

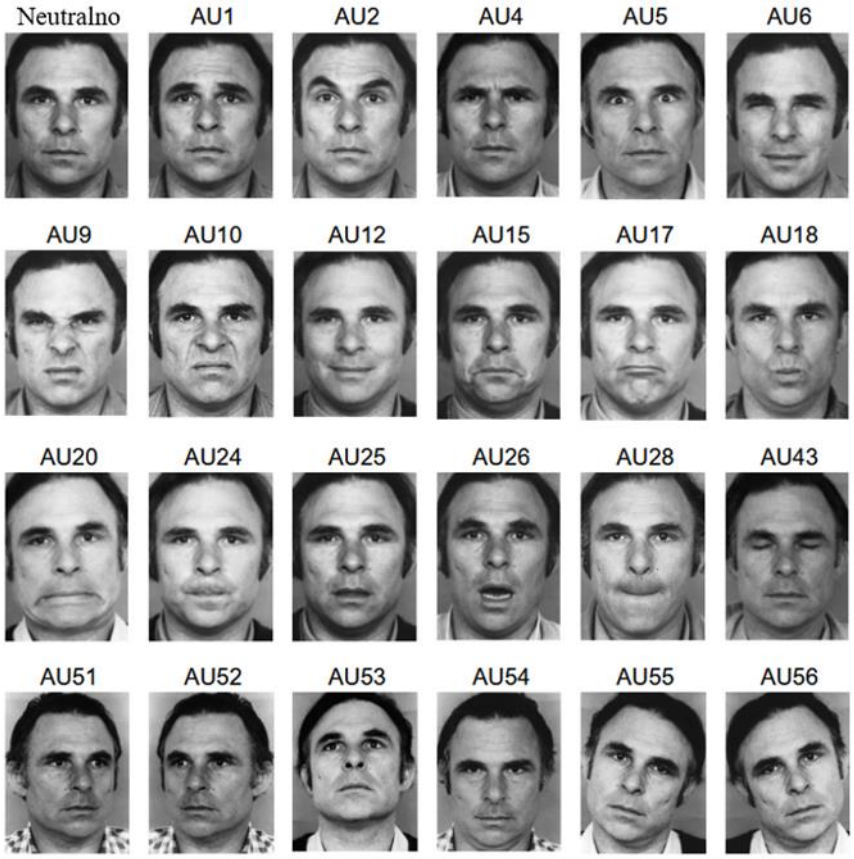
Tablica 5. Raspodjela slika prema emocijama unutar AffectNet baze podataka [18]

Emocija	Broj slika
Neutralno stanje	75.374
Sreća	134.915
Tuga	25.959
Iznenađenje	14.590
Strah	6.878
Gađenje	4.303
Ljutnja	25.382
Bez lica	82.915
Ukupno	370.316

4.2. EmotioNet

EmotioNet baza podataka razvijena od strane djelatnika Sveučilišta u Ohio-u te je dostupna za istraživačke svrhe. Baza podataka sadrži 950.000 slika s označenim akcijskim jedinicama lica. To je označeno na način da su se u WordNetu pretraživale sve slike koje proizlaze prilikom unosa riječ „osjećaj“ (engl. „feeling“), a kasnije prema ključnim riječima emocija i njihovim sinonimima. Nakon pretraživanja filtrirali su slike koje sadrže lice osobe koristeći razne algoritme i tako kreirali bazu podataka. Kako bi uspješno dodijelili oznaku svakoj slici, koristili su tri dostupne baze podataka, s ručno dodanim oznakama, kako bi istrenirali mrežu za raspodjelu slika u klase. Slike unutar ove baze podataka su klasificirane prema akcijskim jedinicama lica, intenzitetu akcijske jedinice, osnovnim i složenim emocijama [19].

EmotioNet koristi FACS sustav kojeg je razvio Paul Ekman. Sustav kodiranja radnji lica (FACS) sveobuhvatan je, anatomski utemeljen sustav za opisivanje svih vizualno uočljivih pokreta lica. Rastavlja izraze lica na pojedinačne komponente pokreta mišića, koje se nazivaju akcijske jedinice (AU). U sklopu ove baze podataka slike su podijeljene prema akcijskim jedinicama. Kasnije se akcijske jedinice koriste za definiranje pojedine emocije jer određeni mišići lica se kontrahiraju prilikom smijeha, dok se ti isti relaksiraju prilikom tuge.



Slika 19. Anotacije prema FACS sustavu [20]

Tablica 6. Pridruživanje akcijskih jedinica pojedinoj emociji [21]

Kategorija	Akcijske jedinice
Sreća	12, 25
Tuga	4, 15
Strah	1, 4, 20, 25
Ljutnja	4, 7, 24
Iznenađenje	1, 2, 25, 26
Gađenje	9, 10, 17
Neutralno stanje	Ostalo

Anotacija prema akcijskim jedinicama najprije je pretvorena u anotaciju prema emocijama koristeći prethodnu tablicu. Slikama su dodijeljene emocije na način da su aktivne akcijske jedinice uspoređene s onima koje su potrebne za određenu emociju.

4.3. RAF-DB

RAF-DB (engl. Real-world Affective Faces Database) je skup podataka za izraze lica. Sadrži 30.000 slika lica označene osnovnim ili složenim emocijama. Slike u ovoj bazi podataka sadrže velike varijacije u svim ljudskim karakteristikama, poput dobi i spola, ali također i varijacije položaja glave, osvjetljenja i mogućih dodataka poput naočala, kapa i slično. U početku URL slika su prikupljani s Flickra te su uneseni u automatski program za preuzimanje. Zatim je sortirano prema ključnim riječima kako bi se dobila klasifikacija prema šest osnovnih emocija plus neutralno stanje [22]. Baza podataka je podijeljena na skup za treniranje i skup za testiranje, pri čemu je skup za treniranje 5 puta veći od skupa za testiranje, dok je distribucija emocija u oba skupa gotovo identična.

Slika 20. Sadržaj RAF-DB baze podataka [22]

Iako već postoje brojne baze podataka koje nastoje klasificirati slike prema emocionalnim stanjima, važno je prepoznati da postoji nekoliko različitih aspekata koji utječu na kvalitetu njenog sadržaja. Način prikupljanja podataka je bitan faktor, ali od onih koji se odnose na podatke bitna je varijacija osobe sa slike prema dobu, spolu, rasnoj pripadnosti te njezina kulturna pozadina. Sve to može utjecati na rezultate tako da se istrenira mreža za specifičnu skupinu ljudi i da ne bude univerzalno primjenjiva. Odabrane baze podataka su sveobuhvatne i varijantne prema svim prethodno spomenutim parametrima te sadrže samo slike s tri kanala boje. No, ono po čemu se razlikuju je to što RAF-DB sadrži slike koje su centrirane tako da je lice osobe stavljeno u središte kadra, dok to nije slučaj kod druge dvije baze podataka.

5. Faza treniranja mreže

U ovom poglavlju slijedno će se obraditi sva potrebna oprema i alati, a zatim i pojasniti programski kod koji je potreban za uspješno provođenje faze treninga.

5.1. Programska podrška

U sklopu diplomskog rada korišten je programski jezik Python. Python je računalni programski jezik koji se često koristi za izradu web stranica i softvera, automatizaciju repetitivnih zadataka i provođenje raznih analiza nad podacima. To je objektno orijentirani programski jezik opće namjene, što znači da se može koristiti u različite svrhe te nije specijaliziran za bilo kakve specifične probleme. To svojstvo, zajedno s time što je intuitivan pa početnici vrlo brzo napreduju, učinilo ga je jednim od najčešće korištenih programskih jezika danas. Python podržava različite module i pakete što potiče modularnost programa i ponovnu upotrebu koda, a opsežna standardna biblioteka dostupna je u izvornom obliku bez naknade. Budući da nema koraka kompilacije, ciklus uređivanja-testiranja-otklanjanja pogrešaka je nevjerojatno brz. Otklanjanje pogrešaka u Python programima je jednostavno: greška ili loš unos nikada neće uzrokovati pogrešku segmentacije. Umjesto toga, kada tumač otkrije pogrešku, pokreće iznimku. Kada program ne uhvati iznimku, tumač ispisuje trag stoga (engl. Stack traceback). Program za ispravljanje pogrešaka na razini izvora omogućuje pregled lokalnih i globalnih varijabli, procjenu proizvoljnih izraza, postavljanje prijelomnih točaka, prolazak kroz kod redak po redak, i tako dalje. Program za ispravljanje pogrešaka napisan je u samom Pythonu, što svjedoči o Python-ovoj introspektivnoj moći [23]. Ono što je možda najvažnije, Python ima ogromnu zajednicu korisnika. Python-ova popularnost je i uzrok i posljedica njegove zajednice. To znači da bez obzira koji problem se pokušava riješiti, velike su šanse da već postoje eksperti koji rade na rješavanju istoga. Zbog svih navedenih prednosti odabran je za izradu programskog koda ovog rada.

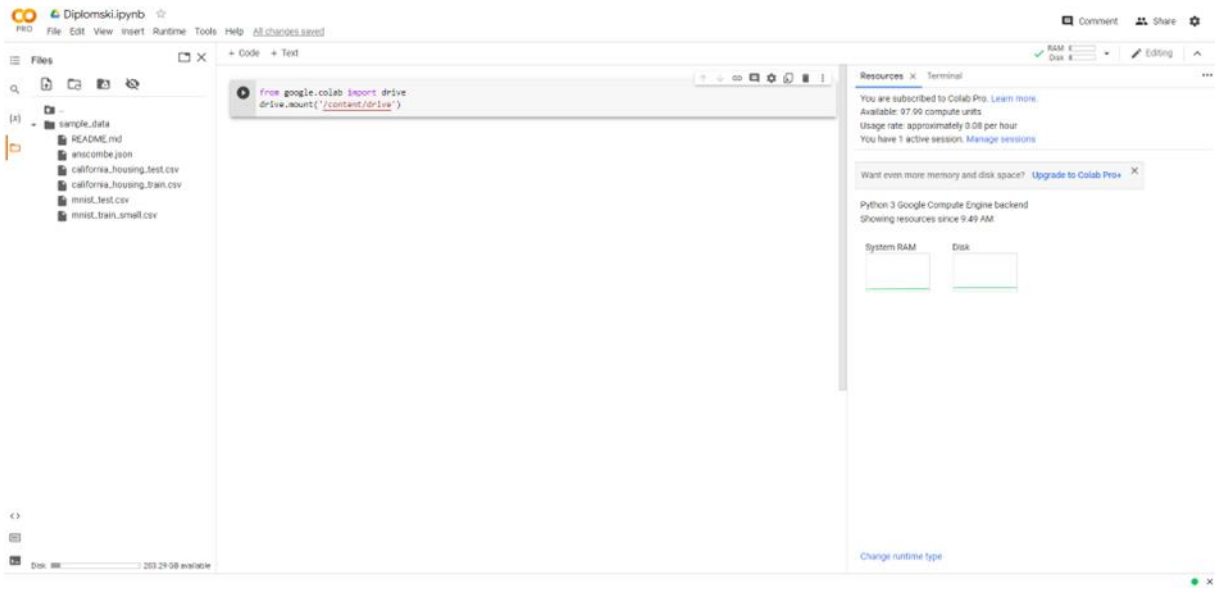
Programski kod pisan je unutar Google Colaboratory okruženja. Google Colaboratory ili skraćeno Colab je proizvod Google Researcha. Colab omogućuje svakom korisniku pisanje i izvršavanje proizvoljnog koda putem preglednika, a posebno je pogodan za strojno učenje, analizu podataka i za korištenje u edukacijske svrhe. Tehnički gledano, Colab je web dostupna Jupyter Notebook usluga za prijenosna računala koja ne zahtjeva postavljanje za korištenje, a pruža besplatan pristup računalnim resursima uključujući GPU-ove. Besplatna verzija Colab-a

osigurava 12GB RAM-a s ograničenim pristupom virtualnim strojevima s velikom količinom memorije koji imaju 25GB RAM-a. U sklopu ovog rada korištena je Colab Pro verzija s velikom količine memorije od 32GB RAM-a. No s obzirom da će se koristiti u svrhu dubokog učenja, važnija je grafička procesorska jedinica odnosno skraćeno GPU. Besplatna verzija Colab-a koristi Nvidia Tesla K80 GPU, dok Colab Pro osigurava upotrebu T4 ili P100 GPU [24].

Tablica 7. Usporedba dostupnih resursa u različitim verzijama Google Colaba [24]

Komponenta	Google Colab	Google Colab Pro	Google Colab Pro+
GPU	Nvidia K80	Nvidia K80, P100 ili T4	Nvidia P100, T4 ili V100
CPU	2 x vCPU	2 x vCPU	2 x vCPU
RAM	12 GB	32 GB	52 GB
Cijena (Euro)	besplatan	9.25	42.25

Korisničko sučelje je prikazano na slici. Kako bi se započelo s radom u Google Colab-u, potrebno je imati Google račun. Odmah nakon izrade računa, omogućen je pristup svim prethodno navedenim značajkama te se mogu započeti trenirati kompleksni modeli dubokog učenja bez brige o potrebnim računalnim resursima. Unatoč svemu spomenutom, Google Colaboratory nije bez svojih ograničenja. Sve očitiji postaju njegovi nedostatci kako raste količina podataka koje je potrebno učitati i obraditi.



Slika 21. Korisničko sučelje Google Colaboratory-a

5.2. Tijek programa

U sklopu ovog poglavlja obraditi će se svi potrebni koraci za učitavanje baza podataka i njihovo treniranje. S obzirom da je Colab dio Google-a, integracija ove dvije aplikacije je vrlo jednostavna te se kod generira automatski prilikom pritiska gumba za povezivanje s Google diskom.

Programski kod 1. Povezivanje Google Colaboratory-a s Google diskom

```
1. from google.colab import drive
2. drive.mount('/gdrive')
```

5.2.1. Faza predprocesiranja

Prilikom preuzimanja prethodno navedenih baza podataka (AffectNet, EmotioNet i RAF-DB) vidljiva je razlika u načinu na koji je određena baza pohranjena. EmotioNet i RAF-DB baze podataka preuzete su u obliku Excel datoteke, a formatirane na način da prvi stupac označava URL slike na mjesto na serveru gdje ju je moguće preuzeti. Nadalje, drugi stupac označava izvorno mjesto slike, odnosno mjesto s kojeg je ona preuzeta i dodana na server te baze podataka. Preostali stupci u RAF-DB bazi podataka sadrže informacije o emociji koju predstavlja ta slika. Ponekad slike nisu anotirane prema emocijama, nego prema akcijskim jedinicama, što je slučaj kod EmotioNet baze podataka. U tom slučaju preostali stupci sadrže informacije o aktivnim akcijskim jedinicama lica po FACS sustavu. Kod AffectNet baze podataka preuzima se datoteka koja sadrži sve slike i anotacije te nije potrebno preuzimati svaku sliku zasebno preko njenog URLa. Zapis o anotaciji svake slike spremljen je u obliku .npy datoteke. Upotrebom naredbe numpy.load() moguće je učitati sve anotacije i spremiti ih u obliku prigodnom za ulaz u mrežu.

Programski kod 2. Učitavanje i obrada anotacija AffectNet baze podataka

```
1. import numpy
2. import pandas as pd
3. import os.path
4. m=numpy.zeros((3999,12))
5. br=0
6. for i in range(0,5500):
7.      name="C:/Users/Marko_Marijic-
   DIPLOMSKI_RAD/Baze_podataka/AffectNet/val_set/val_set/annotations/" +
   str(i)+"_exp.npy"
8.      if os.path.exists(name):
9.          x1= int(numpy.load(name))
10.         m[br][0]=i
11.         m[br][x1+1]=1
12.         br+=1
```

```
13. pd.DataFrame(m).to_csv("Annottaions_affect_net.csv")
```

Nadalje potrebno je sve slike prenijeti na Google Disk kako bi ih bilo moguće učitati koristeći Colab i koristiti u programu. Oznake emocija na slici učitane su i spremljene u obliku rječnika. Rječnik je tip podataka koji je pogodan za korištenje kada program obrađuje velik broj podataka, a ovdje se on koristi u svrhu označavanja slika jer se podaci unutar rječnika dohvaćaju pomoću ključa. Kao ključ postavljeno je ime slike, a vrijednost je brojčana vrijednost emocije.

Programski kod 3. Obrada i mapiranje anotacija RAF-DB baze podataka

```
1. import csv
2. emotions = {
3.     1 : "Iznenadenje",
4.     2 : "Strah",
5.     3 : "Gadenje",
6.     4 : "Sreca",
7.     5 : "Tuga",
8.     6 : "Ljutnja",
9.     7 : "Neutralno"
10.}
11.def get_labels_map(path):
12.  map = {}
13.  with open(path) as file:
14.    tsv_file = csv.reader(file, delimiter=" ")
15.    for line in tsv_file:
16.      size = len(line[0])
17.      map[line[0][:size-4]] = int(line[1])
18.  return map
19.labels = get_labels_map("/gdrive/MyDrive/Diplomski/RAF-
   DB/list_patition_label.txt")
```

Ono što je još potrebno prije nego je moguće prijeći na fazu treninga je učitati sve slike povezane uz te oznake. Slike je najprije potrebno učitati koristeći cv2.imread(), a zatim i prilagoditi veličinu koristeći naredbu cv2.resize(). Cilj je da sve slike budu dimenzija 224×224×3 kako bi sve mreže imale isti ulaz pa tako i isti broj parametara za učenje.

Programski kod 4. Predprocesiranje slika i anotacija

```
1. import numpy as np
2. import cv2
3. import os
4.
5. base_path = "/gdrive/MyDrive/Diplomski/RAF-DB"
6.
7. def load_images_and_labels_from_folder(folder, label_map,x):
8.     resized = np.zeros((x,224,224,3))
9.     labels = np.zeros((x,7))
10.    br=0
11.    for filename in os.listdir(folder):
12.        img = cv2.imread(os.path.join(folder,filename))
```

```
13.         if img is not None:
14.             res = cv2.resize(img, dsize=(224, 224))
15.             reshaped = res.reshape(1,224,224,3)
16.             resized[br][:][:][:]=reshaped
17.             label_key = filename.split("_aligned")[0]
18.             l = np.zeros((1,7))
19.             index = label_map[label_key] - 1
20.             l[0][index] = 1
21.             labels[br][:]=l
22.             br+=1
23.     return resized, labels
24. train_images, train_labels = load_images_and_labels_from_folder(base_path
    + "/Train", labels,12271)
25. test_images, test_labels = load_images_and_labels_from_folder(base_path +
    "/Test", labels,3068)
```

Ukoliko se prethodni koraci uspješno provedu, faza predprocesiranja je gotova i sve je spremno za fazu treninga. Ono što je poznato da su slike za trening spremljene u obliku tenzora koji je nazvan train_images te je oblika (x,y,z,w), od kojih *x* označava broj podataka u bazi odnosno broj slika, a varijable *y, z* i *w* označavaju dimenzije slike (224×244×3). Zbog ograničenih resursa neće se provoditi treniranje sa svim podacima iz mreže. Broj slika korišten u fazi treninga vidljiv je u sljedećoj tablici. Iako pojedine baze podataka sadrže izuzetno veliki broj slika, nasumično je uzet uzorak slične veličine kao cjelokupna baza podataka RAF-DB. Na taj način moći će se usporediti kvaliteta sadržaja baza podataka, bez da na to utječe veličina baze podataka.

5.2.2. Faza treninga

U fazi treninga koristit će se podaci koji su prethodno obrađeni u fazi predprocesiranja. Prvi korak ove faze je izrada modela konvolucijske neuronske mreže. Kao što je prethodno spomenuto korištena je VGG16 konvolucijska neuronska mreža kojoj je na ulazu potrebna slika odnosno tenzor 3. reda dimenzija 224×224×3. Na izlazu mreže nalazi se sedam neurona od kojih svaki označava jedno emocionalno stanje. Za kreiranje modela neuronske mreže koristi se biblioteka Keras. Keras je javno dostupna biblioteka koja omogućuje jednostavan i brz rad s dubokim neuronskim mrežama. Modularnost joj je najveća prednost jer se slojevi grade s naredbom model.add(), a aktivacijske funkcije i mnoštvo drugih alata implementirani su u njoj te je promjena ili dodavanje sloja moguća u vrlo kratkom vremenu.

Programski kod 5. Biblioteke nužne za izradu modela neuronske mreže

```
1. import keras,os
2. from keras.models import Sequential
3. from keras.layers import Dense, Conv2D, MaxPool2D , Flatten
4. from keras.preprocessing.image import ImageDataGenerator
```

```
5. from tensorflow.keras.optimizers import Adam
6. import numpy as np
```

Odabir algoritma za optimizaciju izuzetno je bitan za model dubokog učenja i može jako ubrzati proces učenja. Optimizacijski algoritam Adam se koristi umjesto klasične metode stohastičkog gradijentnog spuštanja te se pokazao kao jedan od najboljih optimizacijskih algoritama u većini slučajeva. Koristi kvadrate gradijenata za skaliranje stopa učenja i promjenjiv prosjek gradijenta čime upotrebljava sve prednosti zamaha.

Programski kod 6. Kreiranje modela VGG16 s optimizacijskim algoritmom Adam

```
1. opt = Adam(learning_rate=0.0001)
2. vgg16.compile(optimizer=opt ,
   loss=keras.losses.categorical_crossentropy, metrics=['accuracy'])
```

Na slici je prikazan ispis strukture kreirane mreže. Vidljiv je izuzetno velik broj parametara što je svojstveno za sve duboke konvolucijske neuronske mreže.

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_13 (Conv2D)           (None, 224, 224, 64)      1792
conv2d_14 (Conv2D)           (None, 224, 224, 64)      36928
max_pooling2d_5 (MaxPooling  (None, 112, 112, 64)      0
2D)
conv2d_15 (Conv2D)           (None, 112, 112, 128)     73856
conv2d_16 (Conv2D)           (None, 112, 112, 128)     147584
max_pooling2d_6 (MaxPooling  (None, 56, 56, 128)       0
2D)
conv2d_17 (Conv2D)           (None, 56, 56, 256)       295168
conv2d_18 (Conv2D)           (None, 56, 56, 256)       590080
conv2d_19 (Conv2D)           (None, 56, 56, 256)       590080
max_pooling2d_7 (MaxPooling  (None, 28, 28, 256)       0
2D)
conv2d_20 (Conv2D)           (None, 28, 28, 512)       1180160
conv2d_21 (Conv2D)           (None, 28, 28, 512)       2359808
conv2d_22 (Conv2D)           (None, 28, 28, 512)       2359808
max_pooling2d_8 (MaxPooling  (None, 14, 14, 512)       0
2D)
conv2d_23 (Conv2D)           (None, 14, 14, 512)       2359808
conv2d_24 (Conv2D)           (None, 14, 14, 512)       2359808
conv2d_25 (Conv2D)           (None, 14, 14, 512)       2359808
max_pooling2d_9 (MaxPooling  (None, 7, 7, 512)         0
2D)
flatten_1 (Flatten)          (None, 25088)             0
dense_3 (Dense)              (None, 4096)              102764544
dense_4 (Dense)              (None, 4096)              16781312
dense_5 (Dense)              (None, 7)                 28679
=================================================================
Total params: 134,289,223
Trainable params: 134,289,223
Non-trainable params: 0
_________________________________________________________________
```

Slika 22. Ispis strukture neuronske mreže VGG16

Ono što je još preostalo za učiniti je definirati kontrolne točke. Nakon svake epohe učenja potrebno je usporediti vrijednosti točnosti testiranja trenutne epohe s prethodnom i u slučaju napretka, spremiti vrijednosti težina trenutnog koraka učenja kao predstavnika ovog modela. Također, definira se i rano zaustavljanje u slučaju da nema napretka u točnosti pet epoha u nizu. U tom slučaju faza treninga se završava.

Programski kod 7. Definiranje kontrolnih točaka i ranog zaustavljanja

```
1. from keras.callbacks import ModelCheckpoint, EarlyStopping
2.
3. checkpoint = ModelCheckpoint("/gdrive/MyDrive/Diplomski/RAF-
   DB/Weights/vgg16_1.h5", monitor='val_accuracy', verbose=1,
   save_best_only=True, save_weights_only=False, mode='auto',
   save_freq="epoch")
4. early = EarlyStopping(monitor='val_accuracy', min_delta=0, patience=5,
   verbose=1, mode='auto')
```

Potrebno je još samo provesti trening mreže sa svim prethodno navedenim parametrima. Biblioteka Keras omogućuje jednostavno treniranje mreže gdje varijabla *x* sadrži sve slike koje se koriste za trening, a varijabla *y* anotacije tih slika. Skup podataka za validaciju nije nužno definirati odvojeno od podataka za trening kao što je to slučaj u sklopu programskog koda 8, već je moguće i imati jednu mapu u kojoj su spremljene sve slike. Opcijom validation_split=num moguće je programski podijeliti slike u omjeru definiranom nepoznanicom *num.*

Programski kod 8. Treniranje konvolucijske neuronske mreže

```
1. hist = vgg16.fit(x=train_images, y=train_labels,
   batch_size=32,epochs=10,validation_data=(test_images,test_labels),verbose
   =1,shuffle=True,callbacks=[checkpoint,early])
```

5.2.3. Faza testiranja

Nakon provedenog treninga, parametri koji su rezultirali najtočnijim rješenjima su pohranjeni u datoteci i spremni za korištenje. S ciljem kako bi osigurali relevantna rješenja, prije učitavanja slike u mrežu provodi se algoritam koji svaku sliku dovedenu na ulaz analizira i izrezuje dio na kojem se nalazi lice osobe.

Programski kod 9. Funkcija detekcije lica osobe

```
1. import cv2
2. from google.colab.patches import cv2_imshow
3. def get_cropped_face_image(path):
4.   img = cv2.imread(path)
5.   gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
6.   face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades +
   'haarcascade_frontalface_default.xml')
7.   faces = face_cascade.detectMultiScale(gray, 1.1, 4)
8.
9.   for (x, y, w, h) in faces:
10.      face = img[y:y + h, x:x + w]
11.
12.  img_224=cv2.resize(face, dsize=(224, 224))
13.  cv2_imshow(img_224)
```

```
14.   img_reshaped = img_224.reshape(1,224,224,3)
15.   return img_reshaped
```

Nakon uvođenja funkcije detekcije lica osobe, sve je spremno za testiranje mreže. Prije samog testiranja, poziva se funkcija koja će od vjerojatnosti sedam različitih emocija odabrati onu koja ima najveću vrijednost i donijeti konačnu odluku. Mogu se testirati pojedinačne slike, no ono što je česta praksa kod testiranja mreže dubokog učenja jest uvesti veći broj slika koje će se provesti kroz mrežu i usporediti rezultate mreže s onom emocijom koja je anotirana uz navedenu sliku. Nakon toga izrađuje se matrica konfuzije. Matrica konfuzije je u suštini tablica koja prikazuje gdje su napravljene pogreške u modelu, a koristi se u problemima klasifikacije. Redovi predstavljaju stvarne klase slike, dok stupci predstavljaju predviđanja. Pomoću ove tablice lako je vidjeti koja su predviđanja pogrešna. Ono što je poželjno je da se na dijagonali nalaze najveći brojevi u tom redu jer bi to značilo kvalitetnije rezultate mreže. Može se predvidjeti da će emocija koja je najzastupljenija u bazi podataka imati bolje rezultate te će se kod te emocije najveći broj nalaziti na dijagonali.

Programski kod 10. Izrada matrice konfuzije

```
1. from sklearn.metrics import confusion_matrix
2. import seaborn as sns
3.
4. def plot_confusion_matrix(actual_classes, predicted_classes,
   sorted_labels):
5.     matrix = confusion_matrix(actual_classes, predicted_classes,
   labels=sorted_labels)
6.     plt.figure(figsize=(15,10))
7.     sns.heatmap(matrix, annot=True, xticklabels=sorted_labels,
   yticklabels=sorted_labels, cmap="Blues", fmt="g")
8.     plt.xlabel('Predicted'); plt.ylabel('Actual'); plt.title('Confusion
   Matrix')
9.     plt.show()
10. predicts = vgg16.predict(test_images)
11.
12. actual_classes = []
13. for l in test_labels:
14.   actual_classes.append(get_most_probable_class(l))
15. predicted_classes = []
16. for l in predicts:
17.   predicted_classes.append(get_most_probable_class(l))
18. sorted_labels = []
19. for l in emotions:
20.   sorted_labels.append(emotions[l])
21.
22. plot_confusion_matrix(actual_classes, predicted_classes, sorted_labels)
```

Naposlijetku, kada se mreža koristi za pojedinačna testiranja slika, potrebno je provesti sljedeći kod. Prije samog učitavanja slike potrebno je provesti programski kod 9 koji definira funkcije za detekciju lica osobe a zatim učitati model i sve težine neuronske mreže.

Programski kod 11. Učitavanje modela i težina neuronske mreže

```
1. from keras.models import load_model
2. vgg16 =
   load_model("/gdrive/MyDrive/Diplomski/Weights/EmotionNet/vgg16_1.h5")
```

Nakon što je učitan model, programski kod 12 učitava sliku iz datoteke, detektira lice osobe i stvara predikciju o emociji koja se nalazi na slici.

Programski kod 12. Testiranje pojedinačne slike

```
1. picture = get_cropped_face_image("/gdrive/MyDrive/Diplomski/RAF-
   DB/marko1.jpg")
2. predicts = vgg16.predict(picture)
3. print(get_most_probable_class(predicts[0]))
```

6. Evaluacija rada mreža

Nakon uspješnog treniranja mreže potrebno je napraviti evaluaciju rješenja kako bi se mogli donijeti kvalitetni zaključci o radu pojedine mreže. Za testiranje mreže korišten je skup podataka koji nije bio korišten u fazi treninga te je prema tome izrađena matrica konfuzije. Nadalje, mreža je testirana i na slike koje su prenesene s osobnog računala i tako je prikazano iz prve ruke ono što se može očekivati prilikom pokretanja ovog modela.

Zbog ograničenih resursa, uzeti su uzorci iz AffectNet i EmotioNet baze podataka veličine od oko 15.000 slika kako bi sve baze bile podjednake veličine. Na taj način direktno se uspoređuje kvaliteta pojedine baze podataka i mogu se donijeti konkretni zaključci.

Tablica 8. Raspodjela emocija unutar skupova iz baza podataka odabranih za treniranje

Emocija	AffectNet	EmotioNet	RAF-DB
Iznenađenje	1.185	2.357	1.619
Strah	492	1.500	355
Gađenje	308	200	877
Sreća	5.819	3.796	5.957
Tuga	2.121	1.200	2.460
Ljutnja	2.079	980	867
Neutralno stanje	3.307	3.038	3.204
UKUPNO	15.311	13.071	15.339

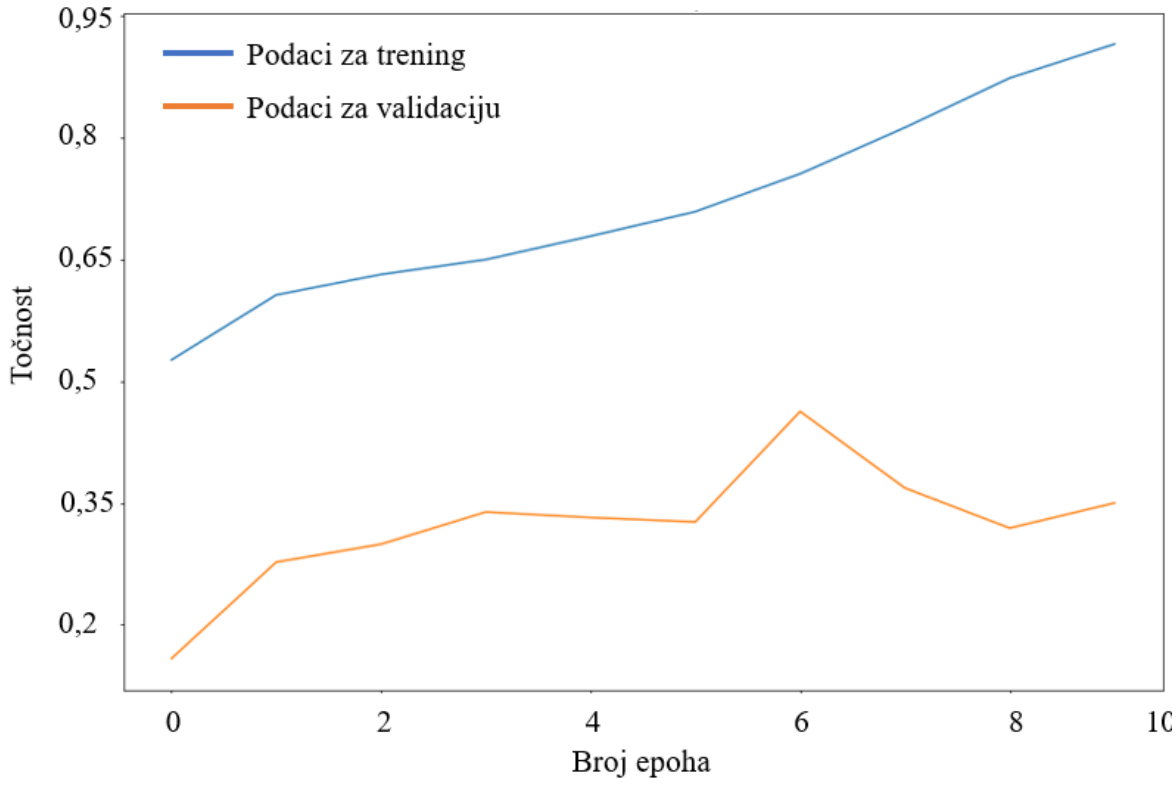
6.1. Evaluacija faze treninga

Točnost i gubitak su dva najvažnija parametra koje je potrebno pratiti prilikom faze treninga. Točnost je bitna za mjerenje izvedbe klasifikacijskog modela. Obično se izražava kao postotak. Točnost je broj predviđanja gdje je predviđena vrijednost jednaka stvarnoj vrijednosti. Često se prikazuje grafički i prati tijekom faze obuke, iako je vrijednost često povezana s ukupnom ili konačnom točnošću modela. S druge strane gubitak uzima u obzir vjerojatnosti ili nesigurnosti predviđanja na temelju toga koliko predviđanje varira od prave vrijednosti. To pruža bolji uvid u to koliko dobro model radi. Za razliku od točnosti, gubitak nije postotak, nego je to zbroj pogrešaka napravljenih za svaki uzorak u skupovima za obuku ili validaciju. Gubitak se često

koristi u procesu obuke kako bi se pronašle "najbolje" vrijednosti parametra za model (npr. težine u neuronskoj mreži). Tijekom procesa treninga cilj je minimizirati ovu vrijednost. Većinom je njihov odnos obrnuto proporcionalan, odnosno s povećanjem točnosti, smanjuje se vrijednost gubitka, ali to ne mora uvijek biti slučaj. Točnost i gubitak imaju različite definicije i mjere različite stvari te ne postoji njihova matematička ovisnost [25].

6.1.1. Točnost i gubitak modela treniranog s AffectNet bazom podataka

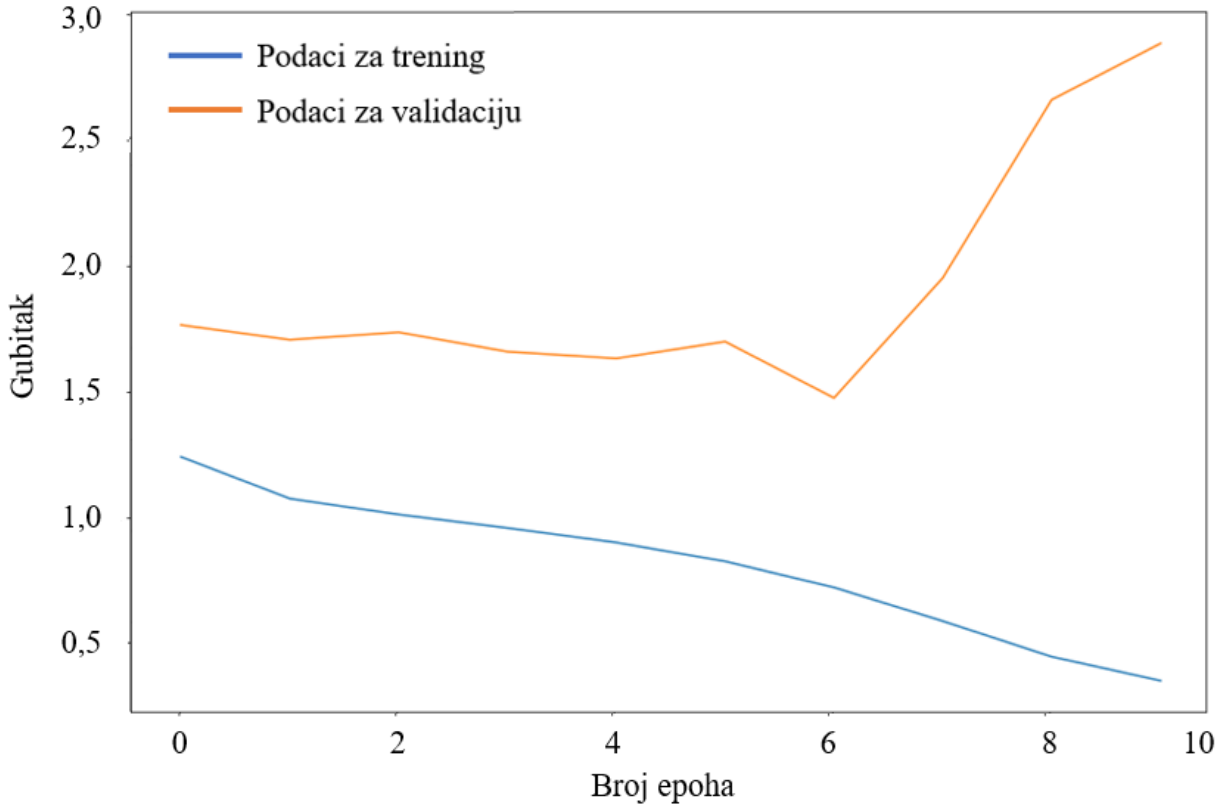
Trening AffectNet baze podataka proveden je prema programskom kodu koji je opisan u prethodnom poglavlju, a također se nalazi i u prilogu. Unutar ovog rada nije korištena cjelokupna baza podataka zbog toga što se radi o izuzetno velikom broju slika te je za obradu takve količine slika potrebno izuzetno jako računalo. Iz toga razloga uzet je uzorak od 15 tisuća slika koje nastoje reprezentirati ukupnu bazu podataka, a predstavljaju manje od 5% ukupnog broja slika s navedenim emocijama koje su dostupne za preuzimanje. Na grafikonima ispod vidljivo je kretanje vrijednosti točnosti i gubitka tijekom faze treninga.



Slika 23. Grafički prikaz kretanja vrijednosti točnosti kroz epohe u fazi treninga koristeći AffectNet bazu podataka

Iz grafikona je vidljivo kako točnost raste s porastom broja epoha, bilo da je riječ o dijelu treninga nad podacima za trening ili o dijelu validacije. Točnost validacije nagibom prati

krivulju koja opisuje točnost podataka za trening uz manje iznimke. U šestoj epohi dogodio se skok te će se težine dobivene u toj epohi uzeti kao konačan rezultat za daljnju evaluaciju. Točnost u šestoj epohi iznosi oko 47%. Nadalje, iz prikazanog dijagrama kretanja vrijednosti gubitka vidljiv je padajući trend do šeste epohe oba skupa podataka, no nakon šeste epohe gubitak podataka za validaciju počinje rasti. To je trenutak kada se počinje javljati pretreniranost mreže te su vidljive promjene na oba dijagrama u toj epohi.

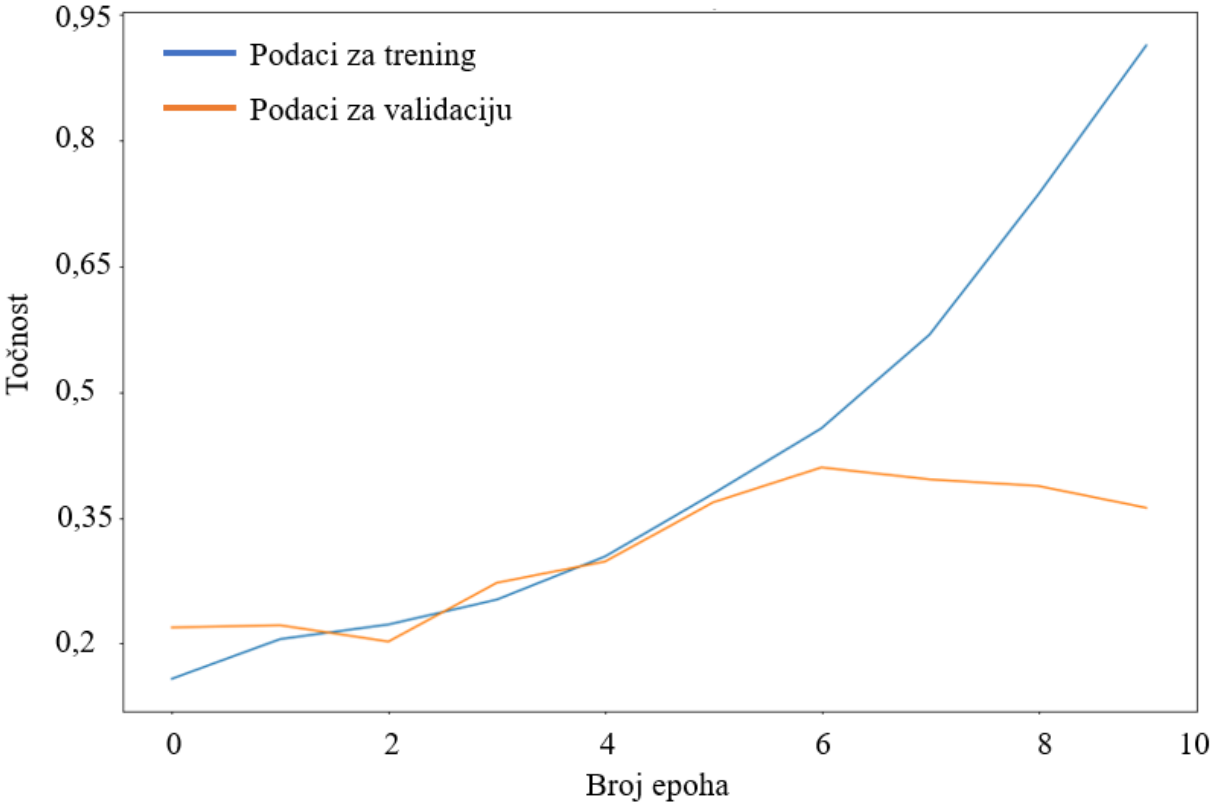


Slika 24. Grafički prikaz kretanja vrijednosti gubitka kroz epohe u fazi treninga koristeći AffectNet bazu podataka

6.1.2. Točnost i gubitak modela treniranog s EmotioNet bazom podataka

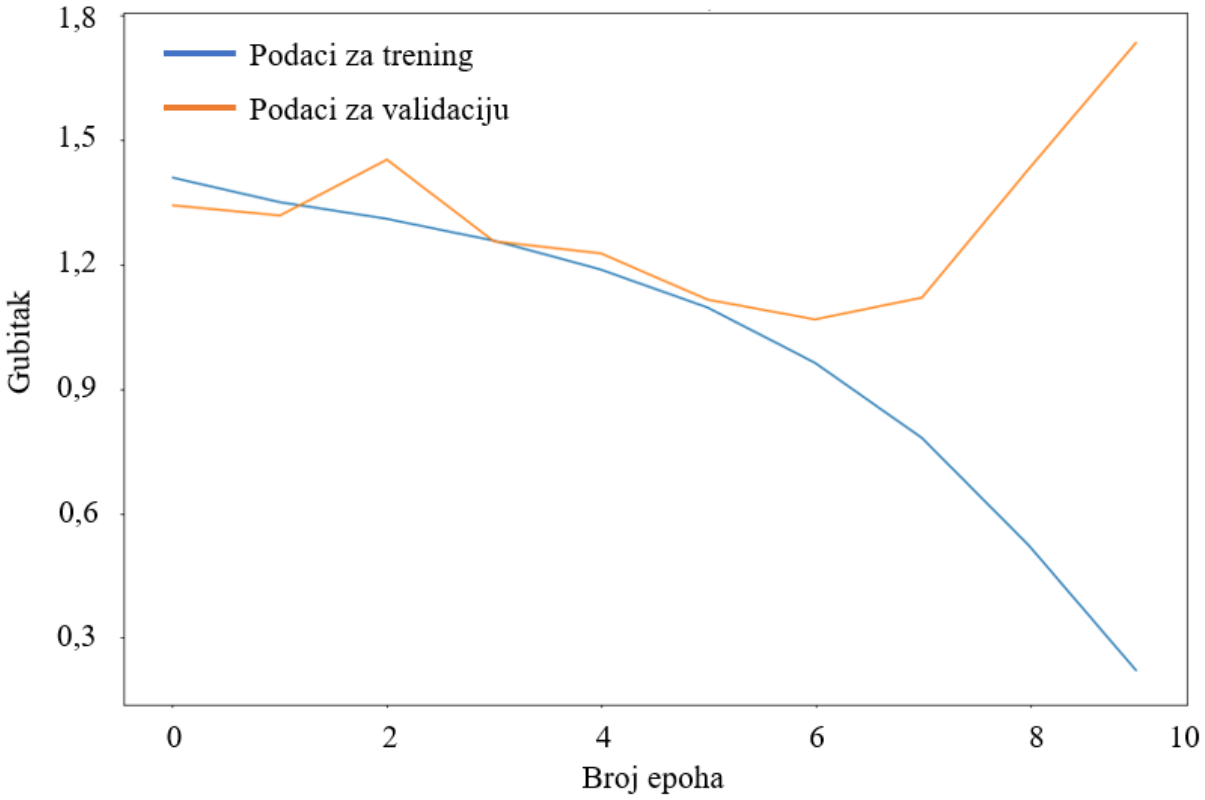
Prije same faze treninga anotacije podataka koje su se odnosile na akcijske jedinice transformirane su u anotacije prema emocijama koristeći tablicu 6. Iz baze podataka koja sadrži više od jedan milijun slika uzet je uzorak od 13 tisuća slika. Već prvim pogledom na podatke vidljiva je razlika u odnosu na ostale baze podataka i moguće probleme prilikom treniranja. Slike koje su preuzete s interneta nisu sve bile istih dimenzija, niti su sve sadržavale samo osobe, već i brojne smetnje i tekstove. Transformacija akcijskih jedinica u emocije je jedna

zasebna tema koju je u daljnjem istraživanju potrebno podrobnije istražiti. Iako tablica 6 prikazuje njihov odnos, prilikom ispisa slike i pripadajuće emocije vidljive su nedosljednosti.



Slika 25. Grafički prikaz kretanja vrijednosti točnosti kroz epohe u fazi treninga koristeći EmotioNet bazu podataka

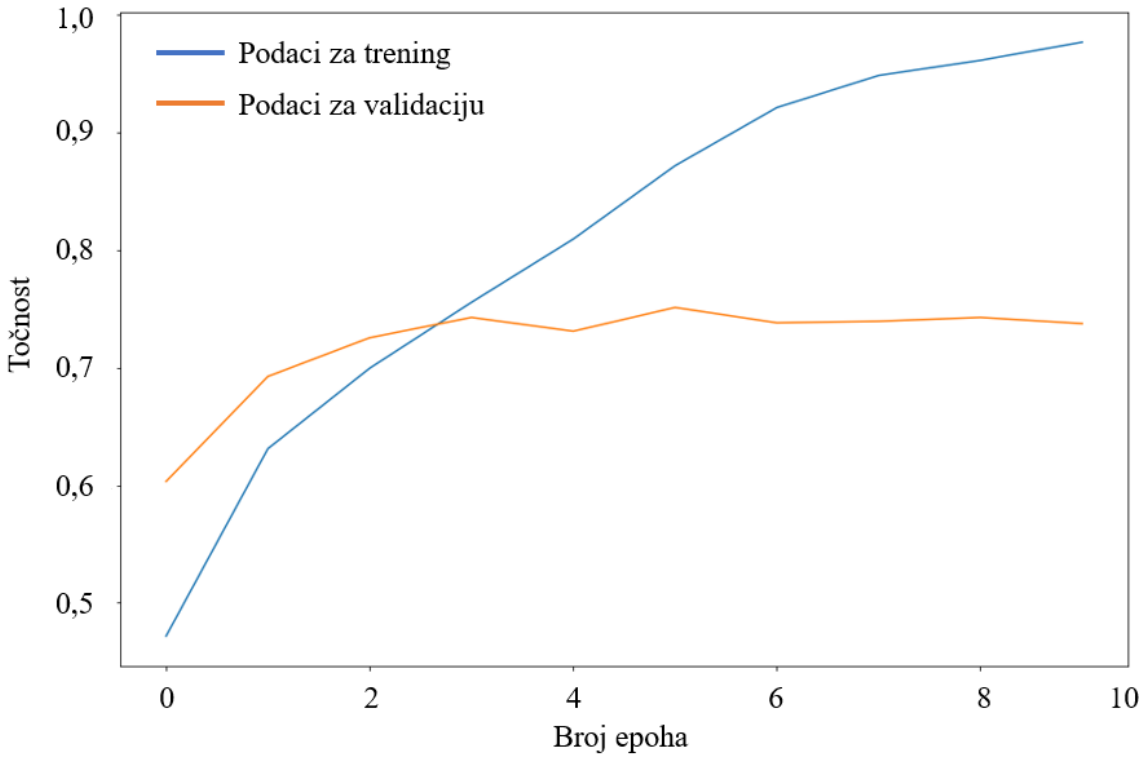
Na slici 25 vidljiv je grafikon koji prikazuje kretanje točnosti te se može vidjeti rast točnosti podataka za trening i validaciju. Kao i u prethodnom modelu, u šestoj epohi se dešava promjena te točnost podataka za validaciju stagnira. Iz tog razloga su za daljnju evaluaciju uzete vrijednosti težina u šestoj epohi, u kojoj je točnost iznosa 43%. Iako se točnost od 43% može činiti mala, treba uzeti u obzir da su podaci neravnomjerno raspoređeni između emocija. Odnos od svega 200 podataka koji prikazuju emociju gađenja, naspram 3.796 podataka koji prikazuju sreću daje naslutiti da će se emocija gađenja jako loše, ako uopće moći prepoznati. Na slici 26 vidljiv je grafikon koji prikazuje kretanje vrijednosti gubitka te se vide sličnosti s prethodnom mrežom. Imajući na umu da su obje mreže trenirane s istim parametrima, samo s različitim bazama podataka, za očekivati je bilo slične rezultate. To govori da o sposobnosti modela mreže da prepozna uzorke i značajke na slikama i krene ih učiti. Iako grafovi točnosti i gubitaka prikazuju puno podataka bitnih za davanje zaključaka, ono što je bitno je provesti evaluaciju na nepoznatim podacima kako bi se mogli dati kvalitetni zaključci.



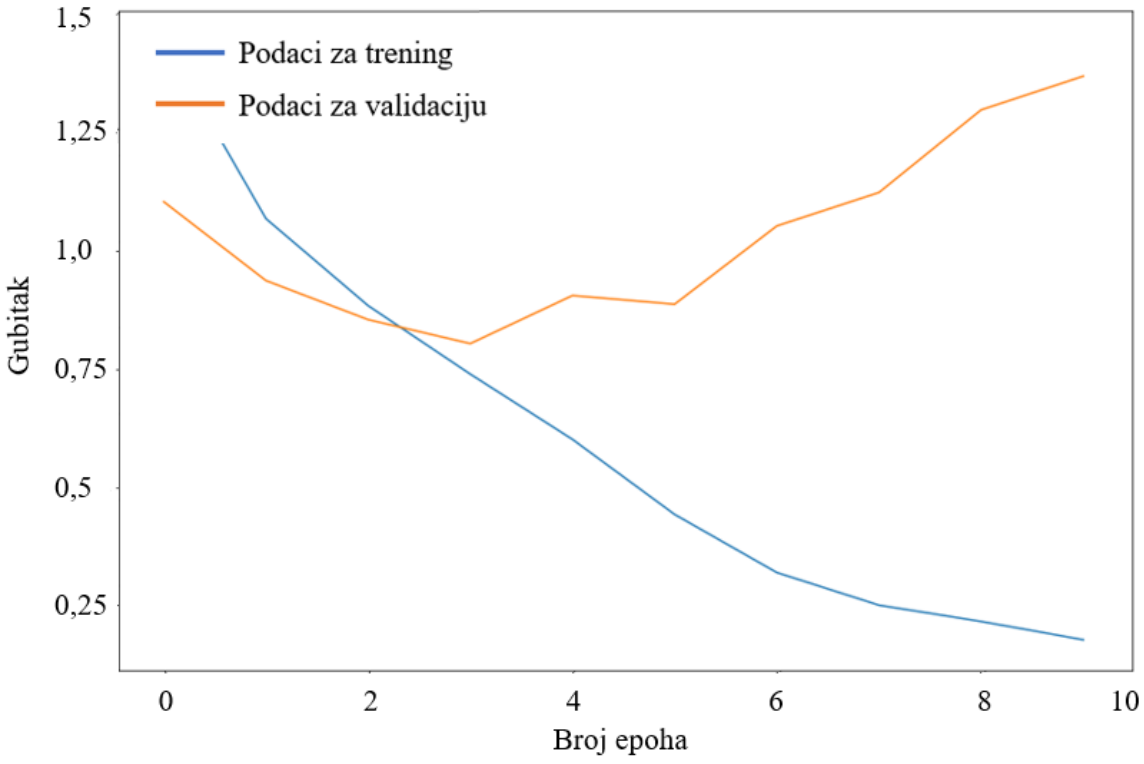
Slika 26. Grafički prikaz kretanja vrijednosti gubitka kroz epohe u fazi treninga koristeći EmotioNet bazu podataka

6.1.3. Točnost i gubitak modela treniranog s RAF-DB bazom podataka

RAF-DB je jedina baza podataka u kojoj su se svi dostupni podaci upotrijebili za treniranje bez potrebe za uzimanjem manjih uzoraka. Iako manjeg volumena u odnosu na druge dvije baze, sadrži izuzetno kvalitetne slike koje su već poravnate tako da se lice osobe nalazi u središtu slike. Iako manjih dimenzija, omjer slike je ispravan te se obradom neće pokvariti odnosi unutar slike. Na sljedećim grafikonima vidljivo je kretanje vrijednosti točnosti i gubitka tijekom faze treninga. Točnost se povećavala i na podacima za trening i na podacima za validaciju. Najveći iznos zabilježen je u petoj epohi pa su vrijednosti težina iz te epohe uzete za daljnju evaluaciju. Nakon četvrte epohe vidljiva je stagnacija točnosti te se nakon desete epohe mreža u potpunosti zaustavila, iz razloga jer nije vidljiv daljnji rast točnosti koji je uzet kao referenca za određivanje trenutka zaustavljanja mreže. Gubitak učenja se smanjivao konstantno tijekom cjelokupne faze treninga, a gubitak validacije je nakon treće epohe krenuo rasti, što se podudara s time da je i točnost stagnirala. Vidljiva su velika poboljšanja u odnosu na prethodne dvije baze podataka te je za očekivati i bolje rezultate ove mreže u odnosu na druge u daljnjoj evaluaciji.



Slika 27. Grafički prikaz kretanja vrijednosti točnosti kroz epohe u fazi treninga koristeći RAF-DB bazu podataka



Slika 28. Grafički prikaz kretanja vrijednosti gubitka kroz epohe u fazi treninga koristeći RAF-DB bazu podataka

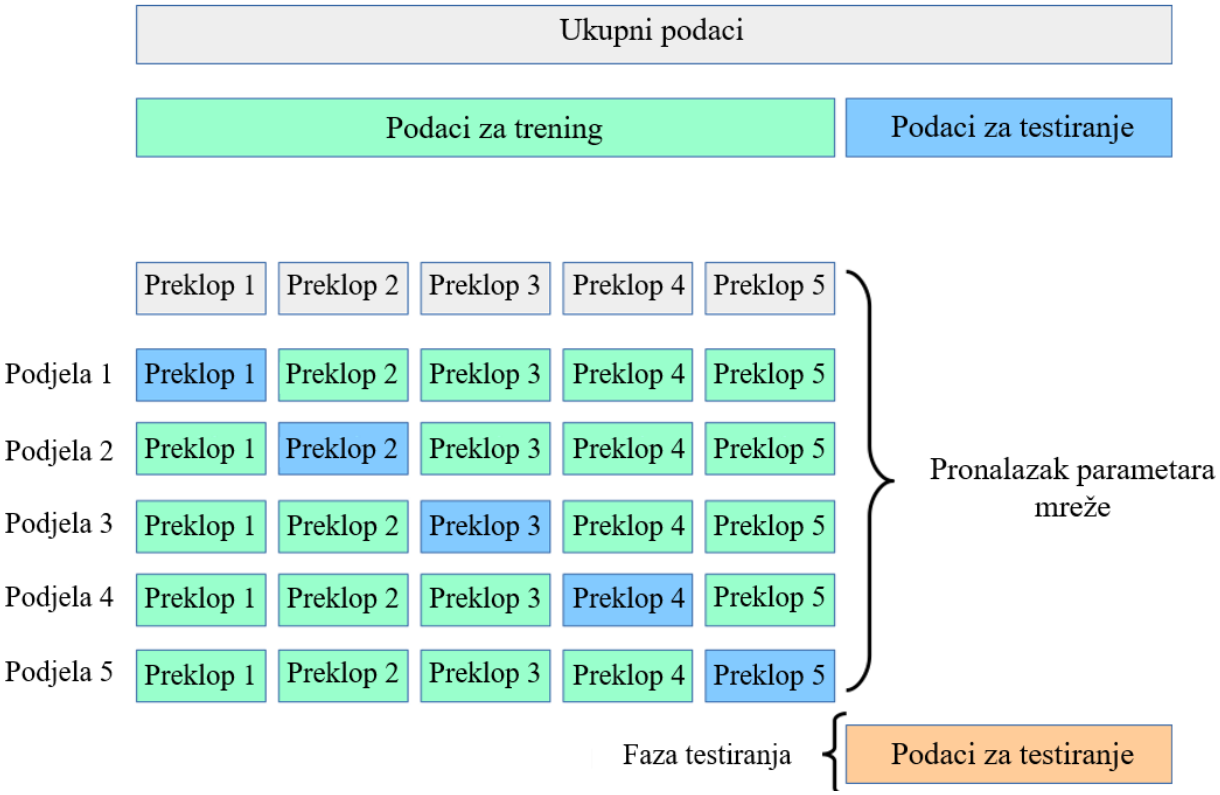
6.2. K- struka unakrsna validacija

K-struka unakrsna validacija podataka definirana je kao metoda za procjenu izvedbe modela na još nepoznatim podacima. Ova tehnika nalazi primjenu tamo gdje je broj podataka malen i gdje je stavljen naglasak na dobru procjenu pogreške treniranja i generalizacije što podrazumijeva dobro razumijevanje teorijskih odrednica pretreniranosti i premale istreniranosti mreže. To je tehnika ponovnog uzorkovanja bez zamjene, odnosno cilj je podesiti parametre tako da se može istrenirati model s optimalnim vrijednostima. Prednost ovog pristupa je da se svaki primjer koristi za obuku i provjeru valjanosti samo jednom što daje nižu procjenu varijance izvedbe modela od drugih metoda. Koristeći ovu tehniku nastoji se izbjeći pretreniranost mreže, što se može dogoditi kada se za treniranje koriste svi dostupni podaci. Korištenjem K-struke unakrsne validacije može se testirati model na k različitih skupova podataka, što osigurava generalizaciju modela.

Koraci K-struke unakrsne validacije [26]

1. Skup podataka podijeli se na podatke za obuku i podatke za test
2. Skup podataka za obuku se zatim dijeli na *k* preklopa
3. Od k preklopa, *k-1* koristi se za fazu treninga
4. Preostali preklop *k* koristi se za validaciju odnosno testiranje mreže
5. Model se obučava podacima za obuku(*k-1* preklopa) i validacijskim podacima(*k-ti* preklop). Sprema se izvedba modela.
6. Koraci 3, 4 i 5 ponavljaju se sve dok se svaki od *k* preklopa ne koristi u svrhu testiranja mreže. Prema tome je metoda dobila naziv k-struka unakrsna validacija.
7. Srednja vrijednost i standardna devijacija izvedbe modela se izračunavaju uzimanjem svih rezultata modela izračunatih u koraku broj 5 za svaki od *k* modela.
8. Konačno, odabire se kombinacija preklopa koja daje optimalna rješenja.
9. Model se zatim trenira pomoću skupa podataka za treniranje, a izvedba modela izračunava se na testnom skupu podataka.

Vrijednost za *k* odabire se tako da svaka skupina uzoraka podataka za trening/validaciju bude dovoljna da bude statistički reprezentativna za širi skup podataka. Vrijednosti 5 i 10 vrlo su česte u polju primijenjenog strojnog učenja [26]. Na slici 29 prikazan je osnovni koncept K-struke unakrsne validacije.



Slika 29. Osnovni princip k-struke unakrsne validacije [26]

Upotrebom k-struke unakrsne validacije prilikom faze treniranja smanjuje se pristranost modela odnosno osigurava se generalizacija. No isto tako sam po sebi algoritam za trening je računski intenzivan, pa njegovo pokretanje 5 ili 10 puta iziskuje puno više vremena od samo jednog prolaska kroz podatke.

6.2.1. Trening modela upotrebom RAF-DB baze podataka s 5-strukom validacijom

S obzirom na dobivene rezultate odlučeno je provesti k-struku validaciju samo na RAF-DB bazi podataka. Iako se ovom metodom dobivaju bolje točnosti mreže, iziskuje puno računalnih resursa, a dobiveni rezultati kada je riječ o ovakvim veličinama baza podataka su neznatno bolji. Iako gubitak može znatno varirati čak i više od 20% u odnosu na prosječnu vrijednost, treba primijetiti kako se točnost modela mijenja svega nekoliko posto. Malene varijacije u točnostima validacije govore o kvaliteti baze podataka, odnosno o pokrivenosti svih varijacija između osoba. Usporedbom slučajno odabranih pet različitih podjela na bazi podataka RAF-DB, može se zaključiti kako je baza kvalitetna i sveobuhvatna. Model se podjednako dobro istrenirao koristeći svih pet podjela, a odstupanja u točnosti su minimalna (vidi tablicu 9.). U daljnjoj evaluaciji rješenja korištena je mreža s početnim podatcima jer je njezina točnost približno jednaka prosječnoj točnosti nakon 5-struke validacije.

Tablica 9. Usporedba rezultata 5-struke validacije

Redni broj podjele	Gubitak validacije	Točnost validacije
1.	1,527635	73,0117
2.	1,425917	72,3924
3.	1,907632	74,2503
4.	1,195249	73,2073
5.	1,681602	72,3834
Prosječna vrijednost	1,5476	73,0490

6.3. Matrica konfuzije

Matrica konfuzije je tehnika mjerenja performansi za probleme klasifikacije strojnog učenja. To je tablični zapis broja točnih i netočnih predviđanja koje je napravio klasifikator (tablica 10). Koristi se za mjerenje izvedbe klasifikacijskog modela. Matrice konfuzije naveliko se koriste jer daju bolju sliku o izvedbi modela kod problema klasifikacije. Na primjer, u točnosti klasifikacije nema informacija o broju pogrešno klasificiranih instanci. Mogući je slučaj da podaci imaju dvije klase gdje 85% podataka pripada klasi A, a 15% pripada klasi B. Također, pretpostavka je da model klasifikacije ispravno klasificira sve instance klase A i pogrešno klasificira sve instance klase B. U ovom slučaju, model je 85% točan. Međutim, klasa B je pogrešno klasificirana, što je nepoželjno. [27]

Tablica 10. Primjer matice konfuzije [27]

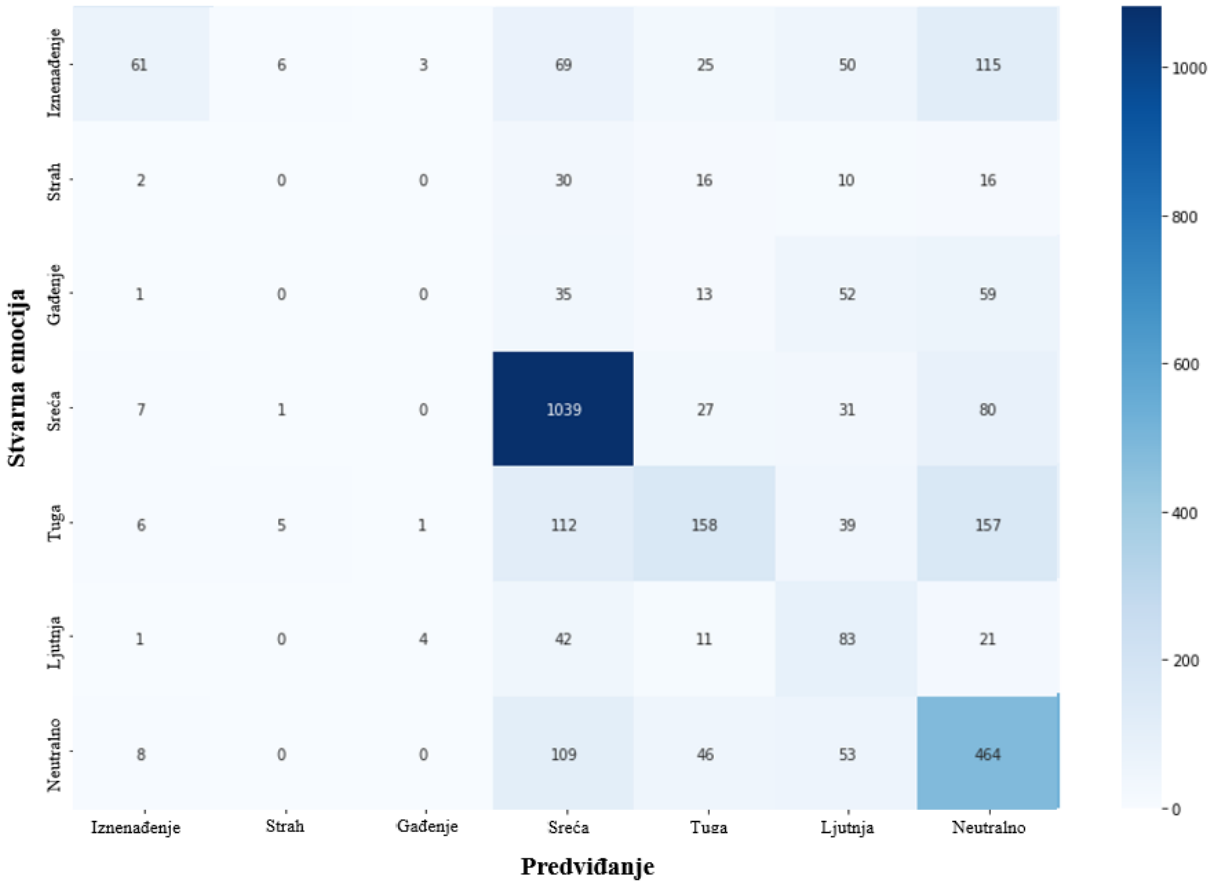
Stvarna vrijednost	Negativna vrijednost	Točno predviđena negativna vrijednost	Krivo predviđena pozitivna vrijednost
	Pozitivna vrijednost	Krivo predviđena negativna vrijednost	Točno predviđena pozitivna vrijednost
		Negativna vrijednost	Pozitivna vrijednost
		Previđanje	

Matrica konfuzije, s druge strane, prikazuje ispravno i netočno klasificirane instance za sve klase i stoga će dati bolji uvid u izvedbu mreže. Dobar model je onaj koji ima visoke stope točnih predviđanja, a niske stope krivih predviđanja. Ako je skup podatak neuravnotežen kao u

ovim slučajevima, uvijek je bolje upotrijebiti matricu konfuzije kao kriterij procjene za model strojnog učenja.

6.3.1. Matrica konfuzije AffectNet baze podataka

AffectNet baza podataka prilikom faze treninga imala je točnost od 47%, što nije dovoljno za davanje konkretnih zaključaka. Iz tog razloga izradila se matrica konfuzije. Na slici 30 prikazana je matrica konfuzije koja koristi set od 3.068 slika. Raspodjela emocija unutar testnog seta je slična kao i kod seta za trening, gdje pretežno prevladavaju slike koje sadrže emociju sreće. Matrica konfuzije pokazala je sasvim očekivana rješenja. Emocija sreće je emocija koja sadrži najveći postotak točnih predikcija koji iznosi preko 95%, a iza nje odmah slijedi emocija neutralno stanje. Imajući na umu da su baze podataka sadržavale najviše slika s ta dva emocionalna stanja, za očekivati je bilo ovakve rezultate. Ova baza podataka može još dobro predvidjeti ljutnju u 50% slučajeva te tugu u trećini slučajeva. Ostale emocije baza ne predviđa najbolje, pogotovo gađenje i strah koje je u potpunosti eliminirala što uopće ne čudi jer je zbrojeno njihov udio u ukupnom broju slika za trening bio manji od 5%.

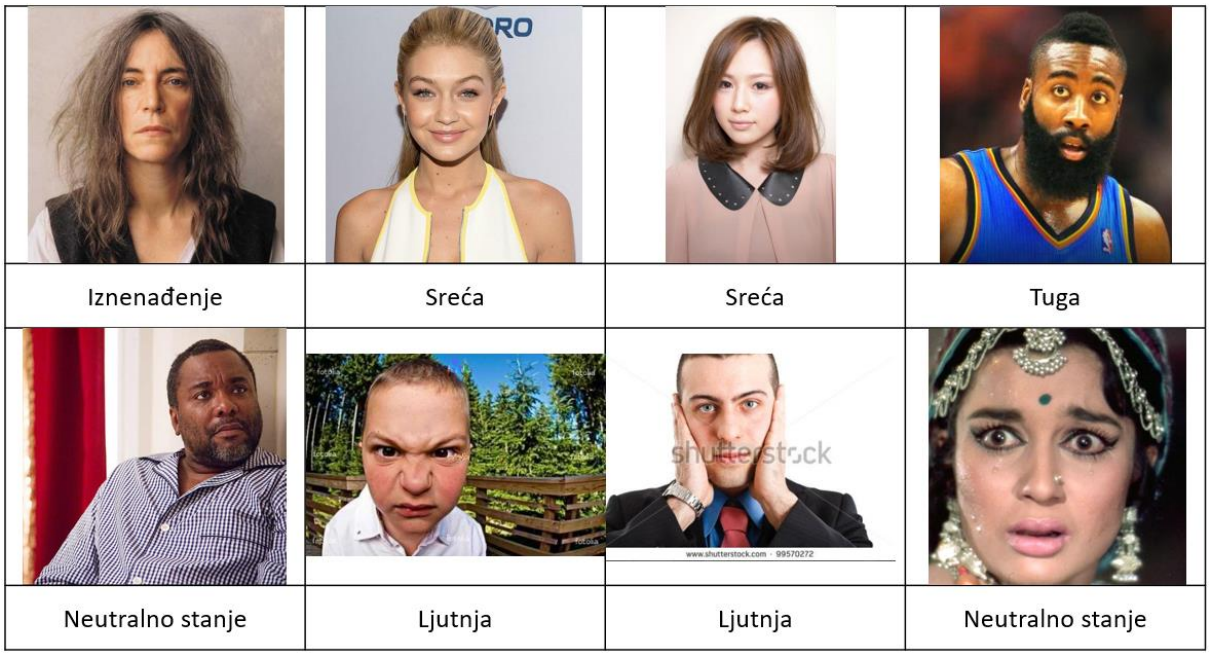


Slika 30. Matrica konfuzije modela treniranog s AffectNet bazom podataka

6.3.2. Matrica konfuzije EmotioNet baze podataka

Matrica konfuzije EmotioNet baze podataka nešto se razlikuje od ostalih. Iako bi bilo poželjno da najveće brojke prevladavaju na dijagonali, kod modela treniranog s EmotioNet bazom podataka to nije slučaj. Razlog tomu je već spomenuta transformacija anotacija prema akcijskim jedinicama u anotaciju prema emocijama. Već prilikom same analize baze podataka, bilo je jasno i za očekivati ovakve rezultate. Iako je mreža prilikom treninga imala točnost validacije od 43%, vidljivo je iz matrice konfuzije da je to daleko od realnog stanja. Mreža gotovo niti jednu emociju ne prepoznaje dobro, osim neutralnog stanja. Iako je baza sadržavala dosta slika koje su bile označene da sadrže sreću, evidentno je da su slike pogrešno označene jer mreža ne pokazuje dobra rješenja. Mreža je većinom kao predikciju imala neutralno stanje, a jedan od mogućih razloga je pretreniranost mreže, no s obzirom da su ove težine dobivene nakon šeste epohe, mala je vjerojatnost da je to slučaj. S obzirom na ovako loše rezultate potrebna je podrobnija analiza, te će se izabrati nekoliko nasumičnih slika i vidjeti odgovara li anotacija emociji koja se nalazi na slici.

Slika 31. Matrica konfuzije modela treniranog s EmotioNet bazom podataka



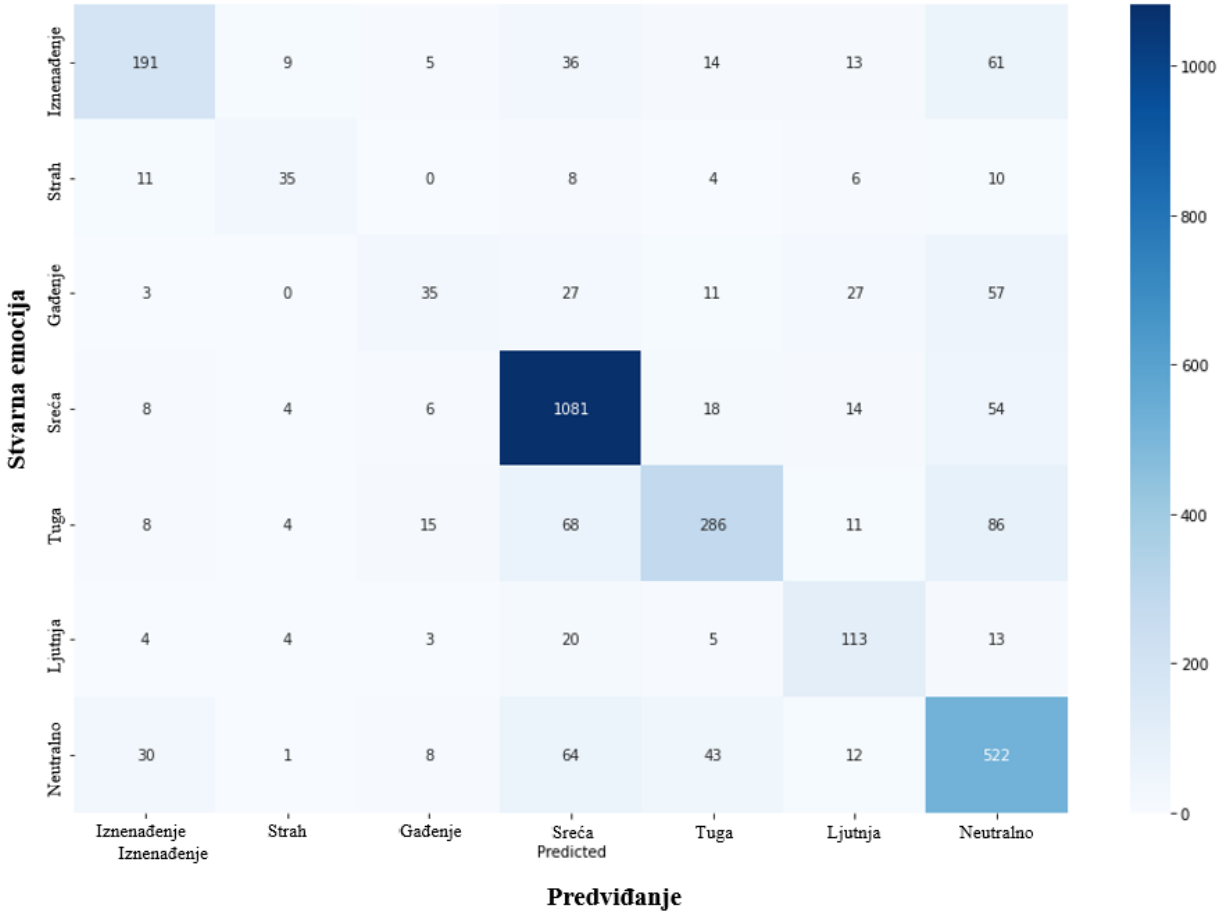
Slika 32. Transformirane anotacije prema emocijama EmotioNet baze podataka [19]

Iz slike 32 vidljive su evidentne pogreške u anotaciji prilikom transformacije. Iako tablica 6 prikazuje odnos anotacija prema akcijskim jedinicama lica i anotacija prema emocijama, ona nije potpuna jer nedostaju brojne akcijske jedinice. Također treba uzeti u obzir da unutar baze podataka EmotioNet nisu označene sve akcijske jedinice lica, nego samo njih 12 što eliminira dosta emocija, a usput povećava greške jer sve slike koje se ne mogu svrstati u jednu od šest osnovnih emocija, svrstavaju se pod neutralno stanje. Tako neutralno stanje sadrži broje slike koje bi možda trebale biti smještene pod neku drugu emociju. Na primjeru slike 32 može se vidjeti da nisu sve slike pogrešno anotirane, ali isto tako da su neke potpuno pogrešne poput slike u gornjem lijevom kutu koja sigurno ne označava iznenađenu osobu, nego neutralno stanje ili slike osobe u donjem desnom kutu koja bi svakako trebala označavati strah.

6.3.3. Matrica konfuzije RAF-DB baze podataka

Izgled matrice konfuzije kod modela treniranog s RAF-DB bazom podataka nakon provedene faze testiranja vidljiv je na sljedećoj slici. Vidi se poprilična centriranost rezultata oko dijagonale što govori kako je dobivena točnost od 73% valjana. Emocija sreće sadržavala je najviše slika u ovoj bazi podataka, stoga je i najviše testnih slika upravo te emocije. Točnost mreže da prepozna emociju sreće iznosi preko 95% te je viša nego srednja točnost od 73%. Mreža također dobro prepoznaje tugu, ljutnju, iznenađenje i neutralno stanje jer su najveće vrijednosti na dijagonali i barem su dvostruko veće od predviđanja bilo koje druge emocije u

tom redu. Emociju straha mreža nešto manje dobro raspoznaje, dok emociju gađenja gotovo uopće ne raspoznaje. Razlog tomu je i najmanja zastupljenost te emocije unutar baze podataka. Može se primijetiti uzorak da gotovo sve mreže loše raspoznaju emociju gađenja zbog najmanje zastupljenosti. Isti zaključak se može povući i za strah koji je u ovom slučaju emocija s najmanje podataka u bazi za trening.



Slika 33. Matrica konfuzije modela treniranog s RAF-DB bazom podataka

6.4. Evaluacija rada mreža na proizvoljnim slikama

U sklopu ovog poglavlja provesti će se faza testiranja na slikama osobne izrade kojima je pridružena pripadajuća emocija. Ono što je na kraju bitno, osim izrade matrice konfuzije, jest provesti testiranje na slikama koje su anotirane od strane autora kako bi evaluacija bila potpuna i kako bi se moglo znati što očekivati od mreža prilikom njihove upotrebe. Nakon što su fotografije prikupljene, a zatim anotirane, nad njima su proveden algoritam za detekciju lica. Nakon što je lice stavljeno u fokus te im se veličina prilagodila ulazu u mrežu, dobiveni su

sljedeći rezultati koji će biti prikazani u tablici 11. Na podatcima koji su prilagođeni tako da je sreća najzastupljenija među emocijama, ostvareni su dobri rezultati na mrežama treniranim s RAF-DB i AffectNet bazama podataka. Kada se pogledaju matrice konfuzije baza podataka, odmah je na prvu vidljivo zašto EmotioNet raspoznaje sreću, a za sve ostale emocije predviđa neutralno stanje.

Tablica 11. Prikaz predikcije mreža na fotografijama

 Sreća	Sreća (AffectNet)
	Sreća (EmotioNet)
	Sreća (RAF-DB)
 Sreća	Sreća (AffectNet)
	Sreća (EmotioNet)
	Sreća (RAF-DB)
 Iznenađenje	Iznenađenje (AffectNet)
	Neutralno (EmotioNet)
	Iznenađenje (RAF-DB)

Ljutnja

Ljutnja (AffectNet)
Sreća (EmotioNet)
Ljutnja (RAF-DB)

Tuga

Tuga (AffectNet)
Neutralno (EmotioNet)
Tuga (RAF-DB)

Sreća

Sreća (AffectNet)
Neutralno (EmotioNet)
Sreća (RAF-DB)

Iako različitih točnosti, mreže trenirane s EmotioNet i RAF-DB bazam podataka nad ovim podatcima pokazali su se kao pouzdane te su ostvarile 100% točnost. Za daljnju evaluaciju potrebno je računalo s više resursa kako bi se u fazi treninga mogao iskoristiti cjelokupan sadržaj svake baze podataka te omogućiti još kvalitetnija usporedba.

7. Zaključak

Ručno prepoznavanje emocija od strane ljudi često ne predstavlja veliki problem, pogotovo kod jasnih ekspresija lica. Međutim strojevima to predstavlja veliki problem, zbog potencijalne iznimne složenosti slike, jer ona najčešće sadrži i brojne pozadinske parametre. Napretkom arhitekture dubokog učenja te uviđanjem sve veće važnosti i široke primjenjivosti analize ljudskog ponašanja, ali također i potreba za obradom veće količine podataka koja je strojno moguće brže izvediva, ovo područje istraživanja postaje sve zanimljivije široj populaciji.

Upravo je primarni cilj ovog rada bilo ispitati i usporediti različite baze podataka za automatsko prepoznavanje emocija. U tu svrhu korištene su baze podataka AffectNet, EmotioNet i RAF-DB. Baze podataka sadrže slike osoba koje su anotirane ili prema emocijama ili prema akcijskim jedinicama. Nakon što su sve anotacije prilagođene kako bi rezultati bili usporedivi, provedena je faza treninga. Nakon provedenog treninga mreže su bile testirane te su izrađene matrice konfuzije za svaku pojedinu mrežu. Najbolje rezultate ostvarila je mreža trenirana s RAF-DB bazom podataka koja je od početka dala naslutiti kako će biti jedan od favorita za najkvalitetniju bazu od ovih tri navedenih. Slike u bazi podataka RAF-DB unaprijed su uređene tako da su izrezane na željenu veličinu, a lice osobe centrirano je u sredinu slike. Iz tog razloga postojala je manja vjerojatnost da mreža uzme u obzir raznolike šumove te da se stvarno fokusira na lice osobe i sve značajke koje može iz njega izvući. Rezultate EmotioNet baze podataka treba uzeti s dozom opreza jer je pretvorba anotacija akcijskih jedinica lica u anotaciju prema emocijama unijela određeni postotak pogreške koji je utjecao na ukupnu točnost mreže. Potrebno je podrobnije istraživanje kako bi se unijela što manja greška u sustav prilikom te transformacije. Iz AffectNet baze podataka nasumično je izuzet uzorak koji iznosi 5% od ukupnog broja slika u bazi te je dobivena točnost manja u odnosu na onu koja bi se mogla dobiti kada bi se faza treninga provela s cijelim skupom podataka. Automatsko prepoznavanje emocija obradom vizualnih informacija, odnosno u ovom slučaju fokusiranjem samo na lice osobe, jedno je područje koje je dobilo veliki zamah u zadnjem desetljeću, no još nije dosegnulo svoj vrhunac. Iako VGG16 pokazuje izvrsne rezultate u ovom području, potrebno je ispitati i druge modele mreža, ali i kombinirati više baza podataka kako bi se dobili optimalni rezultati i mreža koja bi se mogla komercijalno upotrebljavati u raznolike svrhe.

LITERATURA

[1] W. Mellouk i W. Handouzi, „Facial emotion recognition using deep learning: review and insights“, *Procedia Computer Science*, sv. 175, str. 689–694, 2020, doi: 10.1016/j.procs.2020.07.101.

[2] K. Vemou i A. Horvath, „TechDispatch #1/2021 - Facial Emotion Recognition“. https://edps.europa.eu/data-protection/our-work/publications/techdispatch/techdispatch-12021-facial-emotion-recognition_en (pristupljeno 10. listopad 2022.).

[3] Dr. A. Rosebrock, *Deep Learning for Computer Vision with Python*, 1st Edition (1.1.0). PyImageSearch, 2017.

[4] Y. LeCun, Y. Bengio, i G. Hinton, „Deep learning“, *Nature*, sv. 521, izd. 7553, str. 436–444, svi. 2015, doi: 10.1038/nature14539.

[5] javaTpoint, „Supervised Machine Learning“. https://www.javatpoint.com/supervised-machine-learning (pristupljeno 17. listopad 2022.).

[6] A. Mollahosseini, D. Chan, i M. H. Mahoor, „Going deeper in facial expression recognition using deep neural networks“, u *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*, Lake Placid, NY, USA, ožu. 2016, str. 1–10. doi: 10.1109/WACV.2016.7477450.

[7] A. T. Lopes, E. de Aguiar, A. F. De Souza, i T. Oliveira-Santos, „Facial expression recognition with Convolutional Neural Networks: Coping with few data and the training sample order“, *Pattern Recognition*, sv. 61, str. 610–628, sij. 2017, doi: 10.1016/j.patcog.2016.07.026.

[8] M. Mohammadpour, H. Khaliliardali, S. Mohammad. R. Hashemi, i Mohammad. M. AlyanNezhadi, „Facial emotion recognition using deep convolutional networks“, u *2017 IEEE 4th International Conference on Knowledge-Based Engineering and Innovation (KBEI)*, Tehran, pros. 2017, str. 0017–0021. doi: 10.1109/KBEI.2017.8324974.

[9] G. Yolcu *i ostali*, „Facial expression recognition for monitoring neurological disorders based on convolutional neural network“, *Multimed Tools Appl*, sv. 78, izd. 22, str. 31581–31603, stu. 2019, doi: 10.1007/s11042-019-07959-6.

[10] Y. Li, J. Zeng, S. Shan, i X. Chen, u *Occlusion Aware Facial Expression Recognition Using CNN With Attention Mechanism*, sv. 28, 5 sv., IEEE Trans. Image Process, 2019, str. 2439–2450.

[11] D. Ž. U. Andrijić, „Topologija umjetnih neuronskih mreža“, str. 2.

[12] javaTpoint, „Artificial Neural Network Tutorial“. https://www.javatpoint.com/artificial-neural-network (pristupljeno 10. listopad 2022.).

[13] J. Brownlee, „How to Choose an Activation Function for Deep Learning“. https://machinelearningmastery.com/choose-an-activation-function-for-deep-learning/ (pristupljeno 13. listopad 2022.).

[14] A. Kumar, „Model Complexity and Overfitting in Machine Learning“. https://vitalflux.com/model-complexity-overfitting-in-machine-learning/ (pristupljeno 19. listopad 2022.).

[15] J. McDermott, „Convolutional Neural Networks - Image Classification w. Keras“. https://www.learndatasci.com/tutorials/convolutional-neural-networks-image-classification/ (pristupljeno 16. studeni 2022.).

[16] Great Learning Team, „Introduction to VGG16 | What is VGG16?“ https://www.mygreatlearning.com/blog/introduction-to-vgg16/ (pristupljeno 05. listopad 2022.).

[17] G. Rohini, „Everything you need to know about VGG16“. https://medium.com/@mygreatlearning/everything-you-need-to-know-about-vgg16-7315defb5918 (pristupljeno 20. rujan 2022.).

[18] A. Mollahosseini, B. Hasani, i M. H. Mahoor, „AffectNet: A Database for Facial Expression, Valence, and Arousal Computing in the Wild“, *IEEE Trans. Affective Comput.*, sv. 10, izd. 1, str. 18–31, sij. 2019, doi: 10.1109/TAFFC.2017.2740923.

[19] C. F. Benitez-Quiroz, R. Srinivasan, i A. M. Martinez, „EmotioNet: An Accurate, Real-Time Algorithm for the Automatic Annotation of a Million Facial Expressions in the Wild“, u *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, lip. 2016, str. 5562–5570. doi: 10.1109/CVPR.2016.600.

[20] CVPR 2020, „EmotioNet Challenge“. https://cbcsl.ece.ohio-state.edu/enc-2020/index.html (pristupljeno 20. listopad 2022.).

[21] E. A. Clark *i ostali*, „The Facial Action Coding System for Characterization of Human Affective Response to Consumer Product-Based Stimuli: A Systematic Review“, *Front. Psychol.*, sv. 11, str. 920, svi. 2020, doi: 10.3389/fpsyg.2020.00920.

[22] S. Li, W. Deng, i J. Du, „Reliable Crowdsourcing and Deep Locality-Preserving Learning for Expression Recognition in the Wild“, u *2017 IEEE Conference on Computer Vision*

and Pattern Recognition (CVPR), Honolulu, HI, srp. 2017, str. 2584–2593. doi: 10.1109/CVPR.2017.277.

[23] „The Python Tutorial“. https://docs.python.org/3/tutorial/ (pristupljeno 22. studeni 2022.).

[24] D. Banys, „Alternative to Colab Pro: Comparing Google’s Jupyter Notebooks to Gradient Notebooks“. https://blog.paperspace.com/alternative-to-google-colab-pro/ (pristupljeno 09. rujan 2022.).

[25] Ai Wiki, „Accuracy and Loss“. https://machine-learning.paperspace.com/wiki/accuracy-and-loss (pristupljeno 11. listopad 2022.).

[26] A. Kumar, „K-Fold Cross Validation - Python Example“. https://vitalflux.com/k-fold-cross-validation-python-example/ (pristupljeno 15. studeni 2022.).

[27] S. Anuganti, „What is confusion matrix?“ https://medium.com/analytics-vidhya/what-is-a-confusion-matrix-d1c0f8feda5 (pristupljeno 18. studeni 2022.).

PRILOZI

I. Programski kod

Programski kod

Preuzimanje podataka iz EmotioNet baze podataka na Google Disk

```
1. from google.colab import drive
2. drive.mount('/gdrive')
3.
4. # Read emotion net urls
5. from google.colab import auth
6. import requests # request img from web
7. from PIL import Image
8. import shutil
9. import json
10. auth.authenticate_user()
11.
12. import gspread
13. from google.auth import default
14. creds, _ = default()
15.
16. gc = gspread.authorize(creds)
17.
18. worksheet = gc.open('emotion-net-urls').sheet1
19.
20. # get_all_values gives a list of rows.
21. rows = worksheet.get_all_values()
22.
23. # Convert to a DataFrame and render.
24. import pandas as pd
25. df = pd.DataFrame.from_records(rows)
26. labels = []
27. image_labels = {}
28. index = 0
29. path = "/gdrive/MyDrive/Diplomski/EmotionNet/downloaded/"
30. for row in df.values.tolist():
31.    if index == 0:
32.      labels = row[2:18]
33.    else:
34.      # download and store image
35.      url = row[0]
36.      name = "image_" + str(index) + ".jpg"
37.      print(f'Downloading {url} and storing as {name}'.format(url=url,
    name=name))
38.      res = requests.get(url, stream = True)
39.      if res.status_code == 200:
40.        full_name = path + name
41.        with open(full_name, 'wb') as f:
42.          res.raw.decode_content = True
43.          shutil.copyfileobj(res.raw, f)
44.
45.      # find classification
46.      values = row[2:18]
47.      index_of_class = values.index('1')
48.      classification = labels[index_of_class]
49.      image_labels[name] = classification
50.
```

```
51.    index += 1
52.
53. json = json.dumps(image_labels)
54. f = open("/gdrive/MyDrive/Diplomski/EmotionNet/image_labels.json", "w")
55. f.write(json)
56. f.close()
57.
58. def get_values(X):
59.   v = []
60.   for y in X:
61.     if y == "1":
62.       v.append(1)
63.     else:
64.       v.append(0)
65.   return v
66.
67. df = pd.DataFrame.from_records(rows)
68. class_labels = []
69. image_labels = {}
70. index = 0
71. path = "/gdrive/MyDrive/Diplomski/EmotionNet/downloaded/"
72. for row in df.values.tolist():
73.    if index == 0:
74.      class_labels = row[2:18]
75.    else:
76.      # find classification
77.      name = "image_" + str(index) + ".jpg"
78.      values = row[2:18]
79.      v = get_values(values)
80.      image_labels[name] = v
81.
82.    index += 1
83.
84. j = json.dumps(image_labels)
85. f = open("/gdrive/MyDrive/Diplomski/EmotionNet/image_labels_with_int.json",
    "w")
86. f.write(j)
87. f.close()
```

Faza treninga baza podataka nakon prijenosa slika i anotacija na Google Disk

```
1. from google.colab import drive
2. drive.mount('/gdrive')
3. ####################################
4.
5. # Incijalizacija mreze
6. import keras,os
7. from keras.models import Sequential
8. from keras.layers import Dense, Conv2D, MaxPool2D , Flatten
9. from keras.preprocessing.image import ImageDataGenerator
10. from tensorflow.keras.optimizers import Adam
11. import numpy as np
12.
13. vgg16 = Sequential()
```

```
14. vgg16.add(Conv2D(input_shape=(224,224,3),filters=64,kernel_size=(3,3),paddi
    ng="same", activation="relu"))
15. vgg16.add(Conv2D(filters=64,kernel_size=(3,3),padding="same",
    activation="relu"))
16. vgg16.add(MaxPool2D(pool_size=(2,2),strides=(2,2)))
17. vgg16.add(Conv2D(filters=128, kernel_size=(3,3), padding="same",
    activation="relu"))
18. vgg16.add(Conv2D(filters=128, kernel_size=(3,3), padding="same",
    activation="relu"))
19. vgg16.add(MaxPool2D(pool_size=(2,2),strides=(2,2)))
20. vgg16.add(Conv2D(filters=256, kernel_size=(3,3), padding="same",
    activation="relu"))
21. vgg16.add(Conv2D(filters=256, kernel_size=(3,3), padding="same",
    activation="relu"))
22. vgg16.add(Conv2D(filters=256, kernel_size=(3,3), padding="same",
    activation="relu"))
23. vgg16.add(MaxPool2D(pool_size=(2,2),strides=(2,2)))
24. vgg16.add(Conv2D(filters=512, kernel_size=(3,3), padding="same",
    activation="relu"))
25. vgg16.add(Conv2D(filters=512, kernel_size=(3,3), padding="same",
    activation="relu"))
26. vgg16.add(Conv2D(filters=512, kernel_size=(3,3), padding="same",
    activation="relu"))
27. vgg16.add(MaxPool2D(pool_size=(2,2),strides=(2,2)))
28. vgg16.add(Conv2D(filters=512, kernel_size=(3,3), padding="same",
    activation="relu"))
29. vgg16.add(Conv2D(filters=512, kernel_size=(3,3), padding="same",
    activation="relu"))
30. vgg16.add(Conv2D(filters=512, kernel_size=(3,3), padding="same",
    activation="relu"))
31. vgg16.add(MaxPool2D(pool_size=(2,2),strides=(2,2)))
32.
33. vgg16.add(Flatten())
34. vgg16.add(Dense(units=4096,activation="relu"))
35. vgg16.add(Dense(units=4096,activation="relu"))
36. vgg16.add(Dense(units=7, activation="softmax"))
37.
38. opt = Adam(learning_rate=0.0001)
39. vgg16.compile(optimizer=opt , loss=keras.losses.categorical_crossentropy,
    metrics=['accuracy'])
40. vgg16.summary()
41. ####################################
42.
43. emotions = {
44.     1 : "Iznenadenje",
45.     2 : "Strah",
46.     3 : "Gadenje",
47.     4 : "Sreca",
48.     5 : "Tuga",
49.     6 : "Ljutnja",
50.     7 : "Neutralno"
51. }
52.
53. import csv
54.
55. def get_labels_map(path):
```

```
  map = {}

  with open(path) as file:
    tsv_file = csv.reader(file, delimiter=" ")
    for line in tsv_file:
      size = len(line[0])
      map[line[0][:size-4]] = int(line[1])

  return map
####################################

labels =
   get_labels_map("/gdrive/MyDrive/Diplomski/AffectNet/train_set/annotations/A
   ffect_25k_Marko_uredeno_finalno.txt")
####################################

import keras,os
from keras.models import Sequential
from keras.layers import Dense, Conv2D, MaxPool2D , Flatten
from keras.preprocessing.image import ImageDataGenerator
import numpy as np
import cv2
import os

base_path = /gdrive/MyDrive/Diplomski/AffectNet/train_set/images/25k"

#def make_tuples(X, Y):
  #tuples = []
  #for i in range(len(X)):
   # tuples.append((X[i],Y[i]))
  #return tuples

def load_images_and_labels_from_folder_Affect(folder, label_map,x):
    resized = np.zeros((x,224,224,3))
    labels = np.zeros((x,7))
    br=0
    for i in label_map:
        img =
   cv2.imread('/gdrive/MyDrive/Diplomski/AffectNet/train_set/images/25k/'+str(
   i)+'.jpg')
        res = cv2.resize(img, dsize=(224, 224))
        reshaped = res.reshape(1,224,224,3) # return the image with shaping
   that TF wants.
        resized[br][:][:][:]=reshaped
        index = label_map[i] - 1
        labels[br][index]=1
        br+=1
        print(br)
    return resized, labels
#####################################

train_images, train_labels = load_images_and_labels_from_folder(base_path
   + "/Train", labels,12271)
print("Extracted train images")
test_images, test_labels =
   load_images_and_labels_from_folder_Affect(base_path, labels,3068)
```

```
print("Extracted test images")
####################################

from keras.callbacks import ModelCheckpoint, EarlyStopping

checkpoint =
   ModelCheckpoint("/gdrive/MyDrive/Diplomski/AffectNet/Weights/vgg16_1.h5",
   monitor='val_accuracy', verbose=1, save_best_only=True,
   save_weights_only=False, mode='auto', save_freq="epoch")
early = EarlyStopping(monitor='val_accuracy', min_delta=0, patience=5,
   verbose=1, mode='auto')
####################################

import tensorflow as tf

tf.config.run_functions_eagerly(True)
hist = vgg16.fit(x=train_images, y=train_labels,
   batch_size=32,epochs=10,validation_data=(test_images,test_labels),verbose=1
   ,shuffle=True,callbacks=[checkpoint,early]
```

Prikaz kretanja točnosti i gubitka te izrada matrice konfuzije

```
#vizualizacija
import matplotlib.pyplot as plt
from google.colab import files

# summarize history for accuracy
plt.rcParams['figure.figsize'] = [15,10]
plt.plot(hist.history['accuracy'])
plt.plot(hist.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()

# summarize history for loss
plt.plot(hist.history['loss'])
plt.plot(hist.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
####################################

def get_most_probable_class(predictions):
  index = np.argmax(predictions)
  return emotions[index + 1]
####################################

import cv2
from google.colab.patches import cv2_imshow
def get_cropped_face_image(path):
  # Read the input image
```

```
34.  img = cv2.imread(path)
35.
36.  # Convert into grayscale
37.  gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
38.  face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades +
   'haarcascade_frontalface_default.xml')
39.#  face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_alt2.xml')
40.
41.  # Detect faces
42.  faces = face_cascade.detectMultiScale(gray, 1.1, 4)
43.
44.
45.  for (x, y, w, h) in faces:
46.      face = img[y:y + h, x:x + w]
47.
48.  img_224=cv2.resize(face, dsize=(224, 224))
49.  cv2_imshow(img_224)
50.  img_reshaped = img_224.reshape(1,224,224,3)
51.  return img_reshaped
52.####################################
53.
54.# cross validate
55.
56.from sklearn.metrics import confusion_matrix
57.import seaborn as sns
58.
59.def plot_confusion_matrix(actual_classes, predicted_classes,
   sorted_labels):
60.
61.    matrix = confusion_matrix(actual_classes, predicted_classes,
   labels=sorted_labels)
62.
63.    plt.figure(figsize=(15,10))
64.    sns.heatmap(matrix, annot=True, xticklabels=sorted_labels,
   yticklabels=sorted_labels, cmap="Blues", fmt="g")
65.    plt.xlabel('Predicted'); plt.ylabel('Actual'); plt.title('Confusion
   Matrix')
66.
67.    plt.show()
68.
69.predicts = vgg16.predict(test_images)
70.####################################
71.
72.total_classes = len(train_labels)
73.train_emotions = {}
74.for l in train_labels:
75.  c = get_most_probable_class(l)
76.  if c not in train_emotions:
77.    train_emotions[c] = 0
78.  train_emotions[c] += 1
79.
80.for e in train_emotions:
81.  f = "Emotion {} - Num: {} Percent: {}".format(e, train_emotions[e],
   train_emotions[e]/float(total_classes) * 100)
82.  print(f)
83.####################################
```

```
84.
85. actual_classes = []
86. for l in test_labels:
87.   actual_classes.append(get_most_probable_class(l))
88.
89. predicted_classes = []
90. for l in predicts:
91.   predicted_classes.append(get_most_probable_class(l))
92.
93. sorted_labels = []
94. for l in emotions:
95.   sorted_labels.append(emotions[l])
96.
97. plot_confusion_matrix(actual_classes, predicted_classes, sorted_labels)
```

K-struka unakrsna validacija s RAF-DB bazom podataka

```
1.  from google.colab import drive
2.  drive.mount('/gdrive')
3.  ####################################
4.
5.  # Incijalizacija mreze
6.  import keras,os
7.  from keras.models import Sequential
8.  from keras.layers import Dense, Conv2D, MaxPool2D , Flatten
9.  from keras.preprocessing.image import ImageDataGenerator
10. from tensorflow.keras.optimizers import Adam
11. import numpy as np
12.
13. vgg16 = Sequential()
14. vgg16.add(Conv2D(input_shape=(224,224,3),filters=64,kernel_size=(3,3),paddi
    ng="same", activation="relu"))
15. vgg16.add(Conv2D(filters=64,kernel_size=(3,3),padding="same",
    activation="relu"))
16. vgg16.add(MaxPool2D(pool_size=(2,2),strides=(2,2)))
17. vgg16.add(Conv2D(filters=128, kernel_size=(3,3), padding="same",
    activation="relu"))
18. vgg16.add(Conv2D(filters=128, kernel_size=(3,3), padding="same",
    activation="relu"))
19. vgg16.add(MaxPool2D(pool_size=(2,2),strides=(2,2)))
20. vgg16.add(Conv2D(filters=256, kernel_size=(3,3), padding="same",
    activation="relu"))
21. vgg16.add(Conv2D(filters=256, kernel_size=(3,3), padding="same",
    activation="relu"))
22. vgg16.add(Conv2D(filters=256, kernel_size=(3,3), padding="same",
    activation="relu"))
23. vgg16.add(MaxPool2D(pool_size=(2,2),strides=(2,2)))
24. vgg16.add(Conv2D(filters=512, kernel_size=(3,3), padding="same",
    activation="relu"))
25. vgg16.add(Conv2D(filters=512, kernel_size=(3,3), padding="same",
    activation="relu"))
26. vgg16.add(Conv2D(filters=512, kernel_size=(3,3), padding="same",
    activation="relu"))
```

```
vgg16.add(MaxPool2D(pool_size=(2,2),strides=(2,2)))
vgg16.add(Conv2D(filters=512, kernel_size=(3,3), padding="same",
   activation="relu"))
vgg16.add(Conv2D(filters=512, kernel_size=(3,3), padding="same",
   activation="relu"))
vgg16.add(Conv2D(filters=512, kernel_size=(3,3), padding="same",
   activation="relu"))
vgg16.add(MaxPool2D(pool_size=(2,2),strides=(2,2)))

vgg16.add(Flatten())
vgg16.add(Dense(units=4096,activation="relu"))
vgg16.add(Dense(units=4096,activation="relu"))
vgg16.add(Dense(units=7, activation="softmax"))

opt = Adam(learning_rate=0.0001)
vgg16.compile(optimizer=opt , loss=keras.losses.categorical_crossentropy,
   metrics=['accuracy'])
vgg16.summary()
####################################

emotions = {
    1 : "Iznenadenje",
    2 : "Strah",
    3 : "Gadenje",
    4 : "Sreca",
    5 : "Tuga",
    6 : "Ljutnja",
    7 : "Neutralno"
}

import csv

def get_labels_map(path):
  map = {}

  with open(path) as file:
    tsv_file = csv.reader(file, delimiter=" ")
    for line in tsv_file:
      size = len(line[0])
      map[line[0][:size-4]] = int(line[1])

  return map
####################################

labels = get_labels_map("/gdrive/MyDrive/Diplomski/RAF-
   DB/list_patition_label.txt")
####################################

import keras,os
from keras.models import Sequential
from keras.layers import Dense, Conv2D, MaxPool2D , Flatten
from keras.preprocessing.image import ImageDataGenerator
import numpy as np
import cv2
import os

```

```
base_path = "/gdrive/MyDrive/Diplomski/RAF-DB"

#def make_tuples(X, Y):
  #tuples = []
  #for i in range(len(X)):
   # tuples.append((X[i],Y[i]))
  #return tuples

def load_images_and_labels_from_folder(folder, label_map,x):
    resized = np.zeros((x,224,224,3))
    labels = np.zeros((x,7))
    br=0
    for filename in os.listdir(folder):
        img = cv2.imread(os.path.join(folder,filename))
        if img is not None:
            res = cv2.resize(img, dsize=(224, 224))
            reshaped = res.reshape(1,224,224,3) # return the image with shaping that TF wants.
            resized[br][:][:][:]=reshaped

            label_key = filename.split("_aligned")[0]

            l = np.zeros((1,7))
              index = label_map[label_key] - 1
              l[0][index] = 1
              labels[br][:]=l
              br+=1
    return resized, labels
####################################

train_images, train_labels = load_images_and_labels_from_folder(base_path + "/Train", labels,12271)
print("Extracted train images")
test_images, test_labels = load_images_and_labels_from_folder(base_path + "/Test", labels,3068)
print("Extracted test images")
####################################

# We create our model only once
def create_model():
    model = get_model()
    return model, model.get_weights()

# we initialize it multiple times
def init_weight(same_old_model, first_weights):

    same_old_model.set_weights(first_weights)
####################################

from sklearn.model_selection import KFold
from keras import backend as K
import gc

def kfold():
  acc_per_fold = []
  loss_per_fold = []
```

```
131.   # Merge inputs and targets
132.   inputs = np.concatenate((train_images, test_images), axis=0)
133.   targets = np.concatenate((train_labels, test_labels), axis=0)
134.
135.   kfold = KFold(n_splits=5, shuffle=True)
136.   fold = 1
137.   best_acc = -1
138.   for train_index, test_index in kfold.split(inputs, targets):
139.     model = get_model()
140.
141.     print("------------------------------------------------")
142.     print(f'Training for fold {fold} ...')
143.
144.     # fit with train split
145.     hist = model.fit(inputs[train_index], targets[train_index],
   epochs=10,verbose=0)
146.
147.     # evaluate on test split
148.     scores = model.evaluate(inputs[test_index], targets[test_index],
   verbose=0)
149.     print(f'Score for fold {fold}: {model.metrics_names[0]} of
   {scores[0]}; {model.metrics_names[1]} of {scores[1]*100}%')
150.
151.     # save the best one
152.     if scores[1] > best_acc:
153.       model.save("/gdrive/MyDrive/Diplomski/RAF-
   DB/Weights/kfold/vgg_best")
154.       best_acc = scores[1]
155.
156.     acc_per_fold.append(scores[1] * 100)
157.     loss_per_fold.append(scores[0])
158.     fold += 1
159.
160.     # release background memory so RAM survives
161.     K.clear_session()
162.     del model
163.     gc.collect()
164.
165.   # == Provide average scores ==
166.   print('------------------------------------------------------------------
   -------')
167.   print('Score per fold')
168.   for i in range(0, len(acc_per_fold)):
169.     print('------------------------------------------------------------------
   ---------')
170.     print(f'> Fold {i+1} - Loss: {loss_per_fold[i]} - Accuracy:
   {acc_per_fold[i]}%')
171.   print('------------------------------------------------------------------
   -------')
172.   print('Average scores for all folds:')
173.   print(f'> Accuracy: {np.mean(acc_per_fold)} (+-
   {np.std(acc_per_fold)})')
174.   print(f'> Loss: {np.mean(loss_per_fold)}')
175.   print('------------------------------------------------------------------
   -------')
176. ####################################
```

```
177.
178. kfold()
```

Testiranje podataka

```
1. from google.colab import drive
2. drive.mount('/gdrive')
3. ###################################
4.
5. # Incijalizacija mreze
6. import keras,os
7. from keras.models import Sequential
8. from keras.layers import Dense, Conv2D, MaxPool2D , Flatten
9. from keras.preprocessing.image import ImageDataGenerator
10. from tensorflow.keras.optimizers import Adam
11. import numpy as np
12.
13. vgg16 = Sequential()
14. vgg16.add(Conv2D(input_shape=(224,224,3),filters=64,kernel_size=(3,3),paddi
    ng="same", activation="relu"))
15. vgg16.add(Conv2D(filters=64,kernel_size=(3,3),padding="same",
    activation="relu"))
16. vgg16.add(MaxPool2D(pool_size=(2,2),strides=(2,2)))
17. vgg16.add(Conv2D(filters=128, kernel_size=(3,3), padding="same",
    activation="relu"))
18. vgg16.add(Conv2D(filters=128, kernel_size=(3,3), padding="same",
    activation="relu"))
19. vgg16.add(MaxPool2D(pool_size=(2,2),strides=(2,2)))
20. vgg16.add(Conv2D(filters=256, kernel_size=(3,3), padding="same",
    activation="relu"))
21. vgg16.add(Conv2D(filters=256, kernel_size=(3,3), padding="same",
    activation="relu"))
22. vgg16.add(Conv2D(filters=256, kernel_size=(3,3), padding="same",
    activation="relu"))
23. vgg16.add(MaxPool2D(pool_size=(2,2),strides=(2,2)))
24. vgg16.add(Conv2D(filters=512, kernel_size=(3,3), padding="same",
    activation="relu"))
25. vgg16.add(Conv2D(filters=512, kernel_size=(3,3), padding="same",
    activation="relu"))
26. vgg16.add(Conv2D(filters=512, kernel_size=(3,3), padding="same",
    activation="relu"))
27. vgg16.add(MaxPool2D(pool_size=(2,2),strides=(2,2)))
28. vgg16.add(Conv2D(filters=512, kernel_size=(3,3), padding="same",
    activation="relu"))
29. vgg16.add(Conv2D(filters=512, kernel_size=(3,3), padding="same",
    activation="relu"))
30. vgg16.add(Conv2D(filters=512, kernel_size=(3,3), padding="same",
    activation="relu"))
31. vgg16.add(MaxPool2D(pool_size=(2,2),strides=(2,2)))
32.
33. vgg16.add(Flatten())
34. vgg16.add(Dense(units=4096,activation="relu"))
35. vgg16.add(Dense(units=4096,activation="relu"))
```

```
vgg16.add(Dense(units=7, activation="softmax"))

opt = Adam(learning_rate=0.0001)
vgg16.compile(optimizer=opt , loss=keras.losses.categorical_crossentropy,
   metrics=['accuracy'])
vgg16.summary()
####################################

emotions = {
    1 : "Iznenadenje",
    2 : "Strah",
    3 : "Gadenje",
    4 : "Sreca",
    5 : "Tuga",
    6 : "Ljutnja",
    7 : "Neutralno"
}

import csv

def get_labels_map(path):
  map = {}

  with open(path) as file:
    tsv_file = csv.reader(file, delimiter=" ")
    for line in tsv_file:
      size = len(line[0])
      map[line[0][:size-4]] = int(line[1])

  return map
####################################

from keras.models import load_model
vgg16 =
   load_model("/gdrive/MyDrive/Diplomski/AffectNet/Weights_Marko/vgg16_1.h5")
####################################

def get_most_probable_class(predictions):
  index = np.argmax(predictions)
  return emotions[index + 1]
####################################

import cv2
from google.colab.patches import cv2_imshow
def get_cropped_face_image(path):
  # Read the input image
  img = cv2.imread(path)

  # Convert into grayscale
  gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
  face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades +
   'haarcascade_frontalface_default.xml')
#  face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_alt2.xml')

  # Detect faces
  faces = face_cascade.detectMultiScale(gray, 1.1, 4)
```

```
89.
90.
91.  for (x, y, w, h) in faces:
92.      face = img[y:y + h, x:x + w]
93.
94.  img_224=cv2.resize(face, dsize=(224, 224))
95.  cv2_imshow(img_224)
96.  img_reshaped = img_224.reshape(1,224,224,3)
97.  return img_reshaped
98.  ####################################
99.
100. marko =
   get_cropped_face_image("/gdrive/MyDrive/Diplomski/Slike_za_testiranje/marko
   .jpg")
101. predicts = vgg16.predict(marko)
102. print(get_most_probable_class(predicts[0]))
```