Korištenje mekih biometrijskih karakteristika za autentikaciju osoba

Šimović, Fran

SVEUČILIŠTE U ZAGREBU
FAKULTET ORGANIZACIJE I INFORMATIKE
V A R A Ž D I N

Fran Šimović

KORIŠTENJE MEKIH BIOMETRIJSKIH KARAKTERISTIKA ZA AUTENTIKACIJU OSOBA

DIPLOMSKI RAD

Varaždin, 2023.

SVEUČILIŠTE U ZAGREBU
FAKULTET ORGANIZACIJE I INFORMATIKE
V A R A Ž D I N

Fran Šimović
Matični broj: 45345/16–R
Studij: ***Organizacija poslovnih sustava***

KORIŠTENJE MEKIH BIOMETRIJSKIH KARAKTERISTIKA ZA AUTENTIKACIJU OSOBA

DIPLOMSKI RAD

Mentor/Mentorica:
Izv. prof. dr. sc. Petra Grd

Varaždin, rujan 2023.

Fran Šimović

Izjava o izvornosti

Izjavljujem da je moj diplomski rad izvorni rezultat mojeg rada te da se u izradi istoga nisam koristio drugim izvorima osim onima koji su u njemu navedeni. Za izradu rada su korištene etički prikladne i prihvatljive metode i tehnike rada.

Autor/Autorica potvrdio/potvrdila prihvaćanjem odredbi u sustavu FOI-radovi

_____

Sažetak

Ovaj diplomski rad bavi se korištenjem mekih biometrijskih značajki u svrhu autentifikacije. Meka biometrija su diskretne fizičke karakteristike ili karakteristike ponašanja koje se mogu koristiti za identifikaciju ili razlikovanje pojedinaca. Rad govori o upotrebi meke biometrije u strojnom učenju, važnosti privatnosti i sigurnosti te alatima koji se koriste za razvoj modela strojnog učenja, uključujući Python, NumPy, Keras, Tensorflow, Matplotlib, Scikit-learn i Jupyter. Rad također pokriva koncepte dubokog učenja i konvolucijskih neuronskih mreža, uključujući slojeve uključene u izgradnju takvog modela, funkcije optimizacije i gubitka te funkcije aktivacije. Rad završava raspravom o obučavanju modela pomoću modela MobileNet i iznosi autorove zaključke.

Ključne riječi: strojno učenje, sigurnost, TensorFlow, Python, duboko učenje, biometrija

Sadržaj

1. Uvod 1
2. Meka biometrija 2
2.1. Privatnost i sigurnost 4
3. Alati 6
3.1. Python 6
3.2. NumPy 7
3.3. Keras 7
3.4. Tensorflow 8
3.5. Matplotlib 9
3.6. Scikit-learn 10
3.7. Jupyter notebooks 10
4. Strojno učenje 11
4.1. Duboko učenje 12
4.2. Konvolucijske mreže 13
4.2.1. Konvolucijski sloj 17
4.2.2. Gusti sloj 18
4.2.3. Dropout sloj 21
4.2.4. Funkcija optimizacije 22
4.2.5. Funkcija gubitka 23
4.2.6. Aktivacijska funkcija 25
5. Treniranje modela za klasifikaciju spola, dobi i etničke pripadnosti 27
5.1. MobileNet model 30
5.2. Skup podataka 31
5.2.1. Distribucija podataka 32
5.3. Treniranje modela i priprema dataseta 35
5.3.1. Kreiranje python env varijable 35
5.3.2. Učitavanje i označavanje dataseta 36
5.3.3. Kreiranje i obuka modela 38
5.3.3.1. Treniranje modela za predviđanje spola 38
5.3.3.2. Treniranje modela za predviđanje etničke pripadnosti 43
5.3.3.3. Treniranje modela za predviđanje dobi 45
5.3.4. Iscrtavanje performansi modela 46
5.3.4.1. Performanse modela za predikciju spola 46
5.3.4.2. Performanse modela za predikciju etničke pripadnosti 49
5.3.4.3. Performanse modela za predikciju dobi 50
5.3.5. Testiranje modela 52
5.3.6. Spremanje modela 55

5.3.7. Aplikacija za klasifikaciju fotografija osoba 56
6. Zaključak 64
7. Popis literature 66
8. Popis slika 69
9. Popis tablica 71

1. Uvod

Posljednjih je godina upotreba meke biometrije u strojnom učenju stekla široku popularnost kao sredstvo za klasifikaciju i identifikaciju pojedinaca. To je djelomično zbog sve veće dostupnosti podataka i napretka u algoritmima strojnog učenja, koji su omogućili izdvajanje i analizu ovih karakteristika s većom točnošću i učinkovitosti.

Međutim, korištenje meke biometrije u strojnom učenju također postavlja niz važnih pitanja i izazova. Primjerice, kako možemo osigurati da su podaci o obuci koji se koriste za izradu modela strojnog učenja raznoliki i reprezentativni za populaciju koja se klasificira? Koji su potencijalni rizici i etička razmatranja korištenja ove tehnologije i kako ih možemo ublažiti? Ovo su samo neka od pitanja koja će biti istražena u ovom eseju, uz teme strojnog učenja i meke biometrije.

Od sigurnosti i nadzora do zdravstvene zaštite i maloprodaje, potencijalna upotreba ove tehnologije čini se gotovo neograničenom. U području sigurnosti i nadzora, na primjer, algoritmi strojnog učenja trenirani na mekim biometrijskim karakteristikama mogli bi se koristiti za identifikaciju pojedinaca u gomili ili na sigurnosnim snimkama s većom preciznošću i brzinom nego što je trenutno moguće tradicionalnim metodama. U zdravstvu bi se modeli strojnog učenja mogli koristiti za predviđanje i prevenciju bolesti analizom podataka o pacijentima i identificiranjem obrazaca i čimbenika rizika. U maloprodaji bi se algoritmi strojnog učenja mogli koristiti za personalizaciju korisničkih iskustava i poboljšanje marketinških strategija analizom podataka o kupcima i predviđanjem njihovih preferencija i ponašanja.

Unatoč brojnim potencijalnim prednostima strojnog učenja i meke biometrije, postoji i niz etičkih pitanja i pitanja privatnosti koja se moraju uzeti u obzir. Primjerice, postoji rizik da bi se ova tehnologija mogla koristiti za nepravednu diskriminaciju određenih skupina ili pojedinaca. Također postoji mogućnost zlouporabe ili zlouporabe osobnih podataka prikupljenih u svrhu strojnog učenja. Kao rezultat toga, bitno je da pažljivo razmotrimo etičke implikacije korištenja strojnog učenja i meke biometrije te da uvedemo odgovarajuće zaštitne mjere za zaštitu privatnosti i prava pojedinaca.

Općenito, korištenje strojnog učenja i meke biometrije fascinantno je polje koje se brzo razvija s potencijalom da revolucionira način na koji klasificiramo i identificiramo pojedince. Ovaj rad nastoji dublje proniknuti u potencijalne prednosti i izazove korištenja ove tehnologije i istražiti njezine različite primjene u različitim područjima.

2. Meka biometrija

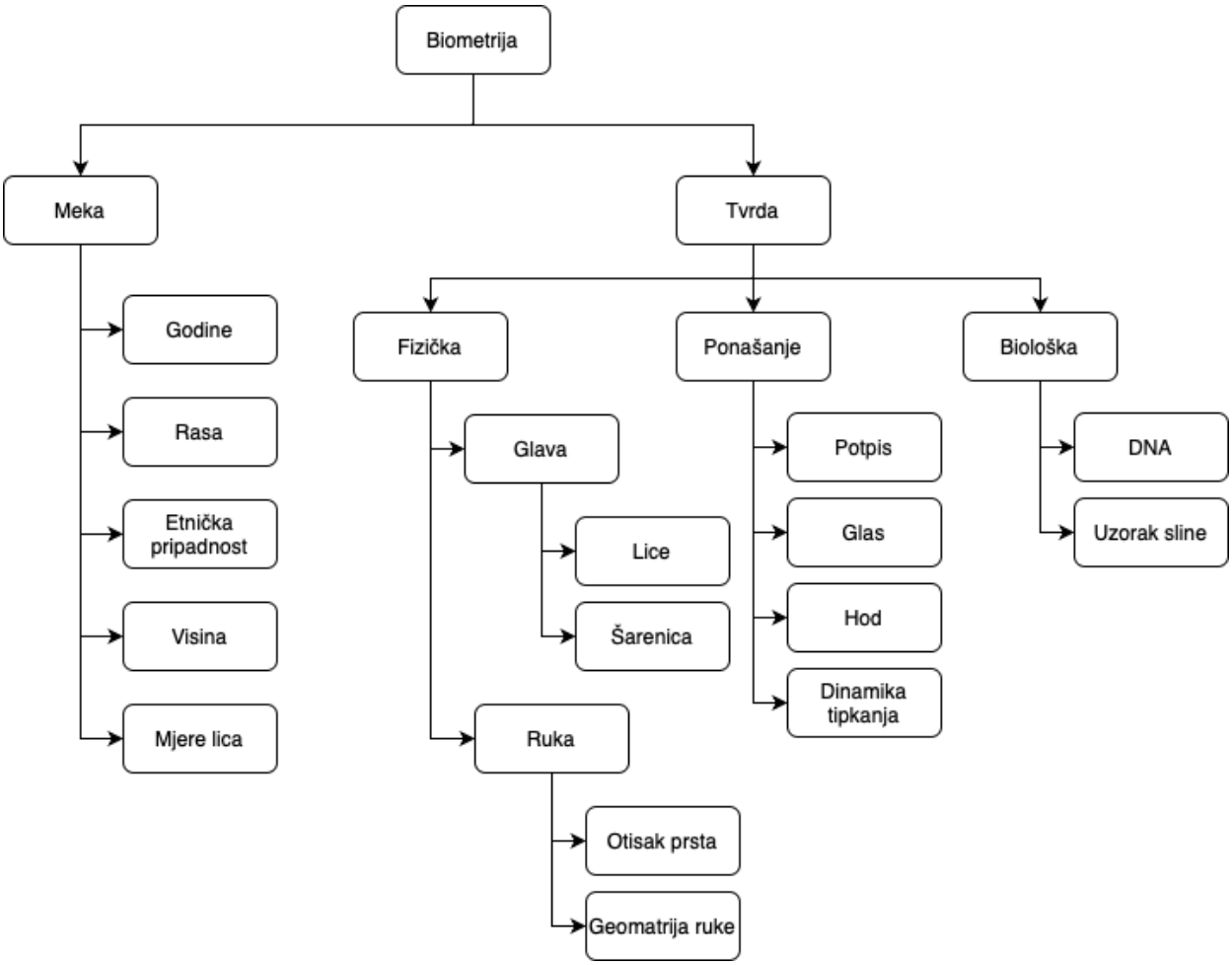
Fizičke karakteristike ili karakteristike ponašanja koje se mogu koristiti za ljudsku identifikaciju ili prepoznavanje, nazivaju se meke biometrije, no nisu jedinstvene ili trajne kao tradicionalne biometrijske karakteristike kao što su, primjerice otisci prstiju ili DNK. Primjeri meke biometrije (prikazano na slici 1) uključuju visinu, težinu, spol, dob, etničku pripadnost, frizuru i crte lica (Maiorana, 2018).

Jedna od glavnih prednosti meke biometrije je ta što se može jednostavno i jeftino snimiti standardnim kamerama ili senzorima, bez potrebe za specijaliziranom opremom ili obučenim operaterima (Zhang, 2019). To ih čini posebno korisnima za aplikacije kao što je nadzor, gdje je važno točno identificirati i pratiti pojedince u stvarnom vremenu, bez potrebe za njihovom suradnjom ili pristankom.

Zbog svoje inherentne varijabilnosti i prisutnosti šuma u procesu mjerenja, jedan je od glavnih izazova u korištenju meke biometrije za identifikaciju, što je klasificira kao manje pouzdanu i točnu za razliku od tradicionalne biometrije (Jain, 2015). Na primjer, visina i težina osobe mogu varirati tijekom vremena zbog promjena u fizičkom stanju, dok na izgled crta lica mogu utjecati čimbenici kao što su starenje, izrazi lica i uvjeti osvjetljenja.

Kako bi prevladali te izazove, istraživači su razvili razne tehnike za poboljšanje točnosti i pouzdanosti mekih biometrijskih sustava. Ove tehnike uključuju upotrebu algoritama strojnog učenja za učenje uzoraka i odnosa iz velikih skupova podataka mekih biometrijskih mjerenja (Ko, 2015), upotrebu višestrukih mekih biometrijskih obilježja u kombinaciji za povećanje jedinstvenosti rezultirajućeg identifikatora (Maiorana, 2018) i korištenje tehnika spajanja za kombiniranje rezultata više biometrijskih modaliteta za poboljšanje ukupne izvedbe (Zhang, 2019).

Jedan od primjera uspješne primjene meke biometrije je korištenje procjene spola i dobi za automatsku identifikaciju pojedinaca u video nadzoru. U ovoj se aplikaciji algoritmi strojnog učenja obučavaju na velikim skupovima podataka slika ili videozapisa pojedinaca s poznatim oznakama spola i dobi, a zatim se koriste za predviđanje spola i dobi pojedinaca na novim slikama ili videozapisima (Maiorana, 2018). Te se informacije mogu koristiti za sužavanje potrage za određenim pojedincem u velikoj bazi podataka nadzornih snimaka ili za upozorenje vlastima o prisutnosti pojedinca od interesa na određenoj lokaciji.



Slika 1: Podjela biometrije (Vlastita izrada prema: https://www.intechopen.com/media/chapter/60675/media/F2.png)

Općenito, meka biometrija predstavlja obećavajuće područje istraživanja i razvoja, s potencijalom da revolucionira način na koji identificiramo i pratimo pojedince u raznim aplikacijama (Zhang, 2019). Uz kontinuirani napredak u strojnom učenju i biometrijskoj tehnologiji, vjerojatno će meka biometrija igrati sve važniju ulogu u budućnosti ljudske identifikacije i prepoznavanja (Ko, 2015).

Meka biometrija ima potencijal revolucionalizirati način na koji identificiramo i pratimo pojedince u raznim aplikacijama. Za razliku od tradicionalnih biometrijskih karakteristika, koje zahtijevaju specijaliziranu opremu i obučene operatere, meka biometrija može se jednostavno i jeftino uhvatiti pomoću standardnih kamera ili senzora (slika 2). To ih čini posebno korisnima za aplikacije kao što je nadzor, gdje je važno točno identificirati i pratiti pojedince u stvarnom vremenu, bez potrebe za njihovom suradnjom ili pristankom. Iako sigurno postoje izazovi u korištenju meke biometrije za identifikaciju, kao što je njihova inherentna varijabilnost i prisutnost šuma u procesu mjerenja, ti se izazovi mogu prevladati upotrebom naprednih

algoritama strojnog učenja i tehnika spajanja. Ukratko, meka biometrija predstavlja obećavajuće područje istraživanja i razvoja, s potencijalom za značajno poboljšanje točnosti i pouzdanosti sustava za identifikaciju i prepoznavanje ljudi.

Slika 2: Autentifikacija osoba po načinu hoda u Kini (Izvor: https://static.independent.co.uk/s3fs-public/thumbnails/image/2018/11/07/14/china-surveillance-gait-recognition.jpg?quality=75&width=1200&auto=webp)

2.1. Privatnost i sigurnost

Kako tehnologije strojnog učenja i nadzora nastavljaju napredovati i postaju sve prisutnije u društvu, postoji sve veća zabrinutost oko implikacija njihove upotrebe na sigurnost i privatnost (Zhang, 2019). Iako ove tehnologije mogu pružiti niz prednosti, poput poboljšane sigurnosti, učinkovitosti i praktičnosti (Maiorana, 2018), one također pokreću niz etičkih i pravnih pitanja koja treba pažljivo razmotriti (Jain, 2015) .

Jedna od glavnih zabrinutosti korištenja strojnog učenja u nadzoru je mogućnost pristranih ili diskriminirajućih ishoda (Ko, 2015). Na primjer, ako se sustav strojnog učenja obučava na skupu podataka koji je na neki način pristran ili neuravnotežen, može proizvesti

pristrane ili diskriminirajuće rezultate kada se primijeni na nove podatke (Zhang, 2019). To može dovesti do neproporcionalnog ciljanja ili profiliranja određenih skupina ljudi, na temelju čimbenika kao što su rasa, spol ili etnička pripadnost, što može imati ozbiljne posljedice za njihovu privatnost i prava (Maiorana, 2018).

Kako bi se riješio ovaj problem, važno je da se sustavi strojnog učenja koji se koriste za nadzor obučavaju na skupovima podataka koji su reprezentativni i uravnoteženi (Ko, 2015), te da se ulažu napori kako bi se osiguralo da su korišteni algoritmi pošteni i nepristrani (Jain, 2015). To može uključivati upotrebu tehnika kao što je pretprocesiranje podataka i korekcija pristranosti (Zhang, 2019) ili usvajanje metrike pravednosti i ograničenja kako bi se osiguralo da algoritmi nisu pristrani prema određenim skupinama (Maiorana, 2018).

Još jedan potencijalni problem u uporabi strojnog učenja i nadzora je mogućnost pogrešne upotrebe ili zlouporabe prikupljenih informacija (Jain, 2015). Na primjer, ako podaci prikupljeni sustavom nadzora nisu pravilno osigurani ili zaštićeni (Ko, 2015), mogu im pristupiti neovlaštene strane, koje bi ih mogle upotrijebiti u opake svrhe kao što su krađa identiteta, prijevara ili ucjena (Zhang, 2019). Kako bi se riješio ovaj problem, važno je da postoje snažne sigurnosne mjere za zaštitu podataka prikupljenih sustavima nadzora (Maiorana, 2018), te da postoje jasne politike i procedure za korištenje i dijeljenje tih podataka (Ko, 2015).

Konačno, postoji i potencijalni problem mogućeg utjecaja strojnog učenja i nadzora na građanske slobode i privatnost (Jain, 2015). Iako ove tehnologije mogu pružiti niz prednosti (Maiorana, 2018), one također imaju potencijal narušavanja privatnosti ljudi i ograničavanja njihove slobode kretanja i izražavanja (Zhang, 2019). Kako bi se riješila ova zabrinutost, važno je da postoje jasni zakoni i propisi kojima se upravlja korištenjem strojnog učenja i nadzora (Ko, 2015) te da postoje jake zaštitne mjere za zaštitu privatnosti i prava ljudi (Jain, 2015).

Općenito, sigurnosne i etičke implikacije korištenja strojnog učenja i nadzora složene su i višestruke (Maiorana, 2018), te zahtijevaju pažljivo razmatranje i upravljanje (Zhang, 2019). Proaktivnim i odgovornim pristupom razvoju i korištenju ovih tehnologija (Ko, 2015) moguće je maksimizirati koristi i minimizirati rizike (Jain, 2015), te osigurati da se one koriste u način koji je u skladu s temeljnim ljudskim pravima i vrijednostima (Maiorana, 2018).

3. Alati

U ovom poglavlju će biti spomenuti i ukratko opisani programski jezik i vezane biblioteke koje se često koriste u izradi i treniranju modela za klasifikaciju. Isto tako, spomenuti alati će se iskoristiti i prilikom rada na praktičnom dijelu.

3.1. Python

Python je programski jezik opće namjene visoke razine koji se koristi za web razvoj, analizu podataka, umjetnu inteligenciju, znanstveno računalstvo i mnoge druge aplikacije. Poznat je po svojoj jednostavnosti, čitljivosti i fleksibilnosti, što ga čini idealnim izborom za početnike i iskusne programere. (Rossum, 2003)

Neke ključne značajke Pythona uključuju:

- Dinamičko pisanje: U Pythonu ne morate navesti tip podataka varijable kada je deklarirate. Interpretator će automatski odrediti tip podataka varijable na temelju vrijednosti koju joj dodijelite.
- Objektno orijentirano programiranje: Python je objektno orijentiran jezik, što znači da podržava stvaranje i korištenje objekata. Objekti su samostalni dijelovi koda kojima se može manipulirati i koristiti za predstavljanje entiteta stvarnog svijeta.
- Opsežna standardna biblioteka: Python dolazi s velikom standardnom bibliotekom koja uključuje module za širok raspon zadataka, kao što je povezivanje s web poslužiteljima, čitanje i pisanje datoteka i rad s podacima.
- Biblioteke trećih strana: uz standardnu biblioteku, za Python su dostupne tisuće biblioteka trećih strana koje pružaju dodatnu funkcionalnost. Ove se biblioteke mogu jednostavno instalirati i uvesti u vaš kôd kako bi se proširile mogućnosti jezika.
- Interaktivni tumač: Python dolazi s interaktivnim tumačem koji vam omogućuje pisanje i izvršavanje koda u stvarnom vremenu. To olakšava testiranje i otklanjanje pogrešaka koda te istraživanje jezika i njegovih značajki.
- Lak za naučiti: Python ima jednostavnu i dosljednu sintaksu, što ga čini lakim za učenje i čitanje. Također ima veliku i aktivnu zajednicu korisnika i programera, koji doprinose jeziku i pružaju podršku i resurse za učenike. (Rossum, 2003)

Python je popularan izbor za mnoge vrste projekata, uključujući web razvoj, analizu podataka, strojno učenje, znanstveno računalstvo i još mnogo toga. Naširoko se koristi u industriji, akademskoj zajednici i vladinim organizacijama diljem svijeta. (Rossum, 2003)

3.2. NumPy

NumPy je Python biblioteka za rad s velikim, višedimenzionalnim nizovima i matricama numeričkih podataka. Omogućuje učinkovite operacije na nizovima i matricama, kao i funkcije za izvođenje matematičkih operacija na tim strukturama podataka. NumPy je ključna biblioteka za znanstveno računalstvo s Pythonom i naširoko se koristi u strojnom učenju i znanosti o podacima (eng. *data science)*. (Oliphant, 2006)

Neke od ključnih značajki NumPy uključuju:

- Objekt N-dimenzionalne liste
- Funkcije za izvođenje elementnih operacija na listama
- Matematičke, logičke operacije, operacije oblika i statističke operacije nad listama
- Operacije linearne algebre i generiranje slučajnih brojeva

NumPy je dizajniran da bude učinkovit i fleksibilan, a implementiran je u C i Python kako bi se maksimalno iskoristila izvedba oba jezika. NumPy liste su učinkovitije i praktičnije za korištenje od Pythonovih ugrađenih listi za mnoge vrste numeričkih računalnih zadataka. (Oliphant, 2006)

3.3. Keras

Keras je biblioteka otvorenog koda (eng. *open-source*) za neuronske mreže visoke razine napisana u Pythonu. Osmišljen je tako da bude jednostavan za korištenje i da programerima omogući brzu izgradnju i eksperimentiranje s modelima dubokog učenja. Keras je popularan izbor za izgradnju neuronskih mreža jer pruža jednostavno sučelje prilagođeno korisniku za definiranje i obuku modela dubokog učenja. (Gulli & Pal, 2017)

Keras je izgrađen na temelju drugih biblioteka dubokog učenja niže razine, kao što su TensorFlow, Theano ili Microsoft Cognitive Toolkit (kraće CNTK). Apstrahira velik dio složenosti izgradnje i treniranja neuronskih mreža, omogućujući programerima da se usredotoče na dizajn i eksperimentiranje svojih modela. (Gulli & Pal, 2017)

Neke ključne značajke Kerasa uključuju:

- API jednostavan za korištenje za definiranje i obuku modela dubokog učenja
- Podrška za više pozadina, uključujući TensorFlow, Theano i CNTK
- Širok raspon unaprijed definiranih slojeva neuronske mreže i uobičajenih funkcija gubitka
- Alati za vizualizaciju i otklanjanje pogrešaka (eng. *debugging*) modela dubokog učenja
- Mogućnost jednostavnog spremanja i učitavanja obučenih modela

Keras se široko koristi u istraživanju i industriji za izgradnju modela dubokog učenja, a popularan je izbor za izgradnju i eksperimentiranje s konvolucijskim neuronskim mrežama (CNN) i mrežama dugog kratkoročnog pamćenja (LSTM). Također se koristi za zadatke obrade prirodnog jezika, kao što je prijevod jezika i klasifikacija teksta. (Gulli & Pal, 2017)

3.4. Tensorflow

TensorFlow je *open-source* softverska biblioteka za strojno učenje i umjetnu inteligenciju. Razvio ga je Google i koristi se za širok raspon zadataka kao što su obuka i implementacija modela strojnog učenja, izvođenje analize podataka i izvođenje eksperimenata strojnog učenja. TensorFlow je dizajniran da bude fleksibilan, učinkovit i prenosiv, a može raditi na različitim platformama uključujući središnju procesorsku jedinicu (eng. *Central processing unit* - CPU), grafičku procesorsku jedinicu (eng. *Graphics processing unit* - GPU) i jedinice za obradu tenzora (eng. *Tensor Processing Units* - TPU). (Shukla & Fricklas, 2018)

Podaci su predstavljeni kao tenzori, koji su višedimenzionalni nizovi koji se mogu obraditi nizom različitih algoritama. TensorFlow omogućuje korisnicima izradu i obuku modela strojnog učenja korištenjem različitih tehnika, uključujući duboko učenje, gradijentno spuštanje i stohastički gradijentni spust. Također nudi niz alata i biblioteka za vizualizaciju i otklanjanje pogrešaka, kao i niz unaprijed izgrađenih modela koji se mogu koristiti kao početna točka za nove projekte. (Shukla & Fricklas, 2018)

TensorFlow je dizajniran kako bi bio jednostavan za korištenje, s jednostavnim i intuitivnim aplikacijskim programskim sučeljem (eng. *Application Programming Interface* – API) koji korisnicima omogućuje izradu i obuku modela strojnog učenja s minimalnim kodom. Također pruža niz API-ja više razine koji olakšavaju izvođenje uobičajenih zadataka kao što su učitavanje podataka, modeli obuke i procjena njihove izvedbe. (Shukla & Fricklas, 2018)

Fleksibilan je, dopuštajući korisnicima da izgrade i treniraju modele koristeći širok raspon tehnika i algoritama. To uključuje podršku za duboko učenje, konvolucijske neuronske

mreže, rekurentne neuronske mreže i druge tehnike koje se obično koriste u strojnom učenju i umjetnoj inteligenciji. (Shukla & Fricklas, 2018)

Učinkovit je, s brojnim tehnikama optimizacije ugrađenim u biblioteku osiguravajući time da se modeli mogu uvježbati i pokrenuti što je brže moguće. Također može iskoristiti hardversko ubrzanje, kao što su GPU i TPU, kako bi dodatno ubrzao obuku i zaključivanje. (Shukla & Fricklas, 2018)

TensorFlow je prijenosan, s podrškom za širok raspon platformi i uređaja. Može se koristiti na stolnim i prijenosnim računalima, kao i na mobilnim uređajima poput telefona i tableta. Također se može implementirati u oblaku, što olakšava skaliranje i pokretanje modela strojnog učenja u velikom broju.

TensorFlow se široko koristi u raznim aplikacijama, uključujući prepoznavanje slika i videa, obradu prirodnog jezika i sintezu govora. Također se koristi u istraživanju i razvoju, a mnoge akademske institucije i tvrtke koriste TensorFlow za istraživanje novih ideja i tehnologija u strojnom učenju i umjetnoj inteligenciji. (Shukla & Fricklas, 2018)

3.5. Matplotlib

Matplotlib je biblioteka za vizualizaciju podataka u Pythonu koja korisnicima omogućuje stvaranje širokog raspona statičnih, animiranih i interaktivnih vizualizacija u Pythonu. To je jedna od najčešće korištenih biblioteka za vizualizaciju podataka u znanosti o podacima, a posebno je korisna za stvaranje linijskih dijagrama, raspršenih dijagrama, stupčastih dijagrama, traka pogrešaka i drugih vrsta dijagrama. (Tosi, 2009)

Matplotlib je dizajniran da bude jednostavan za korištenje, s jednostavnim i intuitivnim API-jem koji korisnicima omogućuje stvaranje vizualizacija s minimalnim kodom. Također pruža brojne mogućnosti prilagodbe, tako da korisnici mogu prilagoditi izgled svojih dijagrama prema svojim potrebama. Matplotlib se može koristiti u raznim kontekstima, uključujući za znanstvena istraživanja, analizu podataka i vizualizaciju podataka za prezentacije i izvješća. (Tosi, 2009)

3.6. Scikit-learn

Scikit-learn je biblioteka za strojno učenje za Python koja pruža niz alata za zadatke kao što su klasifikacija, regresija, grupiranje, smanjenje dimenzionalnosti i odabir modela. Izgrađen je na temelju drugih popularnih Python biblioteka kao što su NumPy i Pandas, a dizajniran je da bude jednostavan za korištenje i učinkovit, s jednostavnim i intuitivnim API-jem koji korisnicima omogućuje izradu i obuku modela strojnog učenja s minimalnim kodom.

Scikit-learn je jako korištena biblioteka u Python podatkovno znanstvenoj sferi, a osobito je popularna zbog opsežne dokumentacije i primjera, kao i velike i aktivne zajednice korisnika i programera. Prikladan je za širok raspon zadataka strojnog učenja i može se koristiti u različitim kontekstima, uključujući znanstveno istraživanje, analizu podataka i proizvodna okruženja. (Hackeling, 2017)

3.7. Jupyter notebooks

Jupyter notebooks su interaktivni dokumenti koji sadrže mješavinu koda, teksta i bogatih medijskih elemenata kao što su crteži i slike. Često se koriste za analizu podataka, znanstveno računalstvo i strojno učenje, ali se mogu koristiti za širok raspon aplikacija.

Jupyter notebooks stvaraju se i uređuju pomoću sučelja temeljenog na webu, koje korisnicima omogućuje pisanje i izvršavanje koda, kao i dodavanje teksta, jednadžbi i drugih bogatih medijskih elemenata. Bilježnice su napisane u programskim jezicima Julia, Python ili R i mogu se pokretati na lokalnom računalu ili na udaljenom poslužitelju. (Jupyter, 2023)

Jedna od ključnih značajki Jupyter notebook-ova je da omogućuju korisnicima pokretanje koda u blokovima ili "ćelijama" i pregled rezultata koda u istom dokumentu. To olakšava testiranje koda i otklanjanje pogrešaka, kao i dokumentiranje i dijeljenje rezultata koda s drugima.

Jupyter notebooks naširoko se koriste u znanosti o podacima i znanstvenom računalstvu jer pružaju prikladan i interaktivan način rada s podacima i izvođenja složenih analiza. Također su popularne u nastavnim i obrazovnim svrhama jer korisnicima omogućuju miješanje koda, teksta i elemenata obogaćenog (eng. *rich*) medija u jednom dokumentu. (Driscoll, 2019)

4. Strojno učenje

Strojno učenje je područje umjetne inteligencije koje uključuje korištenje algoritama i statističkih modela kako bi se računalima omogućilo učenje i donošenje predviđanja ili odluka bez da su eksplicitno programirana za to (Mitchell, 1997). Ova vrsta učenja postiže se korištenjem velikih količina podataka i primjenom računalnih tehnika za automatsko identificiranje uzoraka i odnosa u podacima (Cristianini, 2004).

Kod strojnog učenja postoji nekoliko vrsta, uključujući nadzirno i nenadzirano učenje, te podražajno učenje (I. Goodfellow, 2016). U nadziranom učenju, algoritam se trenira na označenom skupu podataka, koji uključuje i ulazne podatke i odgovarajući točni izlaz. Cilj procesa učenja je napraviti predviđanja o novim, neviđenim podacima na temelju obrazaca i odnosa naučenih iz podataka o obuci. Neki od primjera zadataka koji se mogu rješavati korištenjem nadzirnog učenja, jesu otkrivanje neženjene pošte i klasifikacija slika.

Učenje bez nadzora uključuje korištenje algoritama za otkrivanje uzoraka u skupu podataka bez odgovarajućih izlaznih podataka (Motoda, 1998). Ova vrsta učenja često se koristi za eksplorativnu analizu podataka i može se koristiti za identificiranje klastera ili grupa unutar podataka. Primjeri zadataka koji se mogu rješavati korištenjem učenja bez nadzora uključuju otkrivanje anomalija i procjenu preciznosti.

Podržano učenje uključuje korištenje algoritama za učenje putem pokušaja i pogrešaka poduzimanjem radnji u okruženju i primanjem nagrada ili kazni na temelju rezultata (Barto, 2018). Ova vrsta učenja često se koristi u robotici i sustavima upravljanja, gdje je cilj maksimizirati signal nagrade odabirom radnji koje dovode do željenog ishoda.

Jedna od glavnih prednosti strojnog učenja je njegova sposobnost da automatski uči i poboljšava podatke, bez potrebe za eksplicitnim programiranjem (Murphy, 2012). To ga je učinilo vrijednim alatom za širok raspon primjena, uključujući obradu prirodnog jezika, prepoznavanje slike i govora te prediktivno modeliranje.

Mnogi su izazovi uključeni u primjenu strojnog učenja, uključujući potrebu za velikim količinama označenih podataka, rizik od pretjeranog prilagođavanja podacima o obuci i poteškoće u objašnjavanju odluka koje donose modeli strojnog učenja (Breiman, 2001). Unatoč tim izazovima, strojno učenje postalo je ključni alat u mnogim područjima i vjerojatno će nastaviti igrati središnju ulogu u razvoju umjetne inteligencije i aplikacija koje se temelje na podacima.

4.1. Duboko učenje

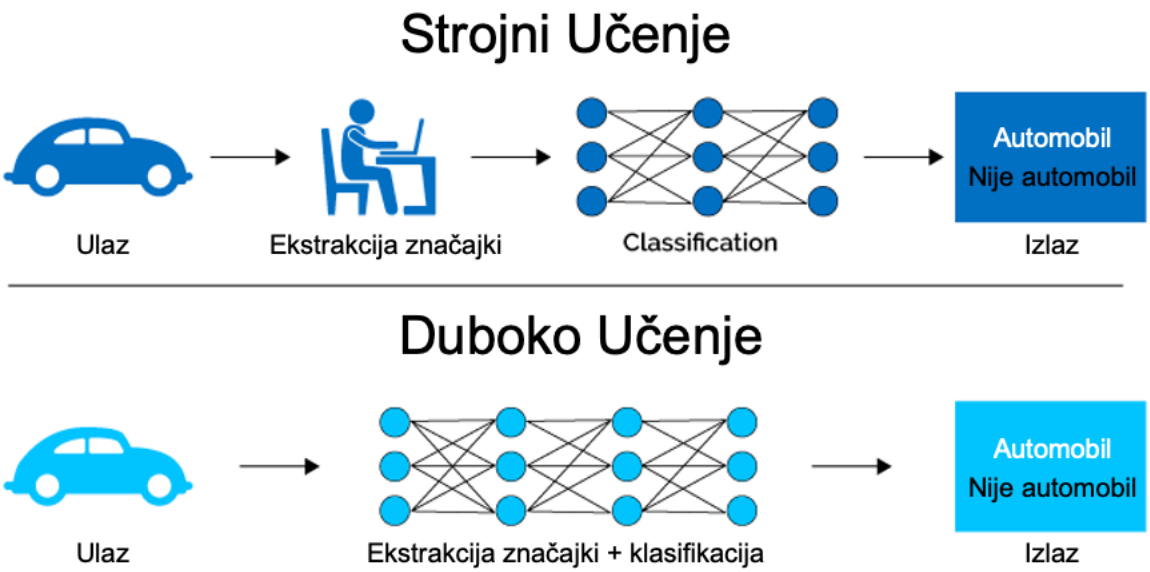
Duboko učenje je potpolje strojnog učenja koje je inspirirano strukturom i funkcijom mozga, posebno neuronskim mrežama koje čine mozak (Y. LeCun Y. B., 2015). Uključuje korištenje umjetnih neuronskih mreža, algoritama koji su dizajnirani za prepoznavanje uzoraka i odnosa u podacima (I. Goodfellow, 2016).

U umjetnoj neuronskoj mreži postoje jedinice zvane "neuroni" i organizirane su u slojeve. Svaki neuron prima ulaz od drugih neurona u prethodnom sloju, obrađuje ulaze podatke koristeći skup težina i pristranosti i proizvodi izlazne podatke. Težine i pristranosti su parametri koje neuronska mreža uči tijekom procesa obuke (Y. LeCun Y. B., 2015).

Izlaz neurona izračunava se pomoću aktivacijske funkcije, koja određuje izlaz neurona na temelju ulaza koji prima (Y. LeCun Y. B., 2015). Postoji nekoliko različitih tipova aktivacijskih funkcija koje se mogu koristiti, uključujući sigmoid (Zisserman, 2014), tanh (Ba, 2014) i ReLU (Rectified Linear Unit) (K. He, 2015). Izbor aktivacijske funkcije može imati značajan utjecaj na rad neuronske mreže (Schmidhuber, 2015).

Duboko učenje odnosi se na korištenje dubokih neuronskih mreža, a to su neuronske mreže s mnogo slojeva (Haykin, 1999). Ovi slojevi omogućuju mreži da nauči složene obrasce i odnose u podacima, što može biti teško za plitke mreže s manje slojeva (Hinton, 2014).

Ukratko, duboko učenje uključuje korištenje umjetnih neuronskih mreža za prepoznavanje uzoraka i odnosa u podacima. Neuroni u neuronskoj mreži primaju ulazne podatke, obrađuju ga pomoću težina i pristranosti te proizvode izlazne podatke koristeći aktivacijsku funkciju. Duboko učenje odnosi se na korištenje dubokih neuronskih mreža s mnogo slojeva, koje su sposobne naučiti složene obrasce u podacima. Na slici 3 prikazana je razlika procesuiranju strojnog i dubokog učenja.

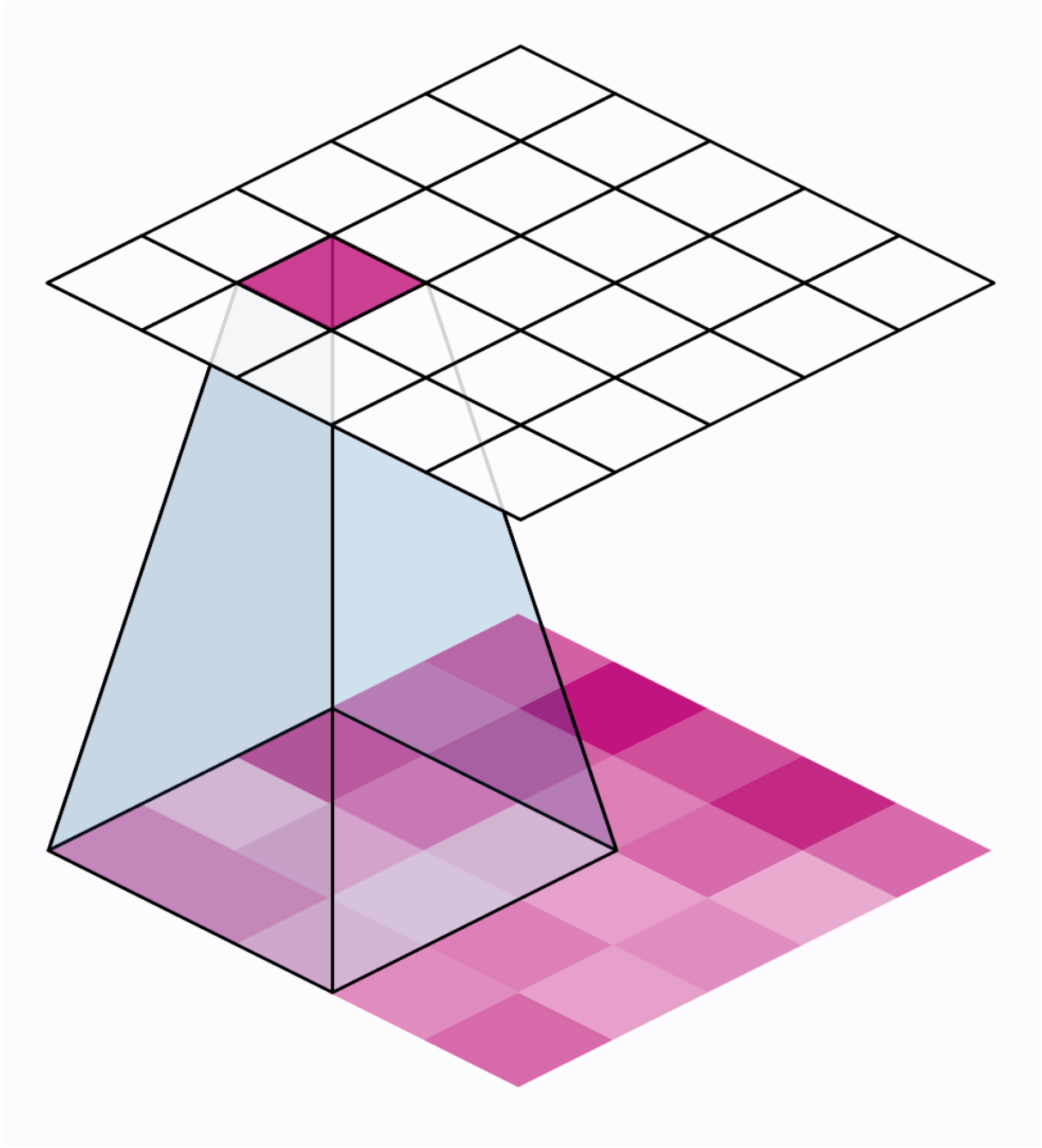


Slika 3: Razlika između strojnog i dubokog učenja (Vlastita izrada prema: https://i0.wp.com/semiengineering.com/wp-content/uploads/2018/01/MLvsDL.png?ssl=1)

4.2. Konvolucijske mreže

Konvolucijske neuronske mreže (kraće CNN), vrsta su umjetne neuronske mreže koja je posebno prikladna za zadatke prepoznavanja i obrade slike (Fukushima, 1980). Inspirirani su načinom na koji vizualni sustav obrađuje informacije u mozgu i pokazali su se iznimno učinkovitima u širokom rasponu primjena, uključujući klasifikaciju slika (A. Krizhevsky, 2012), detekciju objekata (J. Redmon, 2016) i generiranje slika (Osindero).

Jedan ključni aspekt CNN-a je njihova upotreba konvolucijskih slojeva, koji su dizajnirani da automatski uče prostorne hijerarhije značajki iz ulaznih podataka (Y. LeCun B. B.). Ovi slojevi primjenjuju skup filtara koji se mogu naučiti na ulazne podatke, koji se koriste za izdvajanje značajki više razine iz neobrađenih podataka. Filtri se obično primjenjuju u kliznom prozoru, pomičući se preko ulaznih podataka i proizvodeći skup mapa značajki kao izlaz. Filtri se uče kroz proces obuke, omogućujući CNN-u da automatski nauči značajke koje su najrelevantnije za zadatak (L. Fei-Fei, 2004).



Slika 4: Klizni prozor CNN-a (Izvor: https://i.imgur.com/LueNK6b.gif)

Još jedan važan aspekt CNN-a je korištenje udruživanja (eng. *pooling*) slojeva, koji se koriste za smanjenje prostornih dimenzija ulaznih podataka. Ti se slojevi obično koriste između konvolucijskih slojeva i služe za smanjenje veličine ulaznih podataka, čineći mrežu računalno učinkovitijom i također pomažući u smanjenju prekomjernog prilagođavanja. Postoji nekoliko vrsta *pooling* slojeva, uključujući maksimalni *pooling* i prosječni *pooling*, koji pojedinačno uzimaju maksimalnu ili prosječnu vrijednost ulaznih podataka unutar određene regije (Osindero). Slika 4 prikazuje udruživanje slojeva, odnosno smanjivanje prostornih dimenzija ulaznih podataka.

CNN-ovi također često koriste potpuno povezane slojeve, koji su slični onima koji se nalaze u tradicionalnim neuronskim mrežama i koriste se za izradu konačnih predviđanja na temelju naučenih značajki. Ovi slojevi obično slijede konvolucijske i skupne slojeve i koriste se za izradu konačnih predviđanja na temelju naučenih značajki.

Jedna od glavnih prednosti CNN-a je njihova sposobnost da automatski uče i izvlače značajke iz ulaznih podataka, bez potrebe za ručnim inženjeringom značajki. To im omogućuje da se koriste za širok raspon zadataka, uključujući klasifikaciju slika (A. Krizhevsky, 2012), detekciju objekata (I. Goodfellow, 2016) i generiranje slika (Osindero), među ostalima.

Posljednjih godina CNN-ovi su postigli vrhunske rezultate na mnogim zadacima i postali su glavni izbor za mnoge istraživače i praktičare koji rade na zadacima obrade slike. Neki značajni primjeri uključuju korištenje CNN-ova za klasifikaciju slika na skupu podataka ImageNet, koji je postigao najbolje rezultate u godišnjem natjecanju ImageNet Large Scale Visual Recognition Challenge (kraće ILSVRC) (J. Deng, 2009), i korištenje CNN-ova za otkrivanje objekata u skupu podataka Microsoft COCO (T.-Y. Lin, 2014).

Postoje mnoge podteme unutar područja CNN-a, uključujući korištenje različitih arhitektura i tehnika obuke, razvoj novih vrsta slojeva i modula i primjenu CNN-a na širok raspon zadataka i domene (Bengio, 2015).

Jedno područje aktivnog istraživanja je razvoj novih CNN arhitektura, kao što su Residual Networks (kraće ResNets) i Inception Networks, koje su postigle najsuvremenije rezultate na različitim zadacima. Drugo važno područje je razvoj tehnika za poboljšanje učinkovitosti i djelotvornosti CNN-a, kao što je prijenos učenja i uljepšavanje (eng. *fine-tuning*) prethodno obučenih modela (Osindero).

Također se obavlja značajni rad na primjeni CNN-a na širok raspon zadataka i domena, uključujući analizu medicinskih nalaza, obradu prirodnog jezika, pa čak i stvaranje glazbe. Svestranost i učinkovitost CNN-a učinila ih je vrijednim alatom za istraživače i znanstvenike koji rade na širokom spektru problema.

Jedan od glavnih izazova u korištenju CNN-a je potreba za velikom količinom označenih podataka o obuci. To može biti značajna prepreka njihovom usvajanju u određenim okruženjima, kao što je kada zadatak uključuje rijetke ili specijalizirane parametre ili kada je podatke teško prikupiti ili označiti. Kako bi odgovorili na ovaj izazov, istraživači su razvili tehnike kao što je aktivno učenje, koje omogućuje CNN-u da selektivno odabere koje će podatke sljedeće označiti, i prijenos učenja, koji omogućuje CNN-u da iskoristi znanje iz povezanih zadataka ili domena na poboljšati performanse.

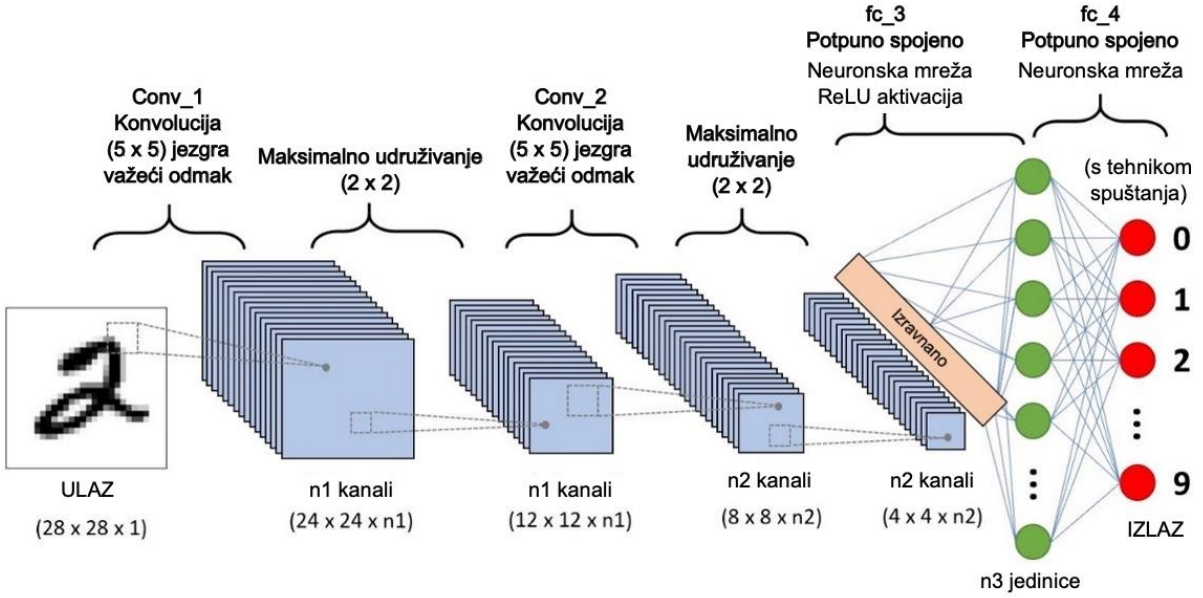
Još jedno važno područje istraživanja u CNN-ovima je razvoj novih vrsta slojeva i modula, kao što su mehanizmi za samopažnju (eng. *self-attention*) (K. Cho, 2014), koji

omogućuju CNN-u da se selektivno fokusira na određene regije ili značajke ulaznih podataka. Također je bilo značajnog rada na razvoju novih tipova slojeva udruživanja, kao što je *pooling self-attention* mehanizama (H. Cheng, 2019), što omogućuje CNN-u da uhvati velike nepravilnosti u ulaznim podacima.

Neosporivo je da su konvolucijske neuronske mreže (CNN) potpuno revolucionirale polje strojnog učenja. Sa svojom sposobnošću automatskog učenja hijerarhijskih prikaza podataka, oni su dosljedno nadmašivali druge modele u zadacima kao što su klasifikacija slika i detekcija objekata. Zapravo, nije pretjerano reći da su CNN-ovi u potpunosti nadmašili sve druge pristupe u ovim područjima, što ih čini glavnim izborom za istraživače i praktičare.

No prednosti CNN-a ne prestaju. Ne samo da su vrlo učinkovite u izvlačenju značajki iz mrežnih podataka, već su i iznimno učinkovite, zahtijevaju manje parametara i manje računanja od drugih vrsta neuronskih mreža. Zbog toga su posebno prikladni za korištenje u okruženjima s ograničenim resursima, kao što su mobilni uređaji ili računalnim aplikacijama.

Sa svim ovim prednostima, nije ni čudo da su CNN-ovi postali dominantna sila u polju strojnog učenja. Konvolucijske neuronske mreže su jedini smisleni izbor za bilo koji zadatak koji uključuje mrežaste podatke i sigurno je reći da će nastaviti dominirati tim područjem u budućnosti. Na slici 5 nalazi se vizualizacija arhitekture konvolicijskih mreža.



Slika 5: Arhitektura konvolucijskih mreža (Vlastita izrada prema: https://miro.medium.com/max/1400/1*uAeANQIOQPqWZnnuH-VEyw.jpeg)

4.2.1. Konvolucijski sloj

Konvolucijski slojevi ključna su komponenta konvolucijskih neuronskih mreža (CNN), koje su vrsta umjetne neuronske mreže posebno dizajnirane za obradu podataka s mrežastim podacima kao što je slika (Y. LeCun L. B.). Konvolucijski slojevi rade konvolucije na ulaznim podacima, što uključuje primjenu skupa filtara koji se mogu naučiti na ulaz kako bi se izdvojile značajke i uzorci (Singh, 2017). Na slici 6 prikazana je primjena filtera na slici automobila.

Filtri koji se koriste u konvolucijskim slojevima obično su male veličine i klize preko ulaznih podataka, izvodeći množenja po elementima i zbrajajući rezultate kako bi se proizveo novi skup mapa značajki (X. Zhang, 1998). Filtri se obično dijele na svim mjestima u ulaznim podacima, što omogućuje CNN-u da nauči značajke nepromjenjive u prijevodu (Singh, 2017).

Veličina filtara i korak, ili broj piksela koje filtar pomiče u svakom koraku, mogu se prilagoditi za kontrolu veličine mapa značajki koje proizvodi konvolucijski sloj (Singh, 2017). Veći filtri i manji koraci rezultiraju većim kartama značajki, dok manji filtri i veći koraci rezultiraju manjim mapama značajki.

Slika 6: Primjer konvolucije ulazne slike (Vlastita izrada prema: https://i.imgur.com/IYO9lqp.png)

Konvolucijski slojevi također mogu koristiti odmak (eng. *padding*), što uključuje dodavanje dodatnih redaka i stupaca piksela oko ulaznih podataka kako bi se očuvala veličina mapa značajki (Singh, 2017). To može biti korisno za održavanje prostornih informacija u kartama značajki, posebno kada se koriste manji filtri ili veći koraci.

4.2.2. Gusti sloj

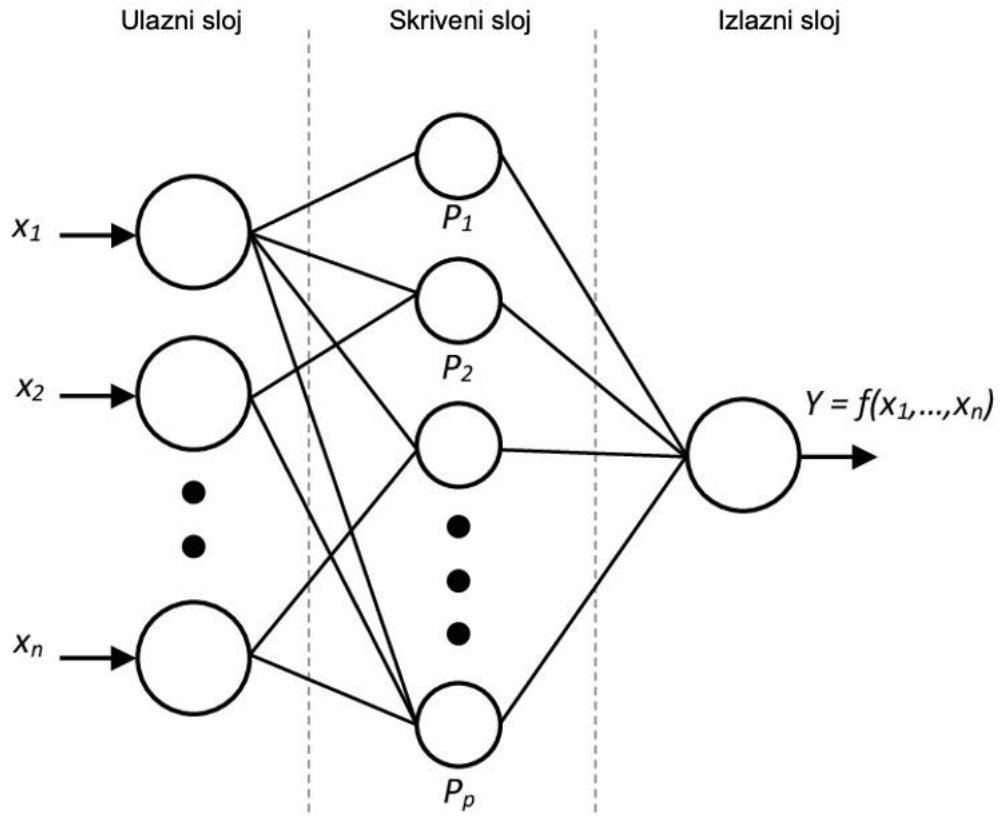
Gusti slojevi, poznati i kao potpuno povezani slojevi, vrsta su slojeva koji se obično koriste u konvolucijskim neuronskim mrežama (CNN). Nazivaju se gustim slojevima jer je svaki neuron u sloju povezan sa svakim neuronom u prethodnom sloju (Singh, 2017).

Gusti slojevi obično se koriste pred kraju CNN-a, nakon što su konvolucijski i skupni slojevi izvukli značajke iz ulaznih podataka (Singh, 2017). Izlaz gustog sloja koristi se za izradu predviđanja, kao što je klasificiranje slike u jednu od nekoliko unaprijed definiranih klasa (Singh, 2017).

Težine i pristranosti gustog sloja modificiraju se tijekom procesa obuke, korištenjem optimizacijskog algoritma kao što je stohastičko gradijentno spuštanje (Y. LeCun Y. B., 2015). Izlaz gustog sloja izračunava se korištenjem aktivacijske funkcije, koja određuje izlaz neurona na temelju ulaza koji prima (Y. LeCun Y. B., 2015). Postoji nekoliko različitih vrsta aktivacijskih funkcija koje se mogu koristiti, uključujući sigmoidnu, tanh i ReLU (Rectified Linear Unit) (Y. LeCun Y. B., 2015).

Gusti slojevi također mogu uključivati tehnike regulacije, kao što je ispadanje, koje nasumično postavlja dio neurona na nulu tijekom treninga kako bi se spriječilo prekomjerno prilagođavanje (Singh, 2017). To može pomoći u poboljšanju izvedbe generalizacije CNN-a na nevidljivim podacima.

Na slici 7 prikazana je arhitektura višeslojne neuronske mreže s jednim skrivenim slojem. Ulazni sloj sadrži broj neurona jednak broju ulaznih parametara. Ponderi se primjenjuju na ulaze jer se oni prosljeđuju skrivenim neuronima. Na skrivenom sloju ponderirane vrijednosti unose se u prijenosnu funkciju koja se zatim distribuira u izlazni sloj. Opet se ponderi primjenjuju na vrijednosti poslane u izlazni sloj. Ponderirane vrijednosti iz skrivenog sloja dolaze do izlaznog neurona i kombiniraju se prije ulaska u drugu prijenosnu funkciju. Izlaz prijenosne funkcije je izlaz mreže (Brooks & Tucker, 2014).



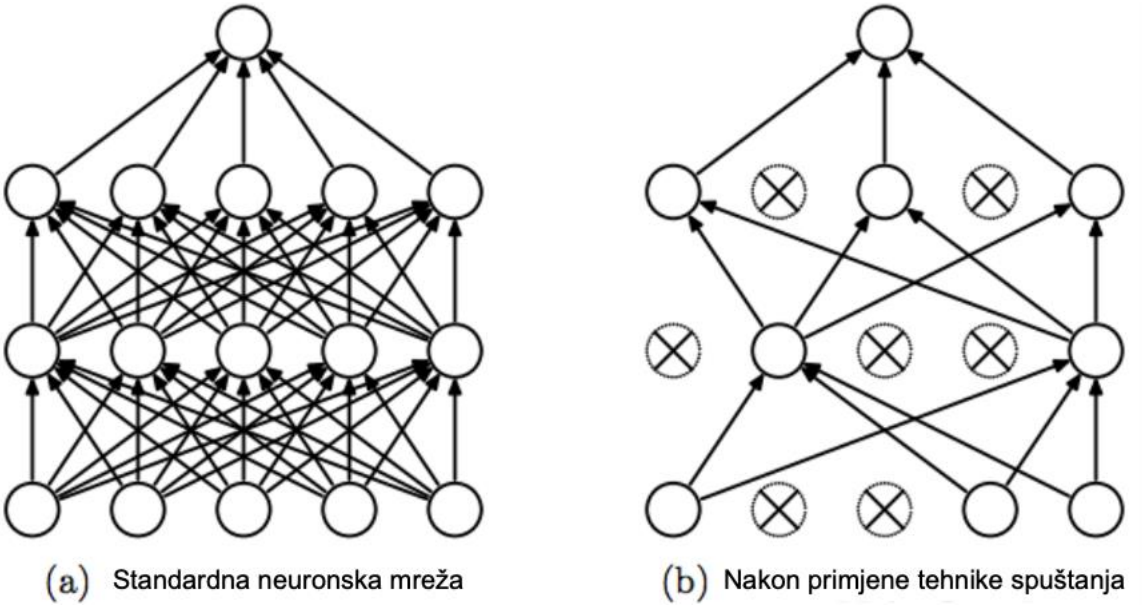
Slika 7: Arhitektura gustog sloja (Vlastita izrada prema: https://www.researchgate.net/profile/Hadley-Brooks/publication/270274130/figure/fig3/AS:667886670594050@1536247999230/Architecture-of-a-multilayer-neural-network-with-one-hidden-layer-The-input-layer.png)

4.2.3. Dropout sloj

Tehnika napuštanja (eng. *dropout*) je tehnika regularizacije koja se obično koristi u konvolucijskim neuronskim mrežama (CNN) kako bi se spriječilo prekomjerno prilagođavanje i poboljšala izvedba generalizacije modela na nevidljivim podacima (Szegedy, 2015). Djeluje nasumično postavljanjem dijela neurona u mreži na nulu tijekom treninga (N. Srivastava, 2014).

Dropout se može implementirati kao zaseban sloj u CNN-u, poznat kao sloj ispadanja. Stopa ispadanja, ili udio neurona koji su postavljeni na nulu, obično je postavljen na vrijednost između 0,2 i 0,5 (Singh, 2017). Tijekom treninga, sloj ispadanja nasumično postavlja dio neurona na nulu prema navedenoj stopi ispadanja. Tijekom zaključivanja, ispadajući sloj obično se isključuje i koriste se svi neuroni. Slika 8 prikazuje isključivanje slučajnih neurona u fazi treninga.

Dropout se pokazao učinkovitim u smanjenju prekomjernog prilagođavanja dubokih neuronskih mreža (N. Srivastava, 2014). Djeluje tako da sprječava da se mreža previše oslanja na bilo koji određeni neuron ili grupu neurona, što može pomoći u poboljšanju izvedbe generalizacije modela.



Slika 8: Neuronska mreža s dropout slojem (Vlastita izrada prema: https://miro.medium.com/max/1044/1*iWQzxhVlvadk6VAJjsgXgg.png)

4.2.4. Funkcija optimizacije

Funkcija optimizacije je važna komponenta konvolucijskih neuronskih mreža (CNN) i igra ključnu ulogu u procesu obuke. Odgovorna je za ažuriranje težina i pristranosti CNN-a na temelju pogreške između predviđenog izlaza i stvarnog izlaza (Y. LeCun Y. B., 2015).

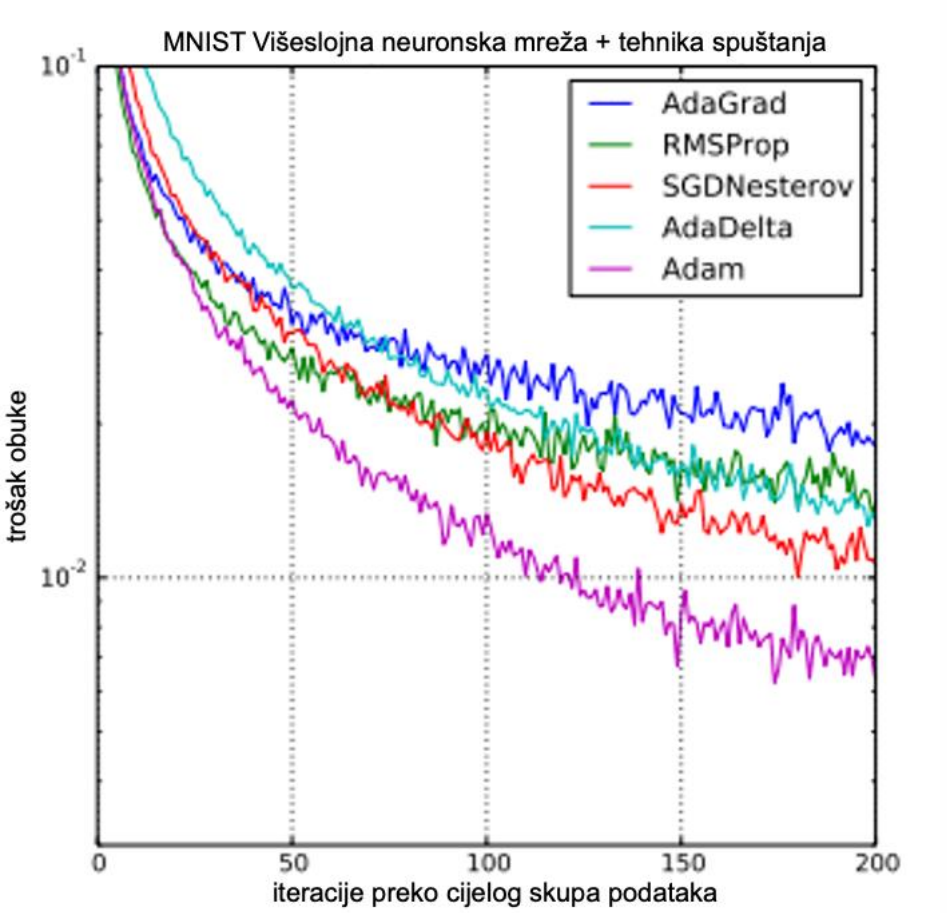
Postoji nekoliko različitih tipova funkcija optimizacije koje se mogu koristiti u CNN-ovima, uključujući stohastičko gradijentno spuštanje (SGD), Adam i RMSprop (Singh, 2017). Svaki od ovih optimizatora ima vlastiti skup hiperparametara koji se mogu prilagoditi za kontrolu brzine učenja i konvergencije CNN-a (Singh, 2017).

Stopa učenja je hiperparametar koji određuje veličinu koraka pri kojem funkcija optimizacije ažurira težine i pristranosti CNN-a (Y. LeCun Y. B., 2015). Veća stopa učenja može dovesti do brže konvergencije, ali također može učiniti CNN sklonijim oscilacijama i može rezultirati neoptimalnim rješenjem. Manja stopa učenja može pomoći u poboljšanju stabilnosti CNN-a, ali također može rezultirati sporijom konvergencijom.

Konvergencija CNN-a odnosi se na točku u kojoj je pogreška između predviđenog izlaza i stvarnog izlaza minimizirana (Y. LeCun Y. B., 2015). Funkcija optimizacije igra ključnu ulogu u postizanju konvergencije ažuriranjem težina i pristranosti CNN-a na način koji smanjuje pogrešku.

Prema (Bock, Weiß, & Goppold, 2018) ADAM-Optimizer je jedan od najpopularnijih algoritama za optimizaciju gradijentnog spuštanja. Implementiran je u uobičajenim okvirima neuronske mreže, poput TensorFlow-a. Eksperimentalno pokazuju da je ADAM-Optimizer brži od bilo koje druge funkcije optimizacije (Slika 9).

Ukratko, funkcija optimizacije je važna komponenta CNN-a i odgovorna je za ažuriranje težina i pristranosti CNN-a na temelju pogreške između predviđenog izlaza i stvarnog izlaza. Postoji nekoliko različitih tipova funkcija optimizacije koje se mogu koristiti, a stopa učenja i drugi hiperparametri mogu se prilagoditi za kontrolu konvergencije CNN-a.



Slika 9: Odnos različitih funkcija optimizacije (Izvor: https://www.researchgate.net/publication/324808725_An_improvement_of_the_convergence_proof_of_the_ADAM-Optimizer)

4.2.5. Funkcija gubitka

Funkcija gubitka važna je komponenta konvolucijskih neuronskih mreža (CNN) i igra ključnu ulogu u procesu obuke. To je mjera pogreške između predviđenog izlaza CNN-a i stvarnog izlaza, a koristi se za određivanje koliko dobro CNN obavlja zadaću (Y. LeCun Y. B., 2015).

Postoji nekoliko različitih vrsta funkcija gubitka koje se mogu koristiti u CNN-ovima, uključujući srednju kvadratnu pogrešku (MSE), unakrsni entropijski gubitak i zglobni gubitak (eng. *joint loss*) (Singh, 2017). Izbor funkcije gubitka ovisi o vrsti problema koji se rješava i prirodi izlaznih podataka.

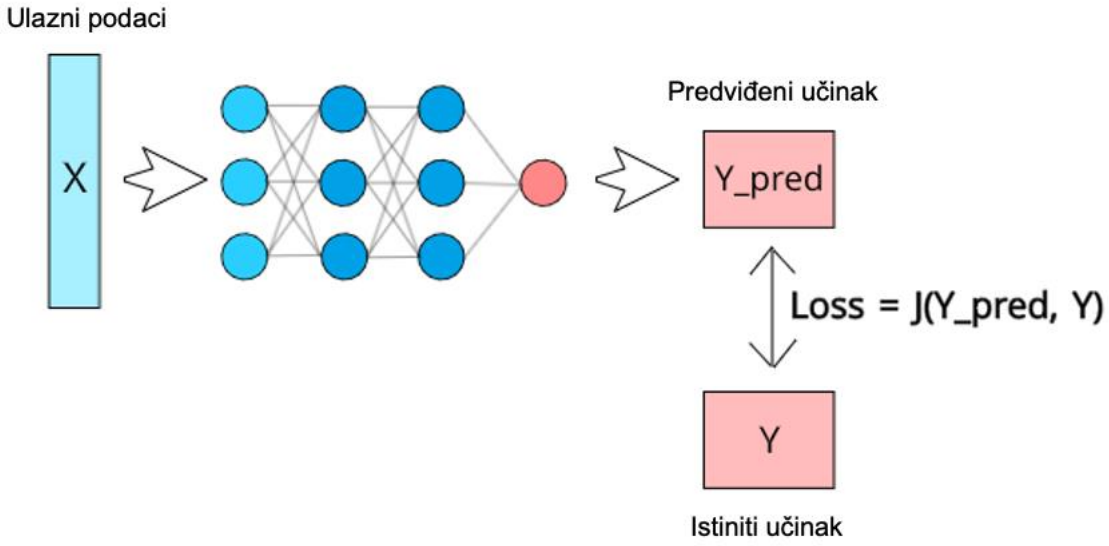
MSE funkcija gubitka obično se koristi za probleme regresije, gdje su izlazni podaci kontinuirani (Singh, 2017). Izračunava prosječnu kvadratnu razliku između predviđenog izlaza

i stvarnog izlaza. Funkcija gubitka MSE osjetljiva je na ekstremne vrijednosti i na nju može utjecati veličina podataka.

Funkcija gubitka entropije obično se koristi za probleme klasifikacije, gdje su izlazni podaci diskretni (Y. LeCun Y. B., 2015). Izračunava razliku između predviđenog izlaza i stvarnog izlaza na način koji je robusniji prema ekstremima i na nju ne utječe skala podataka.

Zglobna funkcija gubitka često se koristi za strojeve za potporu vektora (SVM) i koristi se za osposobljavanje CNN-a za izvođenje binarne klasifikacije (Y. LeCun Y. B., 2015). Izračunava razliku između predviđenog izlaza i stvarnog izlaza na način koji je sličan unakrsnom entropijskom gubitku, ali je otpornija na ekstremne vrijednosti i na nju ne utječe veličina podataka.

Slika 10 opisuje funkciju gubitka (J) kao funkcija koja uzima dva parametra, predviđeni učinak i istinski učinak. Ova funkcija će u biti izračunati koliko loše radi model uspoređujući ono što model predviđa sa stvarnom vrijednošću koju bi trebao ispisati. Ako je `Y_pred` jako daleko od `Y`, vrijednost gubitka bit će vrlo visoka. Međutim, ako su obje vrijednosti gotovo slične, vrijednost gubitka bit će vrlo niska (slika 10).



Slika 10: Formula izračuna funkcije gubitka (Izvor: https://medium.com/deep-learning-demystified/loss-functions-explained-3098e8ff2b27)

4.2.6. Aktivacijska funkcija

Aktivacijske funkcije važna su komponenta konvolucijskih neuronskih mreža (CNN), budući da su odgovorne za određivanje izlaza svakog neuron a u mreži. U ovom odlomku biti će istražena uloga aktivacijskih funkcija u CNN-ovima.

Jedna od ključnih uloga aktivacijskih funkcija u CNN-ovima je uvođenje nelinearnosti u mrežu. Ovo je važno jer većina podataka iz stvarnog svijeta pokazuje nelinearne odnose, a CNN sa samo linearnim aktivacijskim funkcijama ne bi mogao točno modelirati te odnose. Aktivacijske funkcije kao što su ispravljena linearna jedinica (ReLU) i sigmoidna funkcija obično se koriste u CNN-ovima za uvođenje nelinearnosti. (Feng & Lu, 2019)

U nedavnom radu objavljenom u IEEE Transactions on Neural Networks and Learning Systems (Klambauer, 2017) uspoređuju izvedbu nekoliko popularnih aktivacijskih funkcija u CNN-ovima, uključujući ReLU, sigmoidnu i hiperboličku tangentnu (Tanh) funkciju. Otkrili su da ReLU funkcija dobro funkcionira u smislu brzine obuke i točnosti klasifikacije, što je čini popularnim izborom za korištenje u CNN-ovima.

Drugi važan aspekt aktivacijskih funkcija u CNN-ovima je mogućnost kontrole raspona izlaznih vrijednosti. Aktivacijske funkcije kao što su sigmoidna i Tanh funkcija imaju ograničeni raspon izlaznih vrijednosti, što može biti korisno za zadatke kao što je binarna klasifikacija. Međutim, za zadatke kao što je klasifikacija slika, aktivacijske funkcije s neograničenim rasponima, kao što je funkcija ReLU, mogu biti prikladnije jer mogu bolje uhvatiti širok raspon mogućih vrijednosti piksela na slici (Nair, 2010).

Aktivacijske funkcije igraju ključnu ulogu u izvedbi CNN-a uvođenjem nelinearnosti i kontroliranjem raspona izlaznih vrijednosti. Nedavna istraživanja pokazala su da je ReLU funkcija popularan izbor zbog svojih dobrih performansi u smislu brzine treninga i točnosti klasifikacije.

Prema Feng-u i Shengnan-u u radu Analiza izvedbe različitih aktivacijskih funkcija u umjetnim neuronskim mrežama (Feng & Lu, 2019) neke od bitnih nelinearnih aktivacijskih funkcija koje olakšavaju modelu generalizaciju ili prilagodbu različitim podacima i razlikovanje izlaza jesu:

- Sigmoid funkcija

$$f(x_i) = \frac{1}{1 + e^{-x_i}}$$

- Tanh funkcija

$$\tanh(x_i) = \frac{2}{1 + e^{-2x_i}} = 2\ sigmoid(2x_i) - 1$$

- ReLU funkcija

$$f(x_i) = \max(0, x_i) = \begin{cases} x_i, x_i > 0 \\ x_i, x_i < 0 \end{cases}$$

- Leaky ReLU funkcija

$$f(x_i) = \begin{cases} x_i, x_i > 0 \\ \alpha_i x_i, x_i < 0 \end{cases}$$

- ELU funkcija

$$f(x_i) = \begin{cases} x_i, & x_i > 0 \\ \alpha_i(e^{x_i} - 1), & x_i \leq 0 \end{cases}$$

5. Treniranje modela za klasifikaciju spola, dobi i etničke pripadnosti

Prvi korak u obučavanju modela strojnog učenja za određivanje spola, dobi i etničke pripadnosti osobe na temelju slike lica jest prikupljanje skupa podataka slika. Ovaj skup podataka trebao bi uključivati raznolik raspon subjekata u smislu dobi, etničke pripadnosti i spola, kako bi se osiguralo da model može točno identificirati pojedince u različitim situacijama.

Nakon što se skup podataka prikupi, mora se označiti kako bi se modelu strojnog učenja pružili potrebni podaci za obuku. Ovaj proces obično uključuje ručno identificiranje svake osobe u skupu podataka i dodjeljivanje oznake svakoj slici u skladu s tim.

Nakon što je skup podataka označen, može se koristiti za obuku modela strojnog učenja. Ovaj proces uključuje unos skupa podataka u model i podešavanje parametara modela kako bi se optimizirala njegova izvedba. To može uključivati korištenje tehnika kao što je unakrsna provjera valjanosti kako bi se osiguralo da model može točno predvidjeti sve tri karakteristike pojedinaca koje prije nije vidio.

Potrebno je odabrati prethodno trenirani model, tako da bi povećali preciznost i smanjili količinu obuke. Odabrati ćemo model iz Keras model zoo-a.

Tablica 1: Keras model zoo tablica

Model	Veličina (MB)	Top-1 Preciznost	Top-5 Preciznost	Parametri	Dubina	Vrijeme (ms) po koraku (CPU)	Vrijeme (ms) po koraku (GPU)
Xception	88	79.0%	94.5%	22.9M	81	109.4	8.1
VGG16	528	71.3%	90.1%	138.4M	16	69.5	4.2
VGG19	549	71.3%	90.0%	143.7M	19	84.8	4.4
ResNet50	98	74.9%	92.1%	25.6M	107	58.2	4.6
ResNet50V2	98	76.0%	93.0%	25.6M	103	45.6	4.4
ResNet101	171	76.4%	92.8%	44.7M	209	89.6	5.2
ResNet101V2	171	77.2%	93.8%	44.7M	205	72.7	5.4
ResNet152	232	76.6%	93.1%	60.4M	311	127.4	6.5
ResNet152V2	232	78.0%	94.2%	60.4M	307	107.5	6.6
InceptionV3	92	77.9%	93.7%	23.9M	189	42.2	6.9
InceptionResNetV2	215	80.3%	95.3%	55.9M	449	130.2	10.0
MobileNet	16	70.4%	89.5%	4.3M	55	22.6	3.4
MobileNetV2	14	71.3%	90.1%	3.5M	105	25.9	3.8
DenseNet121	33	75.0%	92.3%	8.1M	242	77.1	5.4
DenseNet169	57	76.2%	93.2%	14.3M	338	96.4	6.3
DenseNet201	80	77.3%	93.6%	20.2M	402	127.2	6.7
NASNetMobile	23	74.4%	91.9%	5.3M	389	27.0	6.7
NASNetLarge	343	82.5%	96.0%	88.9M	533	344.5	20.0
EfficientNetB0	29	77.1%	93.3%	5.3M	132	46.0	4.9
EfficientNetB1	31	79.1%	94.4%	7.9M	186	60.2	5.6
EfficientNetB2	36	80.1%	94.9%	9.2M	186	80.8	6.5

Model	Veličina (MB)	Top-1 Preciznost	Top-5 Preciznost	Parametri	Dubina	Vrijeme (ms) po koraku (CPU)	Vrijeme (ms) po koraku (GPU)
EfficientNetB3	48	81.6%	95.7%	12.3M	210	140.0	8.8
EfficientNetB4	75	82.9%	96.4%	19.5M	258	308.3	15.1
EfficientNetB5	118	83.6%	96.7%	30.6M	312	579.2	25.3
EfficientNetB6	166	84.0%	96.8%	43.3M	360	958.1	40.4
EfficientNetB7	256	84.3%	97.0%	66.7M	438	1578.9	61.6
EfficientNetV2B0	29	78.7%	94.3%	7.2M	-	-	-
EfficientNetV2B1	34	79.8%	95.0%	8.2M	-	-	-
EfficientNetV2B2	42	80.5%	95.1%	10.2M	-	-	-
EfficientNetV2B3	59	82.0%	95.8%	14.5M	-	-	-
EfficientNetV2S	88	83.9%	96.7%	21.6M	-	-	-
EfficientNetV2M	220	85.3%	97.4%	54.4M	-	-	-
EfficientNetV2L	479	85.7%	97.5%	119.0M	-	-	-
ConvNeXtTiny	109.42	81.3%	-	28.6M	-	-	-
ConvNeXtSmall	192.29	82.3%	-	50.2M	-	-	-
ConvNeXtBase	338.58	85.3%	-	88.5M	-	-	-
ConvNeXtLarge	755.07	86.3%	-	197.7M	-	-	-
ConvNeXtXLarge	1310	86.7%	-	350.1M	-	-	-

(Izvor: https://keras.io/api/applications/)

Odabrati ćemo MobileNet model zbog svoje male veličine i velike brzine. Pošto ne koristimo uređaj s GPU-om, korištenje drugih modela nije izvedivo ili nije isplativo.

Konačno, nakon što je model obučen, može se testirati na zasebnom skupu podataka kako bi se procijenila njegova izvedba. Ovaj proces uključuje korištenje modela za izradu predviđanja na skupu podataka slika na kojima nije prošao obuku i usporedbu predviđanja modela s poznatim oznakama za te slike.

5.1. MobileNet model

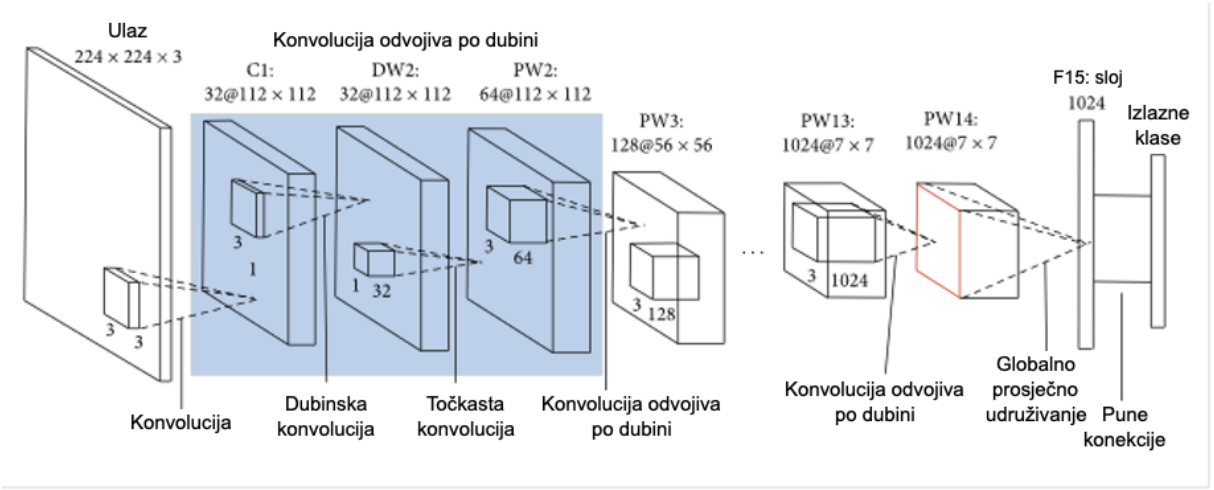
MobileNet je model učinkovitih konvolucijskih neuronskih mreža (CNN) koje je razvio Google za mobilne i ugrađene vidne aplikacije (Howard A. G., 2017). Uvježbani su na skupu podataka ImageNet (J. Deng, 2009), koji je veliki skup podataka slika s oznakama za klasifikaciju, a dizajnirani su da budu mali i brzi, a istovremeno održavaju dobru točnost na skupu podataka ImageNet.

Jedna ključna značajka MobileNets-a je njihova odvojivih konvolucija koji se odvajaju po dubini (Chollet, 2017). U tradicionalnom CNN-u, konvolucijski sloj sastoji se od skupa filtara koji se konvolviraju s ulaznim podacima kako bi proizveli skup izlaznih značajki. Ti se filtri obično uče tijekom obuke i mogu izdvojiti značajne značajke iz ulaznih podataka. U konvoluciji koja se može odvojiti po dubini, filtri su podijeljeni u dva odvojena sloja: sloj po dubini i sloj po točkama. Sloj po dubini izvodi konvoluciju na svakom ulaznom kanalu neovisno, dok sloj po točkama kombinira izlaz sloja po dubini koristeći konvoluciju 1x1.

Korištenje konvolucija koje se mogu odvojiti po dubini omogućuje MobileNet-u značajno smanjenje broja parametara u modelu, što zauzvrat smanjuje količinu memorije i računalnih resursa potrebnih za izvođenje modela (Sandler, 2018). Zbog toga je MobileNet odličan izbor za mobilne uređaje, koji često imaju ograničene resurse. Konvolucije koje se mogu odvojiti po dubini pokazale su se posebno učinkovitima za aplikacije mobilnog vida, budući da mogu značajno smanjiti veličinu i računsku složenost modela bez žrtvovanja točnosti (Howard A. G., 2019).

Uz upotrebu konvolucija koje se mogu odvojiti po dubini, MobileNet također koristi tehniku koja se naziva kanalno skaliranje (Chen, 2014), koja pomaže u daljnjem smanjenju broja parametara u modelu. Skaliranje po kanalima uključuje skaliranje izlaza svakog kanala u konvolucijskom sloju parametrom koji se može naučiti. To omogućuje modelu da prilagodi važnost svakog kanala na temelju podataka koje obrađuje, što može pomoći u poboljšanju ukupne točnosti modela.

MobileNet je naširoko prihvaćen u različitim aplikacijama, uključujući detekciju objekata (Ren, 2015), klasifikaciju slika (A. Krizhevsky, 2012) i segmentaciju (Long, 2015). Nudi dobar kompromis između točnosti i učinkovitosti, što ga čini prikladnim za korištenje na mobilnim i ugrađen (eng. *embedded*) uređajima. Slika 11 prikazuje vizualizaciju arhitekture MobileNet modela.



Slika 11: Arhitektura MobileNet modela (Vlastita izrada prema: https://www.mdpi.com/2073-4433/14/2/280)

5.2. Skup podataka

Prvo, važno je osigurati da skup podataka bude raznolik i reprezentativan za populaciju na koju će se primjenjivati model strojnog učenja. To znači da bi skup podataka trebao sadržavati niz godina, etničkih skupina i drugih relevantnih demografskih podataka, umjesto da bude jako usmjeren prema jednoj određenoj skupini.

Drugo, kvaliteta slika u skupu podataka je jako važna. Ako su slike mutne ili na drugi način niske kvalitete, modelu strojnog učenja bit će ih teško točno klasificirati. Stoga je važno pažljivo pripremiti skup podataka kako bi uključivao samo slike visoke kvalitete.

Treće, veličina skupa podataka također je važan faktor. Općenito, što je veći skup podataka, to će bolje funkcionirati model strojnog učenja. Međutim, također je važno osigurati da skup podataka bude uravnotežen, s otprilike jednakim brojem slika muškaraca i žena. To će pomoći modelu da izbjegne pristranosti i točno klasificira slike obiju skupina.

Konačno, važno je pažljivo označiti slike u skupu podataka, tako da ih model strojnog učenja može naučiti točno klasificirati. To može biti dugotrajan zadatak, ali je ključan za izgradnju modela strojnog učenja visokih performansi.

Odabrani skup podataka je u formatu dokumenta gdje su vrijednosti odvojene zarezom (eng. *Comma separated values*, kraće CSV), s ukupno 5 kolona. Kolone su redom: godina, etnička pripadnost, spol, ime slike, pikseli. Posljednja kolona nosi vrijednosti koje su zapisane kao lista numeričkih vrijednosti svih piksela određene slike. U ovom skupu podataka se ukupno nalazi 27305 zapisa, no detaljnijom analizom možemo vidjeti kako se u koloni pikseli nalazi 23315 unikatnih vrijednosti, što nam govori da je to konačni broj slika ovog skupa. Skup podataka je preuzet s Kaggle web stranice (Arora, 2021).

5.2.1. Distribucija podataka

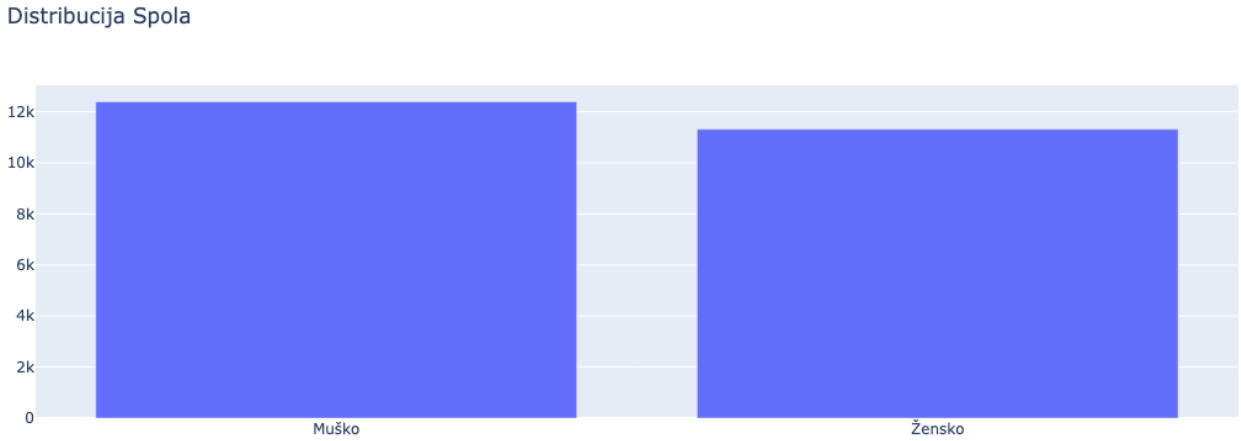
Korištenjem alata Plotly koji služi za analitiku i vizualizaciju podataka, iscrtat ćemo distribuciju podataka za sve tri meke biometrije, odnosno za tri različite klasifikacije.

U našim sljedećim odjeljcima koristit ćemo dijagrame distribucije kako bismo stekli uvide u sastav našeg skupa podataka i kako su dob, etnička pripadnost i spol zastupljeni među podatkovnim točkama.

Kako bi dobili čitljiviju distribuciju podataka za spolnu klasifikaciju potrebno je dodijeliti vrijednostima spolova njihove nazive

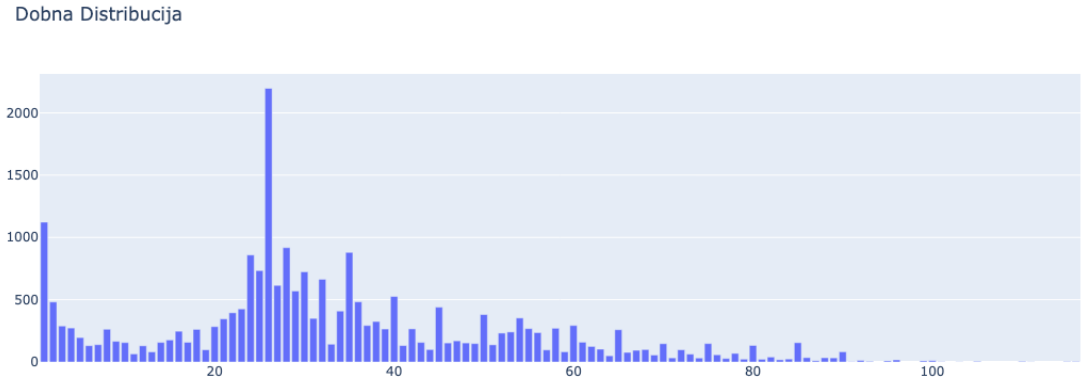
```
gender_dist =
data['gender'].value_counts().rename(index={0:'Muško',1:'Žensko'})
```

Klasifikacija spola ukupno sadrži 12391 slika muških osoba, te 11314 slika ženskih osoba (Slika 12).



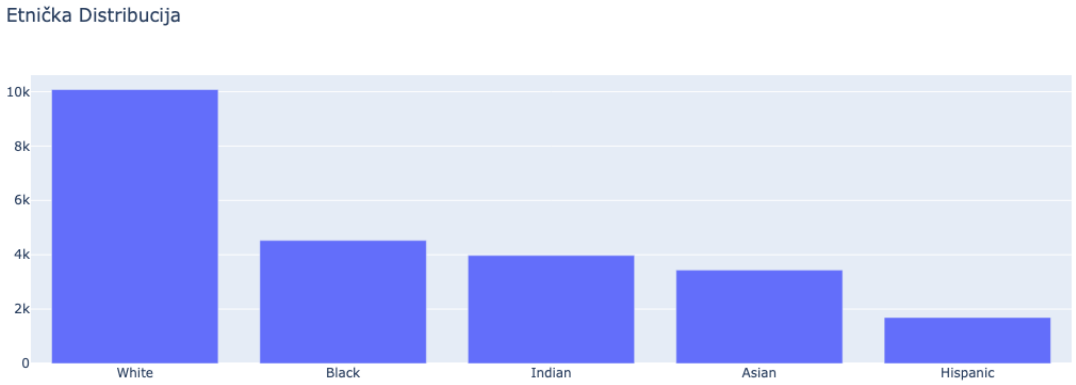
Slika 12. Distribucija spola (Izvor: Vlastita izrada)

Što se tiče dobne distribucije tu možemo vidjeti kako su najzastupljenije slike osoba koje se nalaze u dvadesetim i tridesetim godinama, a iza njih dolaze osobe do 6 godine života (Slika 13).



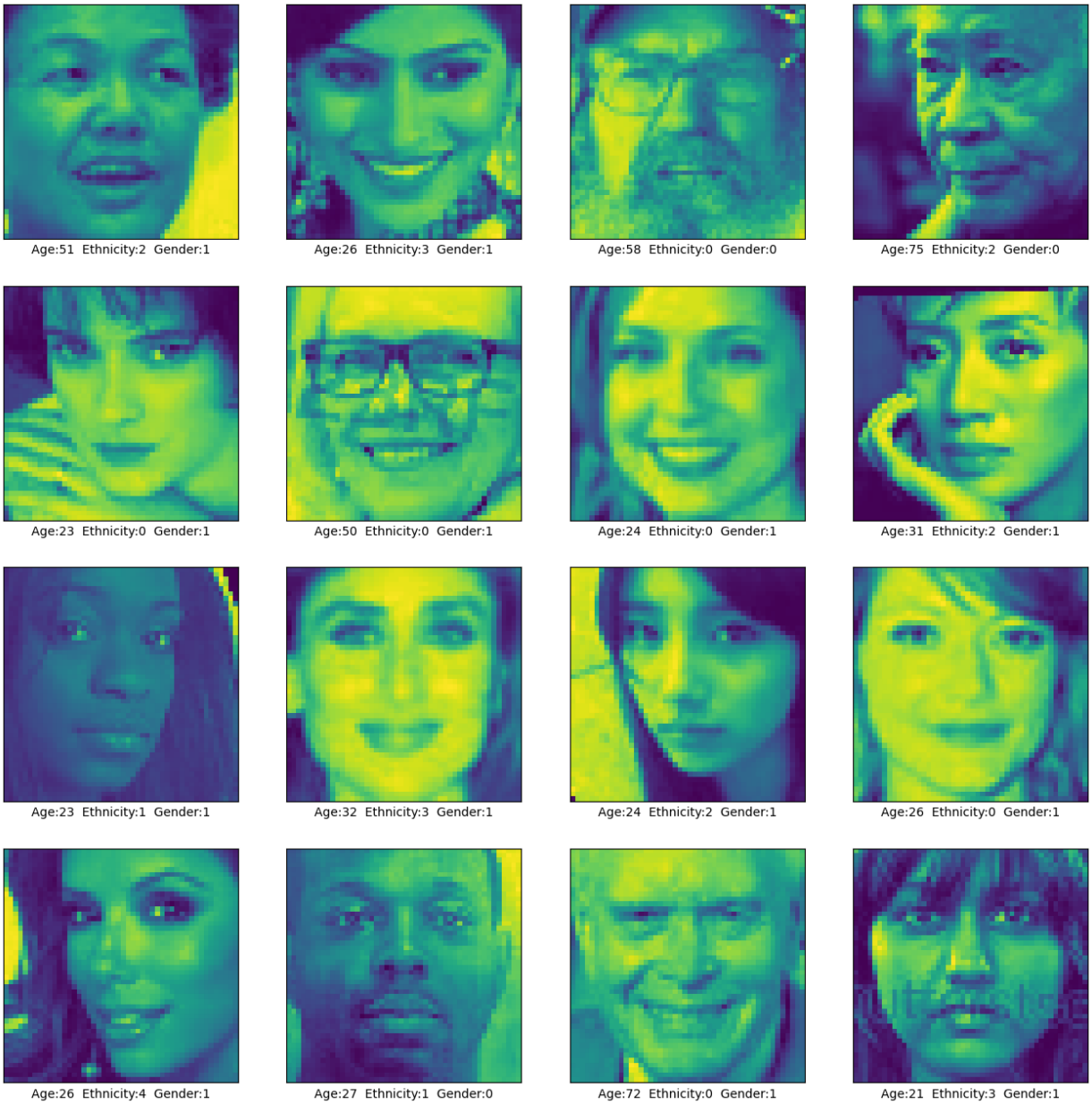
Slika 13. Dobna distribucija (Izvor: Vlastita izrada)

Naposljetku imamo distribuciju po etničkoj pripadnosti, tj. rasnoj pripadnosti. Tu možemo vidjeti kako je skup podataka neujednačen, pa imamo veći broj jedne rase u odnosu na ostale (Slika 14).



Slika 14. Etnička distribucija (Izvor: Vlastita izrada)

U nastavku, na slici 15, prikazan je uzorak od 16 slika iz dataset-a korištenog u fazi treniranja modela. Slike su odabrane nasumično. Na prvu, ispisom nekoliko slučajno odabranih fotografija, možemo zaključiti da je dataset ispunjen pretežno fotografijama lica, bez torza ili pozadine.



Slika 15. Uzorak slika iz dataseta-a (Izvor: Vlastita izrada)

5.3. Treniranje modela i priprema dataseta

Nakon odabira modela i osiguravanja raznolikog skupa podataka slijedi faza treniranja. U toj fazi će biti pojašnjeno označavanje dataseta, kreiranje i obuka modela, te na posljetku završiti s iscrtavanjem performansi i testiranjem modela.

5.3.1. Kreiranje python env varijable

Kako bi trenirali model u kontroliranom okruženju potrebno je kreirati python okruženje (eng. *enviroment*) varijablu. Python enviroment je izolirani odsječak python okruženja na koji ne utječe ostatak instaliranih modula na računalu.

Da biste stvorili Python okruženje, morat ćete imati instaliran Python na vašem računalu. Ako nemate instaliran Python, možete preuzeti najnoviju verziju sa službene web stranice Python (https://www.python.org/downloads/).

Nakon što instalirate Python, možete stvoriti Python okruženje pomoću modula venv koji je uključen u Python standardnu biblioteku. Da biste stvorili novo okruženje, otvorite prozor terminala i dođite do direktorija u kojem želite stvoriti okruženje. Zatim upišite sljedeću naredbu:

```
python3 -m venv envname
```

Zamijenite `envname` imenom koje želite dati svom okruženju. Ovo će stvoriti novi direktorij s navedenim nazivom koji sadrži Python izvršnu datoteku i sve potrebne datoteke za okruženje.

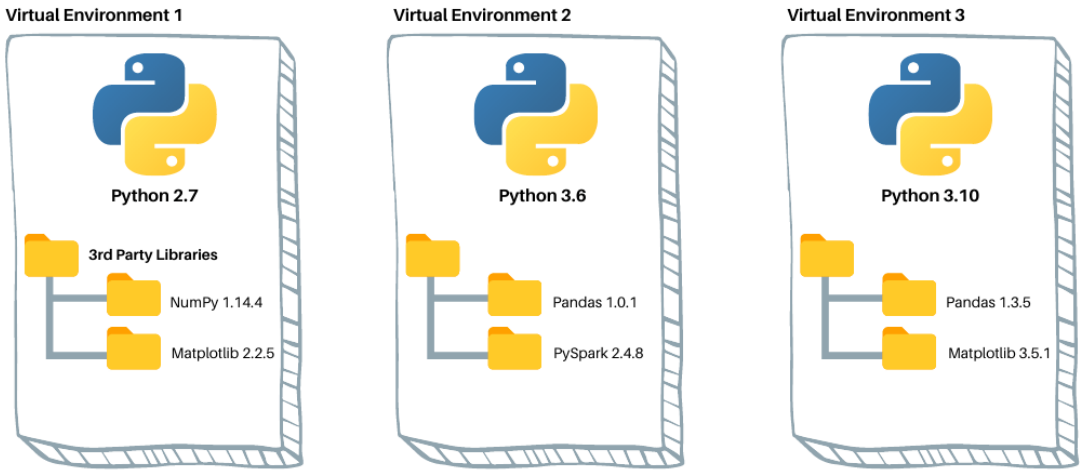
Za aktiviranje okruženja možete koristiti sljedeću naredbu:

```
source envname/bin/activate
```

Ovo će promijeniti upit terminala tako da uključuje naziv okruženja, što pokazuje da je okruženje aktivno. Da biste deaktivirali okruženje, jednostavno upišite `deactivate` i pritisnite Enter.

Tada možete koristiti naredbu `pip` za instaliranje svih paketa koji su vam potrebni za vaš projekt. Ovi paketi bit će instalirani lokalno unutar okruženja i neće utjecati na pakete instalirane u globalnom Python okruženju. To vam omogućuje održavanje zasebnih instalacija paketa za različite projekte i pomaže u izbjegavanju sukoba između paketa (slika 16).

Nakon kreiranja python enviromenta, potrebno je instalirati potrebne module.



Slika 16: Ilustracija python enviroment-a (Izvor: https://www.dataquest.io/wp-content/uploads/2022/01/python-virtual-envs1.webp)

5.3.2. Učitavanje i označavanje dataseta

Prvi korak u pripremi ovog skupa podataka za strojno učenje je definiranje nekih varijabli i konstanti koje će se koristiti u cijelom programu. Pripremna faza podrazumijeva transformaciju naših neobrađenih podataka o pikselima. Naš skup podataka, pažljivo organiziran unutar CSV datoteke pod nazivom `gender_ethnicity_age.csv`, sadrži mnoštvo informacija, uključujući dob, etničku pripadnost, spol, nazive slika i prikaze piksela.

Kako bismo postavili temelje za sofisticiranu meku biometrijsku analizu, okrenuli smo se Python biblioteci Pandas. Započeli smo uvozom i strukturiranjem našeg skupa podataka u Pandas DataFrame, varijabla pod nazivom `data`. Ovaj početni korak bio je ključan jer je pružio čvrstu osnovu za kasniju manipulaciju i istraživanje podataka.

Sljedeća naredba prikazuje funkcije korištenu za otvaranje CSV datoteke:

```
data = pd.read_csv('./dataset/gender_ethnicity_age.csv')
```

Međutim, bit naše transformacije podataka nalazi se u stupcu `pixels`. U početku su vrijednosti piksela bile pohranjene kao nizovi, promišljeno odvojeni razmacima unutar svakog retka. Kako bismo te podatke učinili podložnima analizi, orkestriran je proces u više koraka. Ključna transformacija pokrenuta je implementacijom lambda funkcije:

```
data['pixels'] = data['pixels'].apply(lambda x: np.array(x.split(),
dtype="float32"))
```

Korak `x.split()` izvršava dijeljenje vrijednosti piksela, koje su izvorno postojale kao nizovi, u popis koji se sastoji od pojedinačnih numeričkih vrijednosti. Nakon toga funkcija `np.array()` kreira novi niz s parametrom `dtype` postavljenim na vrijednost `float32`, tim parametrom osiguravamo preciznost svakog elementa u nizu.

Nad novim podaci o pikselima koji su se nakon transformacije sada nalaze u nizovima čije su vrijednosti float32, pokrećemo funkcije `head()` koja nam daje uvid u prvih 5 zapisa transformiranih podataka:

Tablica 2. Prvih 5 zapisa iz CSV datoteke

	age	ethnicity	gender	img_name	pixels
0	1	2	0	20161219203650636.jpg.chip.jpg	[129.0, 128.0, 128.0, 126.0, 127.0, 130.0, 133...
1	1	2	0	20161219222752047.jpg.chip.jpg	[164.0, 74.0, 111.0, 168.0, 169.0, 171.0, 175....
2	1	2	0	20161219222832191.jpg.chip.jpg	[67.0, 70.0, 71.0, 70.0, 69.0, 67.0, 70.0, 79....
3	1	2	0	20161220144911423.jpg.chip.jpg	[193.0, 197.0, 198.0, 200.0, 199.0, 200.0, 202...
4	1	2	0	20161220144914327.jpg.chip.jpg	[202.0, 205.0, 209.0, 210.0, 209.0, 209.0, 210...

Pokretanjem sljedećeg koda dobivamo ukupan boj kolona i zapisa, koji za kolone iznosi 5, dok za zapise iznosi 23 705:

```
print('Ukupan broj zapisa: {}'.format(len(data)))
print('Ukupan broj kolona: {}'.format(len(data.columns)))
```

Idući korak u procesu pripreme je normalizacija podataka. Primjenom transformacije:

```
data['pixels'] = data['pixels'].apply(lambda x: x/255)
```

mijenjamo skalu vrijednosti piksela, u rasponu od 0 do 255, na standardiziranu skalu između 0 i 1. Ovo proces normalizacije osigurava ujednačenost i olakšava smislene usporedbe tijekom naše analize.

U završnom koraku pripreme podataka za fazu treniranja modela izvršavamo posljednju transformaciju podataka. Prvo izdvajamo podatke o pikselima iz stupca `pixels` u našem skupu podataka. Ti se podaci naknadno pretvaraju u NumPy niz kako bi se olakšale učinkovite numeričke operacije.

```
X = np.array(data['pixels'].tolist())
X = X.reshape(X.shape[0],48,48,1)
```

Središnja transformacija događa se kada preoblikujemo podatke o pikselima. Izvorno u 1D formatu, ti se podaci pretvaraju u strukturirani 3D format. Ovo preoblikovanje ga usklađuje s uobičajenim strukturama slikovnih podataka, osiguravajući kompatibilnost s raznim tehnikama obrade slike i strojnog učenja. Preoblikovani format uključuje sljedeće dimenzije:

- Prva dimenzija predstavlja broj slika unutar našeg skupa podataka.
- Sljedeće dimenzije označavaju visinu i širinu svake slike u pikselima (48x48).
- Konačna dimenzija pokazuje da radimo sa slikama u sivim tonovima, što je označeno vrijednošću `1`, koja predstavlja jedan kanal za intenzitet piksela.

5.3.3. Kreiranje i obuka modela

U ovom odjeljku radimo kreiranje i treniranje modela za prepoznavanje spola. Da bismo to postigli, koristimo funkciju `train_test_split` uvezenu iz modula `sklearn.model_selection` što je dio biblioteke `sklearn`.

U sljedećoj sekciji koja objašnjava proces kreiranja i treniranja modela za predviđanje spola, detaljnije će se obraditi cijeli postupak, dok će se za iduća dva objasniti samo razlike koje su primijenjene u definiranju arhitekture modela.

5.3.3.1. Treniranje modela za predviđanje spola

Naš početni korak je definiranje ciljne varijable za naš model predviđanja spola. Izvlačimo atribut `gender` iz našeg skupa podataka, označavajući ga kao `y`. Ovaj atribut obuhvaća oznake spola povezane sa svakom podatkovnom točkom i služit će kao referenca za učenje našeg modela.

U fazi podjele podataka koristimo funkciju `train_test_split` kako bismo podijelili naše podatke u skupove za obuku i testiranje. Ova je podjela ključna za procjenu izvedbe i mogućnosti generalizacije našeg modela predviđanja spola.

```
y = data['spol']
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.22, random_state=37)
```

Unutar funkcije navedeno je nekoliko parametara:

- `X` označava ulazne podatke, koji se sastoje od prikaza piksela slika koje smo prethodno pripremili.
- `y` je ciljna varijabla, koja sadrži oznake spola.
- `test_size=0.22` - dodjeljujemo 22% našeg skupa podataka skupu za testiranje, ostavljajući preostalih 78% za obuku. Ovo odvajanje osigurava da se naš model procjenjuje na nevidljivim podacima, što je ključni aspekt procjene modela.
- `random_state=37` - postavljamo nasumično početno mjesto (37 u ovom slučaju) kako bismo osigurali ponovljivost. Ovo nasumično stanje osigurava da proces dijeljenja podataka ostane dosljedan u različitim izvođenjima koda.

Ishod ove operacije dijeljenja podataka je stvaranje četiri različita skupa:

- `X_train` varijabla o podacima koja sadrži pikselske prikaze slika korištenih za obuku modela.
- `X_test` varijabla o podacima testiranja, koji se sastoje od slika zadržanih za procjenu modela.
- `y_train` varijabla oznake spola, usmjeravajući model tijekom procesa učenja..
- `y_test` varijabla za skup koji obuhvaća oznake spola koje odgovaraju podacima testiranja.

Ovo dijeljenje podataka postavlja početnu točku za razvoj i procjenu našeg modela predviđanja spola. Odvajanjem podataka na ovaj način osiguravamo da se izvedba našeg modela može procijeniti na prethodno nevidljivim podacima, pružajući robusnu mjeru njegovih prediktivnih mogućnosti unutar našeg modela.

Sljedeći koda se bavi izradom modela predviđanja spola pomoću TensorFlow Kerasa. arhitektura modela definirana je kao sekvencijalna neuronska.

```
model = tf.keras.Sequential([
    L.InputLayer(input_shape=(48,48,1)),
    L.Conv2D(32, (3, 3), activation='relu', input_shape=(32, 32, 3)),
    L.BatchNormalization(),
    L.MaxPooling2D((2, 2)),
    L.Conv2D(64, (3, 3), activation='relu'),
    L.MaxPooling2D((2, 2)),
    L.Flatten(),
    L.Dense(64, activation='relu'),
    L.Dropout(rate=0.5),
    L.Dense(1, activation='sigmoid')])
```

Arhitektura modela definirana je kao sekvencijalna neuronska mreža sa sljedećim slojevima:

- `L.InputLayer`: ulazni oblik postavljen je na (48, 48, 1), što znači da model očekuje slike u sivim tonovima dimenzija 48x48 piksela.
- `L.Conv2D`: dva konvolucijska sloja (Conv2D) uključena su u ReLU aktivacijske funkcije. Ovi slojevi imaju za cilj izdvojiti relevantne značajke iz ulaznih slika.
- `L.BatchNormalization`: normalizacija serije primjenjuje se za normalizaciju aktivacija i povećanje stabilnosti treninga.
- `L.MaxPooling2D`: dva sloja maksimalnog skupljanja koriste se za smanjivanje uzorkovanja mapa značajki, smanjujući računsku složenost.
- `L.Flatten`: spljošteni sloj pretvara 2D mape značajki u 1D format, pripremajući podatke za sljedeće guste slojeve.
- `L.Dense`: koriste se dva gusta (potpuno povezana) sloja, od kojih prvi ima 64 neurona i ReLU aktivacijsku funkciju. Drugi sloj, s jednim neuronom i sigmoidnom aktivacijskom funkcijom, služi kao izlazni sloj za binarno predviđanje spola.
- `L.Dropout`: Dropout je uveden kako bi se spriječilo prekomjerno prilagođavanje nasumičnim deaktiviranjem 50% neurona u gustom sloju tijekom treninga.

```
model.compile(optimizer='sgd',
loss=tf.keras.losses.BinaryCrossentropy(), metrics=['accuracy'])
```

Ovdje je prikaz konfiguracije modela za treniranje sastavljen s optimizatorom Stochastic Gradient Descent (SGD), binarnom funkcijom unakrsnog entropijskog gubitka, koja je prikladna za binarnu klasifikacije, što je za ovaj model idealno i točnošću kao metrikom procjene. Dodatno, prilagođena funkcija povratnog poziva definirana je za praćenje gubitka valjanosti tijekom obuke. Zaustavlja obuku ako gubitak valjanosti padne ispod određenog praga.

```
class myCallback(tf.keras.callbacks.Callback):
    def on_epoch_end(self, epoch, logs={}):
        if(logs.get('val_loss')<0.2650):
            print("\nDosegnuto 0,2650 val_loss pa se obuka otkazuje!")
            self.model.stop_training = True
callback = myCallback()
model.summary()
```

Ovaj kod definira povratnu funkciju pod nazivom `myCallback` unutar okvira TensorFlow Keras. Primarna svrha ove povratne funkcije je praćenje gubitka valjanosti tijekom procesa obuke modela i, ako je ispunjen određeni uvjet, rano zaustavljanje obuke. Značajke navedenog koda:

- `on_epoch_end`: metoda koja se poziva na kraju svake epohe obuke. Potrebna su dva argumenta, `self` (odnosi se na objekt povratnog poziva) i dnevnici (rječnik koji sadrži različite metrike obuke).
- `if(logs.get('val_loss') < 0,2650)`: unutar metode `on_epoch_end` provjeravamo je li gubitak valjanosti (`val_loss`) manji od 0,2650, što je unaprijed definirani prag. Ovaj prag odabran je za praćenje performansi modela tijekom treninga.
- `print("\nDosegnuto 0,2650 val_loss pa se obuka otkazuje!")`: ako je uvjet ispunjen (tj. gubitak valjanosti padne ispod 0,2650), ispisuje se poruka koja obavještava da se proces obuke otkazuje.
- `self.model.stop_training = True`: povratni poziv postavlja atribut `stop_training` modela na `True`, učinkovito zaustavljajući proces obuke.

Ova povratna funkcija koristi se za rano zaustavljanje, osiguravajući prekid obuke kada model dosegne zadovoljavajuću razinu izvedbe, kao što je naznačeno gubitkom valjanosti. Pomaže u sprječavanju prekomjernog treniranja i štedi resurse tijekom obuke, čineći proces obuke modela učinkovitijim i djelotvornijim.

Pozivanjem metode `model.summary()` ispisuje se sažetak niza mreže, što je u biti sveobuhvatan pregled arhitekture modela, uključujući pojedinosti kao što su vrste slojeva, izlazni oblici i broj parametara.

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_17 (Conv2D)          (None, 46, 46, 32)        320

 batch_normalization_7 (Bat  (None, 46, 46, 32)        128
 chNormalization)

 max_pooling2d_17 (MaxPooli  (None, 23, 23, 32)        0
 ng2D)

 conv2d_18 (Conv2D)          (None, 21, 21, 64)        18496
```

```
 max_pooling2d_18 (MaxPooli  (None, 10, 10, 64)        0
 ng2D)

 flatten_7 (Flatten)         (None, 6400)              0

 dense_14 (Dense)            (None, 64)                409664

 dropout_7 (Dropout)         (None, 64)                0

 dense_15 (Dense)            (None, 1)                 65

...
Total params: 428673 (1.64 MB)
Trainable params: 428609 (1.64 MB)
Non-trainable params: 64 (256.00 Byte)
```

U nastavku će biti prikazan završni korak u fazi treniranja modela, a to je samo pokretanje.

```
history = model.fit(X_train, y_train, epochs=20, validation_split=0.1,
batch_size=64, callbacks=[callback])
```

Proces obuke uključuje sljedeće ključne komponente:

- `X_train` i `y_train`: podaci o obuci i odgovarajuće oznake spola, koje će se koristiti za obuku modela.
- `epohe=20`: proces obuke je podijeljen u 20 epoha, što znači da će cijeli skup podataka obuke model obraditi 20 puta. Svaka epoha predstavlja jedan potpuni prolaz kroz podatke.
- `validation_split=0.1`: dio podataka o obuci (10%) odvaja se kao skup za provjeru valjanosti. Ovaj skup za provjeru valjanosti koristi se za praćenje performansi modela tijekom obuke i osiguravanje njegove dobre generalizacije na nevidljive podatke.
- `batch_size=64`: tijekom svake epohe obuke, podaci su podijeljeni u serije, pri čemu svaka serija sadrži 64 podatkovne točke. To pomaže u optimiziranju procesa treninga i učinkovitom upravljanju memorijskim resursima.
- `callbacks=[callback]`: prilagođena funkcija povratnog poziva (`myCallback`) definirana ranije prosljeđuje se kao povratni poziv. Ova funkcija nadzire gubitak

valjanosti i može rano zaustaviti obuku ako je ispunjen određeni uvjet, povećavajući učinkovitost obuke.

```
Epoch 1/20
260/260 [==============================] - 12s 43ms/step - loss: 0.5353
- accuracy: 0.7391 - val_loss: 0.4032 - val_accuracy: 0.8307
Epoch 2/20
260/260 [==============================] - 14s 55ms/step - loss: 0.4001
- accuracy: 0.8217 - val_loss: 0.3602 - val_accuracy: 0.8453
...
Epoch 19/20
260/260 [==============================] - 10s 40ms/step - loss: 0.1797
- accuracy: 0.9273 - val_loss: 0.2852 - val_accuracy: 0.8821
Epoch 20/20
260/260 [==============================] - 11s 41ms/step - loss: 0.1708
- accuracy: 0.9319 - val_loss: 0.2937 - val_accuracy: 0.8816
```

Ispis je skraćen zbog sličnosti. Vidimo da je treniranje modela stalo na 20-toj epohi, iako je postojala funkcija koje je zadužena za zaustavljanje treninga onog trenutka kada dosegne zadani cilj. Cilj u ovom slučaju nije ostvaren, stoga je model prestao s treningom u posljednjoj epohi.

Generalno, ovaj je model jednostavan CNN koji je vrlo prikladan za zadatke klasifikacije slika. Podešavanjem parametara modela, kao što su broj i veličina slojeva, moguće je poboljšati izvedbu modela na zadatku klasifikacije.

5.3.3.2. Treniranje modela za predviđanje etničke pripadnosti

U procesu treniranja modela za predviđanje etničke pripadnosti, postoje mnoge sličnosti u pripremi, stoga će se navesti samo razlike koje su učinjene u postupku. Za razliku od spola gdje smo izvlačili atribut `gender` moramo izvući atribut `ethnicity` iz skupa podataka.

```
y = data['ethnicity']
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.22, random_state=37)
```

U `train_test_split` funkciju prosljeđujemo iste parametre kao i u modelu za predviđanje spola.

Što se tiče kreiranja konfiguracije samog modela za predikciju etničke pripadnosti, razlike koja je načinjena u arhitekturi u odnosu na model za predikciju spola je ta da imamo gusti sloj s 5 neurona, koji predstavljaju 5 različitih klasa etničke pripadnosti. Nasuprot tome,

model predviđanja spola ima jedan neuron s funkcijom sigmoidne aktivacije, prikladan za binarnu klasifikaciju (muški ili ženski).

```
…
L.Dense(5)
```

Za predviđanje etničke pripadnosti koristimo rijetku kategoričku funkciju unakrsnog entropijskog gubitka, koja je prikladna za probleme klasifikacije više klasa. Dok za predviđanje spola koristimo binarnu unakrsnu entropiju jer je to zadatak binarne klasifikacije.

```
loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True)
```

Funkcija povratnog poziva razlikuje se između dva modela. U modelu predviđanja etničke pripadnosti obuka se zaustavlja kada točnost validacije prijeđe 79,5%, dok u modelu predviđanja spola obuka prestaje kada gubitak validacije padne ispod 0,2650.

Ove razlike odražavaju specifične zahtjeve i karakteristike dvaju zadataka predviđanja: spola (binarno) i etničke pripadnosti (više klasa).

```
history = model.fit(X_train, y_train, epochs=16, validation_split=0.1,
batch_size=64, callbacks=[callback])
```

U slučaju modela predviđanja spola, odlučili smo se za prošireniji režim treninga, koji obuhvaća 20 epoha. Svaka epoha uključuje potpuni prolaz kroz skup podataka za obuku. Ovaj izbor omogućio je modelu da postupno poboljša svoju sposobnost razlikovanja muških i ženskih pojedinaca.

S druge strane, model predviđanja etničke pripadnosti slijedio je kraću trajektoriju obuke, obuhvaćajući 16 epoha. Ova se odluka temeljila na jedinstvenim karakteristikama zadatka predviđanja etničke pripadnosti, gdje je model trebao klasificirati pojedince u jednu od pet različitih kategorija etničke pripadnosti.

Sljedeći čimbenik koji razlikuje ova dva modela leži u odabiru funkcija gubitka. Model predviđanja spola koristio je binarnu kros-entropiju, funkciju gubitka prilagođenu za zadatke binarne klasifikacije. To je imalo smisla jer je cilj bio utvrditi je li pojedinac muško ili žensko, što je binarni ishod.

Nasuprot tome, model predviđanja etničke pripadnosti koristio je rijetki kategorički unakrsni gubitak entropije. Ova funkcija gubitka je prikladna za zadatke klasifikacije više klasa, kao što je identificiranje etničke pripadnosti pojedinaca među pet različitih kategorija. To je omogućilo modelu da uči i učinkovito klasificira slike u više različitih grupa.

Za model predviđanja spola, prilagođeni povratni poziv pratio je gubitak valjanosti, intervenirajući kada je pao ispod unaprijed definiranog praga od 0,2650. Nasuprot tome, `myCallback` funkcija modela predviđanja etničke pripadnosti usredotočen je na točnost provjere valjanosti, prekidajući obuku kada dosegne ili premaši 79,5%.

```
Epoch 1/16
260/260 [==============================] - 6s 23ms/step - loss: 2.6373 -
accuracy: 0.4430 - val_loss: 1.0764 - val_accuracy: 0.6052
...
Epoch 15/16
260/260 [==============================] - 6s 23ms/step - loss: 0.7067 -
accuracy: 0.7567 - val_loss: 0.8344 - val_accuracy: 0.7696
Epoch 16/16
260/260 [==============================] - 6s 23ms/step - loss: 0.7004 -
accuracy: 0.7546 - val_loss: 0.7802 - val_accuracy: 0.7637
```

Ispis je skraćen zbog sličnosti. Kao što smo vidjeli prethodno u ispisu epoha modela za predviđanje spola, ovaj model također prestaje s treningom u posljednjoj epohi. Isto tako možemo vidjeti kako ni u ovom slučaju funkcija zadužena za zaustavljanje treniranja modela nije iskorištena jer cilj nije dosegnut.

5.3.3.3. Treniranje modela za predviđanje dobi

Kao i u prošlom odlomku, ovdje će biti opisane isključivo razlike u konstrukciji modela za predviđanje dobi. Unutar modela predviđanja dobi uspostavili smo duboku neuronsku mrežu s temeljem u konvolucijskim slojevima (CNN). Ova arhitektura, inspirirana je prethodnim modelima, međutim, odstupa na neke ključne načine.

```
model = tf.keras.Sequential([
    ...
    L.Conv2D(128, (3, 3), activation='relu'),
    ...
    L.Dense(1, activation='relu')
])
```

Iako zadržava temeljne elemente CNN-a, pored ojačavanja modela s još jednim konvolucijskim slojem, ovaj model uvodi bitnu promjenu kod izlaznog sloja. Za razliku od jednog neurona modela predviđanja spola sa sigmoidnom aktivacijom, ovaj model ima jedan neuron s ReLU aktivacijom. Ova prilagodba omogućuje predviđanje dobi kao kontinuirane vrijednosti.

Izbor funkcije gubitka u modelu predviđanja dobi značajno se razlikuje od funkcija iz prethodnih modela, usklađujući se s prirodom zadatka:

```
model.compile(optimizer='adam', loss='mean_squared_error',
metrics=['mae'])
```

Koristimo gubitak srednje kvadratne pogreške (MSE), prikladan izbor za regresijske zadatke, gdje je cilj predvidjeti kontinuirane vrijednosti poput dobi. Ovo odstupa od binarne unakrsne entropije i rijetke kategoričke unakrsne entropije koja se koristi za predviđanje spola i etničke pripadnosti.

Kako bi obuka bila pravilna potrebno je uključiti funkciju povratka, njena razlika u odnosu na prethodne je ta da nadzire gubitak valjanosti, intervenirajući kada dosegne unaprijed definirani prag od 105. Ovaj je prag pomno odabran kako bi se osigurala točnost modela u procjeni dobi.

Trening modela dobi proteže se kroz 20 epoha, pri čemu svaka epoha predstavlja cijeli ciklus kroz skup podataka o obuci. Kao i u prethodnim modelima, tako i u ovom, model doseže posljednju epohu čime ne iskorištava funkciju za povrat.

Pri procjeni izvedbe modela predviđanja dobi, primarni pokazatelji od interesa su gubitak srednje kvadratne pogreške (MSE) i srednja apsolutna pogreška (MAE). Ove metrike mjere točnost predviđanja dobi i daju uvid u to koliko su procjene modela usklađene sa stvarnom dobi.

Ukratko, model predviđanja dobi se sažima kroz nekoliko značajnih razlika u odnosu na prethodne modele. Uvodi jedinstveni izlazni sloj, koristi srednji kvadrat gubitka pogreške prilagođen za regresiju i ima prilagođeni povratni poziv za kontrolu obuke na temelju gubitka valjanosti. Ove prilagodbe odražavaju specijalizirani fokus modela na predviđanje kontinuiranih vrijednosti dobi u području meke biometrijske analize.

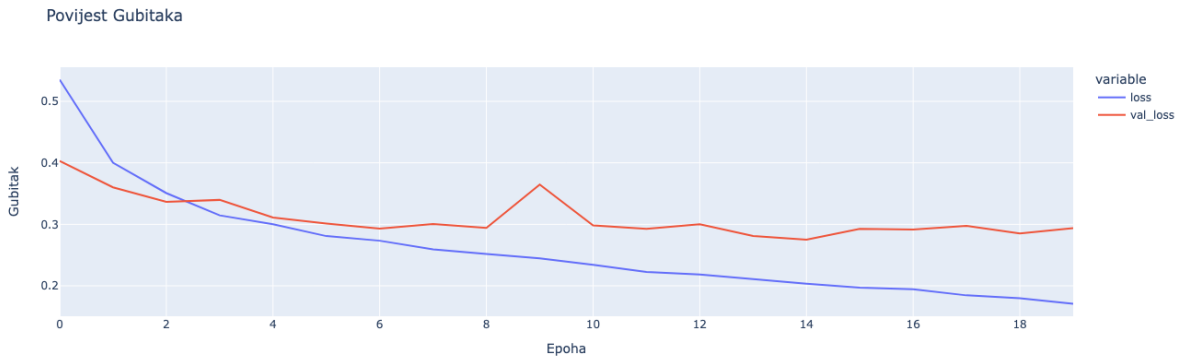
5.3.4. Iscrtavanje performansi modela

Kako bismo stekli sveobuhvatno razumijevanje izvedbe našeg modela predviđanja spola, okrećemo se vizualizacijama koje pruža biblioteka Plotly. Ove vizualizacije bacaju svjetlo na proces učenja modela i njegovu učinkovitost u razlikovanju spola na temelju crta lica.

5.3.4.1. Performanse modela za predikciju spola

Grafikon gubitaka, prikazan u nastavku, pruža povijesnu perspektivu izvedbe modela u vezi s vrijednostima gubitaka u različitim epohama. Prati i gubitke u obuci i gubitke pri validaciji tijekom procesa obuke.

```
fig_loss = px.line(
    history.history, y=['loss', 'val_loss'],
    labels={'index': 'Epoha', 'value': 'Gubitak'},
    title='Povijest Gubitaka'
)
fig_loss.show()
```

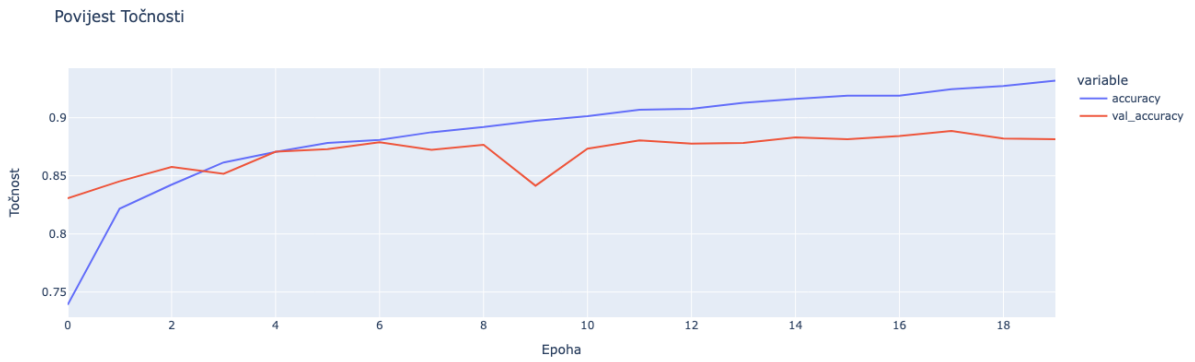


Slika 17. Prikaz pada gubitka modela za predikciju spola (Izvor: Vlastita izrada)

Proučavajući ovaj grafikon, možemo razlučiti koliko je dobro model minimizirao svoju funkciju gubitka tijekom vremena, nudeći uvid u njegovu konvergenciju i potencijal za daljnje usavršavanje.

Jednako je važan grafikon točnosti, ilustriran u nastavku, koji predstavlja putovanje učenja modela u pogledu točnosti klasifikacije.

```
fig_acc = px.line(
    history.history, y=['accuracy', 'val_accuracy'],
    labels={'index': 'Epoha', 'value': 'Točnost'},
    title='Povijest Točnosti'
)
fig_acc.show()
```



Slika 18. Prikaz rasta preciznosti modela za predikciju spola (Izvor: Vlastita izrada)

Ovaj grafikon nam omogućuje da pratimo kako se točnost modela na podacima o obuci i validaciji razvija kako obuka napreduje. Otkriva obrasce poboljšanja ili stagnacije u sposobnosti modela da ispravno klasificira spol na temelju crta lica.

U kontekstu obučavanja modela strojnog učenja, kao što su neuronske mreže, `loss` i `val_loss` obično se odnose na dvije različite vrste funkcija gubitka koje se koriste za praćenje i procjenu izvedbe modela tijekom obuke.

Ukratko, `loss` je funkcija gubitka koja se koristi tijekom obuke za optimizaciju parametara modela, dok je `val_loss` funkcija gubitka koja se koristi za procjenu izvedbe generalizacije modela na zasebnom skupu podataka za validaciju. Razlika između to dvoje može nam pomoći da shvatimo uči li model generalizirati ili se previše uklapa u podatke o obuci. Veliki razmak između `loss` i `val_loss` može ukazivati na pretjerano opremanje, dok mali razmak sugerira dobru generalizaciju.

S druge strane imamo `accuracy` i `val_accuracy`, dvije su različite metrike koje se koriste za mjerenje izvedbe modela strojnog učenja tijekom obuke i provjere.

`accuracy` je mjera koliko dobro model odgovara podacima o obuci. Tijekom treninga cilj je maksimizirati točnost skupa podataka za trening. Visoka točnost obuke ukazuje na to da model uči ispravno klasificirati podatke obuke.

`val_accuracy` je mjera koliko se dobro model generalizira na nove, dosad nepoznate podatke. Odražava sposobnost modela da napravi točna predviđanja na temelju podataka kojima nije bio izložen tijekom obuke. U idealnom slučaju, želja je da točnost obuke i točnost provjere budu visoke i blizu jedna drugoj, jer to sugerira da model ne prilagođava podatke o obuci i da će vjerojatno imati dobre rezultate u scenarijima stvarnog svijeta.

Ukratko, `accuracy` i `val_accuracy` važne su metrike za procjenu izvedbe modela, ali su usredotočene na različite skupove podataka. `accuracy` opisuje podatke o obuci, dok `val_accuracy` podatke o validaciji. Praćenje obje metrike pomaže vam da shvatite koliko dobro model uči iz podataka o obuci i koliko je vjerojatno da će dobro raditi na nevidljivim podacima.

Na kraju, kako bismo pružili kvantitativnu procjenu izvedbe modela, izračunavamo dvije bitne metrike procjene: gubitak testa i točnost testa.

```
loss, acc = model.evaluate(X_test,y_test,verbose=0)
print('Gubitak: {}'.format(loss))
print('Točnost: {}'.format(acc))
```

Priloženi kod ispisuje sljedeće rezultate:

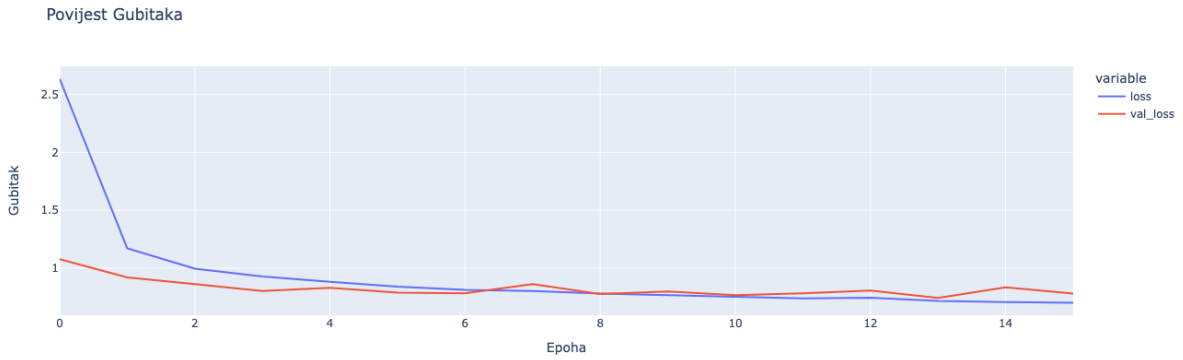
```
Gubitak: 0.2611088156700134
Točnost: 0.8935966491699219
```

Testni gubitak (0,2611) označava izvedbu modela u minimiziranju funkcije gubitka na prethodno nevidljivim testnim podacima. U međuvremenu, točnost testa (0,8936) predstavlja sposobnost modela da ispravno klasificira spol u skupu testnih podataka.

Ove vizualizacije i metrike procjene zajedno nude vrijedan uvid u učinkovitost našeg modela predviđanja spola. Omogućuju nam da procijenimo njegov proces učenja, njegovu sposobnost da minimizira gubitak i njegovu učinkovitost u točnom razlikovanju spola.

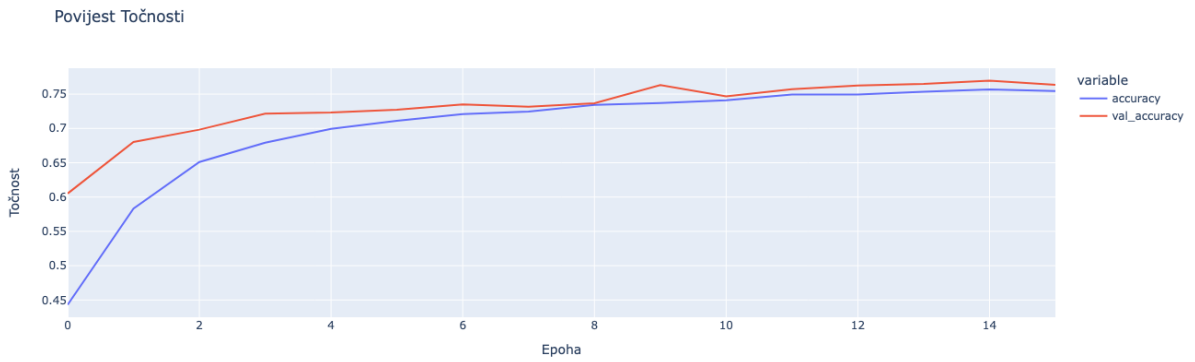
5.3.4.2. Performanse modela za predikciju etničke pripadnosti

Slično kao i u prethodnom odlomku, isti kod se koristi za prikaz grafikona gubitka, te će se prikazati izvedba modela u vezi s vrijednostima gubitaka u različitim epohama..



Slika 19. Prikaz pada gubitka modela za predikciju etničke pripadnosti (Izvor: Vlastita izrada)

Iz navedene slike možemo vidjeti kako je gubitak padao većinu vremena treniranja modela. Ta informacija upućuje na dobre rezultate nakon treniranja.



Slika 20. Prikaz rasta preciznosti modela za predikciju etničke pripadnosti (Izvor: Vlastita izrada)

Iz prikazanog grafa se očito može vidjeti rast preciznosti modela na `accuracy` i `val_accuracy` dijelu dataseta. To nam govori da smo vrlo vjerojatno obučili precizan i vjerodostojan model.

```
Gubitak: 0.7904518246650696
Točnost: 0.7572852969169617
```

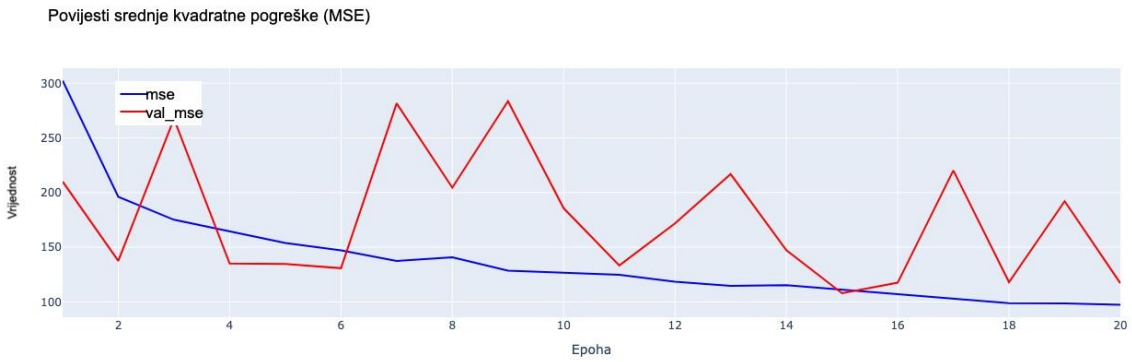
Gubitak testa (0,7905) označava izvedbu modela u smanjenju funkcije gubitka na prethodno neviđenim testnim podacima, dok točnost testa (0,7573) predstavlja njegovu sposobnost da točno predvidi etničku pripadnost u skupu testnih podataka.

5.3.4.3. Performanse modela za predikciju dobi

Grafikoni za prikaz performansi modela za predikciju dobi, razlikuje se u odnosu na one iz prošla dva modela, tj. umjesto prikaza povijesti gubitka, koristit će se povijest srednje apsolutne pogreške (kraće MAE). Isto tako, prikazati će se rezultati srednje kvadratne pogreške (kraće MSE).

MSE se izračunava uzimanjem prosjeka kvadrata razlika između predviđenih i stvarnih vrijednosti, dok se MAE izračunava uzimanjem prosjeka apsolutnih razlika između predviđenih i stvarnih vrijednosti.

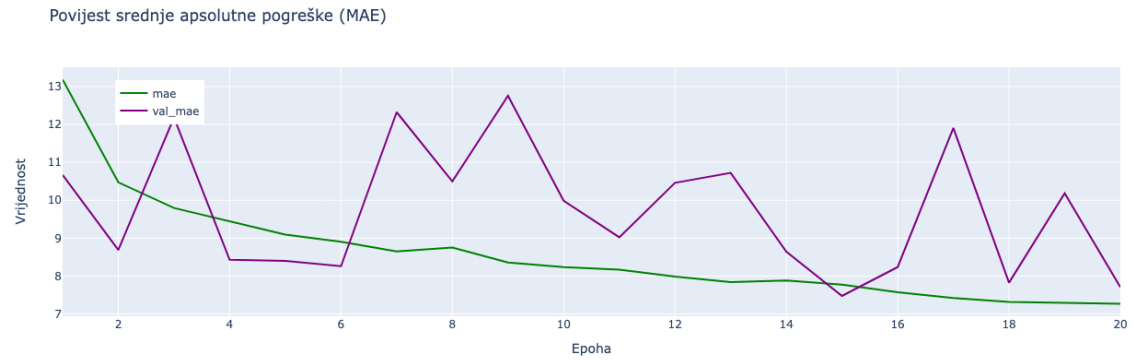
Ukratko, MSE i MAE služe različitim svrhama. MSE je prikladan kada se želi kazniti i uzeti u obzir velike pogreške, čineći ga osjetljivim na odstupanja. MAE je prikladan kada se želi uravnoteženiji prikaz pogrešaka, tretirajući sve pogreške jednako i manje je osjetljiv na odstupanja.



Slika 21. Prikaz povijesti srednje kvadratne pogreške (Izvor: Vlastita izrada)

Na grafikonu iznad, koji prikazuje i MSE za obuku (`mse`) i validacijski MSE (`val_mse`), pojavljuje se intrigantan uzorak. MSE za obuku, koji predstavlja pogreške u skupu podataka za obuku, dosljedno se smanjuje tijekom epoha. Ovo smanjenje ukazuje na to da ovaj model učinkovito uči iz podataka o treningu, poboljšavajući svoju točnost predviđanja dobi dok nastavlja s treningom.

Međutim, validacijski MSE, koji mjeri izvedbu modela na nevidljivim validacijskim podacima, pokazuje značajne fluktuacije u određenim epohama. Ovi skokovi sugeriraju da se model susreće s izazovima pri predviđanju starosti određenih validacijskih uzoraka. Ti skokovi mogu se pripisati ekstremima ili posebno složenim instancama u validacijskom skupu koje model otežano točno predvidi.



Slika 22. Prikaz povijesti srednje apsolutne pogreške (Izvor: Vlastita izrada)

Slično tome, grafikon Povijest srednje apsolutne pogreške prikazuje MAE za obuku (mae) i MAE za provjeru valjanosti (val_mae). MAE treninga stalno se smanjuje, što ukazuje na poboljšanu točnost podataka o treningu. Nasuprot tome, validacijski MAE predstavlja skokove u određenim epohama, odražavajući ponašanje viđeno u validacijskom MSE-u.

Ove fluktuacije u mjernim podacima za provjeru valjanosti mogu izazvati zabrinutost zbog potencijalne pretreniranost, gdje model postaje previše specijaliziran za prilagođavanje nijansi podataka o obuci.

Kako bismo pružili kvantitativnu procjenu izvedbe modela, pokreće se sljedeći kod:

```
mse, mae = model.evaluate(X_test,y_test,verbose=0)
print('Srednja kvadratna pogreška: {}'.format(mse))
print('Srednja apsolutna pogreška: {}'.format(mae))
```

I dobivamo sljedeće rezultate:

```
Srednja kvadratna pogreška: 110.99331665039062
Srednja apsolutna pogreška: 7.570628643035889
```

Srednja kvadratna pogreška (MSE) je 110,9933, što predstavlja izvedbu modela u minimiziranju kvadratne pogreške na prethodno nevidljivim testnim podacima. Srednja apsolutna pogreška testa (MAE) je 7,5706, što ukazuje na prosječnu apsolutnu razliku između predviđene i stvarne vrijednosti dobi na testnom skupu podataka.

5.3.5. Testiranje modela

Zatim koristimo biblioteku Matplotlib za prikaz mreže slika zajedno s njihovim predviđenim i stvarnim oznakama. Počinjemo generiranjem popisa, nasumičnim odabirom iz foldera koristeći funkciju `random.sample`.

```
image_dir = 'dataset/images/'
all_files = os.listdir(image_dir)
image_files = [file for file in all_files if file.endswith(('.jpg',
'.png', '.jpeg'))]
selected_images = random.sample(image_files, 16)
plt.figure(figsize=(16,16))
for i, image_file in enumerate(selected_images):
    image_path = os.path.join(image_dir, image_file)
    img_array = convert_image(image_path)
    detected_faces_array = detect_face(img_array)
    for face in detected_faces_array:
        predictions = predict_attributes(face)
        plt.subplot(4,4,(i%25)+1)
        plt.xticks([])
        plt.yticks([])
        plt.grid(False)
```

```
            plt.imshow(img_array)
            plt.xlabel(
                f"Age: {predictions['Age']}   "
                f"Ethnicity: {predictions['Ethnicity']}   "
                f"Gender: {predictions['Gender']}"
            )
    plt.show()
```

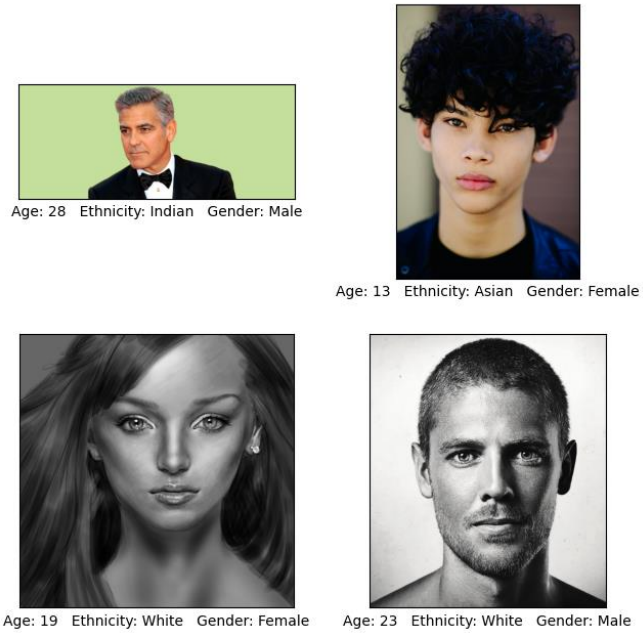
Za početak prvi korak je definiranje rute do foldera gdje se nalaze slike `image_dir`, koje će biti testirane u modelu. Zatim selektiramo samo one koje završavaju s ekstenzijama predviđenim za formate slika. Kada smo definirali skup tada nasumično odaberemo, u ovom slučaju taj boje je postavljen na 16 slika.

Za ispis iteriramo kroz popis od nasumično odabranih slika, gdje svaku fotografiju prosljeđujemo funkciji `predict_attributes` koja je zadužena za izvlačenje klasa. Naposlijetku, predikcije modela dodajemo ispod fotografije.

Kako bi preciznije vidjeli predikciju modela, učitane su fotografije iz 3 različita dataseta. Rezultati za učitane slike iz 3 dataseta, su vidno različiti. Možemo vidjeti kako model za predikciju dobi za prva 2 skupa podataka ne daje najbolje rezultate, te analizom zapažamo da je u nekim slučajevima razlika u predviđenoj i stvarnoj dobi je dosta izražena. Postoji mogućnost da je to izravna posljedica rezultata performansi modela za predikciju dobi. Ranije smo ustanovili kako su validacijski MAE i MSE imali fluktuacije u mjernim podacima za provjeru valjanosti što u konačnici može izazvati pretreniranost. Postoje tu i drugi faktori koji mogu utjecati na ovakav ishod, kao što su rezolucija slike, pozicija lica, sjene, pozadina, kako dataseta korištenog u fazi treninga, tako i fotografija za procjenu.

Iz analize rezultata možemo zaključi kako model najtočnije predviđa klasifikaciju spola, zatim etničku pripadnost, te na kraju, najmanje klasifikaciju dobi. Pored dobre konfiguracije modela za prepoznavanje spola, važno je naglasiti kao je za ovaj ishod model za prepoznavanje spola binarna klasifikacija, što uvelike olakšava predikciju u odnosu na ostala dva modela.

Slika 23. Prikaz klasificiranih slika iz skupa podataka 1 (Izvor: Vlastita izrada)



Slika 24. Prikaz klasificiranih slika iz skupa podataka 2 (Izvor: Vlastita izrada)

Slika 25. Prikaz klasificiranih slika iz skupa podataka 3 (Izvor: Vlastita izrada)

5.3.6. Spremanje modela

Naposljetku koristimo biblioteku TensorFlow za spremanje modela strojnog učenja i njegovih težina. Počinjemo pozivanjem metode spremanja na objektu modela, prosljeđujući naziv datoteke 'model.keras', odnosno s prefiksom za svaki od modela. Ovo će spremiti arhitekturu, težine i konfiguraciju modela u jednu datoteku, koja se kasnije može koristiti za učitavanje i ponovnu upotrebu modela.

Spremanje modela i njegovih težina na disk korisno je za razne svrhe, kao što je postavljanje modela u produkciju, dijeljenje modela s drugima ili nastavak obuke modela

kasnije. Spremanjem modela i njegovih težina na disk moguće je jednostavno učitati i ponovno koristiti model u različitim kontekstima i postavkama.

5.3.7. Aplikacija za klasifikaciju fotografija osoba

Na samome kraju, nakon kreiranja i treniranja modela za predikciju spola, etničke pripadnosti i dobi, te prikazivanja performansi i testiranja modela, napravljena je jednostavna aplikacija koja iskorištava prethodno istrenirane modele. Aplikacija funkcionira na način da je potrebno učitati fotografiju, zatim se ta fotografija, proslijedi u funkciju koja provlači tu fotografiju kroz sva tri modela. Na kraju, ta funkcija vraća predikciju za 3 klasifikacije, koje se ispisuju na ekranu ispod učitane fotografije.

U nastavku ovog odlomka će se detaljnije prikazati i objasniti sama implementacija aplikacije i njezine funkcije koje su se koristile i u prethodnom odlomku, za testiranje modela.

```
import tensorflow as tf
import numpy as np
import cv2
import tkinter as tk

from tkinter import filedialog
from PIL import Image, ImageTk
```

Kod počinje uvozom osnovnih Python biblioteka, uključujući TensorFlow za duboko učenje, NumPy za numeričke operacije, OpenCV (cv2) za zadatke računalnog vida i tkinter za izgradnju grafičkog korisničkog sučelja. Osim toga, uvozi module iz Python Imaging Library (PIL) za manipulaciju slikama.

```
gender_model =
tf.keras.models.load_model('model/gender/gender_model.keras',
compile=False)
ethnicity_model =
tf.keras.models.load_model('model/ethnicity/ethnicity_model.keras',
compile=False)
age_model = tf.keras.models.load_model('model/age/age_model.keras',
compile=False)
```

Ovaj odjeljak učitava unaprijed obučene modele dubokog učenja za predviđanje spola, etničke pripadnosti i dobi. Ovi su modeli prethodno uvježbani i spremljeni u navedene direktorije. Argument `compile=False` koristi se za sprječavanje kompilacije modela jer nije potreban za zaključivanje.

```
gender_labels = {0: "Male", 1: "Female"}
ethnicity_labels = {0: "White", 1: "Black", 2: "Asian", 3: "Indian", 4:
"Hispanic"}
```

Namjera ovih dviju varijabli, `gender_labels` i `ethnicity_labels`, je ta da prevedu predviđanja koja generiraju modeli u njihove odgovarajuće oznake čitljive ljudima. Ove oznake uključuju rodne kategorije (muški i ženski) i etničke kategorije (bijelci, crnci, Azijci, Indijanci, Latinoamerikanci).

```
def convert_image(image_path):
    img = Image.open(image_path)
    img_array = np.array(img)

    return img_array
```

Funkcija `convert_image` pretvara sliku, učitane s putanje, u format prikladan za daljnju obradu. Koristi Python Imaging Library (PIL) za otvaranje i čitanje slike, zatim je pretvara u NumPy polje.

```
def detect_face(image_array):
  img = cv2.cvtColor(image_array, cv2.COLOR_RGB2GRAY)
  face_cascade =
cv2.CascadeClassifier('./haarcascades/haarcascade_frontalface_alt.xml')
  faces = face_cascade.detectMultiScale(img, 1.3, 4)
  all_faces = []
  if len(faces) > 0:
    for (x, y, w, h) in faces:
        face = img[y:y + h, x:x + w]
        all_faces.append(face)

  return all_faces
```

`detect_face` je funkcija odgovorna za otkrivanje lica unutar ulazne slike pomoću biblioteke OpenCV s Haarovim kaskadama. Započinje pretvaranjem ulazne slike, što je NumPy niz koji predstavlja RGB sliku, u sive tonove, a zatim nastavlja s otkrivanjem lica s određenim parametrima. Ishod je popis izrezanih slika lica, ako se bilo koje lice otkrije u unosu.

```
def predict_attributes(image_array):
    img = Image.fromarray(image_array)
    img = img.resize((48, 48))
    img = img.convert('L')
    img_array = np.array(img)
```

```
    img_array = img_array / 255.0
    img_array = img_array.reshape((1, 48, 48, 1))

    gender_prediction = gender_model.predict(img_array)
    predicted_gender =
gender_labels[int(np.round(gender_prediction[0][0]))]

    ethnicity_prediction = ethnicity_model.predict(img_array)
    predicted_ethnicity =
ethnicity_labels[np.argmax(ethnicity_prediction)]

    age_prediction = age_model.predict(img_array)
    predicted_age = int(np.round(age_prediction[0][0]))

    return {
        "Age": predicted_age,
        "Ethnicity": predicted_ethnicity,
        "Gender": predicted_gender
    }
```

Funkcija `predict_attributes` ima središnju ulogu u predviđanju atributa. Prihvaća NumPy niz, koji obično predstavlja detektirano lice, kao ulaz. Funkcija zatim nastavlja s pretprocesiranjem slike, što uključuje promjenu veličine, pretvorbu sivih tonova i skaliranje vrijednosti piksela. Nakon predobrade, koristi unaprijed obučene modele dubokog učenja za predviđanje atributa spola, etničke pripadnosti i dobi iz obrađene slike.

```
def predict_image_attributes(input_image_path):
    img_array = convert_image(input_image_path)
    detected_faces_array = detect_face(img_array)
    predictions = predict_attributes(detected_faces_array[0])
    return predictions
```

Upravljajući cijelim postupkom predviđanja slike, funkcija `predict_image_attributes` započinje pretvaranjem ulazne slike u niz, detekcije lica, a zatim predviđanjem atributa (dob, etnička pripadnost i spol) za svako otkriveno lice pomoću funkcije `predict_attributes` što je u konačnici i izlaz iz funkcije. Kao ulaz se koristi putanja do slike.

```
def display_processed_image(image, predicted_attributes):
    image.thumbnail((300, 300))
    photo = ImageTk.PhotoImage(image)
    preview_label.config(image=photo)
    preview_label.image = photo
    label_text.set(
            f"Age: {predicted_attributes['Age']}   "
            f"Ethnicity: {predicted_attributes['Ethnicity']}   "
            f"Gender: {predicted_attributes['Gender']}"
            )
    remove_button.pack(side=tk.RIGHT)
    resize_window()
    root.geometry("400x400")
```

Pomoću `display_processed_image` funkcije aplikacija prikazuje obrađene slike i predviđene atribute. Mijenja veličinu slika kako bi stale u okvir za prikaz, te ažurira grafičko sučelje s dodanim oznakama predviđenih atributa i otkriva gumb koji korisnicima omogućuje brisanje prikazane slike, kao bi učitali novu sliku.

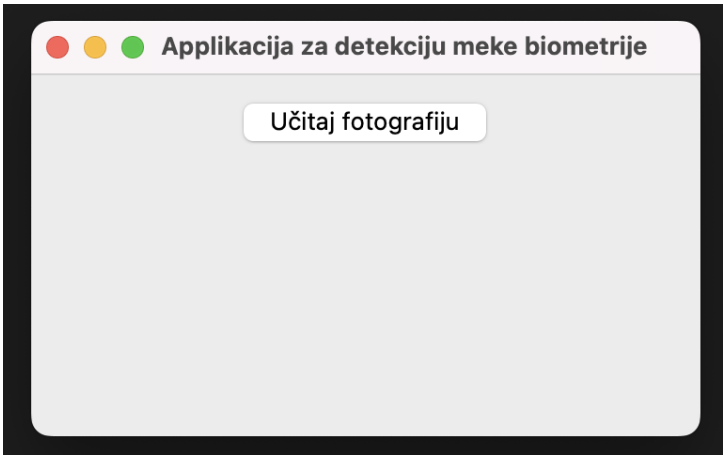
```
def open_image():
    file_path = filedialog.askopenfilename(filetypes=[("Image files",
"*.png *.jpg *.jpeg *.gif *.bmp")])
    if file_path:
        predicted_attributes = predict_image_attributes(file_path)
        image = Image.fromarray(convert_image(file_path))
        display_processed_image(image, predicted_attributes)
```

Funkcija `open_image` je funkcionalnost aplikacije koja omogućuje korisniku da izabere slike iz lokalne pohrane na računalu. Nakon što je odabrana slika, ona pokreće proces predviđanja i zatim prikazuje rezultate unutar sučelja aplikacije.

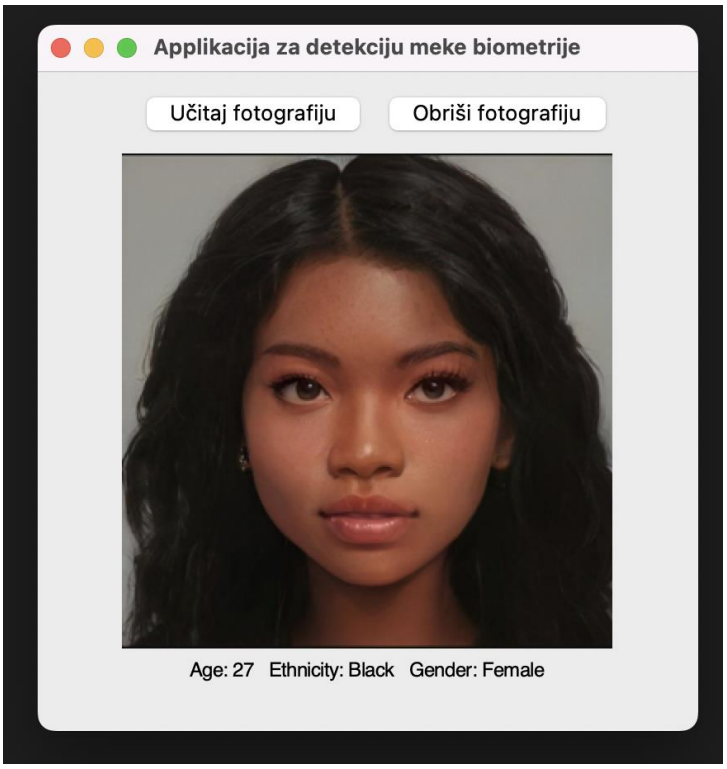
Dodatno aplikacija implementira pomoćne funkcije poput `clear_upload`, koja briše sve prikazane slike i povezane oznake atributa iz sučelja, te skriva gumb 'Obriši fotografiju' nakon izvršenja. S druge strane, pomoćna funkcija, `resize_window`, dinamički prilagođava veličinu prozora aplikacije kako bi se na odgovarajući način prilagodila njezinom sadržaju. Osigurava da sučelje nije pretjerano veliko kada se ne prikazuje slika, održavajući optimalno korisničko iskustvo.

```
root = tk.Tk()
root.title("Applikacija za detekciju meke biometrije")

root.pack_propagate(False)
root.grid_rowconfigure(1, weight=1)
root.grid_columnconfigure(0, weight=1)

button_frame = tk.Frame(root)
button_frame.pack(pady=10)

upload_button = tk.Button(button_frame, text="Učitaj fotografiju",
command=open_image)
upload_button.pack(side=tk.LEFT, padx=10)

remove_button = tk.Button(button_frame, text="Obriši fotografiju",
command=clear_upload)
remove_button.pack_forget()

preview_label = tk.Label(root)
preview_label.pack()

label_text = tk.StringVar()
label_display = tk.Label(root, textvariable=label_text,
font=("Helvetica", 12))
label_display.pack()

root.update_idletasks()
root_width = root.winfo_width()
root_height = root.winfo_height()

min_width = max(preview_label.winfo_width(),
label_display.winfo_width(), button_frame.winfo_width())
min_height = root_height

x = (root.winfo_screenwidth() - min_width) // 2
y = (root.winfo_screenheight() - min_height) // 2
root.geometry(f"{min_width}x{min_height}+{x}+{y}")

root.mainloop()
```

Ovaj blok koda predstavlja grafičko sučelje aplikacije. Stvara Tkinter prozor s gumbima za učitavanje i uklanjanje slike, oznakom za prikaz slike koja se nalazi ispod ova dva gumba i drugom oznakom za prikaz predviđenih atributa ispod slike. Prozor se dinamički dimenzionira kako bi odgovarao tim elementima i automatski se centrira na zaslonu. Zatim se pokreće Tkinter glavna petlja kako bi se omogućila korisnička interakcija i funkcionalnost sučelja. Ovo sučelje omogućuje korisnicima učitavanje slika, pregled predviđanja i brisanje prikazanih slika.

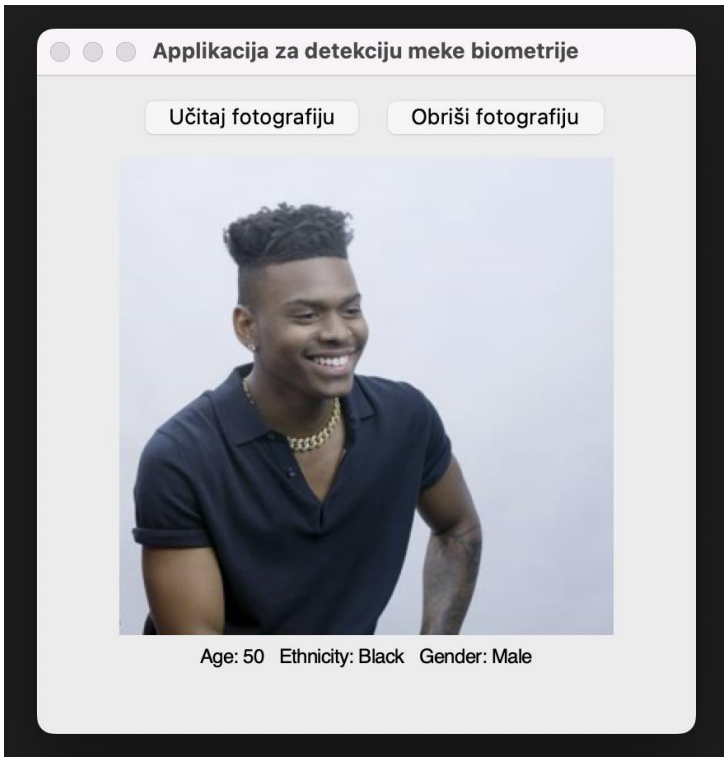


Slika 26. Prikaz aplikacije nakon pokretanja (Izvor: Vlastita izrada)

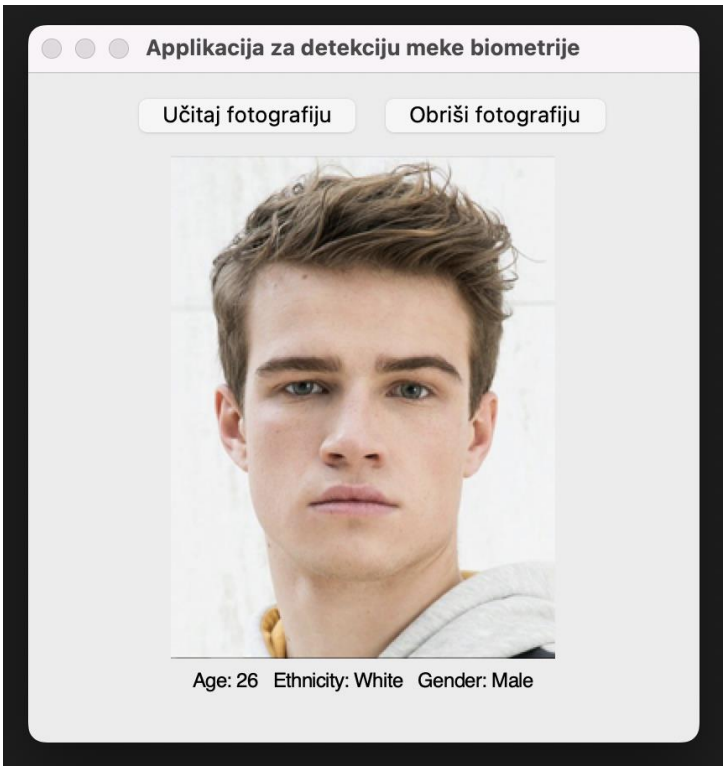


Slika 27. Prikaz aplikacije nakon učitavanja slike 1 (Izvor: Vlastita izrada)

Slika 28. Prikaz aplikacije nakon učitavanja slike 2 (Izvor: Vlastita izrada)



Slika 29. Prikaz aplikacije nakon učitavanja slike 3 (Izvor: Vlastita izrada)



Slika 30. Prikaz aplikacije nakon učitavanja slike 4 (Izvor: Vlastita izrada)

Na slici 26, prikazan je izgled aplikacije nakon što pokrenemo skriptu `python3 ./application.py`. Klikom na gumb 'Učitaj fotografiju', aplikacija na otvara novi prozor gdje odabiremo sliku s lokalne pohrane računala. Odabirom na željenu sliku, aplikacija tu sliku provlači kroz prethodno definirane metode, i na kraju ispod slike dobivamo rezultate predikcije modela spola, etničke pripadnosti i dobi (slika 27, slika 28, slika 29 i slika 30).

6. Zaključak

U ovom radu ispitana je upotreba meke biometrije u strojnom učenju te potencijalne prednosti i izazove korištenja tih karakteristika u svrhu klasifikacije. Vidjeli smo da meka biometrija može pružiti pouzdaniji način identifikacije i razlikovanja pojedinaca u usporedbi s tradicionalnim metodama, a može se izdvojiti iz raznih izvora kao što su slike, video i audio.

Međutim, također je obrađena mogućnost pristranosti u podacima o obuci i potrebi da se osigura njihova raznolikost i reprezentativnost populacije koja se klasificira. Uz to, razmotrena je složenost izdvajanja i preciznog mjerenja mekih biometrijskih karakteristika, što može zahtijevati specijalizirane algoritme i tehnike.

Praktični dio ovog rada obuhvatio je razvoj, obuku i procjenu modela predviđanja spola, etničke pripadnosti i dobi. Proces je započelo preciznom pripremom podataka i pretprocesiranjem, pri čemu je priznato da skup podataka korišten u ovoj studiji ima određena ograničenja, posebice smanjenu kvalitetu slike. Ovo inherentno ograničenje pridonijelo je nekim od uočenih rezultata izvedbe i naglašava važnost korištenja visokokvalitetnih skupova slika u budućim razvojima.

Štoviše, tokom razvoja postalo je očito da pozicioniranje i poravnavanje slika lica igra ključnu ulogu u ukupnoj kvaliteti i točnosti modela predviđanja. Utjecaj orijentacije i pozicioniranja slike na izvedbu modela trebalo bi dodatno istražiti i razmotriti u narednim studijama, što bi potencijalno moglo dovesti do poboljšanja u robusnosti modela.

Konfiguracija modela, koja obuhvaća izbor arhitekture, podešavanje hiperparametara i tehnike optimizacije, pojavila se kao još jedan čimbenik u određivanju prediktivne sposobnosti modela. Buduća istraživanja trebala bi marljivo eksperimentirati s različitim konfiguracijama modela kako bi se identificirali najučinkovitiji pristupi, čime bi se povećala točnost modela i sposobnost generalizacije.

Gledajući unaprijed, javlja se veliki broj mogućnosti za daljnje usavršavanje i poboljšanje modela mekog biometrijskog prepoznavanja. Strategije kao što su povećanje skupa podataka i njezine kvalitete, napredne funkcije gubitka i učenje ansambla imaju potencijal za jačanje točnosti predviđanja. Nadalje, stalni napori u prikupljanju podataka trebali bi biti dopunjeni etičkim razmatranjima kako bi se ublažile pristranosti u predviđanjima i podržali standardi pravednosti.

Zaključno, upotreba meke biometrije u strojnom učenju ima velike mogućnosti za poboljšanje točnosti i pouzdanosti zadataka klasifikacije. Pažljivim razmatranjem potencijalnih

prednosti i izazova korištenja ovih karakteristika, moguće je iskoristiti snagu strojnog učenja za poboljšanje našeg svakodnevnog života na bezbroj načina.

7. Popis literature

Maiorana, D. M. (2018). *Soft Biometrics: State of the art and future trends.* IEEE Transactions on Information Forensics and Security.

Zhang, Y. C. (2019). *Soft biometric trait recognition: A review.* IEEE Access.

Jain, J. R. (2015). *Handbook of Biometric Anti-Spoofing.* Springer.

Ko, K. K. (2015). A survey of soft biometric traits and their applications. *Patter Reognition Letters*, str. 1-12.

Y. LeCun, Y. B. (2015). Deep Learning. *Nature*.

Mitchell, T. M. (1997). *Machine Learning.* McGraw-Hill Education.

Cristianini, J. S.-T. (2004). *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods.* Cambridge University Press.

Motoda, X. L. (1998). *Feature Selection for Knowledge DIscovery and Data Mining.* Springer.

Barto, R. S. (2018). *Reinforcement Learning.* MIT Press.

Murphy, K. P. (2012). *Machine Learning: A Probabilistic Perspective.* MIT Press.

Breiman, F. (2001). Random Forests. *Machine Learning*.

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural Networks*.

Haykin, S. (1999). *Neural Networks: A Comprehensive Foundation.* Pearson.

Hinton, G. E. (2014). Deep learning. *Scholarpedia*.

Ba, D. P. (2014). Adam: A method for stochastic optimization. *arXiV*.

Zisserman, K. S. (2014). Very deep convolutional networks for stochastic optimization. *arXiV*.

Fukushima, K. (1980). Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*.

Bengio, X. G. (2015). Understanding the difficulty of training deep feedforward neural networks. *European Conference on Computer Vision.*

Fei-Fei, A. K. (2015). Deep Visual-Semantic Alignments for Generating Image Descriptions. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.*

Osindero, M. M. (n.d.). Conditional Generative Adversarial Nets. *arXiv*.

A. Krizhevsky, I. S. (2012). ImageNet Classification with Deep Convolutional Neural Networks. *Proceedings of the 25th International Conference on Neural Information Processing Systems.*

H. Cheng, J. K. (2019). Self-Attention Pooling for Fine-Grained Image Classification. *Proceedings of the IEEE International Conference on Computer Vision.*

I. Goodfellow, Y. B. (2016). *Deep Learning.* MIT Press.

K. He, X. Z. (2015). Delving deep into rectifiers. *International Conference on Computer Vision.*

T.-Y. Lin, M. M. (2014). Microsoft COCO: Common Objects in Context. *European Conference on Computer Vision.*

L. Fei-Fei, R. F. (2004). Learning Generative Visual Models from Few Training Examples: An Incremental Bayesian Approach Tested on 101 Object Categories. *California Institute of Technology*.

K. Cho, B. v. (2014). Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. *Conference on Empirical Methods in Natural Language Processin.*

J. Redmon, S. D. (2016). You Only Look Once: Unified, Real-Time Object Detection. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.*

J. Deng, W. D.-J.-F. (2009). ImageNet: A Large-Scale Hierarchical Image Database. *IEEE Conference on Computer Vision and Pattern Recognition.*

Y. LeCun, B. B. (n.d.). Backpropagation Applied to Hand.

D. Amodei, R. A. (2016). Deep Speech 2: End-to-End Speech Recognition in English and Mandarin. *International Conference on Machine Learning.*

Y. LeCun, L. B. (n.d.). Gradient-based learning applied to document recognition. *Proceedings of the IEEE.*

Singh, K. K. (2017). A comprehensive review of convolutional neural network. *Advances in Natural and Applied Sciences*.

Szegedy, S. I. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiV*.

N. Srivastava, G. H. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*.

Klambauer, G. U. (2017). Self-normalizing neural networks. *IEEE Transactions on Neural Networks and Learning Systems*.

Nair, V. &. (2010). Rectified linear units improve restricted boltzmann machines. *International Conference on Machine Learning.*

X. Zhang, Y. L. (1998). Efficient backprop. *Neural Networks: Tricks of the Trade*.

Howard, A. G. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications.

Chollet, F. (2017). Xception: Deep learning with depthwise separable convolutions. *IEEE conference on computer vision and pattern recognition.*

Howard, A. G. (2019). Searching for mobilenetv3.

Chen, L. C. (2014). Semantic image segmentation with deep convolutional nets and fully connected crfs.

Ren, S. H. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. In Advances in neural information processing systems.

Long, J. S. (2015). Fully convolutional networks for semantic segmentation. *In Proceedings of the IEEE conference on computer vision and pattern recognition.*

Sandler, M. H. (2018). Mobilenetv2: Inverted residuals and linear bottlenecks. *In Proceedings of the IEEE conference on computer vision and pattern recognition.*

Rossum, G. v. (2003). *An introduction to Python (p. 115).* Bristol: Network Theory Ltd.

Oliphant, T. E. (2006). *Guide to numpy.* USA: Trelgol Publishing.

Gulli, A., & Pal, S. (2017). *Deep learning with Keras.* Packt Publishing Ltd.

Shukla, N., & Fricklas, K. (2018). *Machine Learning with TensorFlow.* Greenwich: Manning.

Tosi, S. (2009). *Matplotlib for Python developers.* Packt Publishing Ltd.

Hackeling, G. (2017). *Mastering Machine Learning with scikit-learn.* Packt Publishing Ltd.

Jupyter. (24. 7 2023). *Jupyter*. Dohvaćeno iz Jupyter: https://jupyter.org/about

Driscoll, M. (2019). *Jupyter Notebook: An Introduction*. Dohvaćeno iz Real Python: https://realpython.com/jupyter-notebook-introduction/

Feng, J., & Lu, S. (6 2019). Performance Analysis of Various Activation Functions in Artificial Neural Networks. *Journal of Physics: Conference Series, 1237*, 1-5.

Arora, N. (2021). *AGE, GENDER AND ETHNICITY (FACE DATA) CSV*. Dohvaćeno iz Kaggle: https://www.kaggle.com/datasets/nipunarora8/age-gender-and-ethnicity-face-data-csv?resource=download

Bock, S., Weiß, M. G., & Goppold, J. (3 2018). *An improvement of the convergence proof of the ADAM-Optimizer.* Dohvaćeno iz Research Gate: https://www.researchgate.net/publication/324808725_An_improvement_of_the_convergence_proof_of_the_ADAM-Optimizer

Brooks, H., & Tucker, N. (2014). Electrospinning Predictions using Artificial Neural Networks. *ResearchGate*.

8. Popis slika

Slika 1: Podjela biometrije (Vlastita izrada prema: https://www.intechopen.com/media/chapter/60675/media/F2.png) 3

Slika 2: Autentifikacija osoba po načinu hoda u Kini (Izvor: https://static.independent.co.uk/s3fs-public/thumbnails/image/2018/11/07/14/china-surveillance-gait-recognition.jpg?quality=75&width=1200&auto=webp) 4

Slika 3: Razlika između strojnog i dubokog učenja (Vlastita izrada prema: https://i0.wp.com/semiengineering.com/wp-content/uploads/2018/01/MLvsDL.png?ssl=1) ..13

Slika 4: Klizni prozor CNN-a (Izvor: https://i.imgur.com/LueNK6b.gif) 14

Slika 5: Arhitektura konvolucijskih mreža (Vlastita izrada prema: https://miro.medium.com/max/1400/1*uAeANQIOQPqWZnnuH-VEyw.jpeg) 16

Slika 6: Primjer konvolucije ulazne slike (Vlastita izrada prema: https://i.imgur.com/IYO9lqp.png) 18

Slika 7: Arhitektura gustog sloja (Vlastita izrada prema: https://www.researchgate.net/profile/Hadley-Brooks/publication/270274130/figure/fig3/AS:667886670594050@1536247999230/Architecture-of-a-multilayer-neural-network-with-one-hidden-layer-The-input-layer.png) 20

Slika 8: Neuronska mreža s dropout slojem (Vlastita izrada prema: https://miro.medium.com/max/1044/1*iWQzxhVlvadk6VAJjsgXgg.png) 21

Slika 9: Odnos različitih funkcija optimizacije (Izvor: https://www.researchgate.net/publication/324808725_An_improvement_of_the_convergence_proof_of_the_ADAM-Optimizer) 23

Slika 10: Formula izračuna funkcije gubitka (Izvor: https://medium.com/deep-learning-demystified/loss-functions-explained-3098e8ff2b27) 24

Slika 11: Arhitektura MobileNet modela (Vlastita izrada prema: https://www.mdpi.com/2073-4433/14/2/280) 31

Slika 12. Distribucija spola (Izvor: Vlastita izrada) 32

Slika 13. Dobna distribucija (Izvor: Vlastita izrada) 32

Slika 14. Etnička distribucija (Izvor: Vlastita izrada) 33

Slika 15. Uzorak slika iz dataseta-a (Izvor: Vlastita izrada) 34

Slika 16: Ilustracija python enviroment-a (Izvor: https://www.dataquest.io/wp-content/uploads/2022/01/python-virtual-envs1.webp) 36

Slika 17. Prikaz pada gubitka modela za predikciju spola (Izvor: Vlastita izrada) 47

Slika 18. Prikaz rasta preciznosti modela za predikciju spola (Izvor: Vlastita izrada) 48

Slika 19. Prikaz pada gubitka modela za predikciju etničke pripadnosti (Izvor: Vlastita izrada) 49

Slika 20. Prikaz rasta preciznosti modela za predikciju etničke pripadnosti (Izvor: Vlastita izrada) 50

Slika 21. Prikaz povijesti srednje kvadratne pogreške (Izvor: Vlastita izrada) 51

Slika 22. Prikaz povijesti srednje apsolutne pogreške (Izvor: Vlastita izrada) 51

Slika 23. Prikaz klasificiranih slika iz skupa podataka 1 (Izvor: Vlastita izrada) 54

Slika 24. Prikaz klasificiranih slika iz skupa podataka 2 (Izvor: Vlastita izrada) 54
Slika 25. Prikaz klasificiranih slika iz skupa podataka 3 (Izvor: Vlastita izrada) 55
Slika 26. Prikaz aplikacije nakon pokretanja (Izvor: Vlastita izrada) 61
Slika 27. Prikaz aplikacije nakon učitavanja slike 1 (Izvor: Vlastita izrada) 61
Slika 28. Prikaz aplikacije nakon učitavanja slike 2 (Izvor: Vlastita izrada) 62
Slika 29. Prikaz aplikacije nakon učitavanja slike 3 (Izvor: Vlastita izrada) 62
Slika 30. Prikaz aplikacije nakon učitavanja slike 4 (Izvor: Vlastita izrada) 63

9. Popis tablica

Tablica 1: Keras model zoo tablica 28
Tablica 2. Prvih 5 zapisa iz CSV datoteke 37