Primjena konvolucijskih neuronskih mreža u detekciji deepfake zapisa

Vinko, Patrik

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Patrik Vinko

PRIMJENA KONVOLUCIJSKIH NEURONSKIH MREŽA U DETEKCIJI DEEPFAKE ZAPISA

DIPLOMSKI RAD

Varaždin, 2024.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Patrik Vinko

Matični broj: 0016141446

Studij: Informacijsko i programsko inženjerstvo

PRIMJENA KONVOLUCIJSKIH NEURONSKIH MREŽA U DETEKCIJI DEEPFAKE ZAPISA

DIPLOMSKI RAD

Mentorica:

Izv. prof. dr. sc. Petra Grd

Varaždin, rujan 2024.

Patrik Vinko

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 potvrdio prihvaćanjem odredbi u sustavu FOI-radovi

_____

Sažetak

Ovaj rad se bavi istraživanjem i primjenom konvolucijskih neuronskih mreža (CNN) za detekciju deepfake zapisa. U teorijskom dijelu rada se govori o konvolucijskim neuronskim mrežama, na koji način funkcioniraju i od kojih vrsta slojeva se sastoje. Također će se govoriti o *deepfake* tehnologiji i na koje načine može manipulirati zapisima. Kako je cilj kreirati model CNN-a koji će moći prepoznavati lažne videozapise, opisat će se dva takva modela, VGG16 i MobileNet, koji će se nakon toga i usporediti. Za svaki model je potrebno vidjeti ako ispravno klasificira zapisa, stoga će se govoriti o nekoliko metrika bitnih za evaluaciju modela. Praktični dio rada obuhvaća razvoj aplikacije koja, nakon odabira i pripreme podataka, gradi CNN model na temelju MobileNet arhitekture. Razvoj će se odvijati u Python okruženju, uz korištenje TensorFlow biblioteke, koja je široko primijenjena u području strojnog učenja i umjetne inteligencije. Nakon obrade podataka, model će se trenirati i testirati, a rezultati testiranja bit će vizualizirani pomoću različitih metrika.

Ključne riječi: deepfake, neuronske mreže, manipulacija zapisa, duboko učenje, treniranje modela, dataset, vizualni podaci

Sadržaj

1. Uvod

Tema ovog rada je primjena konvolucijskih neuronskih mreža u detekciji deepfake zapisa, iznimno aktualno područje u računalnom vidu i umjetnoj inteligenciji.

Deepfake tehnologija, koja omogućuje manipulaciju videozapisa i slika kako bi se stvorili lažni, ali naizgled autentični zapisi, predstavlja sve veći izazov u suvremenom svijetu. Naziv „deepfake“ je spoj dvaju pojmova, a to su „deep learning“, duboko učenje koje se definira kao grana strojnog učenja, te „fake“ koja podrazumijeva da su podaci dobiveni tom tehnologijom zapravo lažni, tj. da nisu kredibilni. „Deepfake“ se kao pojam koristi za naziv tehnologije, ali i kao rezultat same tehnologije. Najveća opasnost *deepfake* tehnologije je brz prijenos dezinformacija koje mogu uništiti reputaciju osobe ili kompanije, pokrenuti ciljanu prijevaru, krađu osobnih podataka, manipulirati tržištem dionica itd. Sve je veća potreba za razvojem pouzdanih detektora. Naime, kako potreba za detekcijom *deepfake* zapisa raste, tako i sami *deepfake* zapisi postaju sve sofisticiraniji i teži za prepoznavanje, što zahtijeva stalni napredak u metodama detekcije. [1]

CNN modeli, zahvaljujući svojoj sposobnosti prepoznavanja složenih vizualnih uzoraka, pokazali su se vrlo učinkovitim. Pomoću toga, modeli automatski uče značajke iz slika, traže anomalije u teksturama kože ili pokretima lica što omogućuje uspješnu detekciju *deepfake* zapisa. Proces detekcije se sastoji od nekoliko koraka, počinje od prikupljanja i pripreme podataka te završava treniranjem i evaluacijom kreiranog modela. Konvolucijske neuronske mreže se koriste u raznim industrijama kao što je medicina, marketing, automobilizam i maloprodaja. Bitne su kod pokretanja zadataka s prepoznavanjem slika i računalnim vidom. [2]

2. Konvolucijske neuronske mreže (CNN)

Umjetna neuronska mreža (ANN) je računalni procesni sustav koji je inspiriran načinom djelovanja živčanog sustava. ANN se sastoji od velikog broja međusobno povezanih čvorova koje nazivamo neuronima. Ti čvorovi rade zajedno kako bi kolektivno učili iz dobivenih ulaznih podataka s ciljem optimizacije izlaznih rezultata. Nakon učitavanja nekog ulaznog vektora u ulazni sloj, vektor se distribuira dalje prema skrivenim slojevima. Zadatak skrivenih slojeva je donošenje odluka iz prethodnih slojeva te evaluacija kako će promjena unutar sebe promijeniti konačni izlaz. Spomenuta evaluacija je zapravo proces učenja koji se provodi na više slojeva koji su skriveni jedan na drugom. Ovaj princip rada s više slojeva se naziva duboko učenje. [3]

Postoje dvije glavne paradigme učenja kod procesiranja slika, a to su nadzirano učenje te nenadzirano učenje. Najveća razlika između ova dva pojma je ta što se kod nadziranog učenja, algoritam uči iz skupa podataka za obučavanje tako što iterativno pravi predikcije na temelju podataka i prilagođava se kako bi došao do točnog odgovora. Iako modeli nadziranog učenja obično daju točnije rezultate u usporedbi s modelima nenadziranog učenja, oni zahtijevaju ljudsku intervenciju kako bi podaci bili ispravno označeni. Suprotno tome, nenadzirano se učenje vrši na način da algoritam radi samostalno bez ikakvih instrukcija ili pomoći u svrhu pronalaska određene strukture (uzoraka) među danim podacima. Obje paradigme imaju različite ciljeve i pristupe rješavanju problema. Nadzirano učenje je fokusirano na predviđanje rezultata novih podataka, dok je nenadzirano učenje fokusirano na vađenje korisnih informacija iz velike količine neoznačenih podataka. Algoritam samostalno analizira podatke kako bi identificirao uzorke i varijacije bez prethodnih uputa. Primjer nadziranog učenja bi mogla biti detekcija spama, vremenske prognoze i predviđanje cijena, dok bi primjer nenadziranog bilo detektiranje raznih anomalija u financijskim podacima – algoritam bi mogao prepoznati odstupanje od normalnog uzorka što može ukazivati na pokušaj prijevare. CNN spada pod paradigmu nadziranog učenja. [40]

ANN često djeluje kao jednostavnija verzija CNN-a. Naime, ANN i CNN su slični po tome što se sastoje od neurona koji se optimiziraju prilikom procesa učenja, neuroni će u obje mreže primati podatke te vršiti procese nad njima. Jedina značajnija razlika je ta što se CNN najviše koristi kod prepoznavanja uzoraka u slikovnim zapisima. Samim time, dopušteno je da se u arhitekturu CNN-a nadodaju značajke koje su slikovno kodirane (engl. *image-encoded*) te će se prema tome izvršavati slikovno usmjereni (engl. *image-focused*) zadaci. CNN se sastoji od različitih skrivenih slojeva od kojih svaki sadrži više čvorova (engl. „Nodes“) koji se povezuju unutar sloja, a potom i između slojeva. Svaki čvor ima svoju težinu i prag (engl.

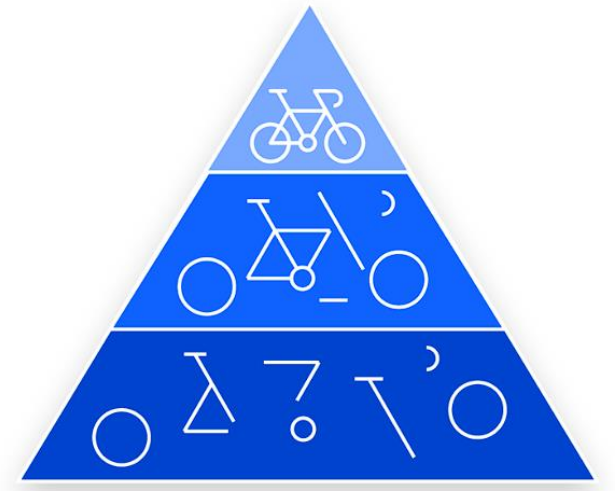
Treshold). Ukoliko se izlaz bilo kojeg čvora nalazi iznad navedene vrijednosti praga, taj čvor se aktivira tako što šalje podatke sljedećem čvoru u mreži. [3]

2.1. Vrste slojeva CNN-a

Konvolucijske neuronske mreže razlikuju 3 glavne vrste slojeva. To su konvolucijski sloj, sloj sažimanje te povezani sloj. Rastom broja slojeva, povećava se i kompleksnost same mreže zbog toga što svaki sloj identificira sve više i više elemenata slike. [2]

Konvolucijski sloj je prvi sloj konvolucijske mreže te je jezgra izgradnje CNN-a. U ovom sloju se događa najviše računanja. Konvolucijskom sloju je potrebno nekoliko elemenata kao što su filteri, *feature map* te naravno ulazni podaci. Ulazni podaci su 2D matrica koja ima svoju dužinu i širinu. Svakoj od tih dimenzija se pridodaje model boja RGB (*Red, Green, Blue*). Filter ili *feature detector* je dvodimenzionalno polje težina koji predstavlja dio slike. Funkcionira na način da se kreće kroz receptivno polje slike te pokušava detektirati ukoliko je neka od značajki (engl. *feature*) prisutna na samoj slici. Taj proces se naziva konvolucijom. Filteri, iako su najčešće 3x3 matrica, mogu varirati u veličini, te se samim time određuje i veličina receptivnog polja slike koje se provjerava. Receptivno polje kreira lakši model za treniranje pošto se radi na manjem dijelu ulaznog podatka. Nakon što se filter primijenio na područje slike, dobiven je skalarni umnožak između svih piksela tog područja i matrice filtera. Skalarni umnožak je tada stavljen u izlazni niz (engl. *output array*). Filter se pomiče za jedan korak te proces ponavlja tako dugo dok filter ne prođe cijelo područje slike. Krajnji proizvod skalarnih umnožaka se naziva *feature map*. [2]

Treba spomenuti da se više filtera primjenjuje paralelno na istu sliku kako bi se pronašle sve značajke ili uzorci. Od jednostavnih značajka poput rubova i tekstura do složenijih značajki poput oblika ili dijelova nekog objekta. Jednostavnije će se značajke pronalaziti na prvim slojevima, dok će se kompleksnost povećavati prolaskom kroz kasnije slojeve. Time je izgrađena hijerarhija ekstrakcije značajki. [4]



Slika 1. Hijerarhija ekstrakcije značajki [2]

Vrijednosti u filteru ostaju iste kod procesa prolaska kroz sliku, ali se mogu mijenjati u fazi treniranja kroz procese kao što su širenje unazad (engl. *backpropagation*) i gradijentni spust (engl. *gradient descent*). [2]

Sloj sažimanja je druga vrsta sloja koji kojem je cilj dodatno smanjiti složenost sustava smanjivanjem broja parametara i računalnu složenost modela. Sloj sažimanja uzima *feature map* kao ulazni podatak te smanjuje njegovu dimenziju. [2]

Postoji nekoliko vrsta slojeva sažimanja. *Max Pooling* je vrsta sloja sažimanja koji bira najveću vrijednost iz područja *feature map*-e koja je prekrivena filterom. Izlaz će tako biti novi *feature map* manje dimenzije s najistaknutijim karakteristikama prethodne *feature map*-e. *Average Pooling* računa prosječnu vrijednost elemenata prisutnih u području *feature map*-e. Za razliku od *Max Pooling* sloja koji daje najistaknutiju vrijednost, *Average (General) Pooling* ima sposobnost obavljanja jednostavnijih matematičkih operacija te tako daje prosječnu vrijednost karakteristika. *Global Pooling* je sloj sažimanja koji smanjuje svaki *feature map* unutar pojedinačnog kanala na jednu vrijednost. Za *Global Max Pooling* to će biti najveća vrijednost, dok će za *Global Average Pooling* to biti prosjek svih vrijednosti. Tako će *feature map* veličine “širina x visina x brojKanala“ biti svedena na “1 x 1 x brojKanala“. [37]

Stochastic Pooling je vrsta sloja sažimanja koja umjesto deterministčkog pristupa, poput prije spomenutih *Max Pooling* i *Average Pooling*, *Stochastic Pooling* odabire vrijednost na temelju distribucije vjerojatnosti izvedene iz vrijednosti unutar *feature map*-a. Npr. ako imamo vrijednosti [1, 2, 3, 4], najprije ćemo izračunati sumu: 1+2+3+4=10. Nakon toga, svakoj vrijednosti ćemo pridružiti vjerojatnost u odnosu na njihov zbroj. Tako bi za vrijednost 1 vjerojatnost bila 1/10 ili 0.1, za 2 bi bila 2/10, za 3 bi bila 3/10 dok bi za 4 bila 4/10. *Stochastic*

Pooling će nasumično odabrati jednu od vrijednosti na temelju njihovih vjerojatnosti. 4 će imati najveću šansu da bude odabran dok će 1 imati najmanju šansu. [38]

Posljednji glavni sloj CNN-a je povezani sloj koji sadrži neurone koji su povezani s neuronima dva susjedna sloja, bez da je povezan s njihovim internim slojevima. Služi u klasifikaciji značajki koje su bili ekstrahirane kroz prethodne slojeve i različite filtere. [2]

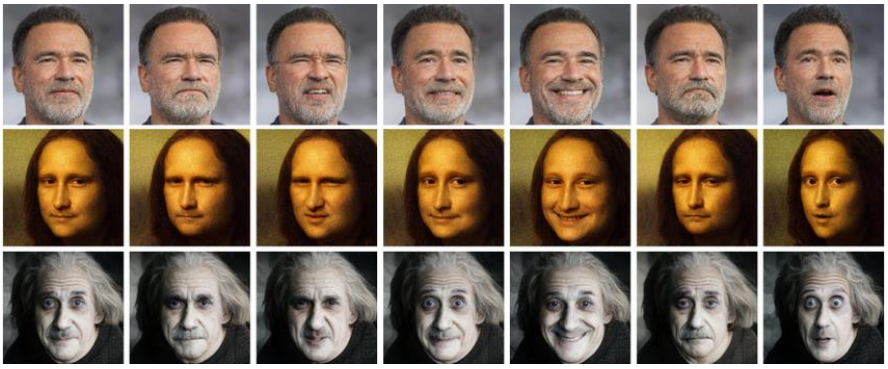
Aktivacijski slojevi služi za dodavanje nelinearnosti modelu neuronske mreže. Nakon što konvolucijski sloj obradi ulazne podatke, rezultat će obično biti linearna kombinacija vrijednosti. Ipak, većina problema nisu linearne prirode te će samim time CNN morati naučiti složenije uzorke. Kako bi to bilo moguće, aktivacijske funkcije poput ReLU, Tanh i Sigmoid se primjenjuju na svaki element izlaza kako bi se uvela nelinearnost. Najpoznatija aktivacijska funkcija ReLU je veoma jednostavna za razumjeti, a glasi: “ReLU(x) = max(0, x)“. Ukoliko je broj pozitivan, taj broj će biti i rezultat funkcije, no ukoliko je broj negativan, rezultat funkcije će biti 0. Na ovaj način će funkcija odrediti ukoliko bi neuron trebao biti aktiviran ili ne. Bez nelinearnosti, mreža bi mogla identificirati samo osnovne uzorke poput ravnih linija i granica. Slike su mnogo složenije od toga jer sadržavaju i nelinearne uzorke poput zakrivljenih linija, oblika, tekstura itd. [39]

Dropout sloj nasumično izbacuje neurone iz neuronske mreže te se samim time mijenja struktura mreže. Ovaj sloj je odličan za rješavanje problema *overfittinga*. Kod *overfittinga*, mreža ne prepoznaje samo uzorke iz podataka, već i statistički šum koji ni na koji način nije povezan s detekcijom *deepfake*-a. Jedan od primjera statističkog šuma bi bili pozadinski podaci. Dolazi do prenaučenosti, tj. neki od neurona se mogu promijeniti na način da poprave greške drugih neurona. Model postaje previše specifičan na skupu podataka za treniranje, a s time gubi sposobnost pravilnog predviđanja na novim i nepoznatim zadacima. Koristeći Dropout, sprječavamo da određeni neuroni popravljaju greške drugih neurona tako što te neurone isključimo, čime se sprječava međusobna prilagodba. Prisutnost svakog neurona u svakoj iteraciji je nepouzdana. [35]

3. Deepfake

Informacija je skup podataka koji omogućava donošenje odluka, glavni je element komunikacije te je temelj obrazovanja. Važnost informacije je neizmjerna te se postavlja pitanje, što ako ono što čujemo i vidimo djeluje kao stvarna informacija, je zapravo produkt složene tehnologije zvane *deepfake* koja ruši integritet podataka, narušava odluke glasača na izborima, kreira lažnu propagandu, uništava karijere i puno više. *Deepfake*, iako je klasificirana kao novija tehnologija, eksponencijalno je napredovala te ju je sve teže detektirati.

Lažne vijesti su postale veliki problem u društvu te se njihov fiktivni način reprezentacije vijesti generira u svrhu kako bi obmanuo javnost. Propagacija lažnih vijesti se dešava brzo zbog popularnosti društvenih mreža kao što su: YouTube, TikTok i Meta (Facebook). Najnoviji tehnološki napretci, kao što su AI (engl. *artificial intelligence*) potpomažu u napretku *deepfake*-a. Tehnologija može generirati humoristične, pornografske i političke videozapise u kojima osoba može govoriti bilo što, najčešće bez njihovog dopuštenja. U današnje vrijeme, gotovo bilo koja osoba koja posjeduje računalo, može kreirati svoj *deepfake* video pomoću gotovih aplikacija koji neće biti detektiran od strane promatrača. Kroz vrijeme, *deepfake* tehnologija postaje sve jednostavnija za korištenje i teža za otkrivanje. [5]



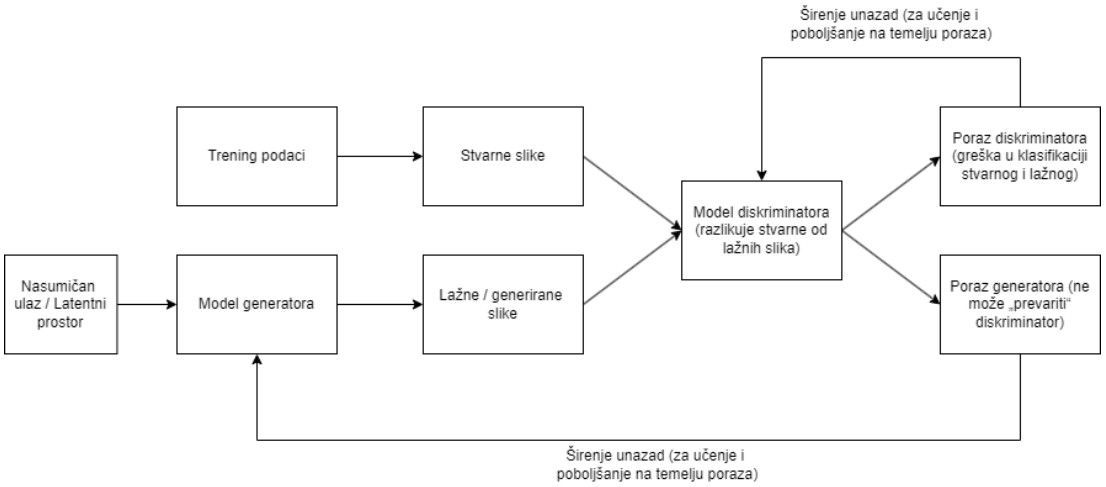
Slika 2. Primjer slika kreiranih deepfake-om [6]

3.1. Tehnologija iza deepfake-a

3.1.1. Generativne kontradiktorne mreže (GAN)

Generativne kontradiktorne mreže (engl. Generative Adversarial Network) poznatija kao GAN, je korištena kod *deepfake* tehnologije za manipuliranje svim vrstama medija, od fotografija i tekstova pa sve do videa. GAN je inovacija u strojnom učenju koja istovremeno trenira dva modela neuronskih mreža, a to su model generatora i model diskriminatora. Sve započinje kod generatora koji nasumično izvlači podatke iz latentnog prostora te pokušava proizvesti lažne slike s ciljem oponašanja stvarnih slika iz skupa podataka za treniranje. Nakon

toga, diskriminator će pokušati razlikovati lažne slike od onih pravih. Kako su oba modela kompetitivna, poboljšanja se provode istovremeno, pri čemu će se generator poboljšavati do onog trenutka kada diskriminator više neće moći raspoznati lažnu sliku od one prave. [7]



Slika 3. GAN arhitektura [7]

Na slici 3. je prikazana Arhitektura GAN-a. Diskriminator je priskrbljen stvarnim slikama iz seta za treniranje, kao i lažnim slikama koje je generirao model generatora. Nakon toga, zadaća je diskriminatora da otkrije koja slike je lažna, a koja prava. Ukoliko diskriminator točno odredi autentičnost slika, to se podrazumijeva da je gubitnik generator te pomoću procesa širenja unazad, dolazi se do točke procesa u kojoj generator generira novu sliku kojom će pokušati zavarati diskriminatora. S druge strane, ako diskriminator ne detektira korektno autentičnost slika, podrazumijeva se da je diskriminator gubitnik te se pomoću procesa širenja unazad dolazi do točke procesa u kojoj generator pokušava shvatiti zašto je slika lažna, odnosno autentična te radi na poboljšanju. [7]

3.1.2. Varijacijski autoenkoder (VAE)

Autoenkoderi su generativni modeli korišteni u nenadziranom učenju. Njihovi modeli sadrže komponente zvane enkoder i dekoder. [8]

Enkoder se sastoji od niza slojeva neuronskih mreža koji iz zapisa vade određene značajke te ih stavljaju u latentni prostor. Dimenzionalnost podataka se smanjuje kako putuju kroz slojeve neuronske mreže. Enkoder te podatke sažima kako bi se mogli uklopiti u niskodimenzionalni latentni prostor. Dekoder se također sastoji od niza slojeva neuronske mreže koji pokušavaju rekreirati izvornu sliku iz latentnog prostora niske dimenzionalnosti. [8].

Suprotno od enkodera, podacima se povećava dimenzionalnost. [9]

Varijacijski autoenkoderi (VAE) su tipovi autoenkodera koji su namjenjeni da bi se nadvladala neka ograničenja tradicionalnih autoenkodera. Kod VAE-ova, umjesto da enkoder

samo pretvara ulazne podatke u jednu točku u latentnom prostoru, on zapravo stvara raspodjelu vjerojatnosti unutar tog prostora. Zatim, dekoder uzima uzorke iz te raspodjele kako bi generirao nove podatke. Ovaj pristup omogućava VAE-ovima da razviju bolje strukturiran i kontinuiran latentni prostor, što je od velike pomoći za generiranje novih podataka. [9]

Kada se mreža izloži velikom broju slika, ona uči prepoznavati ključne značajke na temelju vjerojatnosti, umjesto da se oslanja na jednostavne deterministične metode. Ovo je važno jer omogućava mreži da generalizira naučeno znanje i olakšava, na primjer, kombiniranje značajki s različitih lica. U ranim danima softvera za zamjenu lica, VAE-ovi su bili jezgra tih aplikacija. Međutim, rezultati koje su proizvodili ponekad su bili previše generički i njihova krivotvorenost je bila lako vidljiva. [9]

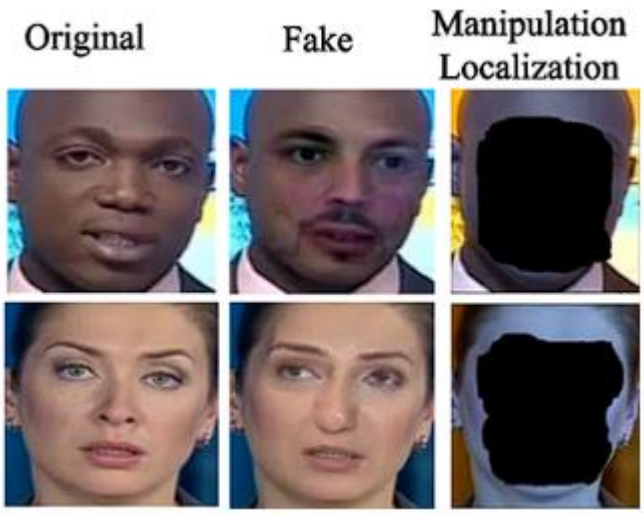
3.2. Metode manipulacije zapisima

3.2.1. Zamjena identiteta

Zamjena identiteta (engl. *identity swap*) uključuje zamjenu lica jedne osobe s licom neke druge osobe. Proces zamjene lica uključuje tri koraka: detekciju lica u izvornoj i ciljnoj snimci, zamjenu ključnih značajki lica kao što su oči, nos i usta te prilagođavanje boje i osvjetljenja kako bi tragovi procesa zamjene ostali neprimjetni. [10]

U zamjenu identiteta spadaju sljedeće metode:

- FaceSwap
- DeepFakes
- FaceShifter



Slika 4.Primjer manipulacije zamjenom identiteta [11]

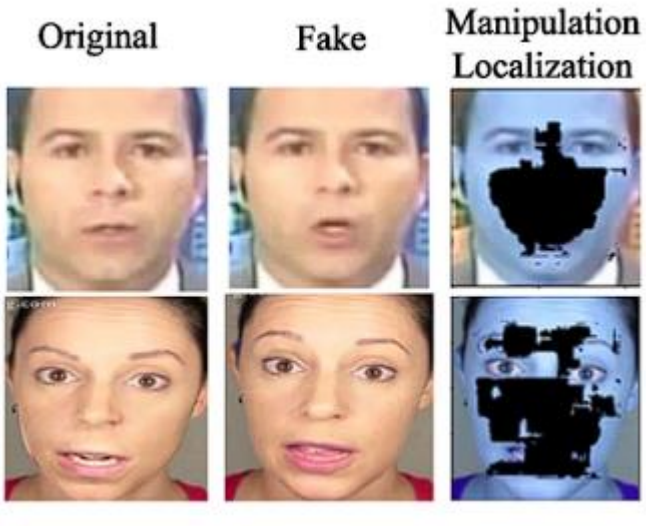
Na slici 4 je prikazan primjer manipulacije zapisa zamjenom identiteta. Prikazan je originalan i krivotvoren zapis te maska manipulacije. Vidljivo je kako je maska primijenjena na cijeloj površini lica, odnosno, zahvaća svaki element lica. Može se reći da je lice prilijepljeno s jedne slike na drugu. Manipulacija zapisa zamjenom identiteta često dovodi do zapisa čije su neusklađenosti veoma uočljive.

3.2.2. Zamjena izraza lica

Deepfake re-enactment, ili zamjena izraza lica, omogućuje prenošenje izraza i pokreta s jednog lica na drugo, dok se identitet ciljanog lica zadržava. Ova tehnologija, koja se može koristiti za stvaranje uvjerljivih lažnih videozapisa, može imati ozbiljne posljedice, pogotovo kada se koristi za manipulaciju informacijama ili narušavanje nečije reputacije. [10]

U zamjenu identiteta spadaju sljedeće metode:

- Neural Textures
- Face2Face



Slika 5. Primjer manipulacije zamjenom izraza lica [11]

Na slici 5 je prikazana manipulacija zapisa zamjenom izraza lica. Najprije je prikazan original, potom krivotvoreni zapis te na kraju maska manipulacija. Vidljivo je da maska ne pokriva cijelu površinu licu kao kod manipulacije zamjenom identiteta, već se primjenjuje na one dijelove lica koji se miču kod izražavanja licem. U slučaju na slici, to su obrve, usta, nos te obrazi. Manipulacija zamjenom izraza lica su kvalitetnije izrađeni te samim time, teže uočljivi krivotvoreni zapisi nego što je to slučaj kod manipulacije zamjene identiteta.

3.3. Prednosti korištenja deepfake-a

Osim brojnih prijetnji, *deepfake* ima široku paletu korisnih primjena u raznim industrijama. U filmskoj industriji, tehnologija omogućava kreiranje digitalnih glasova za glumce koji su izgubili glas zbog bolesti ili za dorade filmskog sadržaja bez potrebe ponovnog snimanja scena. Također, rekreira klasične scene i stvara nove filmove s davno preminulim glumcima, koristi specijalne efekte i napredne obrade lica u postprodukciji te poboljšava amaterske videozapise kako bi izgledali profesionalnije. Osim toga, u obrazovanju se može koristiti u raznim kampanjama koje trebaju doprijeti do publike na globalnoj razini. Naime, *deepfake* omogućuje savršeno prevođenje govora, istovremeno mijenjanje pokreta lica, tijela i usta kako bi se kreirali što stvarniji reklamni sadržaji. U poslovanju, *deepfake* tehnologija transformira e-trgovinu tako što npr. kreira razne virtualne modele s različitom visinom, težinom i bojom kože te prikazuju kako bi izgledali u određenim modnim komadima za pomoć kupcima kod odabira proizvoda. [5]

4. Detekcija *deepfake* zapisa

Deepfake koristi specijalizirane tehnike koje modificiraju određena područja na licu koja se koriste kao baza za superpoziciju (engl. *superimposition*) koja se može definirati kao dodavanje uzoraka ili karakteristika jednog sustava na drugi sustav kako bi se kreirao novi sustav s karakteristikama prethodna dva. Algoritam za kreiranje *deepfake*-ova ne radi savršeno, te će u svoje vrijeme ostaviti neke nepravilnosti kod procesa uređivanja. Neusklađenosti u pitanju su nedosljednost pokreta usana i očiju, razlika u osvjetljenju, promjena tokom procesa kompresija i još mnogo toga. Te neusklađenosti se mogu posebno istaknuti kod treniranja modela za detekciju *deepfake* videa. Konvolucijske neuronske mreže su pokazale veliku sposobnost i skalabilnost za primjene u obradi slike i videa, dok se drugi alati nadziranog učenja mogu koristiti kod konačne klasifikacije kako bi se generirali precizniji modeli za detekciju. [12]

4.1. Prijenosno učenje

Prijenosno učenje se može implementirati za detekciju *deepfake* zapisa. Ono koristi unaprijed modificirane težine neuronske mreže kako bi se sama mreža prilagodila za trening na različitim skupovima podataka te za specifične primjene. Postoji nekolicina unaprijed uvježbanih modela kao što su VGG Net, Xception, Inception, MobileNet i ResNet. [12]

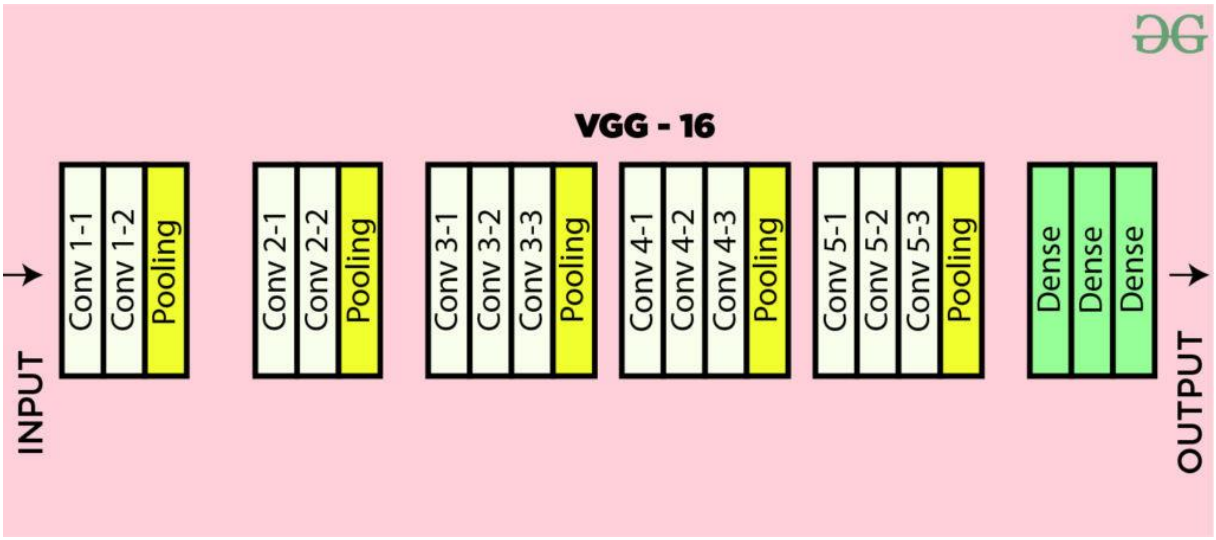
Prijenosno učenje ima brojne prednosti kao što su efektivniji rad s limitiranim podacima, poboljšane performanse, vremenska i troškovna učinkovitost te prilagodljivost. Prijenosno učenje se sastoji od nekoliko koraka. Prvi korak je treniranje modela na specifičnom zadatku koristeći veliki set podataka. Često treniran na opsežnim skupovima podataka, ovaj model identificira opće značajke i uzorke relevantne za brojne slične zadatke. Drugi korak je kreiranje osnovnog (baznog) modela. Sastoji se od slojeva koji koriste ulazne podatke za učenje hijerarhijskih prikaza značajki. U unaprijed treniranom modelu, traži se skup slojeva koji sadrže osnovne značajke bitne za novi zadatak, kao i za prethodni. Posljednji korak je korištenje skupa podataka iz novog zadatka za ponovno treniranje odabranih slojeva. Ovaj postupak je definiran kao podešavanje. Cilj je sačuvati znanje stečeno tijekom prethodnog treniranja, dok se modelu omogućuje prilagodba njegovih parametara kako bi bolje odgovarao zahtjevima novog zadatka. U prijenosnom učenju, postoje dvije glavne komponente: fiksni slojevi te promjenjivi slojevi. Fiksni slojevi su slojevi čiji su parametri nepromijenjeni nakon prvog učenja te se njihovo znanje najčešće koristi kod ekstrakcije značajki i uzoraka iz ulaznog seta

podataka. Promjenjivi slojevi su pak s druge strane oni slojevi koji su ponovno trenirani i modificirani kako bi se prilagodili novom zadatku. [13]

4.2. Modeli

4.2.1. VGG16

Na slici 6, prikazana je arhitektura jednog od poznatijih modela konvolucijske neuronske mreže koji je uvježban uz pomoć prijenosnog učenja. VGG je uveden od strane Visual Geometry Group-e sa sveučilišta Oxford. Model je karakteriziran po jednostavnosti te ga je iz tog razloga lako naučiti, razumjeti i implementirati. [14]



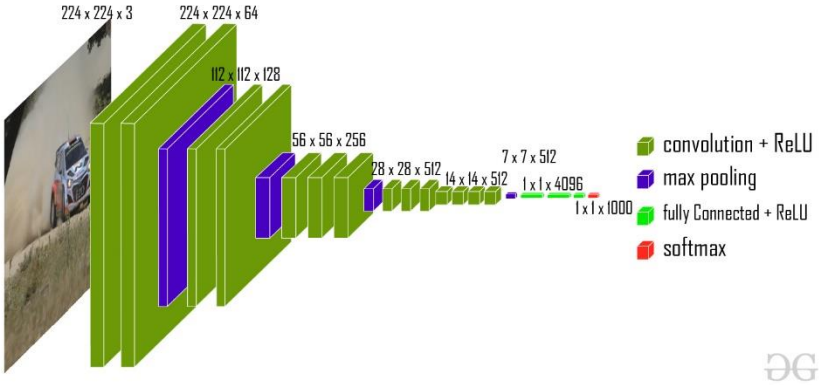
Slika 6. Mapa arhitekture VGG-16 modela [14]

. Kao što možemo vidjeti, VGG-16 se sastoji od 16 slojeva, 13 konvolucijska i 3 povezana sloja. Slojevi su raspoređeni u blokove, od kojih se svaki blok sastoji od nekoliko konvolucijskih slojeva koje zatim slijedi sloj sažimanja radi smanjivanja broja uzoraka. Struktura VGG arhitekture izgleda ovako (dimenzije su u pikselima): [14]

- Ulazni sloj – Koristi se slikama iz seta podataka ImageNet koji su dimenzija 224x224 te imaju RGB kanal što čini sveukupni ulaz dimenzija 224x224x3
- 2 konvolucijska sloja – 64 filtra dimenzija 3x3
- Sloj sažimanja– dimenzija 2x2 s pomakom 2
- 2 konvolucijska sloja – 128 filtera dimenzija 3x3
- Sloj sažimanja - dimenzija 2x2 s pomakom 2
- 2 konvolucijska sloja – 256 filtera dimenzija 3x3
- 2 seta od 3 konvolucijska sloja – 512 filtera dimenzija 3x3

- Sloj sažimanja - dimenzija 2x2 s pomakom 2
- Stog 2 dodatna konvolucijska sloja s filterima dimenzija 3x3 te sloj sažimanja
- Ravnanje (engl. *flatten*) – ravnanje *feature map*-a dimenzija 7x7x512 u vektor veličine 25088
- Povezani sloj – 3 povezana sloja s ReLU aktivacijom
    - Prvi sloj s veličinom ulaza od 25088 i izlaza od 4096
    - Drugi sloj s veličinom ulaza od 4096 i izlaza od 4096
    - Treći sloj s veličinom ulaza od 4096 i izlaza od 1000 – zbog 1000 mogućih klasifikacija
    - Softmax aktivacija se primjenjuje na izlaz trećeg povezanog sloja za klasifikaciju

Jasniji primjer se nalazi na slici 7 u kojoj su jasno vidljive dimenzije svakog sloja u modelu koji ostvaruje 92.7% točnosti. [14]



Slika 7. VGG arhitektura [14]

4.2.2. MobileNet

"MobileNet" je jednostavan i efikasan model koji se najčešće koristi u *real-time* aplikacijama koje se koriste za prepoznavanje objekata, klasifikaciju i prepoznavanje atributa lica itd. "MobileNet" koristi vrste konvolucije koje VGG16 ne koristi, a to su "Depthwise" i "Pointwise" konvolucija. [15]

"Depthwise" konvolucija je konvolucija koja primjenjuje pojedinačni filter za svaki kanal ulaznih podataka, umjesto jednog filtera na sve kanale kao obični kovolucijski sloj. Ovaj proces rezultira s 3 izlazne mape značajki (engl. feature map). Po jedna za svaki kanal. Samim time se omogućava efikasnija obrada podataka. [16]

"Pointwise" konvolucija koristi 1x1 filtere što znači da prolazi kroz svaki piksel u sva 3 kanala. Budući da se "Depthwise" konvolucija primjenjuje zasebno na svaki ulazni kanal, ne

kombinira informacije između kanala kako bi stvorila nove značajke. Kako bi se to omogućilo, koristi se upravo “Pointwise“ konvolucija. Proces se sastoji od izračunavanja linearne kombinacije izlaza “Depthwise“ konvolucije pomoću filtra dimenzija 1x1. Svaki filter će kreirati zasebnu mapu značajki. [15]

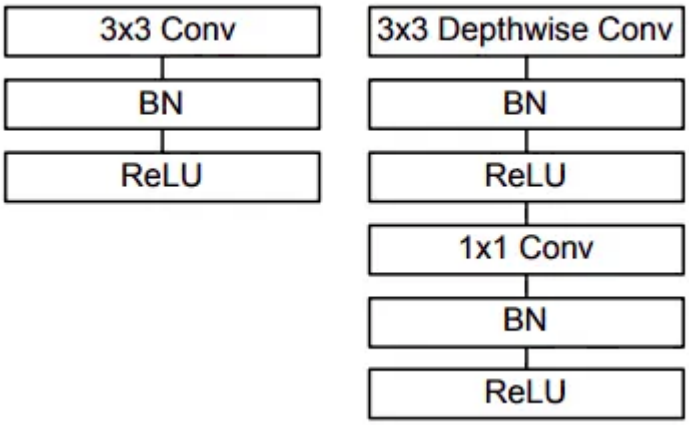
MobileNet također sadržava “Batch Normalization“ slojeve. “Batch Normalization“ značajno poboljšava efikasnost i pouzdanost modela. Normalizira izlaze između slojeva, što omogućava sljedećem sloju da bolje analizira podatke. “Batch Normalization“ uklanja srednju vrijednost grupe i dijeli je standardnom devijacijom. Potom se primjenjuju dodatni parametri kako bi se omogućila čim veća stabilnost mreže. [17]

Arhitektura MobileNet modela je prikazana na slici 8.

Type / Stride	Filter Shape	Input Size
Conv / s2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
Conv dw / s1	$3 \times 3 \times 32$ dw	$112 \times 112 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
Conv dw / s2	$3 \times 3 \times 64$ dw	$112 \times 112 \times 64$
Conv / s1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
Conv dw / s1	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
Conv dw / s2	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
Conv dw / s1	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
Conv dw / s2	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
$5\times$ Conv dw / s1	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
$5\times$ Conv / s1	$1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$
Conv dw / s2	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 1024$	$7 \times 7 \times 512$
Conv dw / s2	$3 \times 3 \times 1024$ dw	$7 \times 7 \times 1024$
Conv / s1	$1 \times 1 \times 1024 \times 1024$	$7 \times 7 \times 1024$
Avg Pool / s1	Pool 7×7	$7 \times 7 \times 1024$
FC / s1	1024×1000	$1 \times 1 \times 1024$
Softmax / s1	Classifier	$1 \times 1 \times 1000$

Slika 8. Arhitektura MobileNet modela

“dw“ na slici označava “depthwise“ i “pointwise“ konvoluciju u jednom sloju. Model također sadržava običnu konvoluciju koja se nalazi i u VGG16 modelu.



Slika 9. Slojevi standardne i "depthwise" konvolucije [15]

Na lijevoj strani slike 9 su prikazani slojevi standardne konvolucije koji sadrže standardni konvolucijski sloj, “Batch Normalization“ te “ReLU“ aktivaciju. Desna strana prikazuje slojeve “depthwise“ konvolucije u kojoj se nalaze “depthwise“ konvolucijski sloj kojeg slijedi “Batch Normalization“ i “ReLU“ aktivacija kao i kod standardne konvolucije, samo što se ova konvolucija sastoji i od “pointwise“ dijela koju opet slijede “Batch Normalization“ i “ReLU“ aktivacija. Ključne razlike su te što standardna konvolucija kombinira informacije iz svih kanala u jednom koraku, dok “depthwise“ konvolucija razdvaja proces na dvije faze. “Depthwise“ koji djeluje pojedinačno na svaki kanal te “pointwise“ koja kombinira informacije između kanala. “Depthwise“ konvolucija se koristi u modernijim modelima poput MobileNet-a radi smanjenja složenosti uz očuvanje performansi.

4.2.3. VGG16 vs MobileNet

MobileNet model naspram VGG16 modelu ima svoje prednosti i mane. MobileNet ima puno manje parametara od VGG16 (4.2 milijuna naspram 138 milijuna) pa je samim time brže i treniranje modela. VGG16 je veoma precizan, ali memorijski i procesorski veoma zahtjevan model koji se iz tog razloga koristi na jačim računalima. Također se koristi na zadacima u kojima je točnost važnija od brzine i ograničenja resursa. MobileNet omogućava bolje performanse. Bolji je model za mobilne uređaje, IoT i druge aplikacije s ograničenim resursima. Performanse su nešto niže u usporedbi s VGG16, ali su prihvatljive za većinu zadataka.

Model	Parameters (Approx.)	Top-5 Accuracy (ImageNet)	Year Developed	Notable Features
VGG16	138 million	92.7%	2014	Simple architectures with 16 weight layers
ResNet50	25.6 million	92.2%	2015	Residual connections, skip connections
Xception	22 million	95.5%	2017	Depth-wise separable convolutions, efficient
MobileNet	4.2 million	89.9%	2017	Depth-wise separable convolutions, lightweight

Slika 10. Usporedbe 4 različita modela [18]

Na slici 10 je vidljiva usporedba 4 popularna modela. Prikazan je broj parametara, točnosti prema “ImageNet“ predefiniranim težinama, godina kada je model razvijen te karakteristike arhitekture. Između dva modela o kojima se govori u ovom poglavlju, vidljivo je kako je MobileNet 3 godine mlađi model koji je uz pomoć “depthwise“ konvolucije uspio ostvariti mali broj parametara u odnosu na VGG16, dok se sam VGG16 sastoji od 16 težinskih slojeva, koji su puno jednostavniji (i slabije optimizirani), ali pritom ostvaruju nešto bolje rezultate (92.7% naprema 89.9%). [18]

4.3. Setovi podataka za detekciju deepfake-a

Većina online alata za kreiranje *deepfake* zapisa su *open-source* te kao što je već prethodno rečeno, ne zahtijevaju prevelike tehničke vještine. Iz tog razloga, programi poput “Basic DeepfakeMaker“ su jako popularni među širom publikom. Isto tako, ovakvi programi kreiraju najviše setova podataka koji se kasnije mogu koristiti za treniranje CNN modela u svrhu detekcije. U ovom poglavlju su prikazani neki setovi podataka te su prikazane njihove prednosti i mane u usporedbi s ostalim setovima. [19]

4.3.1. FaceForensics++

FaceForensics++ je opsežni set podataka čije su glavne manipulacije licem kreiranje pomoću metoda “DeepFakes“, “Face2Face“, “FaceSwap“, “FaceShifter“ te “Neural Textures“ dok su podaci različitih veličina i razina kompresije. “DeepFakes“ i “FaceSwap“ kreiraju manipulirane sekvence lica s slabijom kvalitetom zbog nepodudaranja boja i granica dok “Face2Face“ i “Neural Textures“ kreiraju sekvence nešto bolje kvalitete, no s različitim rezolucijama. U skupu se nalazi 1000 stvarnih YouTube videa te 4000 lažna videa od kojih je 60% s ženama, dok je 40% s muškarcima. Videozapisi su prikazani u 480p, 720p te 1080p. Pošto je ovo pionirski set podataka, sadrži videozapise različitih kvaliteta te približno istog broja muškaraca i žena čija su lica frontalna te vidljiva u svakom trenutku. Set ima i neke nedostatke kao što su niske kvalitete videozapisa s visokom kompresijom te vidljive granice lažnih maski.

Još jedan nedostatak je lošija obrada boja, što rezultira time da su izvorne boje lica različite od ciljanih boja lica. [41]

4.3.2. DFDC

Velike svjetske tehnološke kompanije kao što su Microsoft, Amazon i Meta su se uplele u područje *deepfake*-a tako što su zajedno pokrenule novu inicijativu zvanu “Deepfake Detection Challenge“ na platformi Kaggle koja se sastoji od velike zajednice podatkovnih znanstvenika, kako bi riješili problem *deepfake*-a. Set sadrži oko 120 000 videozapisa od 10 sekundi koji su snimljeni od strane pravih glumaca što ga čini jednim od najvećih setova podataka za prepoznavanje krivotvorenih videozapisa. Videozapisi su generirani tehnikama temeljenim na GAN-u, specijalnim tradicionalnim algoritmima *deepfake*-a ili ne-naučenim tehnikama (tehnike koje ne koriste strojno učenje ili neuronske mreže za generiranje sadržaja), s rezolucijama od 320x240 sve do 3840x2160 te frekvencijama monitora od 15 do 30 FPS (engl. Frames per second). U usporedbi s prijašnjim setom, FaceForensics++, videozapisi dolaze od strane plaćenih glumaca umjesto s YouTube-a, sama veličina uzorka je veća, zajedno s različitosti poza i rasa. Međutim, glavni nedostatak je što kvaliteta podataka varira zbog različitih sposobnosti generiranja *deepfake*-ova. Kao rezultat toga, neki uzorci pate od problema neusklađenosti granica, a izvorna i ciljana lica imaju različite rezolucije. [42]

4.3.3. Celeb-DF

Celeb-DF set podataka također koristi videozapise s YouTube-a, tj. oko 590 videozapisa poznatih osoba te 5639 manipuliranih videa generiranih uz pomoć tehnike FaceSwap. Set obuhvaća sve rasne, dobne i spolne skupine te dodatno vizualno poboljšava lažne videozapise kako ne bi bili prepoznatljivi ljudskom oku. Ta vizualna poboljšanja uključuju povećanje visoke rezolucije područja lica te primjena algoritama prijenosa boja i netočnih maski za lice. Glavni nedostatak ovog skupa podataka je ograničena količina uzoraka i nedostatak raznolikosti jer su svi originalni uzorci preuzeti s YouTube videozapisa poznatih osoba. Također, etnička raznolikost je mala, posebno kada su u pitanju azijska lica. [43]

4.3.4. DFFD

Diverse FakeFace Dataset sadrži 100 000 lažnih fotografija generiranih pomoću ProGAN modela te 200 000 pomoću StyleGAN modela. Kolekcija se sastoji od 47.7% muškaraca te 52.3% žena između 21 i 50 godina starosti. [44]

Kao što se može izvući iz naziva ProGAN i StyleGAN su dodatna poboljšanja prethodno spomenutog modela GAN koji se najčešće koristi kod kreacije deepfake zapisa. [20]

5. Metode evaluacije modela za detekciju deepfake-a

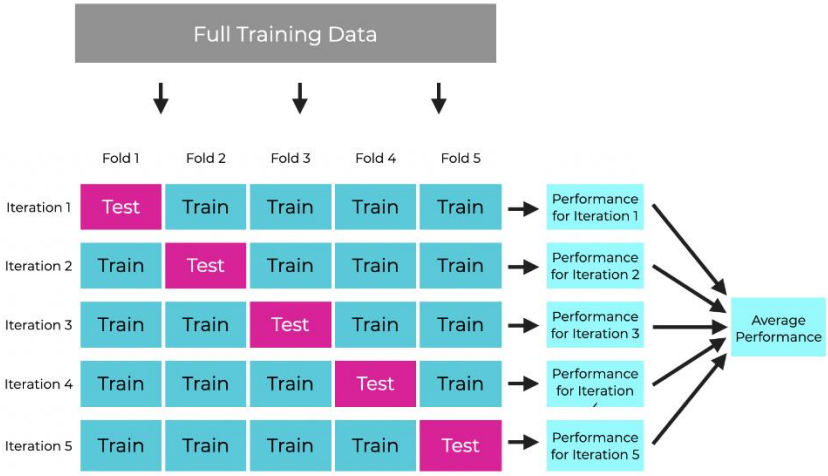
Modeli CNN-a, pa i samog strojnog učenja nisu ručno kodirani, već im se daje veliki set podataka pomoću kojeg samostalno identificira značajke kako bi dolazio do nekih zaključaka. Kako bi se znala stvarna prediktivna moć modela, potrebno je evaluirati model određenim metrikama. [21]

5.1. Unakrsna validacija

Evaluacija modela ključna je za procjenu njegove učinkovitosti i generalizacije na nove podatke. Unakrsna validacija (engl. *Cross-validation*) je često korištena tehnika koja pomaže u procjeni modela tijekom faze treniranja, osiguravajući da model neće biti previše prilagođen trenutačnim podacima. To je tehnika u kojoj se cijeli skup podataka ne koristi za treniranje modela, već se jedan dio rezervira za testiranje. Redovita evaluacija omogućuje bolje razumijevanje performansi modela i identificiranje područja koja zahtijevaju poboljšanje.

5.1.1. K-Fold unakrsna validacija

Jedna od najčešće korištenih vrsta unakrsne validacije je K-Fold Cross Validation. U ovoj metodi, izvorni skup podataka dijeli se na k podskupova, poznatih kao *fold*-ovi. Postupak se ponavlja k puta, pri čemu se jedan *fold* koristi za testiranje, a preostalih k-1 *fold*-ova za treniranje modela. Na taj način svaki podatak služi i za testiranje i za treniranje modela, što pomaže u boljoj generalizaciji modela i smanjenju stope pogreške. [23]

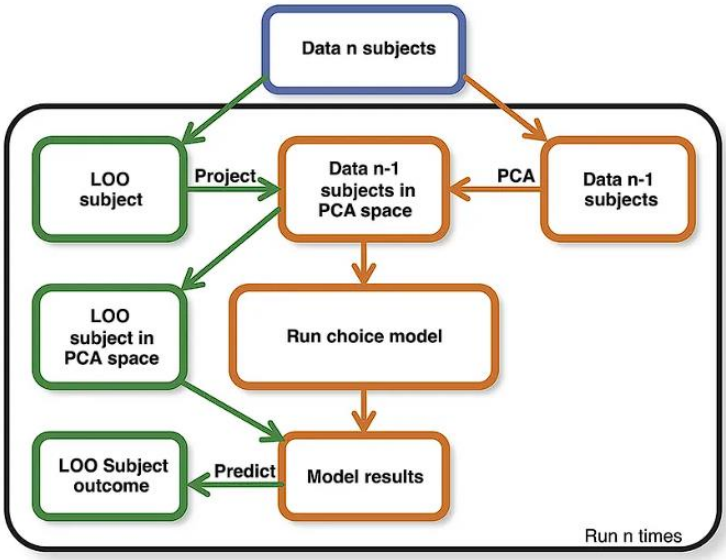


Slika 11. K-Fold unakrsna validacija [23]

Slika 11 prikazuje K-Fold unakrsnu validaciju te kako se set podataka dijeli na set za treniranje i testiranje. Naime, broj *fold* objekata određuje implementator modela. Svaki od tih *fold*-ova će biti u jednoj iteraciji testni set podataka, dok će ostali biti set podataka za trening. Broj *fold*-ova će odrediti i broj iteracija.

5.1.2. Leave-one-out unakrsna validacija

“Leave-One-Out“ unakrsna validacija (LOOCV) je jedna od popularnih metoda za evaluaciju modela u strojnom učenju i sustavima preporuka. Njena glavna karakteristika je da u svakom koraku iz skupa podataka izostavlja točno jednu podatkovnu točku, koja zatim služi kao validacijski set. Model se trenira na preostalim podacima, a postupak se ponavlja za svaku podatkovnu točku u skupu. Na taj način, svaka točka podataka dobiva priliku da bude korištena za validaciju, čime se procjenjuje sposobnost modela da generalizira na nove nepoznate podatke. [22]



Slika 12. Proces LOO unakrsne validacije [22]

Na slici 12 je prikazan proces “Leave-one-out“ unakrsne validacije. Proces započinje s prikupljenim podacima od n subjekata. Kroz LOO, jedan subjekt se izostavlja iz skupa podataka, što znači da će biti korišten za validaciju, dok će preostalih n-1 subjekata biti korišteno za treniranje modela. Ovaj pristup omogućuje da svaki podatkovni subjekt jednom bude korišten kao "testni" podatak. Nakon što je jedan subjekt izostavljen, na preostalih n-1 subjekata primjenjuje se Principal Component Analysis (PCA), koja smanjuje dimenzionalnost podataka te ističe one značajke koje su bitne za predviđanje rezultata. Nakon što je PCA primijenjen na podatke za treniranje, izostavljeni subjekt se primjenjuje u isti PCA prostor kako

bi mogao biti evaluiran. Pokreće se odabrani model za predikciju, koji trenira podatke s n-1 subjekata i koristi se za predviđanje rezultata izostavljenog subjekta. Nakon što se izostavljeni subjekt predvidi, predikcija mu se bilježi, a proces se ponavlja za svakog subjekta, sve dok svi subjekti nisu jednom izostavljeni. Na ovaj način, LOO omogućuje procjenu modela korištenjem svakog subjekta kao testnog podatka. PCA još dodatno optimizira proces tako što smanjuje složenost podataka i omogućava modelu da se fokusira na najbitnije značajke. [22]

5.2. Metrike

5.2.1. Točnost

Točnost (engl. *accuracy*) je metrika koja je predstavljena kao broj točnih predviđanja naprema ukupnom broju predviđanja. Formula za točnost je:

Accuracy = (TP+TN) / (TP+TN+FP+FN).

TP – True Positive – izlaz u kojem je i stvarni i predviđeni rezultat DA

TN – True Negative – izlaz u kojem je i stvarni i predviđeni rezultat NE

FP – False Positive – izlaz u kojem je stvarni rezultat NE, a predviđeni rezultat DA

FN – False Negative – izlaz u kojem je stvarni rezultat DA, a predviđeni rezultat NE

Ovakva ili slična simbolika će se koristiti u formulama drugih metrika.

Problem metrike točnosti je ta što je jednostavna te ne može dati ispravnu evaluaciju modela kod nebalansiranih setova podataka. Pod nebalansirane setove podataka se misli da će model točno razvrstati podatke dominantnih klasa dok neće dobro svrstati podatke manje zastupljenih klasa. Točnost takvog modela će svejedno biti vrlo visoka, iako to neće biti slučaj u stvarnosti. [21]

5.2.2. Preciznost i opoziv

Preciznost (engl. *Precision*) je metrika koja je predstavljena kao broj pozitivno točnih (TP) rezultata u odnosu na sumu točno pozitivnih (TP) i netočno pozitivnih (FP) rezultata. Svrha ove metrike je da analizira sve pozitivne rezultate. Formula za preciznost je:

Precision = TP / (TP+FP)

Glavni nedostatak ove metrike je da zanemaruje sve negativne rezultate, bili oni točni ili netočni. [21]

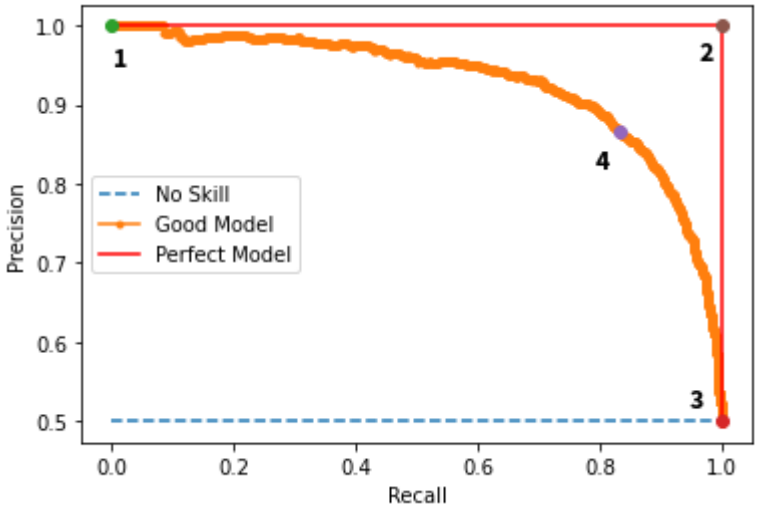
Opoziv (engl. *Recall*) je metrika koja je predstavljena kao broj točno pozitivnih (TP) rezultata u odnosu na sumu točno pozitivnih (TP) i netočno negativnih (FN) rezultata. Svrha

ove metrike je da analizira broj točnih pozitivnih uzoraka u odnosu na sve točne. Formula za preciznost je:

Recall = TP / (TP+FN)

Glavni nedostatak ove metrike je da često dovodi do veće lažno pozitivne stope. [21]

Preciznost i opoziv su veoma korisne metrike kada su setovi podataka nebalansirani. Preciznost će mjeriti relevantnost podataka dok će opoziv mjeriti koliko je zaista relevantnih rezultata vraćeno. Drugim riječima, postoje tri moguća slučaja. Prvi slučaj je maksimiziranje opoziva i ignoriranje preciznosti – želimo izvaditi čim više relevantnih podataka iako ćemo često vaditi i podatke koji nisu relevantni. Drugi slučaj je maksimiziranje preciznosti i ignoriranje opoziva – ovim ćemo pokušati minimizirati broj nerelevantnih podataka koji bi potencijalno izvukli, makar ćemo pritom izvući i manji broj relevantnih podataka. Treći slučaj je održavanje ravnoteže između preciznosti i opoziva čime ćemo maksimizirati broj relevantnih podataka te minimizirati broj nerelevantnih podataka. Treći slučaj je idealna situacija koju je gotovo nemoguće postići. Krivulja preciznosti i opoziva je prikazana na slici 13.



Slika 13. Krivulja preciznosti i opoziva [28]

Točke na slici (1-4) prikazuju različite pragove filtera pomoću kojeg smo izvadili podatke. Prag filtera u točki 1 je 1, u točki 3 je 0, u točki 4 je neka vrijednost između 0 i 1 dok točka 2 korespondira idealnom slučaju. [24]

5.2.3. F1 Score

F1 score je sredina preciznosti i opoziva. Ukoliko se jedna od ovih mjera poveća, druga mjera će se smanjiti jer djeluju obrnuto proporcionalno. Također, pojačava utjecaj manje vrijednosti na ukupnom izračunu kako bi se postiglo uravnoteženo mjerenje. F1 score s vrijednošću 1 označava savršen algoritam, dok F1 score s vrijednošću 0 označava algoritam

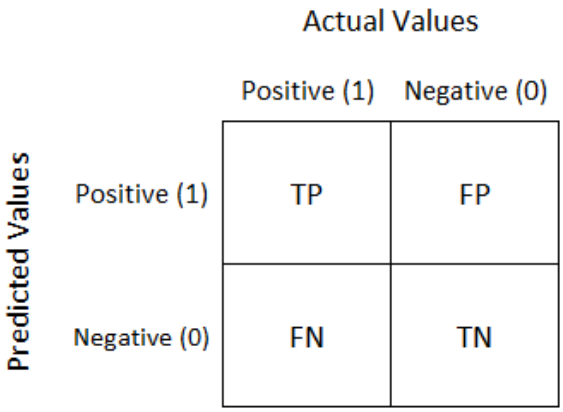
koji je promašen u svakom pogledu, bilo to za preciznost, opoziv ili oboje. Koristi harmonijski prosjek koji je zapravo recipročna vrijednost aritmetičke sredine recipročnih vrijednosti zadanog skupa. Harmonijski prosjek se koristi u situacijama kada se želi prosječna stopa. Formula za F1 score je:

F1 score = (2×Precision×Recall) / (Precision+Recall) [21]

F1 score može biti varljiv prilikom procjene algoritma ili usporedbe više algoritama, jer ne daje detaljne informacije o tome je li veći utjecaj na konačni rezultat imao opoziv ili preciznost. Na primjer, ako se dva modela razlikuju po tome što jedan loše izračunava preciznost, a drugi opoziv, mogli bi imati isti F1 score, iako se modeli suočavaju s različitim problemima u klasifikaciji. [25]

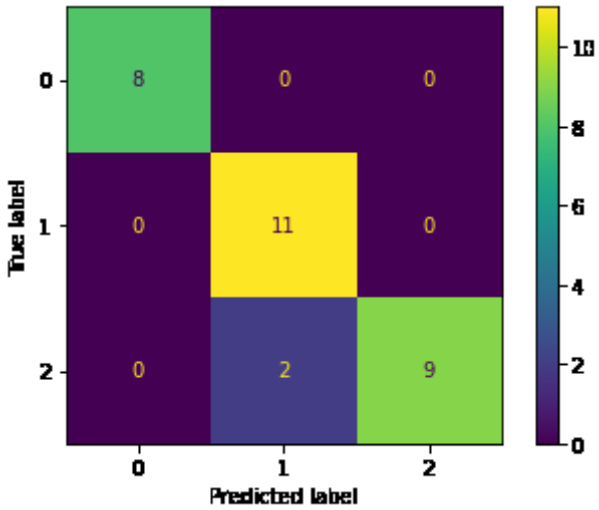
5.2.4. Matrica konfuzije

Matrica konfuzije je matrica dimenzija N x N gdje N predstavlja ciljani broj klasa. Predstavlja broj stvarnih izlaza te predviđenih izlaza.



Slika 14. Matrica konfuzije [29]

Ovakve matrice su odlične za mjerenje opoziva, preciznosti, točnosti, te najvažnije AUC-ROC krivulje koje će biti objašnjene kasnije. Na primjeru s stvarnim vrijednostima, izračunati ćemo točnost, preciznost i opoziv.

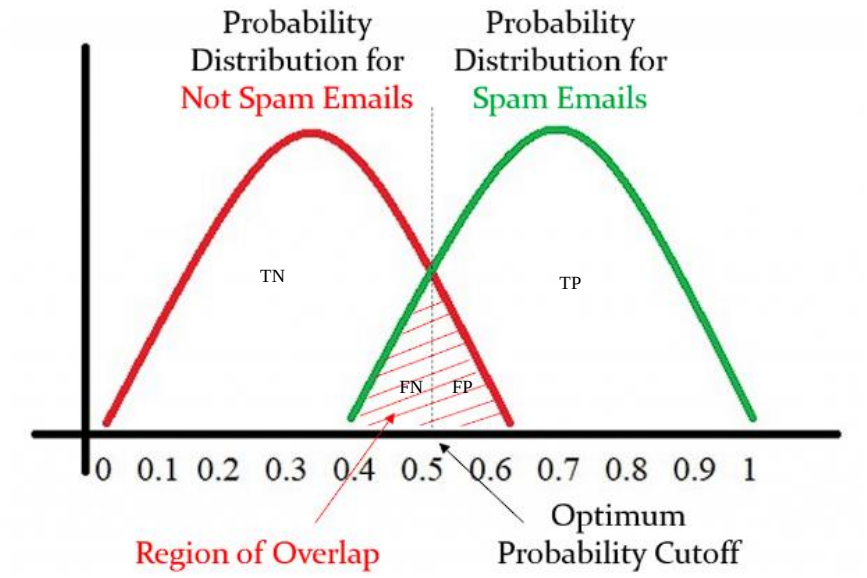


Slika 15. Matrica konfuzije s vrijednostima [21]

Na slici 15. možemo vidjeti oznake imenovane brojevima od 0 do 2, koje su vidljive i na strani stvarnih izlaza i predviđenih izlaza. Možemo vidjeti da se vrijednost 0 točno predviđala 8 puta, vrijednost 1 se točno predviđala 11 puta dok se vrijednost 2 točno predvidjela 9 puta. Imamo slučaj kada se stvarna vrijednost 2 predvidjela kao vrijednost 1 dva puta. [21]

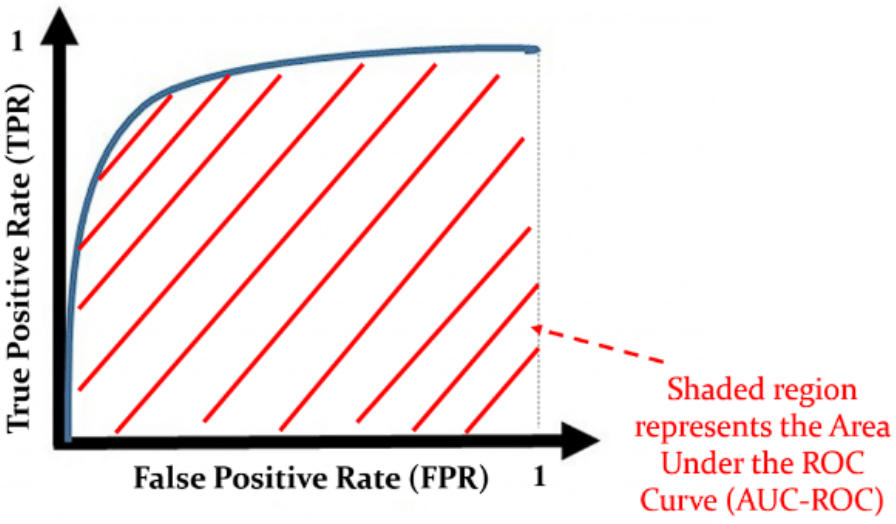
5.2.5. AUC-ROC krivulja

AUC (Area Under Curve) je metrika evaluacija koja je korištena za analiziranje modela pri različitim vrijednostima praga (engl. *treshold*) dok ROC (Receiver Operating Characteristic) krivulja je probabilistička krivulja koja mjeri performanse modela te ima dva bitna parametra. TPR (True positive rate) što zapravo odgovara formuli za opoziv te FPR (False positive rate) što je odnos lažno pozitivnih vrijednosti (FP) u odnosu na sumu lažno pozitivnih (FP) i točno negativnih vrijednosti (TN). Cilj ove krivulje je pronaći vrijednost praga u kojoj je maksimiziran TPR te minimiziran FPR. Taj prag će se kasnije koristiti za razlikovanje klasa.



Slika 16. Distribucija mailova [26]

Na slici 16 je prikazana predviđena distribucija mailova te je vrijednost praga 0.5. Iscrtana regija na grafu prikazuje sve rezultate koji su netočno predviđeni. Teoretski, možemo promijeniti vrijednosti praga u bilo što između brojeva 0 i 1. Za bilo koju promjenu, dobit ćemo različit set predviđenih podataka te će se samim time promijeniti TPR i FPR. Ako bi smanjili prag, dobit ćemo veći broj pozitivnih rezultata, odnosno TPR, ali će to rezultirati i u većem FPR-u. Ukoliko pak bi povećali prag, smanjio bi se broj TPR-a, dok bi se isto tako smanjila stopa pogreške, odnosno FPR.

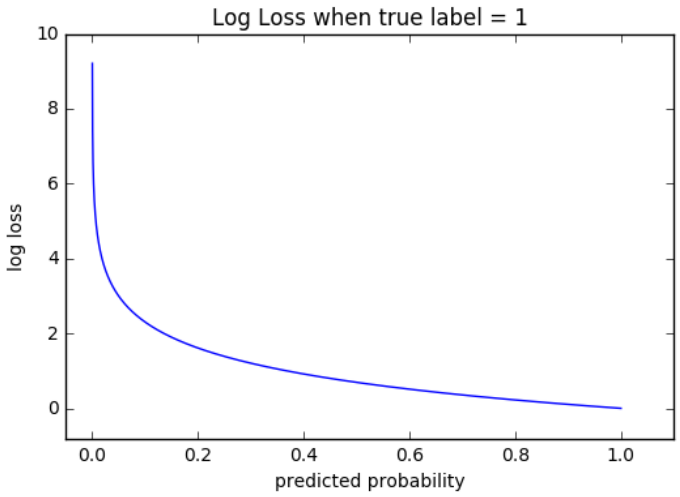


Slika 17. AUC-ROC [26]

Na slici 17 je prikazana ROC krivulja dobro evaluiranog modela. Također je označena većina područja jer je ispod krivulje. Što je takvo područje veće, model bolje odvaja različite klase. [26]

5.2.6. Funkcija gubitka

Funkcija gubitka (engl. *loss function*) predstavlja ključnu mjeru za ocjenu koliko je neuronska mreža uspješna u izvršavanju određenog zadatka. Prilikom procesa širenja unazad (engl. *backpropagation*), neuronska mreža popravlja svoje težine kako bi kasnije bolje prepoznavala, odnosno klasificirala izlazne podatke. Pritom je cilj minimizirati vrijednost funkcije gubitka koja se može interpretirati kao ocjena modela mreže. Svakom pogrešnom predikcijom, model dobiva određeni broj bodova. Što je veća vrijednost funkcije gubitka, to je model manje pouzdan. Postoje dva bitna tipa zadatka u kojima se koristi funkcija gubitka, a to su klasifikacijski i regresijski zadaci. Kod regresijskih zadataka, izlazni vektor je kontinuirana vrijednost, dok će kod klasifikacijskih zadataka isti predstavljati rezultate vjerojatnosti između 0 i 1. [27]



Slika 18. Cross-Entropy loss [27]

Na slici 18. je prikazan cross-entropy loss koji mjeri performanse klasifikacijskog modela čiji je izlaz vjerojatnost između 0 i 1. Primjećujemo da veći „log loss“ inicira na manju vjerojatnost predviđanja točne klasifikacije. Funkcija gubitka penalizira model ukoliko su predviđanja tog modela pogrešna, ali penalizacija je veća ako je model siguran u ta pogrešna predviđanja. Idealan model je onaj model koji ima vrijednost funkcije gubitka jednaku 0, odnosno, vjerojatnost predviđanja jednaku 1. U binarnoj klasifikaciji, gdje postoje samo dvije klase, cross-entropy se računa pomoću ove formule: [27]

$$-\left(y \log(p) + (1-y) \log(1-p)\right)$$

Log – prirodni logaritam (logaritam na bazi e)

y – binarni indikator ako je klasa točno predviđena za neki uzorak

p – predviđena vjerojatnost klasifikacije nekog uzorka za određenu klasu

6. Praktični dio

U ovom dijelu će biti objašnjen praktični dio u kojem je cilj bio izraditi model te ga trenirati i testirati na podacima iz odabranog seta podataka. Korištenjem konvolucijske neuronske mreže, istražit će se načini na koje je moguće prepoznati autentične videozapise od onih lažnih. Fokus će biti na korištenje MobileNet modela koji daje gotovo identične rezultate kao VGG16 s puno manje parametara (VGG-ovih 138 milijuna parametara naprema MobileNet-ovih 4 milijuna). Zbog tako malog broja parametara, treniranje će biti brže te će se model moći trenirati s više različitih hiperparametara. Model će također biti evaluiran, a metrike evaluacije će biti vizualizirane.

6.1. Prikupljanje podataka

Set podataka korišten u praktičnom dijelu je preuzet sa stranice Kaggle pod nazivom "Celeb DF (v2)" na URL-u https://www.kaggle.com/datasets/reubensuju/celeb-df-v2. Set sadržava 590 autentičnih videa te 5639 sintetiziranih videa sa slavnim osoba različitih godina, etničkih skupina i spola. Osim videozapisa, postoji tekstni dokument "List_of_testing_videos.txt" koji sadrži listu videa uglavnom korištenih za testiranje modela. Ukupna veličina seta podataka je oko 10 GB. Uzorci su podijeljeni u 3 direktorija: "Celeb-real", "Youtube-real", "Celeb-synthesis". "Celeb-real" i "Youtube-real" sadrže autentične, dok "Celeb-synthesis" sadrži lažne videozapise.

6.2. Pretprocesiranje podataka

6.2.1. Odabir videozapisa

Prije korištenja preuzetih podataka, potrebno ih je "pripremiti" za samu upotrebu. Proces takve pripreme se naziva pretprocesiranje podataka. Podaci iz seta podataka koji su namijenjeni za testiranje su pomoću tekstualne datoteke premješteni iz izvornog foldera u folder "sample_videos/test/Real" te "sample_videos/test/Fake" ovisno o tome kako je pojedina datoteka klasificirana. Naglasak je stavljen na premještanje, a ne kopiranje datoteke iz razloga što se uzorci za testiranje ne smiju pojavljivati u setu za treniranje. S druge strane, videozapisi za treniranje nisu premješteni, nego samo kopirani u foldere "sample_videos/train/Real" i "sample_videos/train/Fake". Modeli CNN-a koriste slike određenih dimenzija kao ulazni vektor. Ovim dolazimo do prvog koraka pretprocesiranja podataka, a to je ekstrakcija pojedinačnih slika (engl. *frames*) iz videa.

```
def copyFilesToDstFolder(sourceFolder, destinationFolder, sampleNumber=200):
    videoList = os.listdir(sourceFolder)
    for file in random.sample(videoList, sampleNumber):
        srcFile = os.path.join(sourceFolder, file)
        dstFile = os.path.join(destinationFolder, file)
        shutil.copy(srcFile, dstFile)
```

Isječak koda 1.

Isječak 1 prikazuje jednostavnu metodu u kojoj se najprije iz izvornog foldera mapiraju sve datoteke. For petlja će potom uzeti onoliko nasumičnih zapisa koliko se nalazi u parametru “sampleNumber“. Taj parametar je opcionalan, no ukoliko se ne upiše, zadana vrijednost će biti postavljena na 200 uzoraka. Ulaskom u for petlju, datoteka koja će se kopirati dobiva svoju punu putanju dok u liniji iznad datoteci dajemo novu putanju. Pomoću paketa “shutil“ jednostavno je pozvati funkciju copy(x, y) koja kopira datoteku s putanje x u novu datoteku na putanji y.

```
def moveTestFiles(dstFile, isReal, sampleNumber = 40):
    currDir = os.getcwd()
    fileName = 'List_of_testing_videos.txt'
    listFilePath = os.path.join(currDir, fileName)
    with open(listFilePath, 'r') as file:
        lines = file.readlines()
    selectedVideos = []
    for line in lines:
        if len(selectedVideos) >= sampleNumber:
            break
        line = line.strip()
        parts = line.split()
        label = "1" if isReal else "0"
        if parts[0] == label:
            selectedVideos.append(parts[1])
    for video in selectedVideos:
        filePath = os.path.join(currDir, "input", video)
        shutil.move(filePath, dstFile)
```

Isječak koda 2.

Isječak 2 prikazuje metodu za premještaj videozapisa koji će biti korišten kod testiranja modela. Ova metoda će se pozivati prije metode za kopiranje kako se odabrani uzorci za testiranje neće pronaći u skupu za treniranje. Metoda najprije čita redove iz datoteke “List_of_testing_videos.txt“. Primjer reda iz tekstualne datoteke je:

1 Celeb-real/id42_0002.mp4
ili
0 Celeb-synthesis/id1_id0_0007.mp4

Prva dio je indikator ako je videozapis autentičan ili lažan (prvi video je autentičan, dok drugi nije) dok je drugi dio putanja do same datoteke. Tekstovi su odvojeni razmakom. Nakon što je sadržaj datoteke pročitan, kreira se novo polje koje će sadržavati putanje odabranih datoteka. For petlja će ići kroz svaku liniju datoteke, a sama funkcija ima parametar u kojem je vidljivo ako se traže autentični ili lažni videozapisi. Ovisno o tome, oznaka će biti 0 ili 1. Ako prvi dio linije odgovara oznaci, putanja videozapisa će biti spremljena u polje odabranih videozapisa. Nakon što broj odabranih videozapisa dosegne broj definiran u parametru “sampleNumber“, izlazi se iz petlje te se pokreće nova for petlja koja će odabrane videozapise iz polja premjestiti u odredišni direktorij definiran u parametru “dstFile“.

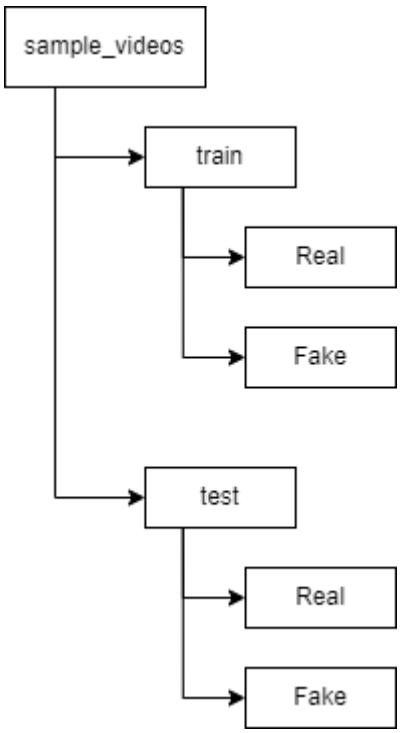
```
if testRealDirCreated:
    moveTestFiles(testReal, True, testSamplesNum)
if testFakeDirCreated:
    moveTestFiles(testFake, False, testSamplesNum)


if realDstFolderCreated:
    copyFilesToDstFolder(sourceReal, destinationReal, samplesNum)
if fakeDstFolderCreated:
    copyFilesToDstFolder(sourceFake, destinationFake, samplesNum)
```

Isječak koda 3

Isječak 3 prikazuje kako se gore spomenute metode dozivaju u programu. Za moveTestFiles, morao se definirati odredišni direktorij, bool vrijednost kojom će se odrediti traže li se autentični zapisi ili ne te broj uzoraka.

Za funkciju copyFilesToDstFolder su se trebali definirati izvorišni te odredišni direktoriji zajedno s brojem uzoraka. Postavljeno je da je broj uzoraka za testiranje 200 dok je za treniranje 20% od toga. Važno je napomenuti da će se mali udio podataka za treniranje koristiti za validaciju, točnije 20% od sveukupnog broja, stoga, omjer podataka za treniranje, validiranje i testiranje će biti 8:1:1.

Nakon procesa premještanja i kopiranja videozapisa, struktura direktorija “sample_videos“ izgleda kao na slici 19.



Slika 19. Struktura direktorija sample_videos

6.2.2. Ekstrakcija slika iz videozapisa

Početak ovog procesa započinje kreiranjem novog direktorija u kojem će biti spremljene slike iz odabranih videozapisa.

```
def extractFramesFromVideos(video_path, image_size, num_frames=10):
  frames = []
  face_detector = MTCNN()

  capture = cv2.VideoCapture(video_path)
  frame_count = int(capture.get(cv2.CAP_PROP_FRAME_COUNT))
  if frame_count < (num_frames * 3):
    return frames
  for frame_num in range(num_frames * 3):
    if frame_num % 3 == 0:
      ret, frame = capture.read()
      if not ret:
        break
      frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
      detectedFaces = face_detector.detect_faces(frame)
      if len(detectedFaces) > 0:
        x, y, width, height = detectedFaces[0]['box']
        if x < 0: x = 0
        if y < 0: y = 0
        if x + width > frame.shape[1]: width = frame.shape[1] - x
        if y + height > frame.shape[0]: height = frame.shape[0] - y

        faceImg = frame[y:y+height, x:x+width]
        faceImg = cv2.resize(faceImg, image_size)
        frames.append(faceImg)

  capture.release()
  return frames
```

Isječak koda 4.

Isječak 4 prikazuje metodu “extractFramesFromVideos“ koja ima nekoliko bitnih procesa za izvlačenje slika iz videa. Prvi proces je ekstrakcija. Najprije se uz pomoć cv2 biblioteke čita videozapis korištenjem funkcije VideoCapture(videoPath). Varijabla kojoj je dodijeljena ova funkcija sadrži brojne bitne značajke o videozapisu. Jedna od tih značajki jest broj slika u videu. Radi raznovrsnosti raznih izraza lica, provjeravamo ukoliko je ukupan broj

uzoraka iz parametra pomnožen s 3 manji od ukupnog broja slika po videozapisu. Ovaj dio koda je bitan radi raznovrsnosti raznih izraza lica jer samim time ne biramo slijedne sličice, već kao u našem slučaju, svaku treću sliku. Svaka treća slika potom ulazi u proces čitanja sadržaja (polje piksela) uz pomoć funkcije “read()“. Ako je sadržaj dobiven iz slike, odnosno “ret == False“, slika se dodatno uređuje. Slika se konvertira iz BGR formata, koji koristi cv2 biblioteka, u RGB format koji će koristiti biblioteka MTCNN. MTCNN je neophodna biblioteka za otkrivanje deepfake zapisa zbog mogućnosti prepoznavanja lica, najbitnijeg dijela na svakom zapisu. Uz pomoć biblioteke, možemo lako doći do broja otkrivenih lica korištenjem funkcije “len“ na ugniježđenu funkciju “detect_faces“. Ukoliko je broj detektiranih lica veći od 0 (što znači da postoji barem jedno lice), uzet će se prvo prepoznato lice. Uz pomoć linije “x, y, width, height = detectedFaces[0]['box']“ se dobivaju koordinate i dimenzije okvira koji obuhvaća lice. Ukoliko se neke od koordinata nalaze van granica, postavljamo ih na 0. U varijablu “faceImg“ se stavlja dio slike koji je izrezan uz pomoć linije “faceImg = frame[y:y+height, x:x+width]“. Naime, ova linija mapira po polju piksela započevši s indeksom visine od “y“ pa sve do “y+height“, isto tako i za širinu uz pomoć varijabla “x“ i “width“. Kako bi slika bila u ispravnom formatu za model, njenu veličinu postavljamo na veličinu zadanu parametrom image_size (u našem slučaju je to 224 x 224 px). Na kraju for petlje, uređenu sliku stavljamo u listu “frames“. Uz pomoć “capture.release“ funkcije, zatvara se video datoteka.

```
for path in realVideoTrainPaths:
  frames = extractFramesFromVideos(path, imageSize, numOfFramesPerVideo)
  if frames != []:
    for frame in frames:
      fileCount = len(os.listdir(framesTrainReal))
      fileName = f'frame_{fileCount + 1}.jpg'
      cv2.imwrite(os.path.join(framesTrainReal, fileName), frame)
```

Isječak koda 5.

U isječku 5 je vidljivo pozivanje prethodno definirane funkcije. U ovom dijelu koda se iterira kroz sve videozapise u direktoriju “sample_videos“, bili oni za treniranje ili testiranje, autentični ili lažni. Isječak prikazuje iteraciju kroz autentične videozapise za treniranje koje smo dobili uz pomoć linije “ realVideoTrainPaths = glob.glob(os.path.join(srcTrainReal, '*.mp4'))“. “realVideoTrainPath“ postaje lista svih videa iz direktorija koji predstavlja varijabla “srcTrainReal“. Vrijednost ove varijable je {putanja do projekta}/sample_video/train/Real. Nakon dobivene liste videozapisa, kroz svaki je bilo potrebno iterirati kako bi se dobio određeni broj slika iz njega. Parametri funkcije su “path“ koji je samo putanja do videozapisa,

“imageSize“ koji predstavlja veličinu izlazne slike nakon uređivanja te “numOfFramesPerVideo“ koji predstavlja broj uzoraka izvučenih iz 1 videozapisa. Veličina slike je postavljena na 224 x 224 piksela, što je već prije spomenuto dok je broj slika po videozapisu postavljen na 10. Ukoliko funkcija vrati prazno polje, trenutna iteracija se preskače, no ukoliko polje sadržava slike, te slike se postavljaju u određeni direktorij pod imenom “frame_#“ u kojem “#“ predstavlja trenutni broj slika u direktoriju inkrementiran za 1. Postupak listanja videozapisa, te na posljetku i ekstrakcije slika iz videozapisa je morao biti izvršen 4 puta, za svaku kombinaciju namjene i autentičnosti podataka. Struktura direktorija “frames“ u kojem se nalaze slike ima identičnu strukturu kao i direktorij “sample_videos“. U direktoriju koji će se koristiti za treniranje se nalazi 4000 slika, 2000 autentičnih te 2000 lažnih, dok se u setu za testiranje nalazi 800 slika, također raspodijeljeno na pola između autentičnih i lažnih. Primjer autentičnog i lažnog medija je vidljiv na slici 20.

Slika 20. Real vs Fake zapis

6.3. Kreiranje modela

Sljedeći korak u procesu detekcije videozapisa je kreiranje samog modela. Model je kreiran na bazi MobileNet-a bez uključivanja slojeva za klasifikaciju. Funkcija za izgradnju modela je vidljiva u isječku 6.

```
def buildModel():
    base_model = MobileNet(weights='imagenet', include_top=False, input_shape=(224,224, 3))
    newLayer = base_model.output
    newLayer = GlobalAveragePooling2D()(newLayer)
    newLayer = Dense(512, activation='relu')(newLayer)
    newLayer = Dropout(0.5)(newLayer)
    predictions = Dense(1, activation='sigmoid')(newLayer)
    model = Model(inputs=base_model.input, outputs=predictions)
    for layer in base_model.layers:
        layer.trainable = True
    optimizer = optimizers.SGD(lr=1e-4)
    model.compile(optimizer=optimizer, loss='binary_crossentropy', metrics=['accuracy'])
    return model
```

Isječak koda 6

Bazni model MobileNet-a se sastoji od mnoštva slojeva. “Depthwise“ i “Pointwise“ konvolucije su primijenjene za smanjenje računske složenosti, dok su “Batch Normalization“ i “ReLU“ aktivacije korištene za stabilizaciju i ubrzanje treninga. “Conv2D“ je sloj koji primjenjuje 2D konvoluciju na ulazne podatke kako bi iz njih izvukao određene značajke uz pomoć filtera.

“ReLU aktivacija“ (*Rectified Linear Activation Function*) je linearna funkcija koja izravno vraća ulaznu vrijednost ako je ona pozitivna, dok u suprotnom vraća nulu. Postala je zadana aktivacijska funkcija za mnoge vrste neuronskih mreža jer je model koji je koristi lakše trenirati i često postiže bolje performanse. [30]

Bazni model ne sadržava svoje klasifikacijske slojeve zbog parametra “include_top=False“. Također je definiran oblik ulaznih podataka (224, 224, 3) gdje su prva dva parametra širina i dužina uzorka dok je treći parametar broj kanala (RGB). Kako se nakon nekoliko pokušaja model nije dokazao na testnim podacima, korištenje predefiniranih težina je bilo neophodno za stvaranje dobrih predikcija. To je postignuto uz pomoć “weights='imagenet'“.

“ImageNet“ je projekt usmjeren na kategorizaciju slika u otprilike 22 000 kategorija temeljenih na definiranom skupu riječi i fraza. Sadrži oko 14 milijuna slika u svojoj bazi podataka te kao što je moguće pretpostaviti, sadrži datoteke s težinama istreniranih neuronskih mreža za određenu kategoriju. [31]

Baznom sloju je dodan sloj “GlobalAveragePooling2D“ koji predstavlja sloj sažimanja, točnije “general pooling“ koji prilikom sažimanja, uzima prosječnu vrijednost piksela receptivnog polja kako bi smanjio dimenzije tog dijela slike. [32]

“Dense“ sloj predstavlja povezani sloj. U modelu se nalaze dva takva sloja, prvi koristi “ReLU“ aktivaciju, dok je drugi “sigmoid“ aktivaciju. “ReDense“ je “Dense“ sloj s ReLU aktivacijom te primarno služi za jednostavno i efikasno poboljšanje performansa već treniranih modela. Koristi kombinaciju nasumičnih težina i ReLU aktivacije za smanjenje gubitka kod treniranja. [33]

Drugi Dense sloj koristi “sigmoid“ aktivaciju koji za manju vrijednost vraća broj bliži 0, dok za veću vrijednost vraća broj bliži 1. Ovaj sloj se koristi za klasifikaciju podataka. [34]

Ostao je i posljednji sloj koji je zapravo implementiran predzadnji, između dva povezana sloja, a to je Dropout. Dropout se koristi radi nasumičnog izbacivanja neurona čime se mijenja struktura mreže kako bi se smanjio *overfitting*.

Nakon što smo kreirali model pomoću baznog modela MobileNet-a, te dodali 4 nova sloja, morali smo kreirati i objekt klase “Model“ u kojem je “input“ bio ulaz baznog modela, dok je “output“ bio sloj za klasifikaciju, na koji su bili dodani slojevi sažimanja i povezani slojevi. Uz pomoć for petlje, postavljeno je da će svi parametri mreže moći biti trenirani, tj. ne želimo zamrznuti ni jedan sloj u mreži. Sljedeći korak je kompiliranje modela uz pomoć optimizacijskoga algoritma SGD (Stochastic gradient descent). Pridodana mu je mala stopa učenja od 0.0001 iz razloga što nismo zamrznuli ni jedan sloj, pa se zbog male stope učenja parametri neće drastično mijenjati u procesu treniranja kao što bi se kod većih stopa te iz tog razloga neće doći do kompletnog prilagođavanja na set za treniranje (bolja generalizacija modela). Funkcija gubitka je već prije spomenuta binarna entropija dok je metrika evaluacije točnost (engl. *accuracy*).

Gradijentni spust je algoritam optimizacije koji se koristi za pronalaženje optimalne vrijednosti funkcije cilja (maksimuma ili minimuma). Njegova svrha u strojnom učenju je pronalazak parametara modela koji daju maksimalnu točnost na setu podataka za treniranje i testiranje. SGD je jedna od varijanta gradijentnog spusta koja unosi element nasumičnosti u proces optimizacije modela. Umjesto da se za svaku iteraciju koristi cijeli skup podataka, SGD koristi slučajno odabrani primjer iz skupa ili manji dio skupa. [36]

Ovime se završava izgradnja modela, a njegovi se slojevi mogu lako prikazati pozivom funkcije “model.summary()“. Također se prikaže broj parametara koji se mogu trenirati i mijenjati te broj parametara koji se ne mogu trenirati zajedno s ukupnim brojem parametara. Početni i završni slojevi modela su pomoću funkcije “summary()“ prikazani na slikama 21 i 22.

Model: "functional"

Layer (type)	Output Shape	Param #
input_layer (InputLayer)	(None, 224, 224, 3)	0
conv1 (Conv2D)	(None, 112, 112, 32)	864
conv1_bn (BatchNormalization)	(None, 112, 112, 32)	128
conv1_relu (ReLU)	(None, 112, 112, 32)	0
conv_dw_1 (DepthwiseConv2D)	(None, 112, 112, 32)	288
conv_dw_1_bn (BatchNormalization)	(None, 112, 112, 32)	128
conv_dw_1_relu (ReLU)	(None, 112, 112, 32)	0
conv_pw_1 (Conv2D)	(None, 112, 112, 64)	2,048
conv_pw_1_bn (BatchNormalization)	(None, 112, 112, 64)	256
conv_pw_1_relu (ReLU)	(None, 112, 112, 64)	0
conv_pad_2 (ZeroPadding2D)	(None, 113, 113, 64)	0
conv_dw_2 (DepthwiseConv2D)	(None, 56, 56, 64)	576
conv_dw_2_bn (BatchNormalization)	(None, 56, 56, 64)	256
conv_dw_2_relu (ReLU)	(None, 56, 56, 64)	0
conv_pw_2 (Conv2D)	(None, 56, 56, 128)	8,192
conv_pw_2_bn (BatchNormalization)	(None, 56, 56, 128)	512
conv_pw_2_relu (ReLU)	(None, 56, 56, 128)	0
conv_dw_3 (DepthwiseConv2D)	(None, 56, 56, 128)	1,152
conv_dw_3_bn (BatchNormalization)	(None, 56, 56, 128)	512
conv_dw_3_relu (ReLU)	(None, 56, 56, 128)	0

Slika 21. Početni slojevi MobileNet-a

conv_dw_11_relu (ReLU)	(None, 14, 14, 512)	0
conv_pw_11 (Conv2D)	(None, 14, 14, 512)	262,144
conv_pw_11_bn (BatchNormalization)	(None, 14, 14, 512)	2,048
conv_pw_11_relu (ReLU)	(None, 14, 14, 512)	0
conv_pad_12 (ZeroPadding2D)	(None, 15, 15, 512)	0
conv_dw_12_bn (BatchNormalization)	(None, 7, 7, 512)	2,048
conv_dw_12_relu (ReLU)	(None, 7, 7, 512)	0
conv_pw_12 (Conv2D)	(None, 7, 7, 1024)	524,288
conv_pw_12_bn (BatchNormalization)	(None, 7, 7, 1024)	4,096
conv_pw_12_relu (ReLU)	(None, 7, 7, 1024)	0
conv_dw_13 (DepthwiseConv2D)	(None, 7, 7, 1024)	9,216
conv_dw_13_bn (BatchNormalization)	(None, 7, 7, 1024)	4,096
conv_dw_13_relu (ReLU)	(None, 7, 7, 1024)	0
conv_pw_13 (Conv2D)	(None, 7, 7, 1024)	1,048,576
conv_pw_13_bn (BatchNormalization)	(None, 7, 7, 1024)	4,096
conv_pw_13_relu (ReLU)	(None, 7, 7, 1024)	0
global_average_pooling2d (GlobalAveragePooling2D)	(None, 1024)	0
dense (Dense)	(None, 512)	524,800
dropout (Dropout)	(None, 512)	0
dense_1 (Dense)	(None, 1)	513

Total params: 3,754,177 (14.32 MB)
Trainable params: 3,732,289 (14.24 MB)
Non-trainable params: 21,888 (85.50 KB)

Slika 22. Izlazni slojevi MobileNet-a

Slike prikazuju početne i završne dijelove modela ne uključujući i 8 konvolucijskih slojeva u središnjem dijelu jer su gotovo identični prikazanim konvolucijskim slojevima. Na slici

16 vidimo kako je ulazni sloj veličine 224 x 224 piksela s 3 kanala. Na kasnijim se slojevima veličina smanjuje, dok se treći argument definira kao broj filtera koji se kroz slojeve povećava. Na slici 17. možemo vidjeti završne konvolucijske slojeve te globalno sažimanje, povezan, dropout i klasifikacijski sloj s brojem neurona u svom argumentu. Vidljivo je da postoji ukupno oko 3.7 milijuna parametara od kojih je samo 21 888 nemoguće trenirati.

6.4. Treniranje modela

Treniranje modela je ključan proces u razvoju sustava za prepoznavanje *deepfake* zapisa. Proces uključuje optimizaciju modela na velikom setu podataka kako bi se postigla što veća generalizacija, tj. što bolja predviđanja na novim podacima. Kroz iterativni postupak treniranja, model će prilagoditi svoje parametre kako bi smanjio razliku između stvarnih i predviđenih rezultata pa samim time i vrijednost funkcije gubitka.

```
train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest',
    validation_split=0.2
)
```

Isječak koda 7.

U isječku 7 je vidljivo kreiranje objekta klase “ImageDataGenerator“ koji se koristi za augmentaciju podataka. Augmentacija podataka je umjetno povećavanje priloženog seta podataka stvaranjem novih primjera kroz različite transformacije slika. “Rescale“ parametar se koristi za normalizaciju piksela slika. Vrijednosti piksela u originalnim slikama su u rasponu od 0 do 255, no kada postavimo “rescale=1./255“ sve će vrijednosti piksela biti u rasponu od 0 do 1. Ovaj parametar pomaže u stabilizaciji procesa treniranja te sveukupno daje bolje rezultate prilikom predviđanja rezultata. “rotation_range=20“ nasumično rotira slike do 20 stupnjeva u bilo kojem smjeru. Ovo pomaže modelu da bude fleksibilan te da prepozna zapise bez obzira na njihovu orijentaciju. “width_shift_range“ i “height_shift_range“ pomiču sliku horizontalno odnosno vertikalno do 20% ukupne širine odnosno veličine jer je sama vrijednost postavljena

na 0.2. Ovo će pak pomoći modelu da bude fleksibilan na različite pozicije objekata na slici. “shear_range=0.2“ nasumično primjenjuje “shear“ transformaciju, koja je zapravo deformiranje slike pomicanjem nekih dijelova te slike u određenom smjeru. “zoom_range=0.2“ slučajno zumira sliku do 20% što omogućava modelu da bude fleksibilan na različite udaljenosti prilikom prepoznavanja objekata na slici. “horizontal_flip“ će nasumično zrcaliti slike što je korisno kod slika u kojima horizontalna orijentacija nije bitna kao u ovom slučaju. “fill_mode='nearest'“ je tip popunjavanja praznina slika koje su bile modificirane različitim transformacijama te su neki pikseli na rubovima postali prazni, a u ovom slučaju, praznine se popunjavaju korištenjem vrijednosti najbližih piksela. Posljednji parametar “validation_split=0.2“ omogućava podjelu podataka na skup za treniranje i validaciju. U ovom slučaju, 20% podataka će se koristiti na validaciju, a 80% na treniranje.

```
train_generator = train_datagen.flow_from_directory(
    framesRoot,
    target_size=(224, 224),
    batch_size=32,
    class_mode='binary',
    subset='training'
)

validation_generator = train_datagen.flow_from_directory(
    framesRoot,
    target_size=(224, 224),
    batch_size=32,
    class_mode='binary',
    subset='validation'
)
```

Isječak koda 8.

Isječak 8 prikazuje korištenje objekta klase “ImageDataGenerator“ opisanog u prethodnom odlomku. Metoda “flow_from_directory“ generira slike za trening i validaciju iz direktorija navedonog u varijabli “framesRoot“ koja zapravo sadrži putanju do direktorija “frames/train“. Metoda očekuje da direktorij bude strukturiran na način da svaki poddirektorij predstavlja jednu klasu. Veličine slika se mijenjaju s obzirom na parametar “target_size“ što je u ovom slučaju 224 x 224 piksela. “batch_size“ je parametar koji definira broj slika na kojima

će se model trenirati u jednom prolasku prije ažuriranja težina. “class_mode='binary'“ postavlja binarni tip klasifikacije podataka zbog toga što su sveukupno prisutne 2 klase. “subset“ služi kao oznaka i definira da li će set podataka biti namijenjen za treniranje ili validaciju. Ovaj parametar se koristi s “validation_split“ parametrom iz objekta tipa “ImageDataGenerator“. Kod “training_generator“ objekta, parametar je postavljen na “training“ dok je u objektu “validation_generator“ postavljen na “validation“.

```
model = modelCNN.buildModel()


STEP_SIZE_TRAIN=train_generator.n // train_generator.batch_size
STEP_SIZE_VALID=validation_generator.n // validation_generator.batch_size


history = model.fit(
  train_generator,
  steps_per_epoch=STEP_SIZE_TRAIN,
  validation_data=validation_generator,
  validation_steps=STEP_SIZE_VALID,
  epochs=15
)
```

Isječak koda 9.

Isječak 9 prikazuje dozivanje metode koja je upisana kod izgradnje modela. Naime, da bi se mogle koristiti metode iz drugih .py datoteka koje su u istom projektu, bilo je potrebno dodati liniju “import modelCNN“ što uvozi cijelu datoteku “modelCNN.py“. Nakon uvođenja datoteke vraćenu vrijednost funkcije koja je model smo stavili u varijablu “model“. Nakon toga, morali smo definirati broj koraka po jednoj iteraciji odnosno epohi (engl. *epoch*). Broj koraka smo dobili tako što smo cjelobrojno podijelili ukupan broj uzoraka u generatoru slika za trening odnosno validaciju s parametom “batch_size“. Rezultate smo smjestili u varijable “STEP_SIZE_TRAIN“ i “STEP_SIZE_VALID“. Nakon toga, možemo početi s treniranjem modela tako što se funkciji klase “Model“ pridodaju generator za treniranje “train_generator“, “steps_per_epoch“ i “validation_steps“ koji smo izračunali prije, generator za validaciju “validation_data“ te broj iteracija “epochs“. Kako bi bitni podaci treniranja ostali zapisani, model se može spremiti u posebnu .h5 datoteku. Prije toga, bitno je definirati točnu putanju i naziv datoteke. Program će najprije provjeriti broj datoteka u direktoriju {projekt}/results/Models te

će datoteka dobiti ime “DeepfakeDetection_#.h5“ gdje # predstavlja trenutni broj datoteka u direktoriju inkrementiran za 1. Posljednji korak je zapisivanje podataka treniranja u datoteku pomoću “json.dump(x,y)“ metode gdje je x lista podataka evaluacije modela zapisanih u JSON formatu, a y datoteka u koji zapisujemo te podatke.

6.5. Testiranje modela

Posljednji korak u detekciji *deepfake* zapisa je evaluacija samog modela na setu podataka za testiranje koji se do sada nije nigdje koristio. Testiranje uključuje procjenu sposobnosti generalizacije, odnosno da točno predviđa ishode novih, nepoznatih podataka. Rezultati daju uvid koliko je model fleksibilan te kako će se ponašati u praksi. Metrike poput točnosti, F1 *score*-a, opoziva i preciznosti dati će bolji uvid u fleksibilnost modela.

```
modelPath = os.path.join(os.getcwd() + '/results/Models/DeepfakeDetection_8.h5')

model = models.load_model(modelPath)

historyPath = os.path.join(os.getcwd() + '/results/Histories/training_history_8.json')

with open(historyPath, 'r') as file:
  history = json.load(file)
```

Isječak koda 10.

Isječak 10 prikazuje učitavanje modela iz .h5 datoteke koja sadržava trenirani model. Osim modela, učitavaju se i podaci treniranja, točnije točnost i funkcija gubitka treninga i validacije kroz epohe. Podaci se stavljaju u varijablu “history“ u obliku JSON formata. Za vizualizaciju se koristi linijski dijagram “plt“ iz biblioteke “matplotlib.pyplot“. Implementacija dijagrama je prikazana na isječku 11.

```
plt.plot(history['loss'], label='Training Loss')
plt.plot(history['val_loss'], label='Validation Loss')
plt.plot(history['accuracy'], label='Training Accuracy')
plt.plot(history['val_accuracy'], label='Validation Accuracy')
plt.legend()
plt.show()
```

Isječak koda 11.

Kod prikazuje iscrtavanje linijskog grafa za stavke funkcije gubitka treninga pomoću “history['loss']“ s oznakom “Training Loss“, funkcije gubitka validacije “history['val_loss']“ s oznakom “Validation Loss“, točnost na podacima treninga “history['accuracy']“ s oznakom “Training Accuracy“ te na kraju točnost na podacima za validaciju “history['val_accuracy']“ s oznakom “Validation Accuracy“. Pomoću “plt.legend()“ se pored grafa dijagrama dodaje legenda kako bi bilo jasno koja linija predstavlja koji podatak, dok će se sam dijagram otvoriti u novom prozoru dozivanjem funkcije “plt.show()“. Dijagram je prikazan na slici 23.

Slika 23. Podaci treniranja i validacije

Na dijagramu se nalaze svi prethodno opisani podaci. Vidi se da se točnosti nalaze u onom rangu u kojem je to prihvatljivo. Imale su mali rast od 1. do 15. epohe. Funkcija gubitka

je na podacima za trening prihvatljiva, tj. vidi se nešto veći pad do 2. epohe, a nakon toga polagano stagnira. Jedini problematični dio je funkcija gubitka nad podacima za validaciju. Ovo se događa zbog lošije generalizacije modela pri kojem dolazi do *overfitting*-a. Neki od načina kako popraviti ovaj problem je dodavanje “Dropout“ slojeva, povećavanje veličine seta podataka za treniranje, povećanje stope učenja kod definiranja kompajlera, dodavanje ranog zaustavljanja kod kojeg će se set podataka prekinuti trening ukoliko se nakon određenog broja epoha neće popraviti “validation_loss“. Iako se ovo čine kao dobra rješenja, točnost nije toliko različita između validacijskog seta i seta za treniranje kao funkcija gubitka što može ukazivati na preveliku regularizaciju s “Dropout“ slojem, ili je moguće da model radi jednak broj pogrešaka kod treniranja kao i kod validacije, samo što su one pogreške kod validacije veće, pa mu daje slabiju ocjenu.

```
test_datagen = ImageDataGenerator(rescale=1./255)


testGenerator = test_datagen.flow_from_directory(
  framesRoot,
  target_size=(224, 224),
  batch_size=32,
  class_mode='binary'
)
```

Isječak koda 12.

Kao i kod treniranja, augmentacija podataka se radi i za treniranje. Ipak, podaci za testiranje ne smiju biti modificirani kao oni za treniranje. Zbog toga ćemo kod “ImageDataGeneratora“ samo normalizirati podatke zbog toga što smo ih normalizirali kod testiranja. Koristit će se ista metoda “flow_from_directory()“ s identičnim vrijednostima parametara

```
y_pred = model.predict(testGenerator)
y_pred = (y_pred > 0.5).astype(int).ravel()

y_true = testGenerator.classes

print(classification_report(y_true, y_pred, target_names=testGenerator.class_indices.keys()))
print(confusion_matrix(y_true, y_pred))
```

Isječak koda 13

Nakon pripreme podataka, model se može početi evaluirati s metrikama: točnost, preciznost, opoziv i F1 score. Linija “y_pred = model.predict(testGenerator)“ koristi model za predikciju na testnom setu podataka koji se učitava uz pomoć “testGenerator“ objekta. Rezultat he set predikcija, gdje svaka predikcija predstavlja vjerojatnost da slika pripada određenoj klasi. Nakon što se dobije predikcija u rangu između 0 i 1, te se predikcije pretvaraju u binarne oznake, tj. 0 ili 1 uz pomoć linije “y_pred = (y_pred > 0.5).astype(int).ravel()“. Ukoliko je predikcija veća od 0.5, klasifikacija je 1, dok je u suprotnom 0. “astype(int)“ pretvara tu vrijednost u cijele brojeve dok “ravel()“ transformira set u jednodimenzionalni niz. Nakon toga, dohvaćaju se stvarne klase za slike iz “testGenerator“ objekta. Te vrijednosti su istinite oznake za određenu sliku koje će se uspoređivati s predikcijama modela. “classification_report“ ispisuje prije spomenute 4 metrike za evaluaciju modela. “target_names=testGenerator.class_indices.keys()“ je parametar unutar “classification_report“ funkcije koji osigurava da izvještaj prikazuje nazive klasa umjesto samo njihovih indeksa kod ispisa.

```
Found 800 images belonging to 2 classes.
25/25 [==============================] - 17s 607ms/step
              precision    recall  f1-score   support

        Fake       0.87      0.90      0.89       400
        Real       0.90      0.87      0.88       400

    accuracy                           0.89       800
   macro avg       0.89      0.89      0.88       800
weighted avg       0.89      0.89      0.88       800
```

Slika 24. Rezultati ispisa metode

Slika 24 prikazuje ispis metode “classification_report“ u kojem su vidljive sve potrebne metrike te kako se odražavaju na pojedinu klasu. Vidljivo je da je pronađeno 800 slika koje pripadaju dvije klase te da je sveukupno testiranje trajalo 17 sekundi. Nadalje, za lažne zapise je izračunata preciznost od 87%, opoziv od 90% te F1 score od 89%. S druge strane, za autentične zapise je izračunata preciznost od 90%, opoziv od 87% i F1 score od 88%. “Support“ je za obje klase na 400 što znači da su skupovi savršeno raspoređeni sa 400 zapisa za svaku klasu. Također su prikazane metrike poput točnosti koja je na 89%, “Macro Avg“ daje prosječne vrijednosti preciznosti (89%), opoziva (89%) i F1 score-a (88%). Ponderirani prosjek (“Weighted Avg“) uzima u obzir broj uzoraka u svakoj klasi prilikom izračuna prosječne preciznosti, odziva i F1-Score-a no kako su brojevi uzoraka raspoređeni, rezultati će biti jednaki prosječnim vrijednostima.

Posljednja linija “print(confusion_matrix(y_true, y_pred))“ ispisuje “Confussion Matrix“ u kojem se nalaze točno i netočno predviđeni rezultati.

```
[[361  39]
 [ 53 347]]
```

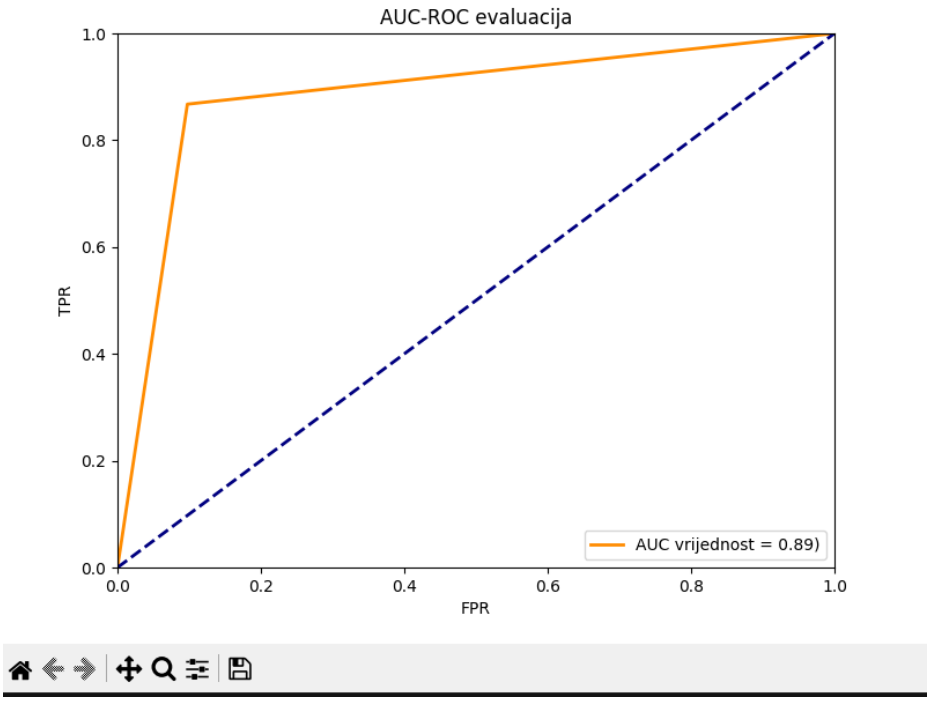
Slika 25. Confussion matrica

Na slici 25 je prikazan ispis “Confussion“ matrice pomoću gore spomenute funkcije. Prvi redak i prvi stupac (361) je broj uzoraka koji je predviđen kao lažan, a uistinu i jest lažan (True Positives), Prvi redak i drugi stupac (39) je broj uzoraka koji je predviđen kao lažan, a zapravo je autentičan (False Positives). Drugi redak i prvi stupac (53) je broj uzoraka koji je predviđen kao autentičan, a zapravo je lažan (False Negatives) dok su drugi red i drugi stupac korektno klasificirani kao autentični (True Negatives).

```
def showAucRoc(y_true, y_pred):
    fpr, tpr, thresholds = roc_curve(y_true, y_pred)
    roc_auc = auc(fpr, tpr)
    plt.figure(figsize=(8, 6))
    plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'AUC vrijednost = {roc_auc:.2f})')
    plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.0])
    plt.xlabel('FPR')
    plt.ylabel('TPR')
    plt.title('AUC-ROC evaluacija')
    plt.legend(loc='lower right')
    plt.show()
```

Isječak koda 14

Isječak 14 prikazuje metodu generiranja AUC-ROC krivulje prema testnim podacima. Uz pomoć funkcije “roc_curve“ čiji su parametri niz predikcija i niz stvarnih oznaka. Vraćene vrijednosti su FPR (False positive rate), TPR (True positive rate) te *tresholds* što predstavlja pragove korištene za binarnu klasifikaciju. Linija “roc_auc=auc(fpr, tpr)“ izračunava površinu područja ispod ROC krivulje. Nakon toga, dajemo dijaloškom okviru veličinu u inčima pa crtamo dijagram uz pomoć FPR i TPR parametara koji su dobiveni iz funkcije “roc_curve“. Postavljamo boju linije na tamno narančastu, širinu krivulje na 2 te oznaku krivulje (u kojem se nalazi AUC). Nakon toga se crta linija za referencu koja prikazuje slučajnu klasifikaciju. To je funkcija y=x, a u kodu spajamo točke (0, 0) s (1, 1). Pri tome se u funkciji “ plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')“ prvi argument odnosi na raspon od x dok je drugi argument raspon od y. Postavljamo tamno plavu (“navy“) boju i postavljamo sam stil linije da bude isprekidan. Postavljamo koordinatni sustav s “xlim“ i “ylim“ te ograničavamo vrijednosti varijabli x i y od 0 do 1. Naslov dijagrama se postavlja uz “plt.title('AUC-ROC evaluacija')“, a legenda se stavlja u donji desni kut uz pomoć “plt.legend(loc='lower right')“.



Slika 26. AUC-ROC krivulja kod evaluacije modela

Na slici 26 možemo vidjeti AUC-ROC krivulju za trenutni model. AUC vrijednost od 0.89 znači da model pokazuje visoke performanse u klasifikaciji između dvije klase. Pošto je vrijednost veoma blizu 1, model vrlo dobro razlikuje između autentičnih i lažnih zapisa, s visokom točnošću. AUC od 0.89 znači da model ispravno predviđa 89% svih uzoraka. Krivulja je zakrivljena prema gornjem lijevom kutu, što znači da model vrlo dobro balansira između osjetljivosti (preciznost u prepoznavanju stvarnih pozitivnih primjera) i specifičnosti (preciznost u prepoznavanju negativnih primjera).

6.6. Pregled netočnih podataka

Opcionalni korak kod kreacije modela je pregled netočno pozitivnih i negativnih primjera koji su “zavarali“ model. Ovo se postiže kreiranjem nove dvije varijable “falsePositives“ i “falseNegatives“.

```
falsePositives = np.where((y_pred == 1) & (y_true == 0))[0]
falseNegatives = np.where((y_pred == 0) & (y_true == 1))[0]
filenames = testGenerator.filenames
```

Isječak koda 15.

Isječak 15 prikazuje punjenje varijabli. “falsePositives“ će sadržavati sve one slike koje su bile predviđene kao lažni zapisi, dok su ustvari autentični. Isto tako, “falseNegatives“ će sadržavati slike koje su bile predviđene kao autentične, dok su ustvari lažne. “filenames“ sadrži nazive svih datoteka u testnom generatoru.

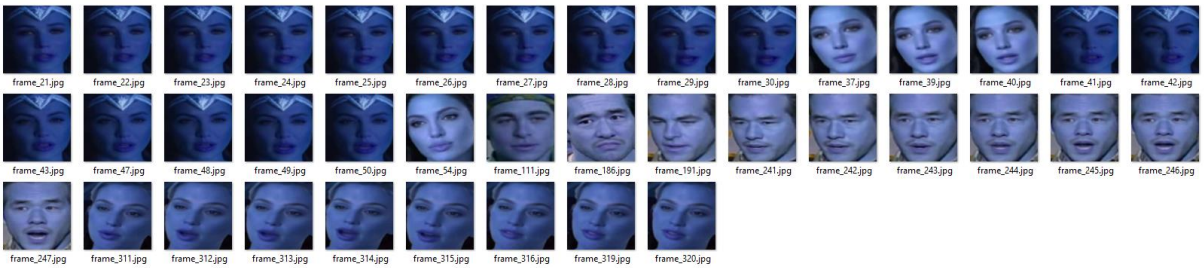
```
for id in falsePositives:
    imgPath = os.path.join(os.getcwd(), "frames", "test", testGenerator.filepaths[id])
    savePath = os.path.join(os.getcwd(), "results", "False Results", "False Positives", filenames[id].split('\\')[1])
    shutil.copy(imgPath, savePath)

for id in falseNegatives:
    imgPath = os.path.join(os.getcwd(), "frames", "test", testGenerator.filepaths[id])
    savePath = os.path.join(os.getcwd(), "results", "False Results", "False Negatives", filenames[id].split('\\')[1])
    shutil.copy(imgPath, savePath)
```

Isječak koda 16.

Isječak koda 16 ima za cilj spremiti slike koje su pogrešno klasificirane tijekom evaluacije modela u zasebne direktorije, što se može koristiti za detaljniju analizu grešaka. Radi se o organizaciji slika lažno pozitivnih i lažno negativnih klasifikacija kako bi se lakše identificirale i vizualizirale pogreške modela. Put do izvornih slika određuje se kroz imgPath, koji vodi do direktorija s testnim slikama. Svaka slika se potom sprema u odgovarajući direktorij unutar mape "{projekt}/results/False Results", gdje se pogrešno klasificirane slike organiziraju u direktorije "False Positives" i "False Negatives". Na taj način se slike koje je model pogrešno klasificirao, mogu naknadno analizirati bez potrebe za ponovnim izvođenjem testiranja.

Slika 27. Netočno pozitivni rezultati

Na slici 27 je prikazano 53 netočno pozitivnih rezultata koje je model pogrešno klasificirao kao lažne. Možemo vidjeti da je većina slika iz istog videa, dok je nekolicina slika jako slabe kvalitete, s neobičnom ekspresijom lica ili sa specifičnim uvjetima osvjetljenja.



Slika 28. Netočno negativni rezultati

Na slici 28 je prikazano 39 netočno pozitivnih rezultati koje je model pogrešno klasificirao kao autentične. Kao i kod netočno pozitivnih rezultata, većina slika je također iz istog videa. Osvjetljenje i izrazi lica ne bi trebali imati preveliki utjecaj na rezultate. Teško je za obje vrste netočnih rezultata protumačiti zajedničke karakteristike slika zbog kojih su netočno klasificirane.

7. Zaključak

Primjena konvolucijskih neuronskih mreža (CNN) za detekciju *deepfake* zapisa predstavlja napredak u borbi protiv sve složenijih metoda manipulacije zapisima. Velika dostupnost programa za izradu *deepfake*-a doprinosi sve većoj zlouporabi u raznim kontekstima.

CNN kao napredni alat iz područja dubokog učenja se pokazao veoma učinkovitim u detekciji lažiranih zapisa. Sposobnost prepoznavanja raznih objekata i uzoraka na slikama CNN-u omogućava prepoznavanje i najmanjih detalja između autentičnih i lažnih zapisa. Koristi se višeslojna struktura kako bi se analizirali detalji na različitim razinama detalja, od rubova i tekstura pa sve do kompleksnijih objekata. Međutim, unatoč visokoj točnosti CNN-ovih modela, problem s tehnologijom *deepfake*-a se nastavlja. Ovi se modeli moraju nastaviti razvijati kako bi išli u korak sa sve naprednijim tehnikama koje se koriste za stvaranje *deepfake* zapisa. Iz dana u dan ljudsko oko sve teže razlikuje što je stvarno, a što lažno. Cilj ovog rada je bio provesti istraživanje o tome kako nam konvolucijske neuronske mreže mogu pomoći u otkrivanju *deepfake*-ova, a potom i kako napraviti samu implementaciju. Implementacija se nalazi na linku *GitHub* repozitorija: https://github.com/Pa3kV/DeepfakeDetectionUsingCNN .

Deepfake je tehnologija koja svakim danom predstavlja veliku prijetnju i predviđa se da će takva tehnologija također predstavljati velike izazove u nadolazećim godinama, posebno u pogledu dezinformacija i sigurnosti. U tu su svrhu mnogi volonteri i organizacije počeli sudjelovati u kreiranju golemih setova podataka u svrhu razvoja sofisticiranih alata za otkrivanje i suzbijanje *deepfake* zapisa. Setovi podataka omogućavaju istraživačima da treniraju algoritme koji mogu bolje prepoznati lažne videozapise i slike, čime se podiže svijest o ovoj tehnologiji i smanjuje njena zloupotreba. Iako su ovi napori tek na početku, suradnja između stručnjaka i zajednice ključna je za zaštitu privatnosti i očuvanje integriteta digitalnih medija u budućnosti.

Popis literature

[1] IBM, N. Barney, “What is deepfake AI? A definition from WhatIs.com”, 2020. [Na internetu]. Preuzeto s: https://www.ibm.com/topics/convolutional-neural-networks [pristupano 9.6.2024.]

[2] IBM, “What are Convolutional Neural Networks?”, bez dat. [Na internetu]. Preuzeto s: https://www.ibm.com/topics/convolutional-neural-networks [pristupano 9.6.2024.]

[3] K. O’Shea and R. Nash, “An Introduction to Convolutional Neural Networks”, 2015.

[4] S. Hesaraki, “Feature Map,” *Medium*, 2023. [Na internetu]. Preuzeto s: https://medium.com/@saba99/feature-map-35ba7e6c689e [pristupano 7.7.2024.]

[5] Westerlund, M., “The Emergence of Deepfake Technology: A Review“, 2019.

[6] Fouzi Harrag, “A Comprehensive Study on Multimedia DeepFakes“, *ResearchGate*, [Slika]. Preuzeto s: https://www.researchgate.net/figure/DeepFake-images-generated-by-editing-StyleGAN-latent-codes-in-the-direction-of-the-six_fig1_367535057 [pristupano 11.7.2024.]

[7] A. O. J. Kwok and S. G. M. Koh, “Deepfake: a social construction of technology perspective”, 2020.

[8] P. Foy, “Generative Modeling: What is a Variational Autoencoder (VAE)?,” *MLQ.ai*, 2021. [Na internetu] https://blog.mlq.ai/what-is-a-variational-autoencoder/ [pristupano 9.8.2024.]

[9] R. Shende, “Autoencoders, Variational Autoencoders (VAE) and β-VAE,” *Medium*, 2023. [Na internetu]. Preuzeto s: https://medium.com/@rushikesh.shende/autoencoders-variational-autoencoders-vae-and-%CE%B2-vae-ceba9998773d [pristupano 9.8.2024.]

[10] Waseem, Saima & Abu-Bakar, Syed & Ahmed, Bilal & Omar, Zaid & Eisa, Taiseer & Dalam, Mhassen, “DeepFake on Face and Expression Swap: A Review“, 2023.

[11] Waseem, Saima & Abu-Bakar, Syed & Omar, Zaid & Ahmed, Bilal & Baloch, Saba & Hafeezallah, Adel, “Multi-attention-based approach for deepfake face and expression swap detection and localization“. [Slika]

[12] A. Karandikar, “Deepfake Video Detection Using Convolutional Neural Network”, 2020.

[13] GeeksforGeeks, “What is Transfer Learning?,” *GeeksforGeeks*, 2019. [Na internetu]. Preuzeto s: https://www.geeksforgeeks.org/ml-introduction-to-transfer-learning/?ref=lbp [pristupano 23.7.2024.]

[14] GeeksforGeeks, “VGG-16 | CNN model,” *GeeksforGeeks*, 2020. [Na internetu]. Preuzeto s: https://www.geeksforgeeks.org/vgg-16-cnn-model/ [pristupano 23.7.2024]

[15] S. PA, “An Overview on MobileNet: An Efficient Mobile Vision CNN,” *Medium*, 2020. [Na internetu]. Preuzeto s: https://medium.com/@godeep48/an-overview-on-mobilenet-an-efficient-mobile-vision-cnn-f301141db94d [pristupano 29.8.2024.]

[16] “Papers with Code - Depthwise Convolution Explained,” *paperswithcode.com*, (bez dat.). [Na internetu]. Preuzeto s: https://paperswithcode.com/method/depthwise-convolution [pristupano 29.8.2024.]

[17] “What is Batch Normalization,” *Deepchecks*, (bez dat.). [Na internetu]. Preuzeto s: https://deepchecks.com/glossary/batch-normalization/ [pristupano 29.8.2024.]

[18] T. Mostafid, “Overview of VGG16, ResNet50, Xception and MobileNet Neural Networks,” *Medium*, 2023. [Na internetu]. Preuzeto s: https://medium.com/@t.mostafid/overview-of-vgg16-xception-mobilenet-and-resnet50-neural-networks-c678e0c0ee85 [pristupano 4.9.2024.]

[19] Liang Yu Gong and Xue Jun Li, “Deepfake Detection Datasets,” *Encyclopedia.pub*, 2024. [Na internetu]. Preuzeto s: https://encyclopedia.pub/entry/55014 [pristupano 7.8.2024.]

[20] “Eight interesting datasets for deepfake detection in 2023,” *INDIAai*, 2023. [Na internetu]. Preuzeto s: https://indiaai.gov.in/article/eight-interesting-datasets-for-deepfake-detection-in-2023 [pristupano 7.8.2024.]

[21] jhimlic1, “Machine Learning Model Evaluation,” *GeeksforGeeks*, 2023. [Na internetu]. Preuzeto s: https://www.geeksforgeeks.org/machine-learning-model-evaluation/ [pristupano 8.8.2024.]

[22] T. R. Pathak, “Understanding and Implementing Leave-One-Out Cross Validation for Measuring Accuracy of Recommendation Systems”, *Medium*, 2024. [Na internetu]. Preuzeto s: https://medium.com/@tejupathak/understanding-and-implementing-leave-one-out-cross-validation-for-measuring-accuracy-of-425b62b01d38 [pristupano 5.9.2024.]

[23] “Cross Validation, Explained - Sharp Sight,” *www.sharpsightlabs.com*, 2023. [Slika]. Preuzeto s: https://www.sharpsightlabs.com/blog/cross-validation-explained/ [pristupano 5.9.2024.]

[24] Aishit Dharwal, “Complete Guide to Understanding Precision and Recall Curves”, 2021. [Na internetu]. Preuzeto s: https://analyticsindiamag.com/developers-corner/complete-guide-to-understanding-precision-and-recall-curves/ [pristupano 8.8.2024.]

[25] K. Akre, “F-score | Definition, Formula, & Facts | Britannica,” *Britannica*, (bez dat.) [Na internetu]. Preuzeto s: https://www.britannica.com/science/F-score [pristupano 8.8.2024.]

[26] P. Rishabh, “AUC-ROC Curve: Visually Explained | Learn Machine Learning” *New Tech Dojo,* 2020. [Na internetu]. Preuzeto s: https://www.newtechdojo.com/auc-roc-curve-visually-explained/ [pristupano 9.8.2024.]

[27] “Loss Functions in Neural Networks & Deep Learning”, *Built In*, (bez dat.) [Na internetu]. Preuzeto s: https://builtin.com/machine-learning/loss-functions [pristupano 26.8.2024.]

[28] Aishit Dharwal, “Complete Guide to Understanding Precision and Recall Curves”, *AIM*, 2021. [Slika]. Preuzeto s: https://analyticsindiamag.com/developers-corner/complete-guide-to-understanding-precision-and-recall-curves/ [pristupano 8.8.2024.]

[29] S. Narkhede, “Understanding Confusion Matrix,” Medium, 2018. [Slika]. Preuzeto s: https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62 [pristupano 9.8.2024.]

[30] J. Brownlee, “A Gentle Introduction to the Rectified Linear Unit (ReLU) for Deep Learning Neural Networks”, *Machine Learning Mastery*, 2019. [Na internetu]. Preuzeto s: https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/ [pristupano 23.7.2024.]

[31] A. Rosebrock, “ImageNet classification with Python and Keras”, *PyImageSearch*, 2016. [Na internetu]. Preuzeto s: https://pyimagesearch.com/2016/08/10/imagenet-classification-with-python-and-keras/ [pristupano 23.7.2024.]

[32] “Create Global Average Pooling Layer,” *Mathworks*, 2024. [Na internetu]. Preuzeto s: https://www.mathworks.com/help/deeplearning/ref/nnet.cnn.layer.globalaveragepooling2dlayer.html [pristupano 24.7.2024.]

[33] Alireza M. Javid, Sandipan Das, Mikael Skoglund, Saikat Chatterjee, “ A ReLU Dense Layer to Improve the Performance of Neural Networks“, 2021.

[34] K. Team, “Keras documentation: Layer activation functions,” *keras.io*., (bez dat.). [Na internetu]. Preuzeto s https://keras.io/api/layers/activations/ [pristupano 19.8.2024.]

[35] H. Yadav, “Dropout in Neural Networks,” *Medium*, 2022. [Na internetu]. Preuzeto s: https://towardsdatascience.com/dropout-in-neural-networks-47a162d621d9 [pristupano 19.8.2024.]

[36] R. Roy, “ML | Stochastic Gradient Descent (SGD),” *GeeksforGeeks*, 2019. [Na internetu]. Preuzeto s: https://www.geeksforgeeks.org/ml-stochastic-gradient-descent-sgd/ [pristupano 1.9.2024.]

[37] GeeksforGeeks, “CNN | Introduction to Pooling Layer”, *GeeksforGeeks*, 2019. [Na internetu]. Preuzeto s: https://www.geeksforgeeks.org/cnn-introduction-to-pooling-layer/ [pristupano 10.9.2024.]

[38] K. Murthy, “Stochastic Pooling ” *Medium*, 2024. [Na internetu]. Preuzeto s: https://medium.com/@crishna0401/stochastic-pooling-eda90d293ced [pristupano 10.9.2024.]

[39] P. Baheti, “12 Types of Neural Networks Activation Functions: How to Choose?”, *V7*, 2022. [Na internetu]. Preuzeto s: https://www.v7labs.com/blog/neural-networks-activation-functions [pristupano 10.9.2024.]

[40] J. Delua, “Supervised vs. unsupervised learning: What’s the difference?” *IBM*, 2024. [Na internetu]. Preuzeto s: https://www.ibm.com/think/topics/supervised-vs-unsupervised-learning [pristupano 10.9.2024.]

[41] ondyari, “FaceForensics++: Learning to Detect Manipulated Facial Images”, *GitHub*, 2022. [Na internetu]. Preuzeto s: https://github.com/ondyari/FaceForensics [pristupano 10.9.2024.]

[42] “Deepfake Detection Challenge Dataset”, *Meta*, (bez dat.). [Na internetu]. Preuzeto s: https://ai.meta.com/datasets/dfdc/ [pristupano 10.9.2024.]

[43] Reuben Suju Varghese, “Celeb DF (v2)”, *Kaggle*, 2023. [Na internetu]. Preuzeto s: https://www.kaggle.com/datasets/reubensuju/celeb-df-v2 [pristupano 10.9.2024.]

[44] “Diverse Fake Face Dataset”, *Msu.edu*, 2020. [Na internetu]. Preuzeto s: https://cvlab.cse.msu.edu/dffd-dataset.html [pristupano 10.9.2024.]

Popis slika