Prepoznavanje osoba temeljem šarenice oka primjenom dubokog učenja

Mikec, Ivan

SVEUČILIŠTE U RIJECI

TEHNIČKI FAKULTET

Sveučilišni prijediplomski studij računarstva

Završni rad

Prepoznavanje osoba temeljem šarenice oka primjenom dubokog učenja

Rijeka, srpanj 2023.

Ivan Mikec
0069091110

SVEUČILIŠTE U RIJECI

TEHNIČKI FAKULTET

Sveučilišni prijediplomski studij računarstva

Završni rad

Prepoznavanje osoba temeljem šarenice oka primjenom dubokog učenja

Mentor: prof. dr. sc. Kristijan Lenac

Komentor: v. asist. dr. sc. Diego Sušanj

Rijeka, srpanj 2023.

Ivan Mikec
0069091110

IZJAVA

Ovom izjavom potvrđujem da sam ovaj završni rad izradio potpuno samostalno, poštujući načela akademske čestitosti i držao se pravila za izradu završnog rada.

Rijeka, 07.07.2023.

Ime i prezime

ZAHVALA

Zahvaljujem se mentoru prof. dr. sc. Kristijanu Lencu i komentoru v. asist. dr. sc. Diegu Sušnju na podršci i znanju koje sam stekao tijekom izrade završnog rada.

SADRŽAJ

1. UVOD

Potreba za sigurnim i pouzdanim rješenjima biometrijske identifikacije postala je kritična u današnjem digitalnom dobu. Lozinke i identifikacijske kartice uobičajeni su oblici tradicionalne identifikacije, no podložni su krađi, lažnom predstavljanju i ljudskoj pogrešci. Kao rezultat svoje posebnosti i ugrađene otpornosti na krivotvorine, rješenja za provjeru autentičnosti temeljena na biometriji privukla su značajnu pozornost. Među brojnim biometrijskim modalitetima prepoznavanje šarenice postalo je jedna od najpreciznijih i najpouzdanijih metoda osobne identifikacije. Ljudska šarenica čini savršen biometrijski identifikacijski alat zbog svojih zamršenih uzoraka i individualnosti, tj. niti jedna ljudska šarenica nije ista. Sustavi za prepoznavanje šarenice iznimno su pouzdani zbog svoje stabilnosti, čak i u različitim svjetlosnim situacijama, te otpornosti na starenje ili male ozljede. Korištenje tehnika dubokog učenja za identifikaciju šarenice glavna je tema ovog rada. Algoritmi dubokog učenja otvorili su nove mogućnosti učenja i potencijal u pogledu prepoznavanja šarenice. Iskorištavanjem algoritama dubokog učenja, sustavi za prepoznavanje šarenice mogu postići iznimnu točnost i robusnost, što ih čini vrlo prikladnima za primjene u sigurnosti, kontroli pristupa i forenzičkim istragama. Posebno su konvolucijske neuralne mreže (eng. Convolutional naural network, skraćeno CNN) pokazale visoke performanse u prepoznavanju šarenica. Sustav može automatski naučiti diskriminirajuća obilježja izravno iz neobrađenih slika šarenice pomoću CNN-a za identifikaciju šarenice, uklanjajući potrebu za ručno projektiranim značajkama. U ovom radu namjeravaju se usporediti i analizirati performanse nekoliko različitih CNN modela na 2 različita skupa podataka (eng. dataset).

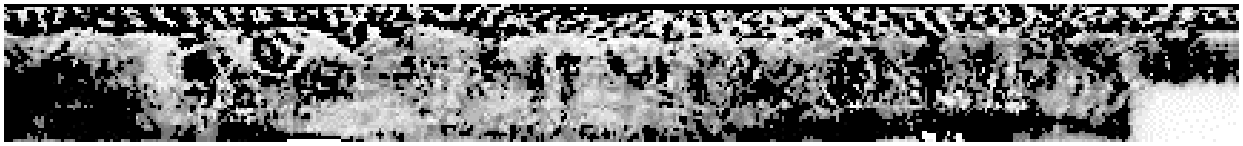
2. OPIS I ANALIZA PROBLEMA

2.1. Opis skupova podataka

Za analizu performansi različitih CNN modela odabrana su 2 različita skupa podataka: Indian Institute of Technology Delhi Iris skup podataka [1] i Casia V4 interval skup podataka [2]. Razlog testiranja više od jednog skupa podataka je zbog nekoliko prednosti koje to pruža. Procjena izvedbe modela na različitim skupovima podataka omogućuje temeljitu procjenu njihovih vještina generalizacije te pomaže u lociranju bilo kakvih pristranosti ili problema s pretreniranjem (eng. overfitting) koji mogu biti jedinstveni za skup podataka i kompromitirati izvedbu modela. Isto tako se time nudi temeljitija i vjerodostojnija procjena njihove izvedbe u različitim postavkama stvarnog svijeta.

2.1.1. IIT Delhi Iris skup podataka

IIT Delhi Iris Database većinom se sastoji od fotografija subjekata studenata i zaposlenika IIT Delhija. Ovaj je skup podataka prikupljen između siječnja i srpnja 2007. (još u tijeku) u Laboratoriju za biometrijska istraživanja pomoću JIRIS, JPC1000 i digitalne CMOS kamere. Skup podataka ima 176 muškaraca i 48 žena koji su svi u dobi između 14 i 55 godina [1]. Sve fotografije u trenutno dostupnoj bazi podataka, koja se sastoji od 224 subjekta, su u bitmap (*.bmp) formatu. Svaki subjekt u skupu podataka ima 5 slika, što znači da se skup podatka sastoji ukupno od 1120 slika rezolucije 432x48 piksela. Ovaj skup podataka dolazi s standardnim (sirovim) slikama, te njihovim normaliziranim verzijama (prikazano na Slici 2.1.).



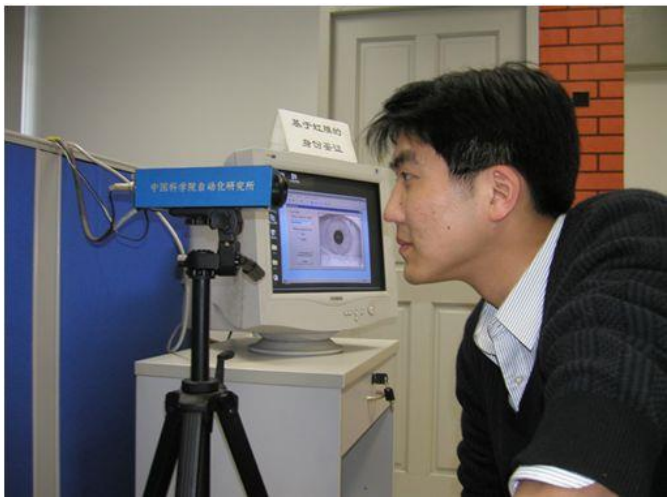
Slika 2.1. Prikaz normalizirane slike iz IIT Delhi Iris skupa podataka [1]

2.1.2. CASIA V4 interval skup podataka

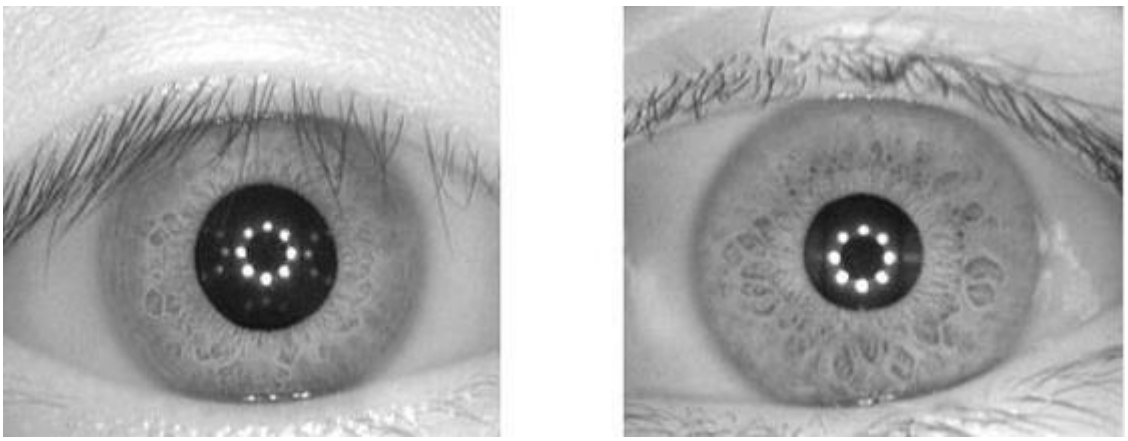
Institut za automatizaciju Kineske akademije znanosti (CASIA) stvorio je CASIA Iris Image skup podataka (CASIA-Iris), koji je ažuriran s CASIA-IrisV1 na CASIA-IrisV4 od 2002. godine.

CASIA-IrisV4 skup podataka se sastoji od 6 podskupa, što je proširenje CASIA-IrisV3 skupa podataka: CASIA-IrisV3, CASIA-Iris-Interval, CASIA-Iris-Lamp, CASIA-Iris-Twins, CASIA-Iris-Distance, CASIA-Iris-Thousand i CASIA-Iris-Syn.

U ovom radu korišten je podskup CASIA-Iris-Interval. Slike šarenica u CASIA-Iris-Interval snimljene su korištenjem posebno dizajnirane kamere namijenjene slikanju šarenica izbliza prikazanoj u Slici 2.2. Kamera je opremljena NIR LED poljem, što joj omogućuje slikanje vrlo jasnih slika (prikazano u Slici 2.3.) [2]. Sve slike su u 8-bitnom JPEG grayscale formatu, rezolucije 320 x 280 piksela. Skup podataka se sastoji od 249 subjekata, te je svakom od njih fotografirano lijevo i desno oko. Broj fotografija po oku varira (između 0 i 15), te su zbog premalih količina slika neki dijelovi skupa podataka morali biti izbačeni.



Slika 2.2. Prikaz CASIA iris kamere razvijene za prikupljanje CASIA-Iris-Interval slika [2]



Slika 2.3. Primjeri slika šarenica iz CASIA-Iris-Interval [2]

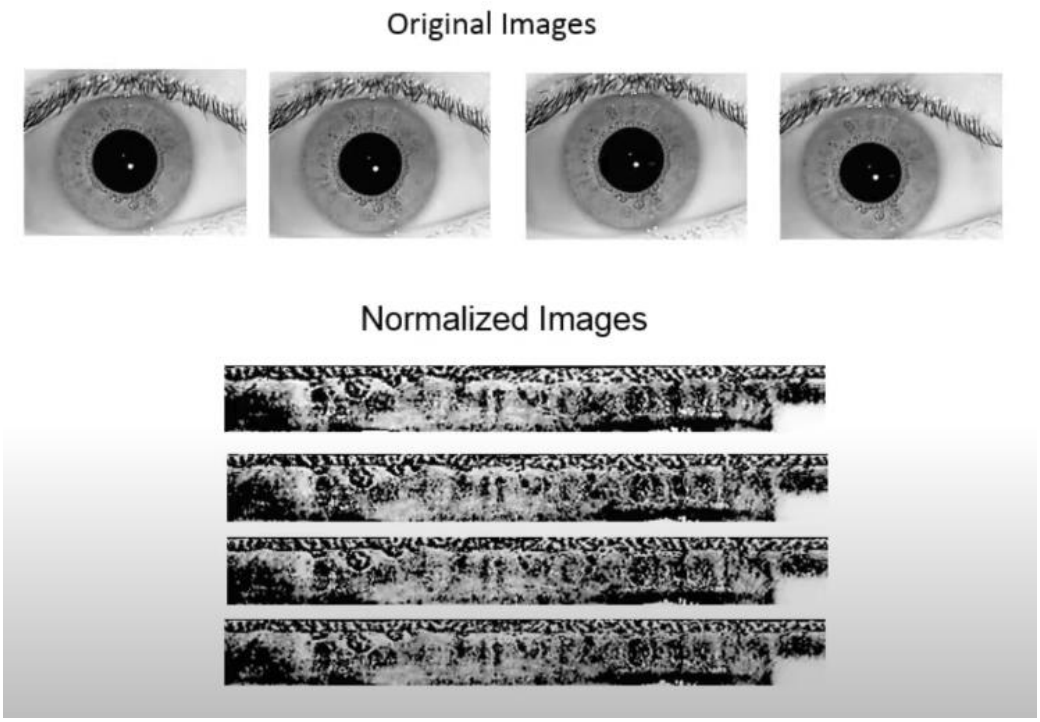
2.2. Pretprocesiranje skupova podataka

Skupovi podataka moraju se prethodno pravilno obraditi prije implementacije u model za prepoznavanje šarenice. Pretprocesiranje ili predobrada je proces poboljšanja kvalitete skupa podataka, standardiziranja njegovog formata i optimiziranja ukupne izvedbe algoritma za prepoznavanje šarenice. IIT Delhi skup podataka je pretprocesuiran prije preuzimanja tako da su mu izvorne slike normalizirane, dok CASIA V4 skup podataka zahtijeva uporabu maski.

2.2.1. Pretprocesiranje IIT Delhi Iris skupa podataka

IIT Delhi skup podataka sastoji se od standardnih slika šarenica, te njihovih normaliziranih verzija. Normalne slike su neobrađene, nepromijenjene sirove slike šarenice koje se dobivaju izravno iz uređaja za snimanje. Ove fotografije zadržavaju izvorne značajke šarenice, kao što su fluktuacije osvjetljenja, rezolucije i položaja očiju.

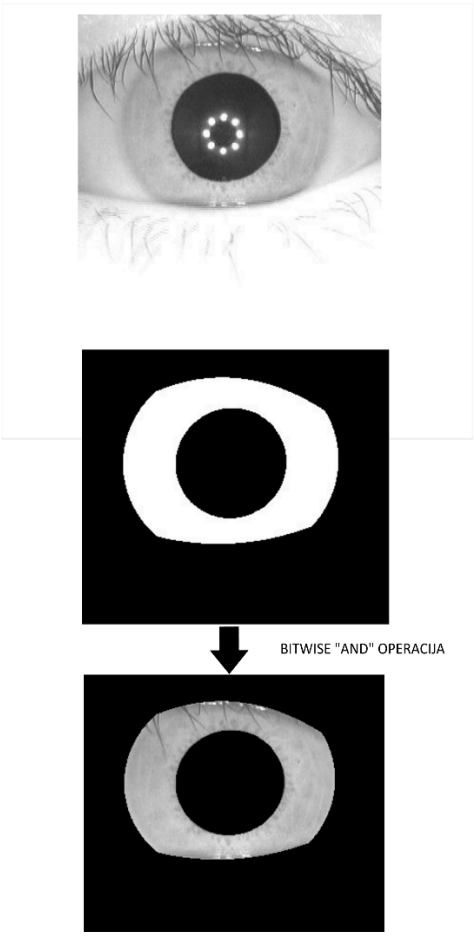
Normalizirane fotografije se razlikuju od standardnih time da je na njima upotrijebljena metoda pretprocesiranja zvana normalizacija. Normalizacija se izvodi na slikama šarenice kako bi se smanjile fluktuacije izazvane stvarima kao što su dilatacija zjenice, okluzija i nepravilni uvjeti osvjetljenja. Normalizacija pokušava smanjiti fluktuacije unutar klase, poboljšati diskriminirajuću sposobnost značajki šarenice i podići ukupnu točnost sustava za prepoznavanje šarenica.



Slika 2.4. Prikaz standardnih slika IIT Delhi skupa podataka i njihovih normaliziranih verzija [1]

2.2.2. Pretprocesiranje CASIA V4 interval skupa podataka

CASIA V4 interval skup podataka treba prethodnu obradu u obliku apliciranja maski koje su za ovaj rad bile pružene od mentora. Maske su bitne za prepoznavanje šarenice jer pomažu u isključivanju područja bez šarenice iz analize, što smanjuje šum i omogućuje algoritmu da se koncentrira samo na relevantne značajke. Očni kapci, trepavice i odrazi primjeri su područja bez šarenice koja mogu iskriviti uzorke šarenice i otežati preciznu identifikaciju. Maske za ovaj skup podataka su stvorene na način da bijelim pikselima u masci predstavljaju područje šarenice te crni pikseli sve ostala mjesta (Prikazano na Slici 2.5.). Pomoću Python skripte maske su aplicirane na sve slike iz skupa podataka, što rezultira slikama u kojima su ostavljeni samo bitne značajke za prepoznavanje šarenice.



Slika 2.5. Prikaz apliciranja maski na slike iz CASIA V4 interval skupa podataka

2.2.3. Razdvajanje skupova podataka u train i test segmente

Korišteni skupovi podataka sastoje se od slika kategoriziranih u zasebne mape po subjektu, pri čemu svaka mapa sadrži određeni broj slika subjekta. Kako bi se osigurala ujednačenost i usklađenost s jedinstvenim učitavačem podataka (eng. dataloader), skupovi su morali proći kroz niz izmjena, uključujući filtriranje subjekata s neadekvatnim brojem fotografijama, podjelu slika u skupine za obuku i validaciju te preimenovanje slika.

Subjekti s manje od pet fotografija nisu uključeni u transformirane skupove podataka kako bi se osigurale najbolje okolnosti za treniranje. Isključivanjem subjekata s niskom zastupljenošću, ova faza filtriranja pokušala je dati prioritet kvaliteti podataka. Željeno je CNN-u pružiti dovoljno primjera za treniranje nametanjem minimalnog praga od pet fotografija po subjekata, smanjujući rizik od pretreniranja i jačajući sposobnost generalizacije.

Nakon filtriranja, fotografije koje su ostale stavljene su u dvije različite mape, "train" i "test". Skup za treniranje primio je oko 80% fotografija, dok je skup za validaciju dobio preostalih 20% nakon podjele 80-20. Ova segmentacija nam je omogućila da treniramo CNN na značajnom dijelu skupa podataka dok smo dio zadržali po strani za nepristranu procjenu. Ovim odvajanjem podataka željeli smo procijeniti izvedbu modela na neviđenim slikama, osiguravajući točnu procjenu i potvrđujući njegove vještine generalizacije.

Fotografije su preimenovane kako bi se stvorio dosljedan format i učinile kompatibilnim s jedinstvenim dataloader-om. Peteroznamenkasti identifikacijski broj subjekta iza kojeg slijedi podvlaka i skup brojeva koji predstavljaju broj slike određenog subjekta čini standardni format (Prikazano u Slici 2.6.). Lijevo ili desno oko također je označeno izbornom podvlakom i slovima "L" ili "R". Osiguravanjem dosljednosti i jednostavnosti tumačenja za prilagođeni dataloader podataka tijekom kasnijih faza obrade, ovaj standard naziva nastojao je pojednostaviti rukovanje podacima.

Slika 2.6. Prikaz preimenovanja slika CASIA V4 interval skupa podataka

Zajedno, ove izmjene pomogle su pojednostaviti kasnije faze obrade podataka, poboljšati kvalitetu obuke i optimizirati skupove podataka za CNN obuku. Nakon transformacije skupova podataka, ostaje nam 224 subjekta, sa 5 fotografija po svakom subjektu u slučaju IIT Delhi Iris skupa podataka, te 496 subjekta s 5 – 13 slika u CASIA V4 interval skupu podataka (tretiramo lijeve i desne oči kao posebne subjekte).

2.3. Učitavanje podataka u model

Prilagođeni dataloader stvoren je za pravilno generiranje skupova podataka za trening i validaciju, što je neophodno za obuku modela dubinskog učenja. Klasa 'CustomDataLoader' čini zadatak praktičnim i višekratno upotrebljivim enkapsulacijom funkcionalnosti potrebne za učitavanje i prethodnu obradu slikovnih podataka za trening i validaciju modela klasifikacije slike.

```python
class CustomDataLoader:
    def __init__(self, data_dir, batch_size, image_size):
        self.data_dir = data_dir
        self.batch_size = batch_size
        self.image_size = image_size

    def load_data(self):
        image_files = os.listdir(self.data_dir)
        image_paths = [os.path.join(self.data_dir, file) for file in image_files]
        labels = [int(((os.path.basename(file))[:5])) for file in image_files]
        num_classes = max(labels)
        labels = tf.one_hot(labels, num_classes)

        # kreira TensorFlow Dataset iz slika i labela
        dataset = tf.data.Dataset.from_tensor_slices((image_paths, labels))

        # Ucitava i predprocesira slike
        dataset = dataset.map(self._load_and_preprocess_image)

        # Shuffle and batch the dataset
        dataset = dataset.shuffle(buffer_size=len(image_paths))
        dataset = dataset.batch(self.batch_size)

        return dataset

    def _load_and_preprocess_image(self, image_path, label):
        # Učita i predprocesira sliku
        image = tf.io.read_file(image_path)
        image = tf.image.decode_jpeg(image,channels=3)
          # Normalizira vrijednosti pixela u [0, 1] raspon

        return image, label
```

Slika 2.7. Prikaz Python koda za prilagođeni dataloader

Tri parametra inicijalizacije za klasu "CustomDataLoader" su "data_dir", "batch_size" i "image_size". Ovi parametri određuju predviđenu veličinu slike za pretprocesiranje, željenu veličinu serije (eng. batch) i putanju prema direktoriju sa slikovnim podacima (Prikazano na Slici 2.8.).

```python
class CustomDataLoader:
    def __init__(self, data_dir, batch_size, image_size):
        self.data_dir = data_dir
        self.batch_size = batch_size
        self.image_size = image_size
```

Slika 2.8. Prikaz konstruktora klase CustomDataLoader

Metoda 'load_data' mjesto je gdje se nalazi primarna funkcionalnost dataloader-a. Koristeći "os.listdir", dohvaća se popis naziva slikovnih datoteka iz priloženog direktorija podataka. Put

direktorija podataka i naziv svake slikovne datoteke kombiniraju se u popis putova do slika. Nakon toga raščlanjuje nazive slikovnih datoteka kako bi izdvojio oznake (eng, label). U ovom slučaju oznaka je predstavljena s prvih pet znakova naziva datoteke. Oznake se zatim transformiraju pomoću "tf.one_hot" u one-hot kodirane tenzore. Putovi slika i oznake zatim se kombiniraju u TensorFlow skup podataka pomoću "tf.data.Dataset.from_tensor_slices".

```
def load_data(self):
    image_files = os.listdir(self.data_dir)
    image_paths = [os.path.join(self.data_dir, file) for file in image_files]
    labels = [int(((os.path.basename(file))[:5])) for file in image_files]
    num_classes = max(labels)
    labels = tf.one_hot(labels, num_classes)

    # kreira TensorFlow Dataset iz slika i labela
    dataset = tf.data.Dataset.from_tensor_slices((image_paths, labels))
```

Slika 2.9. Prikaz load_data metode u CustomDataLoader klasi

Metoda "_load_and_preprocess_image", koja učitava i pretprocesira fotografije, mapirana je na skup podataka. Metoda "_load_and_preprocess_image" koristi "tf.io.read_file" za čitanje slikovne datoteke i "tf.image.decode_jpeg" za njezino dekodiranje. Vraćaju se i učitana slika i pridružena oznaka.

```
def _load_and_preprocess_image(self, image_path, label)
    # Učita i predprocesira sliku
    image = tf.io.read_file(image_path)
    image = tf.image.decode_jpeg(image,channels=3)
      # Normalizira vrijednosti pixela u [0, 1] raspon

    return image, label
```

Slika 2.10. Prikaz _load_and_preprocess_image metode u CustomDataLoader klasi

Koristeći "dataset.shuffle" i veličinu međuspremnika (eng. buffer) koja je jednaka broju putanja slike, skup podataka se zatim nasumično miješa (eng. shuffle). Zatim se pomoću "dataset.batch" odvaja u serije veličine "batch_size". Vraća se skup podataka koji se sastoji od prethodno obrađenih fotografija i oznaka koje idu uz njih.

```
        # Ucitava i predprocesira slike
        dataset = dataset.map(self._load_and_preprocess_image)

        # Shuffle and batch the dataset
        dataset = dataset.shuffle(buffer_size=len(image_paths))
        dataset = dataset.batch(self.batch_size)

        return dataset
```

Slika 2.11. Prikaz nastavka load_data metode

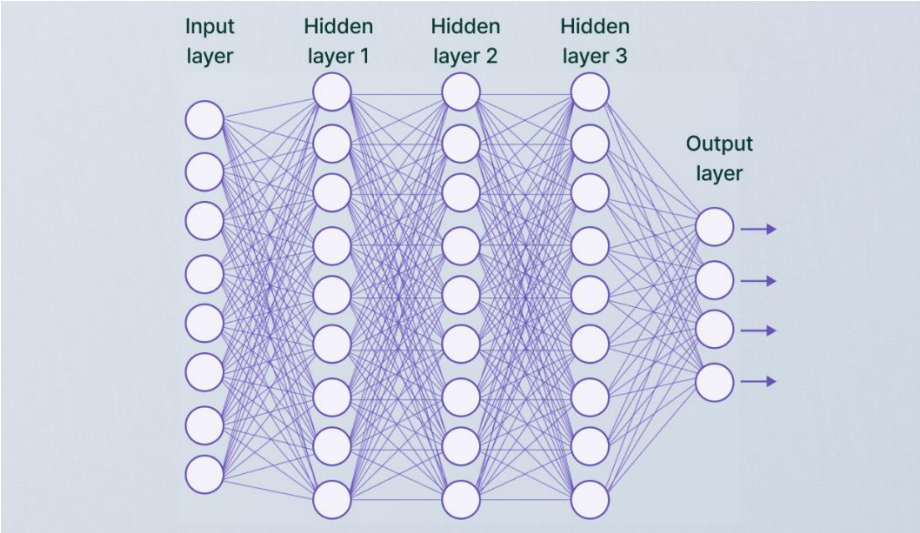
Klasa se zatim poziva u glavnom programu, gdje se definiraju ulazne putanje, veličina serije, te veličine slika (što ovisi o skupu podataka koji se koristi).

```
training_dir = r"<path>\CASIA_interval_dataset\training"
validation_dir = r"<path>\CASIA_interval_dataset\validation"


batch_size = 32
image_size = (280, 320)

training_loader = CustomDataLoader(training_dir, batch_size, image_size)
training = training_loader.load_data()
validation_loader = CustomDataLoader(validation_dir, batch_size, image_size)
validation = validation_loader.load_data()
```

Slika 2.12. Prikaz pozivanje CustomDataLoader klase za kreiranje training i validation skupova podataka

2.4. Konvolucijske neuralne mreže

Duboko učenje (eng. deep learning) je grana strojnog učenja koja se posebno fokusira na duboke neuralne mreže (eng. deep neural networks), koje su umjetne neuralne mreže s nekoliko slojeva (prikazano u Slici 2.13.). Algoritmi dubokog učenja snažniji su i prilagodljiviji od konvencionalnih algoritama strojnog učenja jer izravno uče i izdvajaju složene obrasce i karakteristike iz sirovih podataka.

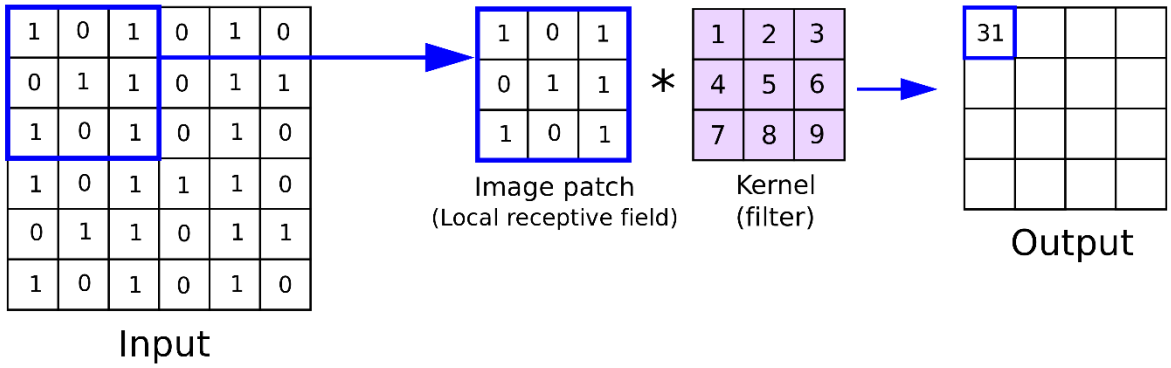


Slika 2.13. Prikaz strukture umjetne neuralne mreže [3]

Konvolucijske neuralne mreže klasa su arhitektura dubokog učenja posebno dizajniranih za analizu vizualnih podataka. CNN-ovi su inspirirani mehanizmima vizualne obrade ljudskog mozga i sastoje se od više međusobno povezanih slojeva umjetnih neurona.

2.4.1. Elementi konvolucijskih neuralnih mreža

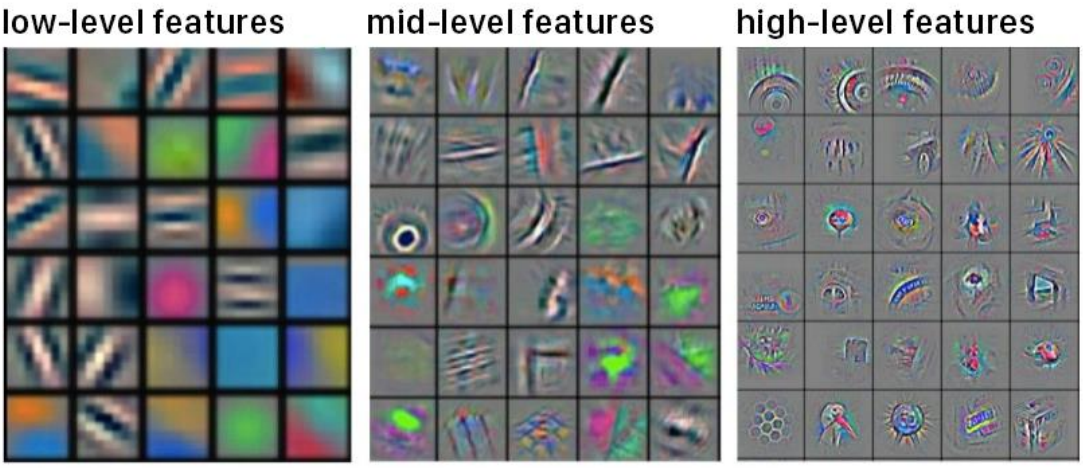
Konvolucijski sloj (eng. convolution layer) je početni sloj u CNN-u. Konvolucijski sloj sastoji se od niza malih matrica zvanih kerneli, koji funkcioniraju kao filtri. Svaki filtar koristi konvoluciju za skeniranje ulaznih podataka. Množenjem po elementima, filtar se konvolvira s ulaznom mapom značajki (eng. feature map). Unutar receptivnog polja, svaki element filtera se množi s odgovarajućim ulaznim elementom. Vrijednosti koje su pomnožene zatim se zbrajaju kako bi se dobila jedna vrijednost koja označava aktivaciju filtra na tom određenom mjestu na karti značajki. Kako bi se uvela nelinearnost, zbroj se zatim šalje kroz aktivacijsku funkciju, kao što je ReLU (Rectified Linear Unit).



Slika 2.14. Prikaz operacije konvolucije u konvolucijskom sloju [4]

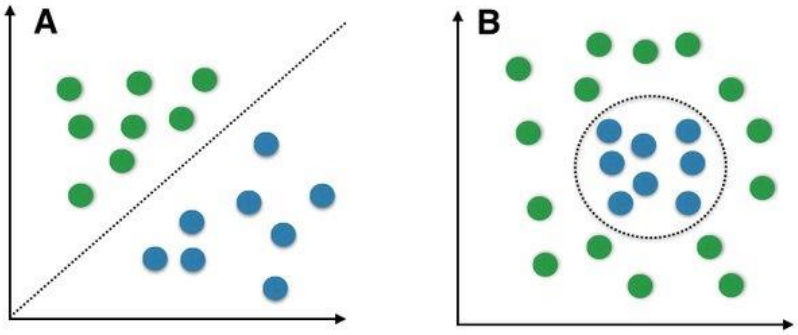
Različiti parametri i hiperparametri utječu na performanse i ponašanje konvolucijskih slojeva. Dubina ili količina kanala u izlaznoj karti značajki ovisi o broju filtara u konvolucijskom sloju. Svaki filtar uči različite obrasce i značajke. Receptivno polje i volumen podataka ovise o veličini filtara (3x3, 5x5 i 7x7 tipične su veličine filtera). Korak (eng. stride) kontrolira koliko veliki korak filtar poduzima kada prelazi ulaznu kartu značajki. Prostorne dimenzije izlazne karte značajki smanjuju se većim korakom.

Budući da bilježi regionalne prostorne obrasce u karti ulaznih značajki, konvolucijski sloj je bitan za CNN-ove. Kao rezultat toga, mreža može automatski naučiti značajke od niske razine do visoke razine. CNN-ovi mogu izdržati manje prijevode u ulazu zahvaljujući funkciji konvolucije. Isti filtri koriste se u cijeloj karti ulaznih značajki, omogućujući mreži da identificira uzorke bez obzira na to gdje se nalaze. Višestruki konvolucijski slojevi mogu se naslagati kako bi CNN-ovi mogli naučiti hijerarhijske prikaze. Dublji slojevi uče kompliciranе i apstraktne elemente, dok niži slojevi hvataju jednostavne značajke poput rubova i tekstura.



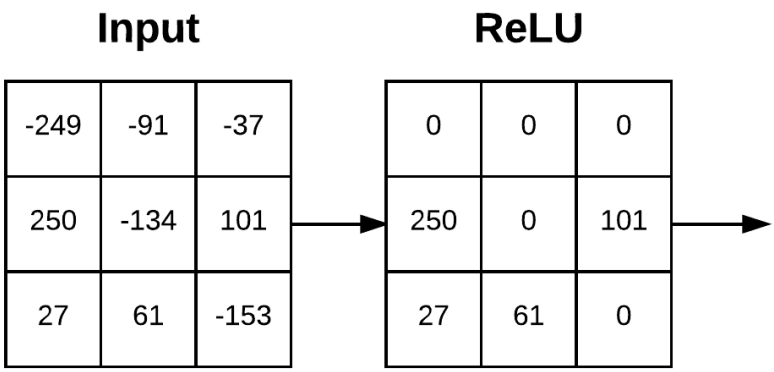
Slika 2.15. Prikaz hvatanja značajka [5]

Aktivacijska funkcija primjenjuje se na kartu značajki nakon konvolucijskog sloja. Izlaz konvolucijskog sloja dobiva nelinearnu funkciju pomoću aktivacijskog sloja. Mreža može naučiti zamršene, nelinearne korelacije između ulaznih značajki i njihovih odgovarajućih izlaznih aktivacija zahvaljujući svojoj nelinearnosti. To omogućuje mreži da pokupi složene prikaze koji su potrebni za preciznu klasifikaciju.



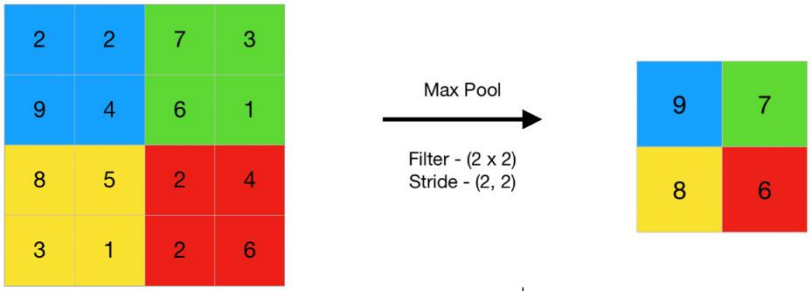
Slika 2.16. Prikaz nelinearnosti [6]

Za uvođenje nelinearnosti, aktivacijski sloj koristi niz aktivacijskih funkcija. Ispravljena linearna jedinica (ReLU), sigmoid i hiperbolički tangens (tanh) neke su od redovito korištenih aktivacijskih funkcija u CNN-ovima. Priroda problema i značajke skupa podataka utječu na odabir aktivacijske funkcije. Raspon izlaza aktivacijske funkcije treba odgovarati planiranom prikazu izlaza. Na primjer, dok je softmax prikladan za višeklasnu klasifikaciju, sigmoid je prikladan za binarnu klasifikaciju.



Slika 2.17. Prikaz mape značajki nakon ReLu funkcije [7]

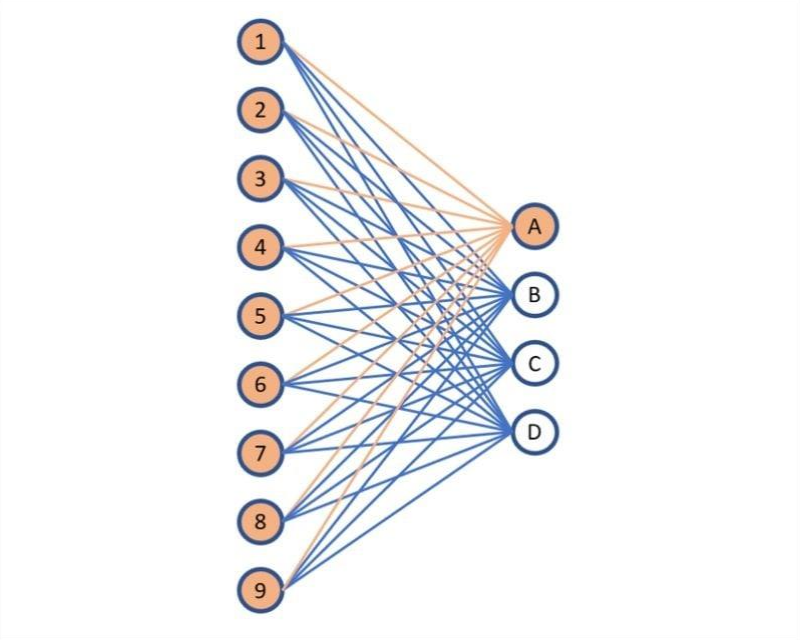
Sloj udruživanja (eng. pooling layer) vitalni je dio CNN-a i bitan je za ekstrakciju značajki i prostorno smanjivanje uzorkovanja. On pomaže u smanjenju računalne složenosti mreže downsampling-om mapa značajki. Iz ulaznih mapa značajki sloj dodatno izdvaja značajke koje se ističu. To se postiže kondenzacijom podataka iz obližnjih područja i identificiranjem istaknutih obilježja koja su manje osjetljiva na manje prostorne translacije. Mapa značajki se downsample-a u sloju udruživanja pomoću različitih tehnika udruživanja. Maksimalno udruživanje, prosječno udruživanje i globalno udruživanje neke su od često korištenih operacija udruživanja. Na ponašanje i izlaz sloja udruživanja utječu brojni parametri. Veličina (eng. pool size) utječe na prostorni doseg operacije (2x2 i 3x3 tipične su veličine), s mapama značajki odvojenim u odjeljke koji se ne preklapaju. Prozor udruživanja (eng. pooling window) prolazi kroz mapu ulaznih značajki u definiranim koracima prema koraku. Veličina izlaza se smanjuje, a downsampling postaje izraženiji s većim korakom.



Slika 2.18. Prikaz operacije maksimalnog udruživanja [8]

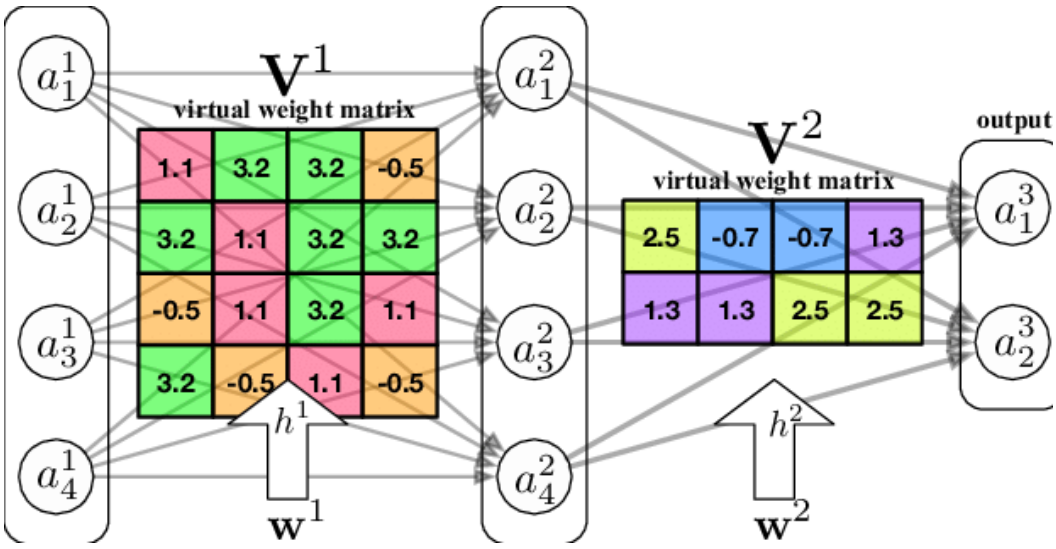
Nakon konvolucijskih slojeva i slojeva udruživanja slijedi potpuno povezani sloj. Potpuno povezani sloj, koji oponaša strukturu konvencionalnih umjetnih neuralnih mreža, povezuje svaki

neuron u prethodnom sloju sa sljedećim slojem, za razliku od prethodnih slojeva, koji rade na lokalnim prostornim podacima. To mreži omogućuje predviđanje visoke razine prikupljanjem globalnih ovisnosti i odnosa između značajki. U skladu s zadatkom potpuno povezani sloj miješa naučene informacije iz prethodnih slojeva i koristi ih za izradu predikcija. Koristi značajke koje su izdvojene za stvaranje izlaza koji odražava prosudbu mreže.



Slika 2.19. Prikaz potpuno povezanog sloja [9]

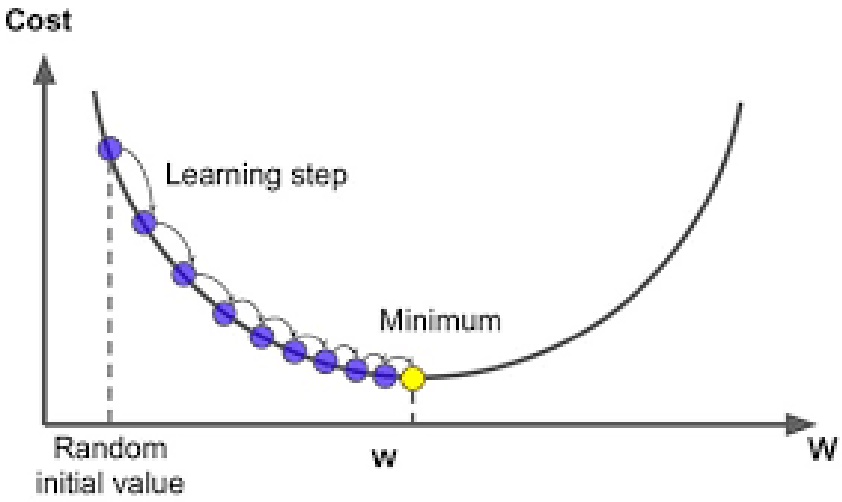
U dubokom učenju, jedan od ključnih elemenata koji omogućuju izvanredne mogućnosti neuralnih mreža je koncept težina. Težine u neuralnoj mreži predstavljaju snagu veza između određenih neurona. Mreža može transformirati dolazne podatke u korisne izlaze zahvaljujući ovim vezama. Ukupna izvedba i ponašanje mreže pod utjecajem je svake vrijednosti težina. Kako bi mreža imala koristi od podataka o obuci i razvila svoje sposobnosti predviđanja tijekom vremena, težine se moraju prilagoditi.



Slika 2.20. Prikaz uloge težina u neuralnoj mreži [10]

U konvencionalnim metodama strojnog učenja, "feature engineering" korišten je za ručnu promjenu težina od strane ljudskih stručnjaka. Iz pristiglih podataka, stručnjaci bi pažljivo odabrali i ručno izradili relevantne značajke, a zatim bi svakoj značajki dodijelili određenu težinu. Znanje o domeni i stručnost ljudskog dizajnera bili su presudni za učinkovitost ovih strategija. Ručna modifikacija težine dala je određeni stupanj interpretabilnosti i kontrole, ali je imala i značajne nedostatke. Prilagođavanje utega bio je naporan postupak koji je oduzimao puno vremena, te zahtijevao puno znanja i "trial and error" iteracija. Također budući da je bilo teško ručno uhvatiti sve relevantne elemente, ručna prilagodba često je imala problema s visoko dimenzionalnim i kompliciranim podacima. Ta su ograničenja dovela do pomaka dubokih neuralnih mreža prema autonomnoj prilagodbi težine.

Duboke neuralne mreže posebno su dobre u automatskom učenju i modificiranju težina tijekom treninga. Mreža obrađuje velike količine označenih podataka tijekom treninga, a opetovane prilagodbe težine koriste se kako bi se poboljšala točnost predviđanja. Propagacija unatrag (eng. backpropagation) je primarni algoritam koji se koristi za podešavanje težina u dubokim neuralnim mrežama. Izračunavanjem gradijenta pogreške mreže s obzirom na svaku težinu propagacija unatrag može odrediti kako bi se svaka težina trebala promijeniti da bi se poboljšala izvedba mreže. Gradijentni spust ravnomjerno smanjuje pogrešku mreže iterativnim podešavanjem težina u smjeru suprotnom od gradijenta (Prikazano na Slici 2.21.). Ova iterativna metoda nastavlja se sve dok mreža ne radi zadovoljavajuće ili dok se ne zadovolji unaprijed određeni kriterij zaustavljanja. Mreža učinkovito ažurira težine pomoću ovih informacija o gradijentu koje se prenose unatrag [11][12][13].

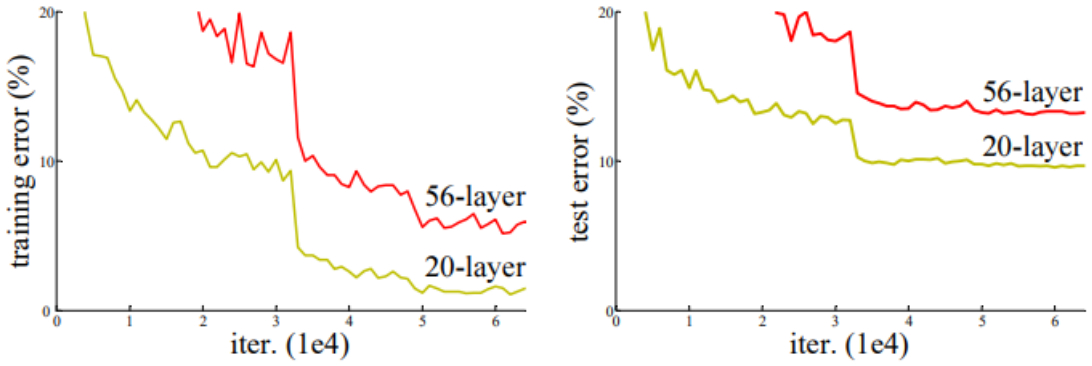


Slika 2.21. Prikaz gradijentnog spusta [14]

2.4.2. ResNet 50

ResNet je revolucionarna arhitektura dubokog učenja koja je utjecala na područje računalnog vida. ResNet je promijenio obuku duboke neuralne mreže uvođenjem ideje rezidualnog učenja (eng. residual learning). Kako se dubina mreže širila, tradicionalne duboke mreže naišle su na probleme s nestajućim ili eksplodirajućim gradijentima. Dizajniran 2012. za ImageNet, model AlexNet bio je 8-slojna konvolucijska neuralna mreža. Slaganjem 3x3 konvolucijskih slojeva, neuralna mreža koje je 2014. stvorila Visual Geometry Group (VGG) sa Sveučilišta u Oxfordu približila su se broju od 19 slojeva. Ali problem "degradacije", koji je uzrokovan slaganjem više slojeva, brzo je smanjio točnost treninga (prikazano u Slici 2.22.).

Dublja mreža ne bi trebala proizvesti veći gubitak uvježbavanja od svoje pliće mreže, ako se ta dublja mreža može konstruirati njezinom plićom mrežom složenom s dodatnim slojevima [15].



Slika 2.22. Prikaz rezultata dobivenih testiranjem 20 slojne mreže i 56 slojne mreže [15]

Ovaj problem je riješen rezidualnim učenjem uvođenjem preslikavanja identiteta ili "skip" veza. Temeljni koncept je da je lakša optimizacija omogućena mrežom koja uči odstupanje između željenog mapiranja i trenutnog mapiranja. Razmotrite podmrežu u modelu višeslojne neuralne mreže koja ima određeni broj (npr. 2 ili 3) naslaganih slojeva. Temeljna operacija koju izvodi ova podmreža označena je kao $H(x)$, gdje je x njen ulaz. Ova podmreža je ponovno parametrizirana korištenjem koncepta rezidualnog učenja, koji omogućuje slojevima da predstavljaju rezidualnu funkciju $F(x) = H(x) - x$. Izlaz y ove podmreže predstavljen je kao:

$$y = F(x) + x \quad (2.1.)$$

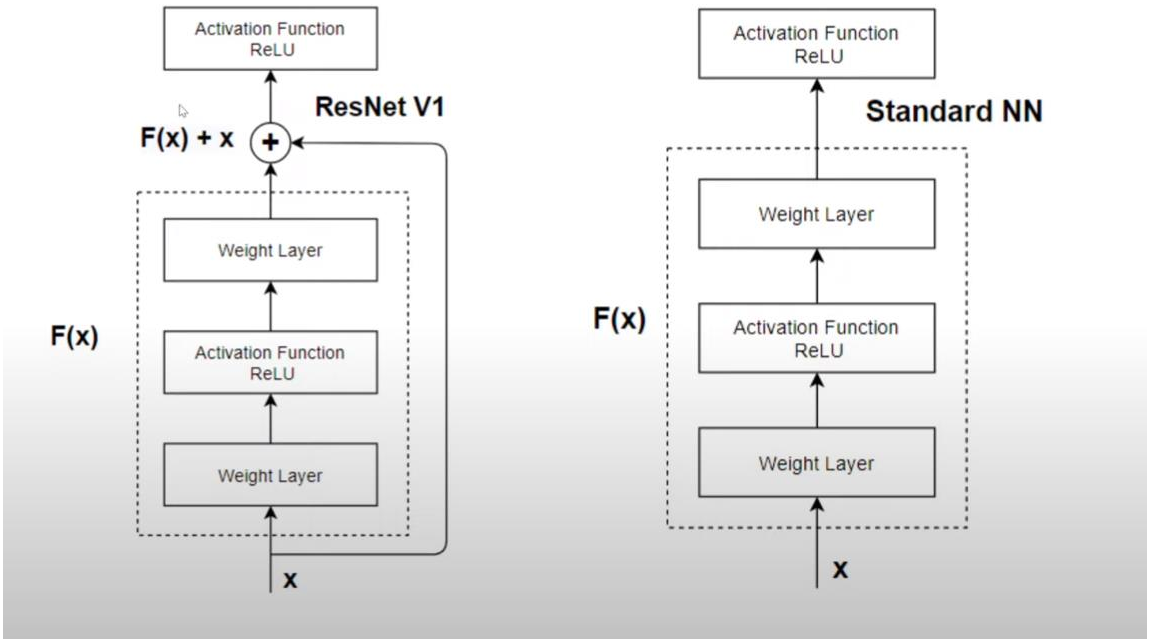
gdje je:

y izlaz iz podmreže

$F(x)$ rezidualna funkcija

x ulaz u podmrežu

Ova podmreža se naziva "rezidualni blok" (Prikazano na Slici 2.23.). Niz rezidualnih blokova složen je kako bi se stvorila duboka rezidualna mreža. Skip veza koja izvodi mapiranje identiteta i povezuje ulaz rezidualnog bloka s njegovim izlazom se odnosi na "$+x$" dio u $y = F(x) + x$ [16].



Slika 2.23. Prikaz rezidualnog bloka u odnosu na tradicionalnu strukturu neuralne mreže

ResNet-50 je duboka mreža koja se sastoji od 50 slojeva, uključujući skip vezu, udruživanje, konvoluciju i potpuno povezane slojeve. Mreža je organizirana u niz stupnjeva, od kojih svaki ima određeni broj rezidualnih blokova i filtara. ResNet-50 je član ResNet obitelji, a kreirao ga je Kaiming He et al. u Microsoft Research-u.

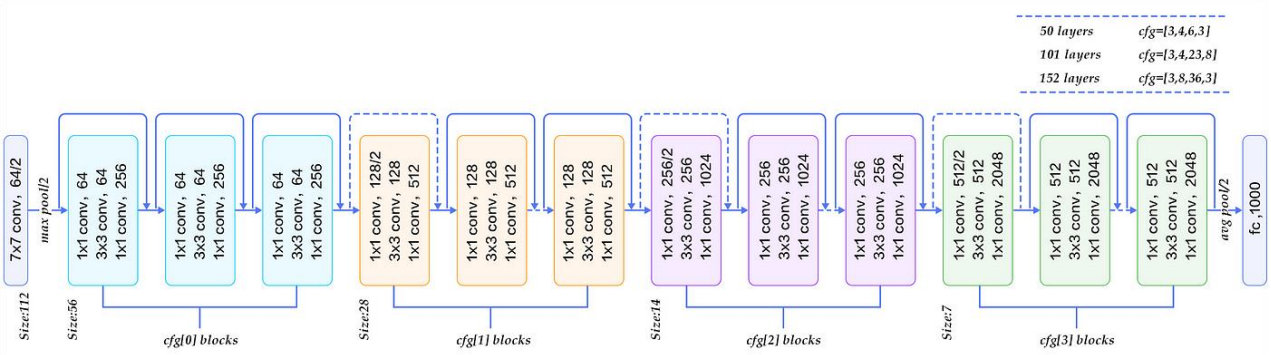
layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Slika 2.24. Prikaz arhitekture ResNeta-50 [17]

Opću arhitekturu možemo sažeti na sljedeći način: Konvolucija sa 64 različita kernela, svaka s korakom veličine 2, i veličinom kernela 7 * 7, što nam daje nam 1 sloj. Nakon toga imamo sloj

maksimalnog udruživanja s veličinom koraka od 2. Sljedeća konvolucija sastoji se od tri sloja: kernel 1*1,64, kernel 3*3,64 i konačno kernel 1*1,256. Ova tri sloja se ponavljaju ukupno 3 puta, što daje 9 slojeva u ovoj fazi. Sljedeće je prikazan kernel od 1 * 1,128, nakon kojeg slijedi kernel od 3 * 3,128 i, na kraju, kernel od 1 * 1,512. Ovaj postupak ponavlja se 4 puta za ukupno 12 slojeva. Nakon toga, imamo kernel veličine 1 * 1,256, nakon kojeg slijede još dva kernela veličine 3 * 3,256 i veličine 1 * 1,1024; ovo se ponavlja šest puta, dajući nam ukupno 18 slojeva. Na kraju je dodan kernel 1 * 1,512, a zatim još dva kernela od 3 * 3,512 i 1 * 1,2048. Ovaj proces je učinjen tri puta, dajući nam ukupno 9 slojeva. Zatim se radi prosječno udruživanje, te je model završen s potpuno povezanim slojem sastavljenim od 1000 čvorova i funkciju softmax što nam daje još jedan sloj.

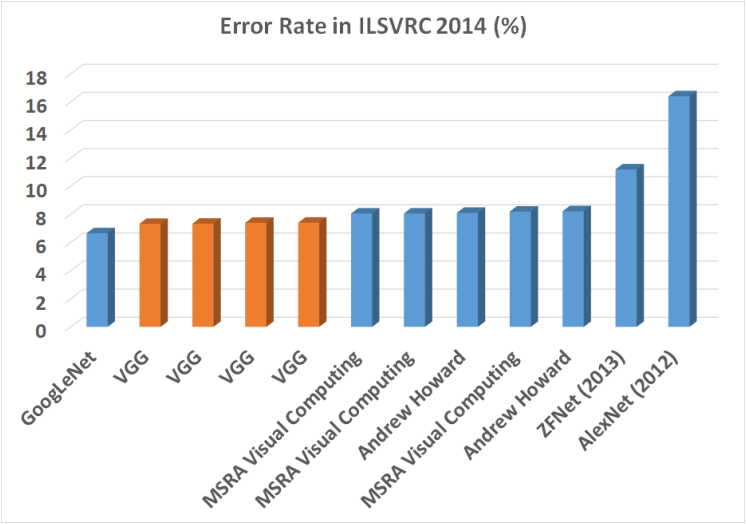
Zbrojivši to, dobivamo 1 + 9 + 12 + 18 + 9 + 1 = 50 slojeva mreže.



Slika 2.25. Logički prikaz arhitekture ResNeta-50 [13]

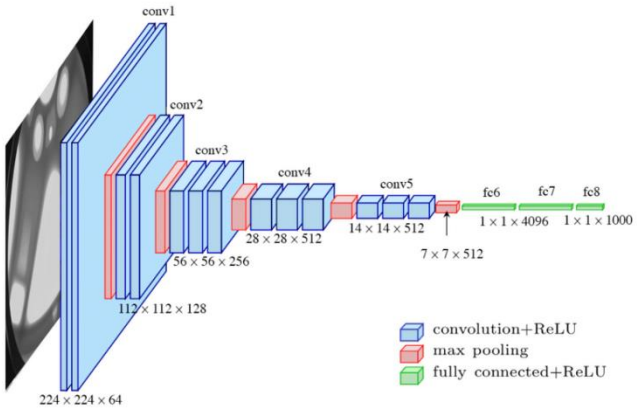
2.4.3. VGG 16

Arhitektura konvolucijske neuralne mreže (CNN) VGG16, koja je dobila naziv od Visual Geometry Group-a na Sveučilištu u Oxfordu, prepoznata je po svojoj dubini, jednostavnosti i izvanrednim performansama. Model VGG prvi su predložili Andrew Zisserman i Karen Simonyan 2013. godine, a 2014. godine napravljen je prototip za ImageNet Challenge. U ImageNetu, skupu podataka koji sadrži više od 14 milijuna fotografija za obuku ove 1000 klasa predmeta, model VGG16 može doseći točnost ispitivanja od 92,7%. VGG16 je bio istaknuti model u tom natjecanju [18].



Slika 2.26. Prikaz rezultata natjecanja ILSVRC 2014 [19]

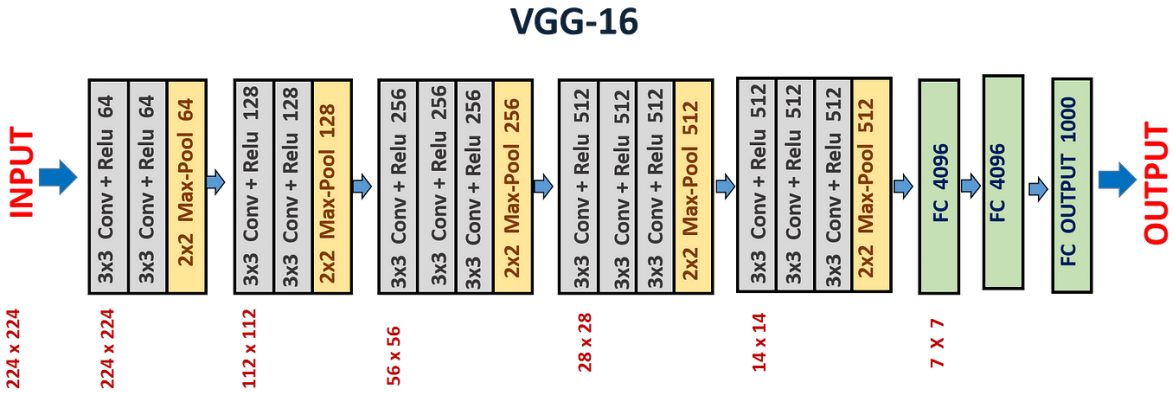
Paradigma od VGG16 znatno se razlikovala od ranijih, uspješnih modela. U usporedbi s AlexNetom, koji je koristio receptivno polje 11x11 s korakom od 4 piksela, VGG16 je koristio mnogo manje receptivno polje 3x3 i korak od 1 piksela. Kombinacijom 3x3 filtera postiže se šire receptivno polje. Konzistentni 3×3 konvolucijski filtri čine mrežu lakom za upravljanje. Kada se koriste brojni manji slojevi za razliku od jednog velikog sloja, funkcije odlučivanja su poboljšane i mreža može brže konvergirati. Jedan od razloga tome je to što je prisutno više nelinearnih aktivacijskih slojeva.



Slika 2.27. Grafički prikaz VGG 16 arhitekture [20]

Postoji 13 konvolucijskih slojeva u VGG16, a nakon svakog dolazi ReLU aktivacijska funkcija. Iz ulazne slike ovi slojevi postupno izdvajaju značajke niske razine u značajke visoke razine. Algoritam VGG16 dodaje slojeve maksimalnog udruživanja s filtrom 2x2 i korakom od 2 nakon svakog skupa konvolucijskih slojeva. Tri potpuno povezana sloja, od kojih svaki ima 4096

neurona, čine posljednji odjeljak VGG16. Funkcija aktivacije softmax koristi se za klasifikaciju u izlaznom sloju, koji ima onoliko neurona koliko je klasa u skupu podataka [21].

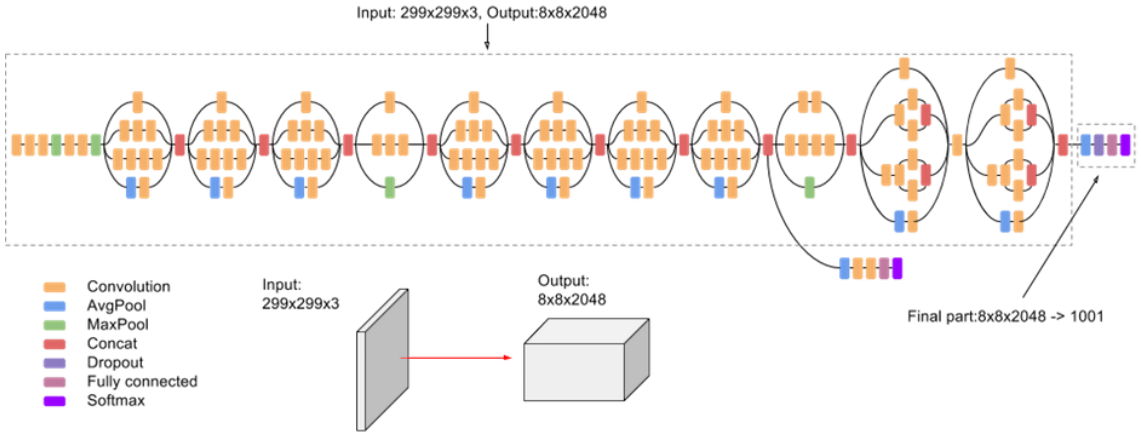


Slika 2.28. Dijagramski prikaz VGG16 arhitekture [22]

Kao i drugi CNN-ovi, VGG16 se trenira korištenjem stohastičkog gradijentnog spuštanja i vračanja unatrag (eng.backpropagation). Prethodna obuka na ekspanzivnim skupovima podataka kao što je ImageNet i fino podešavanje (eng. fine tuning) na određenim zadacima ili skupovima podataka tipične su metode obuke za VGG16. Prethodno obučeni modeli iz VGG16 stekli su popularnost kao mjesto za početak prijenosnog učenja (eng. transfer learning).

2.4.4. Inception V3

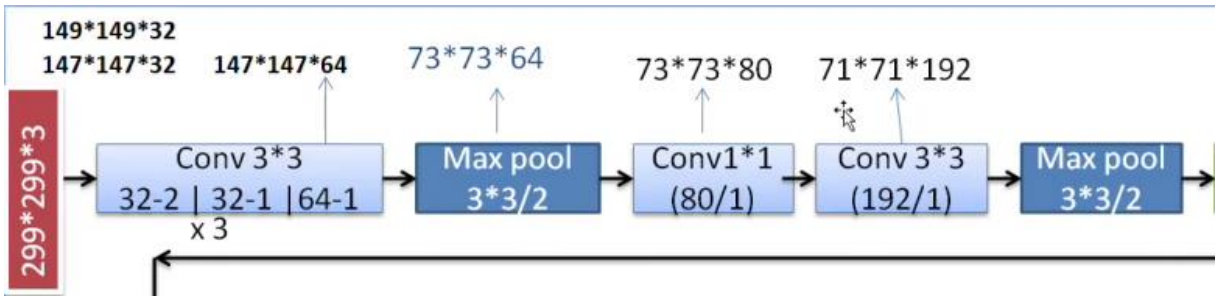
Google Research predstavo je arhitekturu konvolucijske neuralne mreže Inception V3 2015. Ona ima vrhunsku izvedbu na brojnim mjerilima, uključujući ImageNet Large-Scale Visual Recognition Challenge (ILSVRC), a namijenjena je za klasificiranje slika [23].



Slika 2.29. Pojednostavljeni prikaz arhitekture Incetiona V3 [17]

Inception V3 model ima ukupno 42 sloja, što je malo više od Inception V1 i V2 modela.

Prva komponenta mreže koja obrađuje dolaznu sliku je matični (eng. stem) modul. Sastoji se od niza konvolucijskih slojeva koji kontinuirano povećavaju broj kanala dok smanjuju prostorne dimenzije slike. Započinje s tri 3x3 konvolucijska sloja, zatim prelazi na sloj maksimalnog udruživanja, 1x1 konvolucijski sloj, još jedan 3x3 konvolucijski sloj, zatim ponovno sloj maksimalnog udruživanja. Prije prijenosa ulazne slike u početne blokove, zadatak osnovnog sloja je izdvojiti osnovne značajke slike i minimizirati njezine prostorne dimenzije.



Slika 2.30. Prikaz matičnog modula

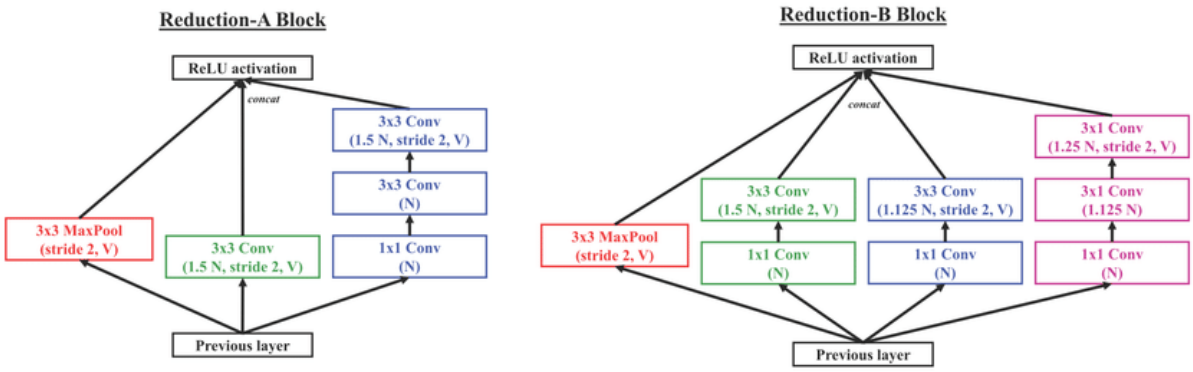
Nakon toga podaci ulaze u "Inception" blok. Inception V3 ima ukupno 11 Inception blokova, od kojih je svaki sastavljen od brojnih simultanih konvolucijskih grana. Temeljni koncept koji stoji iza Inception blokova je istovremena upotreba različitih veličina filtera za prikupljanje podataka na različitim razinama apstrakcije. Kao rezultat toga, mreža može učinkovito koristiti svoje računalne resurse i prikupljati lokalne i globalne informacije. Četiri osnovne grane Inception bloka

su grane 1x1, 3x3, 5x5 i maksimalno udruživanje. Izlaz svake dimenzije kanala zatim se konkatinira kako bi postao ulaz za sljedeći sloj. Mreža može uhvatiti različita svojstva zahvaljujući ovom konkatiniranju.

Tehnike faktorizacije koriste se u Inception V3 kako bi se smanjila računalna složenost modela. Zamjenjuje veće konvolucije poput 5x5 mješavinom konvolucija 1x1 i 3x3. Smanjenjem broja ulaznih kanala, konvolucije 1x1 djeluju kao slojevi smanjenja dimenzionalnosti, dok sljedeće konvolucije 3x3 preuzimaju prostorne informacije. Osim toga, postoje tehnike faktorizacije koje smanjuju broj parametara dijeljenjem konvolucije u dvije asimetrične konvolucije (npr. 3x3 konvolucijski filtar podijeljen je na 1x3 i 3x1 konvolucijske filtre).

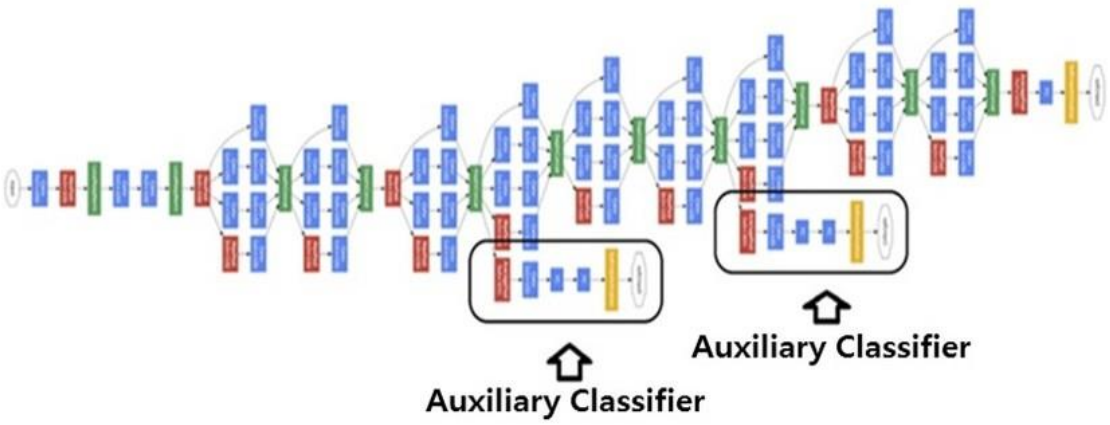
Slika 2.31. Prikaz strukture Inception blokova [24]

Inception V3 koristi redukcijske blokove za povećavanje broja kanala dok istovremeno smanjuje prostorne dimenzije mapa značajki. Između početnih blokova nalaze se dva redukcijska bloka. Kako bi postigli svoj cilj, ovi blokovi kombiniraju maksimalno udruživanje, 1x1 konvoluciju i 3x3 konvoluciju. Svrha redukcijskih blokova je agregirati važne značajke uz odbacivanje irelevantnih informacija.



Slika 2.32. Prikaz strukture različitih tipove redukcijskog bloka

Pomoćni klasifikatori prisutni su na srednjim slojevima u Inception V3 kako bi riješili problem nestajanja gradijenta. Ove dodatne funkcije gubitaka i gradijenti koje uvode ovi pomoćni klasifikatori pomažu u širenju gradijenata dalje u mrežu. Oni služe kao nadzornici i regulatori, pomažući mreži u učenju diskriminirajućih osobina.



Slika 2.33. Prikaz pozicije pomočnih klasifikatora u Inception V3 arhitekturi [25]

Nakon posljednjeg Inception bloka, globalno prosječno udruživanje se primjenjuje kako bi se dobio vektor značajki fiksne veličine. Slijede potpuno povezani slojevi koji izvode klasifikaciju na generiranim značajkama i funkcija aktivacije softmax koristi se za generiranje vjerojatnosti klasa [26][27].

2.4.5. Implementacija modela

Navedeni kod objašnjava kako implementirati VGG16 arhitekturu za treniranje modela.

```
pretrained_model= tf.keras.applications.VGG16(
                    include_top=False,
                    input_tensor = None,
                    input_shape=(img_height,img_width,3),
                    pooling='avg',
                    classes=num_classes,
                    classifier_activation="softmax",
                    weights='imagenet')

for layer in pretrained_model.layers:
        layer.trainable=False

custom_model.add(pretrained_model)
custom_model.add(Flatten())
custom_model.add(Dense(512, activation='relu'))
custom_model.add(Dense(num_classes, activation='softmax'))

custom_model.summary()

custom_model.compile(
    optimizer=Adam(learning_rate=0.001),
    loss='categorical_crossentropy',
    metrics=[
        'accuracy',
        Precision(),
        Recall(),
        tf.keras.metrics.CategoricalAccuracy(name='categorical_accuracy'),
        tf.keras.metrics.AUC(name='auc'),
        tf.keras.metrics.TruePositives(name='tp'),
        tf.keras.metrics.FalsePositives(name='fp'),
        tf.keras.metrics.TrueNegatives(name='tn'),
        tf.keras.metrics.FalseNegatives(name='fn'),
        tfa.metrics.F1Score(num_classes, name='f1_score')
    ]
)
epochs=100
history = custom_model.fit(
  training,
  validation_data=(validation),
  epochs=epochs
)
```

Slika 2.34. Prikaz koda za treniranje modela

Funkcija "tf.keras.applications.VGG16" koristi se za inicijalizaciju VGG16 modela na početku koda. Potpuno povezani slojevi (eng. fully connected layers) modela VGG16 uklanjaju se postavljanjem "include_top=False" jer će prilagođeni potpuno povezani slojevi biti dodani kasnije kako bi bolje zadovoljili naše zahtjeve za kategorizaciju. Model se inicijalizira pomoću težina (eng. weights) naučenih iz ImageNet skupa podataka, koji sadrži milijune označenih fotografija, pomoću opcije "weights="imagenet"". Argument "input_shape" navodi očekivane veličine ulaznih

slika, pri čemu "(img_height, img_width, 3)" označava broj piksela u visinu, broj piksela u širinu i 3 RGB kanala boja (promjenjive vrijednosti ovisne o skupu podataka).

```
custom_model = Sequential()


pretrained_model= tf.keras.applications.VGG16(
                    include_top=False,
                    input_tensor = None,
                    input_shape=(img_height,img_width,3),
                    pooling='avg',
                    classes=num_classes,
                    classifier_activation="softmax",
                    weights='imagenet')
```

Slika 2.35. Prikaz koda za učitavanje VGG16 arhitekture

Nakon toga imamo petlju koja iterira po slojevima od "pretrained_model" i postavlja njihov atribut "trainable" na "False" kako bi se održalo prethodno obučeno znanje u modelu VGG16 i izbjegao pretreniranje tijekom treninga. Težine ovih slojeva su zamrznute kao rezultat, što znači da se ne mijenjaju tijekom treninga.

```
for layer in pretrained_model.layers:
        layer.trainable=False
```

Slika 2.36. Prikaz koda petlje za zamrzavanje težina

"Pretrained_model" (VGG16) zatim se dodaje novogeneriranom "custom_model" koristeći "custom_model.add (pretrained_model)". Umjesto treniranja kompletnog modela VGG16 od nule, ovo nam uključivanje omogućuje korištenje prethodno obučenih slojeva VGG16 kao ekstraktora značajki za naš određeni problem klasifikacije. Nakon toga se dodaju tri dodatna sloja u "custom_model". Izlaz prethodnog sloja prvo se transformira u 1-dimenzionalni vektor slojem "Flatten" kako bi se pripremio za potpuno povezane sloja. Onda je dodan "Dense" sloj koji ima 512 neurona i ReLU aktivacijsku funkciju. Model zatim može predvidjeti najvjerojatniju klasu dodavanjem konačnog sloja "Dense" s "num_classes" (promjenjivo za svaki skup podataka) brojom neurona i funkcijom aktivacije "softmax".

```
custom_model.add(pretrained_model)
custom_model.add(Flatten())
custom_model.add(Dense(512, activation='relu'))
custom_model.add(Dense(num_classes, activation='softmax'))
```

Slika 2.37. Prikaz koda za dodavanje potpuno spojenih slojeva

Metoda "custom_model.summary()" zatim se koristi za pregled arhitekture modela i broja parametara koji se mogu trenirati u svakom sloju. Koristeći funkciju "compile", model je pripremljen za treniranje. Algoritam optimizacije određen je argumentom "optimizer", s vrijednošću "Adam" i stopom učenja od 0.001. "Loss" argument postavljen je na "categorical_crossentropy", što je često korištena funkcija gubitka za probleme klasifikacije s više klasa. Ona izračunava razliku između stvarnih oznaka klase i predviđenih oznaka, usmjeravajući model da minimalizira tu razliku tijekom treninga.

```
custom_model.summary()

custom_model.compile(
    optimizer=Adam(learning_rate=0.001),
    loss='categorical_crossentropy',
    metrics=[
        'accuracy',
        Precision(),
        Recall(),
        tf.keras.metrics.CategoricalAccuracy(name='categorical_accuracy'),
        tf.keras.metrics.AUC(name='auc'),
        tf.keras.metrics.TruePositives(name='tp'),
        tf.keras.metrics.FalsePositives(name='fp'),
        tf.keras.metrics.TrueNegatives(name='tn'),
        tf.keras.metrics.FalseNegatives(name='fn'),
        tfa.metrics.F1Score(num_classes, name='f1_score')
    ]
)
```

Slika 2.38. Prikaz koda za definiranje metrika

Evaluacijske metrike koje će se izračunati tijekom treninga i evaluacije definirane su u argumentu "metrics". Osim metrike "accuracy" koja određuje ukupnu točnost klasifikacije, dodane su sljedeće metrike:

- Precision() - izračunava metriku preciznosti, koja procjenjuje sposobnost modela da pravilno otkrije pozitivne uzorke među svim uzorcima za koje je model očekivao da budu pozitivni

- Recall() - izračunava metriku opoziva, koja procjenjuje točnost modela u odvajanju pozitivnih uzoraka od svih stvarnih pozitivnih uzoraka
- tf.keras.metrics.CategoricalAccuracy(name='categorical_accuracy') - izračunava metriku "categorical_accuracy", koja uspoređuje oznake koje je model predvidio sa stvarnim oznakama da odredi koliko su točna predviđanja modela
- tf.keras.metrics.AUC(name='auc') - koja mjeri kako su dobro predikcije rangirane, umjesto njihovih apsolutnih vrijednosti
- tf.keras.metrics.TruePositives(name='tp') - izračunava ukupan broj pravih pozitiva modela (slučajevi koji su točno klasificirani kao pozitivni nazivaju se pravim pozitivnim)
- tf.keras.metrics.FalsePositives(name='fp') - izračunava ukupan broj lažnih pozitiva modela (lažni pozitivi su slučajevi netočno klasificirani kao pozitivi)
- tf.keras.metrics.TrueNegatives(name='tn') - izračunava ukupan broj pravih negativa modela (slučajevi koji su točno klasificirani kao negativi nazivaju se pravim negativom)
- tf.keras.metrics.FalseNegatives(name='fn') - izračunava ukupan broj lažnih pozitiva modela (lažni negativi su slučajevi netočno klasificirani kao negativi)
- tfa.metrics.F1Score(num_classes, name='f1_score') - izračunava F1 rezultat, što je kombinacija preciznosti i opoziva [28]

$$Preciznost = \frac{TP}{TP + FP} \qquad TP = \text{Pravi pozitiv}$$

$$TN = \text{Pravi negativ}$$

$$Opoziv = \frac{TP}{TP + FN} \qquad FP = \text{Lažni pozitiv}$$

$$FN = \text{Lažni negativ}$$

$$F1 = 2 * \frac{preciznost * opoziv}{preciznost + opoziv}$$

Slika 2.39. Prikaz jednadžbi za preciznost, opoziv i F1 rezultat [29]

Varijabla "epochs" koristi se u kodu za postavljanje broja epoha obuke na 100. Svaka epoha predstavlja potpuni prolaz kroz skup podataka obuke. Model se zatim uvježbava metodom "fit". Za metodu se specificiraju skupovi podataka za trening i validaciju, zajedno s brojem epoha. Metoda ažurira težine modela na temelju izračunatog gubitka i odabranog algoritma optimizacije.

```
epochs=100
history = custom_model.fit(
  training,
  validation_data=(validation),
  epochs=epochs
)
```

Slika 2.40. Prikaz koda za početak treniranja modela

Za dobivanje rezultata od modela VGG16, ResNet50 i Inception V3 korišten je gore prikazan kod, jedina izmjena u kodu koja se događa je mijenjanje "pretrained_model" varijable, s tim da parametri koje prima ostaju isti.

```
pretrained_model= tf.keras.applications.VGG16(
                    include_top=False,
                    input_tensor = None,
                    input_shape=(img_height,img_width,3),
                    pooling='avg',
                    classes=num_classes,
                    classifier_activation="softmax",
                    weights='imagenet')
```

Slika 2.41. Prikaz uključivanja VGG16 arhitekture u model

```
pretrained_model= tf.keras.applications.ResNet50(
                    include_top=False,
                    input_tensor = None,
                    input_shape=(img_height,img_width,3),
                    pooling='avg',
                    classes=num_classes,
                    classifier_activation="softmax",
                    weights='imagenet')
```

Slika 2.42. Prikaz uključivanja ResNet50 arhitekture u model

```
pretrained_model= tf.keras.applications.InceptionV3(
                    include_top=False,
                    input_tensor = None,
                    input_shape=(img_height,img_width,3),
                    pooling='avg',
                    classes=num_classes,
                    classifier_activation="softmax",
                    weights='imagenet')
```

Slika 2.43. Prikaz uključivanja InceptionV3 arhitekture u model

3. REZULTATI USPOREDNE ANALIZE PROBLEMA

Nakon testiranja tri različita CNN modela na dva različita skupa podataka, dobiveni su rezultati prikazani u Tablici 3.1. i Tablici 3.2.

Tablica 3.1. Rezultati performansi CNN modela na Casia V4 interval skupu podataka

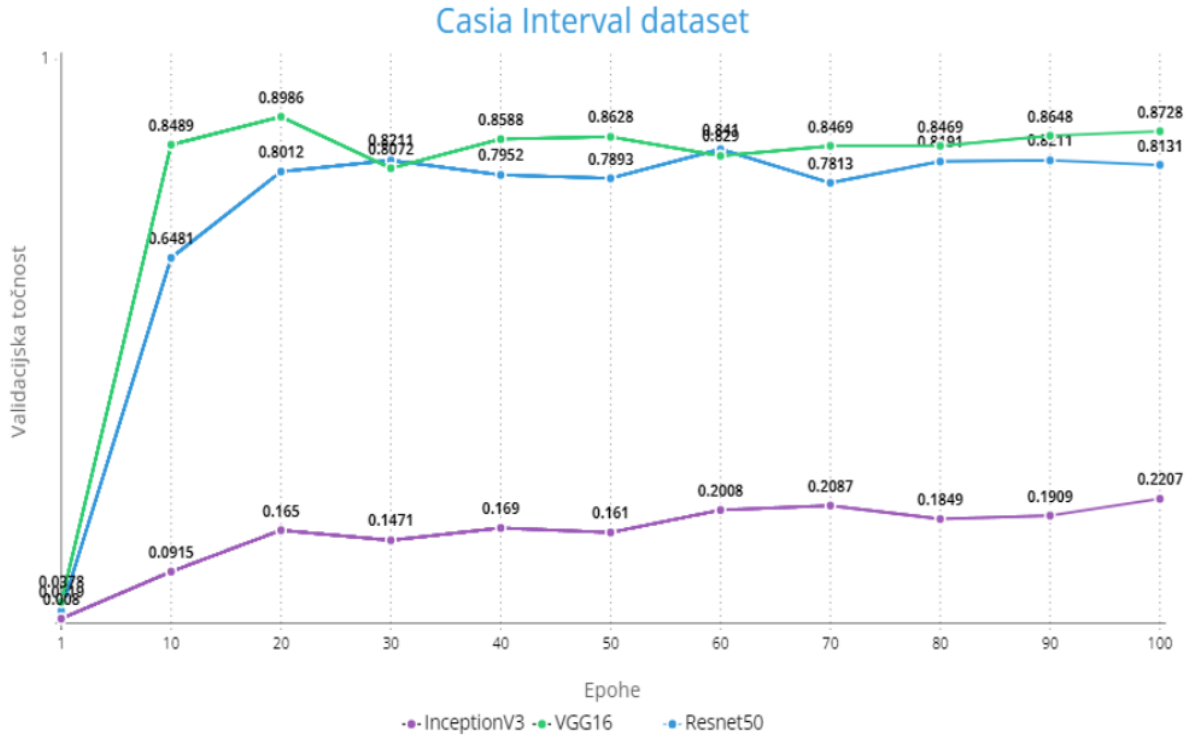
CNN model	Validacijska točnost	Validacijska preciznost	Validacijski opoziv	Validacijski AUC rezultat	Validacijski F1 rezultat
ResNet50	0.8131	0.8337	0.8088	0.9609	0.4298
VGG16	0.8728	0.8826	0.8685	0.9620	0.4622
InceptionV3	0.2207	0.4757	0.0976	0.8877	0.1039

Tablica 3.2. Rezultati performansi CNN modela na IIT Delhi Iris skupu podataka

CNN model	Validacijska točnost	Validacijska preciznost	Validacijski opoziv	Validacijski AUC rezultata	Validacijski F1 rezultat
ResNet50	0.8520	0.8670	0.8514	0.9614	0.8141
VGG16	0.8341	0.8326	0.8288	0.9321	0.7858
InceptionV3	0.6323	0.6479	0.6216	0.8906	0.5467

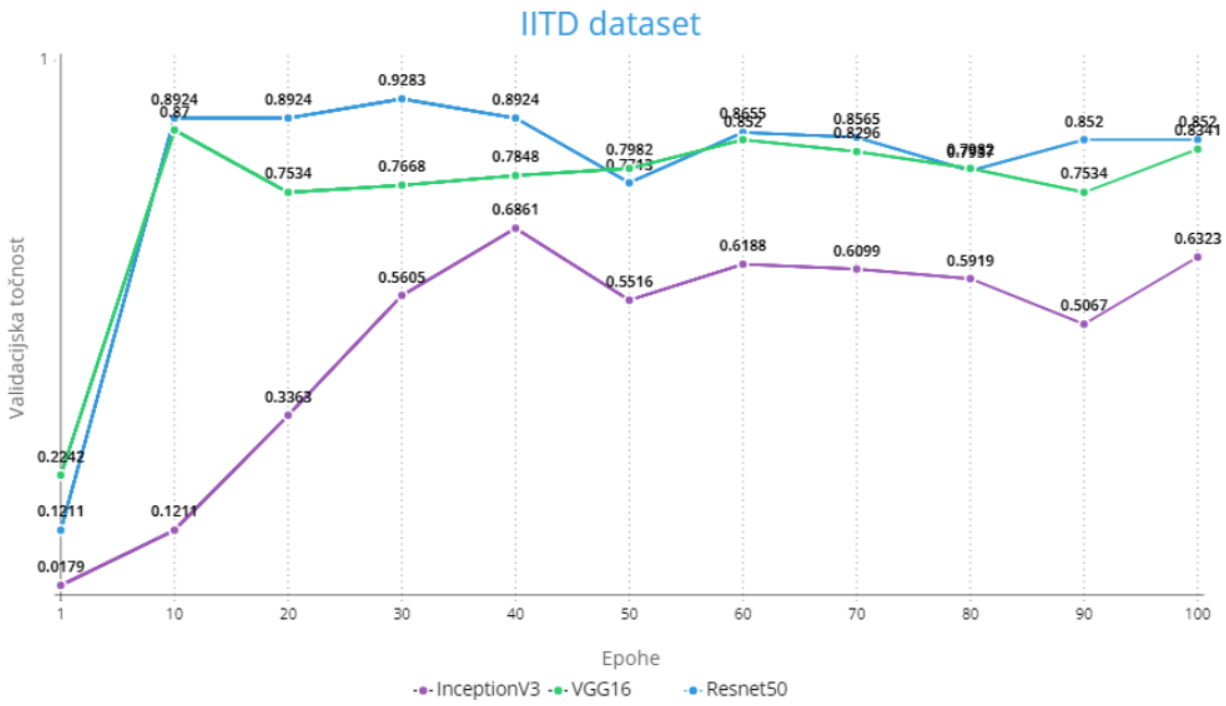
Rezultati performansi ResNet50 modela nad Casia Interval skupom podataka pokazuju prilično dobre performanse. Iznadprosječne vrijednosti za točnost, preciznost, opoziv i AUC pokazuju da model uspješno razlikuje uzorke šarenica u skupu podataka. Relativno nizak F1 rezultat međutim naglašava da ima područja za rast i može ukazivati na neuravnotežen odnos između opoziva i preciznosti. Rezultati pokazuju da VGG16 nadmašuje ResNet50 u smislu izvedbe. VGG16 ima bolju ravnotežu između preciznosti i opoziva koju sugerira viši F1 rezultat.

U usporedbi s druge dvije arhitekture, Inception V3 radi osjetno lošije na skupu podataka. Niski opoziv ukazuje na značajnu stopu lažno negativnih rezultata zbog nemogućnosti modela da točno identificira pozitivne uzorke.



Slika 3.1. Prikaz validacijske točnosti s obzirom na broj epoha za Casia Interval skup podataka

ResNet50 i VGG 16 također rade dobro na skupu podataka IIT Delhi, koji je manji i manje sofisticiran od prvog skupa podataka. Robusnost modela u otkrivanju pozitivnih uzoraka istaknuta je relativno visokim F1 rezultatima u oba modela, što označava uravnoteženi kompromis između preciznosti i opoziva. Rezultati InceptionV3 modela ponovno pokazuju lošije performanse u usporedbi s ResNet50 i VGG16, ali je razlika u rezultatima smanjena u usporedbi na prvi skup podataka. Razlog tome bi mogao biti da Inception V3 ima složeniju arhitekturu od VGG16 i ResNet50. Zbog složenosti Inceptiona V3, može postojati veća šansa za pretreniranje ili probleme pri izdvajanju važnih karakteristika iz slika šarenice.



Slika 3.2. Prikaz validacijske točnosti s obzirom na broj epoha za IIT Delhi skup podataka

4. ZAKLJUČAK

Prepoznavanje šarenice oka jedno od najboljih mjera identifikacije i zaštite zbog svoje jedinstvenosti i teškoće krivotvorstva. U ovom radu prikazane su metode dubokog učenja i konvolucijskih neuralnih mreža koje su se u bliskoj prošlosti pokazale kao jedne od najefikasnijih u rješavanju ove vrste problema.

Rezultati dobiveni testiranjem 3 različita CNN modela na 2 različita skupa podataka pokazuju da CNN-ovi mogu obavljati zadatke klasifikacije šarenice s izvrsnom točnošću. Automatsko izdvajanje značajki i hijerarhijsko učenje koji su prikladni za zadatak kao što je identifikacija šarenice omogućeni su primjenom algoritama dubokog učenja. Rezultati su pokazali da CNN-ovi mogu razlikovati nekoliko razreda šarenica i naučiti složene obrasce. Primijećene razlike u performansama između tri ispitana CNN dizajna mogu se pripisati arhitektonskom dizajnu svakog modela, njihovim hiperparametrima i metodama optimizacije. Korišteni skupovi podataka su također igrali značajnu ulogu. Veličina, kvaliteta i varijabilnost slika šarenice bile su različite između Casia Interval skup podataka i IIT Delhi skup podataka. Kod ResNet50 i VGG16 modela metrika točnosti između skupova podataka se nije značajno razlikovala ali su modeli u manje epoha došli do visoke točnosti kod IIT Delhi skupa podataka koji je normaliziran za razliku od Casia Interval skupa podataka koji to nije. Razlika između performansi s obzirom na skupove podataka još se bolje vidi kod Inception V3 modela koji ima značajne razlike u točnosti i ostalim metrikama s obzirom na skupove podataka. U izgradnji snažnih modela koji se mogu nositi s postavkama iz stvarnog svijeta, rezultati sugeriraju važnost i značaj kvalitete skupova podataka i njihove raznolikosti.

Ključnu ulogu u daljnjem razvoju identifikacije šarenice oka mogla bi imati metoda dubokog učenja. Daljnji pravci istraživanja vjerojatno će biti usmjereni prema inovativnim arhitekturama CNN-a i poboljšanju funkcionalnosti modela klasifikacije šarenica što bi moglo značajno pomoći sustavima za prepoznavanje šarenice i područjima koja ih koriste (biometriji, sigurnosti i kontroli pristupa i sl.).

5. LITERATURA

[1] "IIT Delhi Iris Database (Version 1.0)", s Interneta, https://www4.comp.polyu.edu.hk/~csajaykr/IITD/Database_Iris.htm, (pristupljeno 23.5.2023.)

[2] "Note on Casia-IrisV3", s Interneta,http://www.cbsr.ia.ac.cn/english/IrisDatabase.asp, (pristupljeno 25.5.2023.)

[3] "Unleashing the Power of CNNs: A Very Short Introduction to Convolutional Neural Networks", s Interneta, https://medium.com/@abhishekmishra13k/unleashing-the-power-of-cnns-a-very-short-introduction-to-convolutional-neural-networks-for-7f645abd9b88, (pristupljeno 7.7.2023)

[4] "Convolutional Neural Network", s Interneta, https://siddharthsankhe.medium.com/convolutional-neural-network-dc942931bff8, (pristupljeno 7.7.2023.)

[5] "Deep Collaborative Filtering Combined with High-Level Feature Generation on Latent Factor Model", s Interneta, https://link.springer.com/chapter/10.1007/978-3-030-04167-0_13, (pristupljeno 7.7.2023.)

[6] "Different Text Classification Algorithms and Best Practices in a Nutshell", s Interneta, https://medium.com/@shubham.ksingh/different-text-classification-algorithms-and-best-practices-in-a-nutshell-c2223263f5d, (pristupljeno 7.7.2023.)

[7] "Convolutional Neural Network — A brief introduction", s Interneta, https://becominghuman.ai/convolutional-neural-network-a-brief-introduction-c044302b3271, (pristupljeno 7.7.2023.)

[8] "Computer Vision: MaxPooling and Dropouts", s Interneta, https://aaweg-i.medium.com/computer-vision-2-maxpooling-and-dropouts-f89cfc5fa412, (pristupljeno 7.7.2023.)

[9] "Convolutional Layers vs Fully Connected Layers", s Interneta, https://towardsdatascience.com/convolutional-layers-vs-fully-connected-layers-364f05ab460b, (pristupljeno 7.7.2023.)

[10] "Compressing Neural Networks with the Hashing Trick", s Interneta, https://www.researchgate.net/figure/An-illustration-of-a-neural-network-with-random-weight-sharing-under-compression-factor-1_fig1_275279789 , (pristupljeno 7.7.2023.)

[11] Li, B., Venkatesan, R.: "Convolutional Neural Networks in Visual Computing: A Concise Guide", CRC Press, Oregon ,2017.

[12] "Convolutional neural network", s Interneta, https://en.wikipedia.org/wiki/Convolutional_neural_network, (pristupljeno 23.5.2023.)

[13] "A Comprehensive Guide to Convolutional Neural Networks", s Interneta, https://www.v7labs.com/blog/convolutional-neural-networks-guide, (pristupljeno 19.5.2023.)

[14] "Understanding of Gradient Descent: Intuition and Implementation", s Interneta, https://blog.gopenai.com/understanding-of-gradient-descent-intuition-and-implementation-b1f98b3645ea, (pristupljeno 7.7.2023.)

[15] He, K., Zhang, X., Ren, S., Sun, J.: "Deep Residual Learning for Image Recognition"

[16] "Residual neural network", s Interneta, https://en.wikipedia.org/wiki/Residual_neural_network , (pristupljeno 30.5.2023.)

[17] "ResNet-50: The Basics and a Quick Tutorial", s Interneta, https://datagen.tech/guides/computer-vision/resnet-50/, (pristupljeno 25.5.2023.)

[18] "Understanding VGG16: Concepts, Architecture, and Performance", s Interneta, https://datagen.tech/guides/computer-vision/vgg16/#, (pristupljeno 28.5.2023.)

[19] "[Paper] Review of VGGNet — 1st Runner-Up of ILSVLC 2014 (Image Classification)", s Interneta, https://laptrinhx.com/paper-review-of-vggnet-1st-runner-up-of-ilsvlc-2014-image-classification-2687564632/, (pristupljeno 28.5.2023.)

[20] "The Power of VGG16: A Deep Dive into One of the Most Influential Neural Networks in Image Recognition", s Interneta, https://wisdomml.in/the-power-of-vgg16-a-deep-dive-into-one-of-the-most-influential-neural-networks-in-image-recognition/, (pristupljeno 28.5.2023.)

[21] Simonyan, K., Zisserman, A.: "Very Deep Convolutional Networks for Large-Scale Image Recognition"

[22] "VGG-Net Architecture Explained", s Interneta, https://medium.com/@siddheshb008/vgg-net-architecture-explained-71179310050f, (pristupljeno 28.5.2023.)

[23] "Inceptionv3", s Interneta, https://en.wikipedia.org/wiki/Inceptionv3 , (pristupljeno 1.6.2023.)

[24] "Deep feature representation and multiple metric ensembles for person re-identification in security surveillance system", s Interneta, https://www.researchgate.net/figure/Inception-modules-used-in-our-CNN-architecture-module-a-and-b-were-all-proposed-in_fig3_316525473, (pristupljeno 1.6.2023.)

[25] "Auxiliary Classifier", s Interneta, https://paperswithcode.com/method/auxiliary-classifier, (pristupljeno 1.6.2023.)

[26] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the Inception Architecture for Computer Vision

[27] "Inception V3 Model Architecture", s Interneta, https://iq.opengenus.org/inception-v3-model-architecture/, (pristupljeno 1.6.2023.)

[28] "Popular Machine Learning Metrics. Part 1: Classification & Regression Evaluation Metrics", s Interneta, https://towardsdatascience.com/20-popular-machine-learning-metrics-part-1-classification-regression-evaluation-metrics-1ca3e282a2ce, (pristupljeno 23.5.2023.)

[29] "mAP (mean Average Precision) for Object Detection", s Interneta, https://jonathan-hui.medium.com/map-mean-average-precision-for-object-detection-45c121a31173, (pristupljeno 1.6.2023.)

6 . POPIS OZNAKA I KRATICA

Kratica	Puni naziv na stranom jeziku	Tumačenje na hrvatskom jeziku
AUC	eng. Area under the ROC curve	Područje ispod ROC krivulje
CASIA	eng. Chinese Academy of Sciences Institute of Automation	Institut za automatizaciju Kineske akademije znanosti
CMOS	eng. complementary metal-oxide semiconductor	komplementarni metalno-oksidni poluvodič
CNN	eng. Convolutional neural network	Konvolucijska neuralna mreža
fn	eng. False negative	Lažni negativ
fp	eng. False positive	Lažni pozitiv
IIT	eng. Indian Institute of Techology	Indijski institut tehnologije
ILSVRC	eng. ImageNet Large Scale Visual Recognition Challenge	ImageNet Large Scale Visual Recognition Izazov
LED	eng. Light-emitting diode	Svjetleća dioda
NIR	eng. Near-Infrared	Bliski infracrveni
ReLU	eng. Rectified linear unit	Rektificirana linearna jedinica
ResNet	eng. Residual neural network	Rezidualna neuralna mreža
RGB	eng. Red, green, blue	Crvena, zelena, plava
ROC	eng. Receiver operating characteristic curve	Krivulja radne karakteristike prijemnike
tn	eng. True negative	Pravi negativ
tp	eng. True positive	Pravi pozitiv
VGG	eng. Visual Geometry Group	Vizualna Geometrijska Grupa

7. SAŽETAK I KLJUČNE RIJEČI NA HRVATSKOM I ENGLESKOM JEZIKU

Biometrijska je identifikacija postala ključna pretpostavka sigurnosti poslovnih sustava te njihove materijalne, financijske i informacijske imovine. Rješenja za provjeru autentičnosti temeljena na biometriji privukla su značajnu pozornost u domeni informacijske i poslovne sigurnosti. Među brojnim biometrijskim modalitetima prepoznavanje šarenice postalo je jedna od najpreciznijih i najpouzdanijih metoda osobne identifikacije.

Sustavi za prepoznavanje šarenice iznimno su pouzdani zbog svoje stabilnosti, čak i u različitim svjetlosnim situacijama, te otpornosti na starenje ili male ozljede. Korištenje tehnika dubokog učenja za identifikaciju šarenice glavna je tema ovog rada. Algoritmi dubokog učenja otvorili su nove mogućnosti učenja i potencijal u pogledu prepoznavanja šarenice. Iskorištavanjem algoritama dubokog učenja, sustavi za prepoznavanje šarenice mogu postići iznimnu točnost i robusnost, što ih čini vrlo prikladnima za primjene u sigurnosti, kontroli pristupa i forenzičkim istragama. Posebno su konvolucijske neuralne mreže pokazale visoke performanse u prepoznavanju šarenica. Sustav može automatski naučiti diskriminirajuća obilježja izravno iz neobrađenih slika šarenice pomoću CNN-a za identifikaciju šarenice, uklanjajući potrebu za ručno projektiranim značajkama. Rezultati prikazani u radu sugeriraju da uspješnost prepoznavanja šarenica pomoću CNN-ova bitno ovisi o arhitekturi CNN modela koju odabiremo za rješavanje problema te o kvaliteti i veličini skupova podataka koje koristimo kao ulazne podatke za model.

Ključne riječi: biometrijska identifikacija, sigurnost, prepoznavanje šarenice, duboko učenje, algoritmi, konvolucijske neuralne mreže.

SUMMARY AND KEY WORDS

Biometric identification has become a key assumption for the security of business systems and their material, financial and information assets. Biometrics-based authentication solutions have attracted considerable attention in the domain of information and business security. Among the many biometric modalities, iris recognition has become one of the most accurate and reliable methods of personal identification.

Iris recognition systems are extremely reliable due to their stability, even in different lighting situations, and resistance to aging or minor injuries. The use of deep learning techniques for iris identification is the main topic of this paper. Deep learning algorithms have opened up new learning possibilities and potential in terms of iris recognition. By leveraging deep learning algorithms, iris recognition systems can achieve exceptional accuracy and robustness, making them well suited for applications in security, access control, and forensic investigations. In particular, convolutional neural networks have shown high performance in iris recognition. The system can automatically learn discriminative features directly from raw iris images using CNNs for iris identification, removing the need for manually designed features. The results presented in the paper suggest that the success of iris recognition using CNNs depends significantly on the architecture of the CNN model that we choose to solve the problem and on the quality and size of the datasets that we use as input data for the model.

Keywords: biometric identification, security, iris recognition, deep learning, algorithms, convolutional neural networks.

POPIS SLIKA

POPIS TABLICA