Vizualizacija rada konvolucijske neuronske mreže

Vlahović, Stjepan

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

ZAVRŠNI RAD

Stjepan Vlahović

Zagreb, 2020.

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

ZAVRŠNI RAD

Mentori:

Doc.dr.sc. Tomislav Stipančić, dipl. ing.

Student:

Stjepan Vlahović

Zagreb, 2020.

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

Zahvaljujem se mentoru Tomislavu Stipančiću na puno strpljenja i pomoći koju mi je pružio tokom izrade završnog rada.

Također bi se želio zahvaliti svojim roditeljima, bratu, prijateljima i djevojci na razumijevanju i pomoći koju su mi pružili kroz studiranje.

Stjepan Vlahović

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

Sveučilište u Zagrebu Fakultet strojarstva i brodogradnje	
Datum	Prilog
Klasa:	
Ur.broj:	

ZAVRŠNI ZADATAK

Student: **Stjepan Vlahović** Mat. br.: 0035199416

Naslov rada na hrvatskom jeziku: **Vizualizacija rada konvolucijske neuronske mreže**

Naslov rada na engleskom jeziku: **Visualizing the work of convolution neural network**

Opis zadatka:

Konvolucijske neuronske mreže (engl. Convolution Neural Networks - CNNs) predstavljaju podtip neuronskih mreža koje računalima daju mogućnosti izvršavanja određenih zadataka u sklopu različitih primijena tehnika duboke računalne analize podataka (predviđanje, klasifikacija, detekcija, prepoznavanje i sl.).
Modeli konvolucijskih neuronskih mreža često su teški za razumijevanje od strane prosječnih korisnika (npr. pitanja kako model uči kompleksne ovisnosti prisutne u ulaznim podacima ili zašto je određena procijena dana za vrijeme zaključivanja modela). Stoga se koriste razne tehnike vizualizacije u svrhu boljeg razumijevanja rada konvolucijske neuronske mreže.

U radu je potrebno:
a) objasniti principe rada konvolucijske neuronske mreže,
b) odrediti tehniku za vizualizaciju rada konvolucijske neuronske mreže,
c) razviti modularnu aplikaciju za vizualizaciju rada mreže,
d) testirati na skupu slika razvijenu aplikaciju koristeći ranije trenirani model mreže te objasniti rezultate.

Programska aplikacija mora biti izvedena koristeći programski jezik Python.
Rad predati u pisanom i elektroničkom obliku uz prikladne upute za korištenje razvijene aplikacije.
U radu je potrebno navesti korištenu literaturu i eventualno dobivenu pomoć.

Zadatak zadan:
28. studenog 2019.

Zadatak zadao:

Doc. dr. sc. Tomislav Stipančić

Datum predaje rada:
1. rok: 21. veljače 2020.
2. rok (izvanredni): 1. srpnja 2020.
3. rok: 17. rujna 2020.

Predviđeni datumi obrane:
1. rok: 24.2. – 28.2.2020.
2. rok (izvanredni): 3.7.2020.
3. rok: 21.9. - 25.9.2020.

Predsjednik Povjerenstva:

Prof. dr. sc. Branko Bauer

SADRŽAJ

SADRŽAJ I
POPIS SLIKA II
POPIS TABLICA III
SAŽETAK IV
SUMMARY VI
1. UVOD 1
2. RAZRADA ZADATKA 2
2.1. Programski jezik P*ython* 2
2.2. Baza podataka 2
3. TEORIJSKA OSNOVA ZADATKA 4
3.1. Neuronska mreža 4
3.2. Konvolucijska neuronska mreža 4
3.2.1. Arhitektura konvolucijskih neuronskih mreža 4
3.2.1.1. Konvolucijski sloj 6
3.2.1.2. Sloj sažimanja 7
3.2.1.3. Potpuno povezani slojevi 8
3.2.2. VGG16 9
3.3. Vizualizacija konvolucijskih neuronskih mreža 10
3.3.1. Vizualizacija aktivacije slojeva 10
3.3.2. Vizualizacija težina 10
3.3.3. Preuzimanje slika koje maksimalno aktiviraju neuron 10
3.3.4. Ugradnja slika u dvije dimenzije 11
3.3.5. Zamućivanje dijela slike 11
4. IZVEDBA ZADATKA 12
4.1. Keras 12
4.2. Priprema ulaznih podataka 13
4.3. Keract 13
4.4. Tkinter 14
4.5. Opis rada aplikacije 15
4.6. Mogući problemi 16
5. REZULTATI 17
5.1. Objašnjenje vizualizacije izlaza 17
5.1.1. Objašnjenje vizualizacije 2D aktivacijskih mapa na primjeru kruga 17
5.1.2. Objašnjenje vizualizacije 2D aktivacijskih mapa na primjeru mačke 20
5.2. Tablica izlaza 23
5.2.1. Analiza slike orla 24
6. ZAKLJUČAK 27
LITERATURA 28
PRILOZI 30

POPIS SLIKA

Slika 1 AlexNet arhitektura 5
Slika 2 Arhitektura GoogleNet-a 5
Slika 3 LeNet-5 arhitektura 6
Slika 4 Rad filtera 6
Slika 5 Rad konvolucijskog sloja 7
Slika 6 Izlazni volumen 7
Slika 7 Rezultat MAX operacije 8
Slika 8 Potpuno povezani sloj 8
Slika 9 Softmax funkcija 8
Slika 10 VGG16 arhitektura 9
Slika 11 Uvezeni paketi 12
Slika 12 Keract funkcije 12
Slika 13 Obrada slike 13
Slika 14 Vizualizacija aktivacija 13
Slika 15 Interaktivni prozori aplikacije 14
Slika 16 Upute za aplikaciju 15
Slika 17 Gumb za spremanje 16
Slika 18 1. Slika iz tablice baze podataka 17
Slika 19 Prvi konvolucijski sloj 18
Slika 20 Drugi konvolucijski sloj 18
Slika 21 Peti konvolucijski sloj 19
Slika 22 Deveti konvolucijski sloj 19
Slika 23 Četvrti sloj sažimanja 20
Slika 24 2. slika iz tablice 20
Slika 25 Peti konvolucijski sloj sa istaknutim neuronima na 2D aktivacijskim mapama 1 i 2 (odozgo prema dolje) 21
Slika 26 Deveti konvolucijski sloj sa istaknutim neuronima 2D aktivacijskih mapa 1 i 2 (označene odozgo prema dolje) 22
Slika 27 Četvrti sloj sažimanja 22

POPIS TABLICA

Tablica 1 Baza podataka..... 2
Tablica 2 Tablica 2D aktivacijskih mapa..... 23
Tablica 3 Manipulirane slike orla..... 25

SAŽETAK

U ovom radu će se razvijati modularna aplikacija za vizualizaciju aktivacije slojeva VGG16 konvolucijske neuronske mreže. Prvi dio rada obrađuje teoriju konvolucijskih neuronskih mreža te metode za vizualizaciju i razumijevanje konvolucijskih neuronskih mreža. Nakon toga slijedi dio koji objašnjava kodiranje aplikacije u programu *Python*. Na kraju se analiziraju rezultati modularne aplikacije, vizualizirani izlazni volumen slojeva konvolucijske neuronske mreže raspoređen u 2D prostoru.

Ključne riječi: neuronska mreža, *Python*, vizualizacija, modularna aplikacija.

SUMMARY

In this paper, a modular application will be developed to visualize the activation of VGG16 layers of a convolutional neural network. The first part of the paper deals with the theory of convolution neural networks and methods for visualizing and understanding convolution neural networks. This is followed by the part that explains coding of applications in Python program. Finally, the results of the modular application, a visualized output volume arranged in 2D space, is analyzed.

Key words: neural network, *Python*, visualization, modular application

1. UVOD

Svakim danom sve više raste potreba za interaktivnijim uređajima koji su u stanju prilagoditi se korisniku i njegovim potrebama. To zahtjeva programiranje okrenuto pojedincu, a ne masi, što je s tradicionalnim programiranjem nemoguće postići na velikom broju uređaja. No, tu dolaze neuronske mreže koje omogućavaju programiranje uređaja koji će se prilagođavati i stalno učiti ovisno o individualnim potrebama čovjeka.

Ukoliko se radi o vizualnom ulazu podataka, kao što su slike, koristit će se konvolucijske neuronske mreže koje iskorištavaju činjenicu da se tu radi o slikovnom ulazu. S druge strane, obične neuronske mreže preferiraju ulaz u obliku jednog vektora te se loše skaliraju prema slikama većih rezolucija [2]. Zahvaljujući konvolucijskim neuronskim mrežama cijelo polje računalne vizualizacije je doživjelo velika dostignuća. To je omogućilo razvoj mnogih tehnologija, kao što je tehnologija za prepoznavanje lica u mobitelima. No, bez obzira na veliku pristupačnost različitim bazama podataka i pretreniranim modelima, ponekad je teško razumjeti kako i na koji način korišteni model uči, pogotovo za ljude koji nisu upoznati sa strojnim učenjem. Samo razumijevanje modela potrebno je i za njegovo poboljšavanje i pravilnu implementaciju. Problem se pogotovo javlja kod modela kao *InceptionResNetV2* [12] koji ima preko 200 slojeva. Što se dublje ulazi u mrežu slojevi postaju sve više apstraktni, više ne prepoznaju samo rubove već su sposobni prepoznati oči, lice, njušku i slično. Kako bi se znalo što se točno događa u tim dubljim slojevima koriste se metode za vizualizaciju konvolucijskih neuronskih mreža.

U svrhu boljeg razumijevanja rada konvolucijskih neuronskih mreža, u ovom radu će se izraditi modularna aplikacija koja će koristiti jednu od tih metoda.

2. RAZRADA ZADATKA

U poglavlju prije napravljen je kratak uvid u temu ovog rada. Cilj je pomoću programskog jezika *Python* napraviti aplikaciju koja će vizualizirati izlaze iz slojeva odabrane konvolucijske neuronske mreže. Kako bi se to moglo postići bit će potrebno u mrežu unijeti bazu sa slikama koje će ona obraditi. U ovom dijelu, prije teorije i izvedbe zadatke, ukratko će se upoznati programski jezik *Python* i baza podataka.

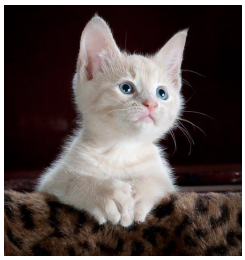
2.1. Programski jezik P*ython*

Python je objektno orijentiran, fleksibilan programski jezik opće namjene. Iz razloga što je interpreterski jezik, znatno je sporiji u odnosu na *C* i *C++*. Razlog zašto će se koristiti u ovom zadatku, ali i zašto se općenito koristi za zadatke ovoga tipa, je zbog velikog broja paketa specijaliziranih za različite projekte. Paketi koji će se ovdje koristiti su *keras*, *pillow*, *matplotlib*, *tkinter*, *os* i *functools*. Oni koji su važniji za rad korištene modularne aplikacije bit će objašnjeni u poglavlju 4.

2.2. Baza podataka

Baza podataka sadržava pažljivo odabrane slike, pomoću kojih će se moći što bolje interpretirati rezultati vizualizacije.

Tablica 1 Baza podataka

 **1. Krugovi**	 **2. Mačka**	 **3. Crvena panda**
 **4. Pas**	 **5. Papiga**	 **6. Orao**

Prva slika [Tablica 1] je odabrana zbog svoje jednostavnosti, koja će omogućiti lakšu interpretaciju vizualiziranog izlaznog volumena. Na ostalim slikama nalaze se životinje, tri sisavca i dvije ptice. One su odabrane jer će se njihovom kombinacijom lakše naći 2D aktivacijske mape koje prepoznaju specifične karakteristike.

3. TEORIJSKA OSNOVA ZADATKA

U ovom poglavlju će se proći kroz teoriju potrebnu za razumijevanje teme zadatka.

3.1. Neuronska mreža

Neuronska mreža se definira kao sustav za računanje sastavljen od jednostavnih elemenata koje zovemo neuroni, a bazira se na strukturi i načinu funkcioniranja ljudskog mozga. Neuroni su organizirani po slojevima koji obrađuju podatke. Slojeve ovisno o funkciji dijelimo na ulazni, skriveni i izlazni sloj. Ulazni sloj prima podatke, skriveni sloj (koji može biti jedan ili više njih) je sloj u kojem se događa sva obrada podataka preko mreže veza koje se zovu težine i biasi, a izlazni sloj izbacuje izlaze. Za više o neuronskoj mreži pogledati [8].

3.2. Konvolucijska neuronska mreža

Konvolucijska neuronska mreža je slična običnoj neuronskoj mreži spomenutoj u poglavlju 3.1, ali za razliku od nje iskorištava činjenicu da je ulaz slika. To dozvoljava programiranje određenih karakteristika u arhitekturu, što čini unaprijednu funkciju puno učinkovitijom te smanjuje broj parametara potrebnih u mreži. Slojevi konvolucijske neuronske mreže imaju neurone posložene u 3 dimenzije: širina, visina i dubina. Ulaz u mrežu će iz toga razloga biti ulazni volumen slike. Dimenzija dubine odgovara kanalu boje (crvena, zelena, plava). Tako je ulaz kod VGG16 modela slika volumena 224x224x3.

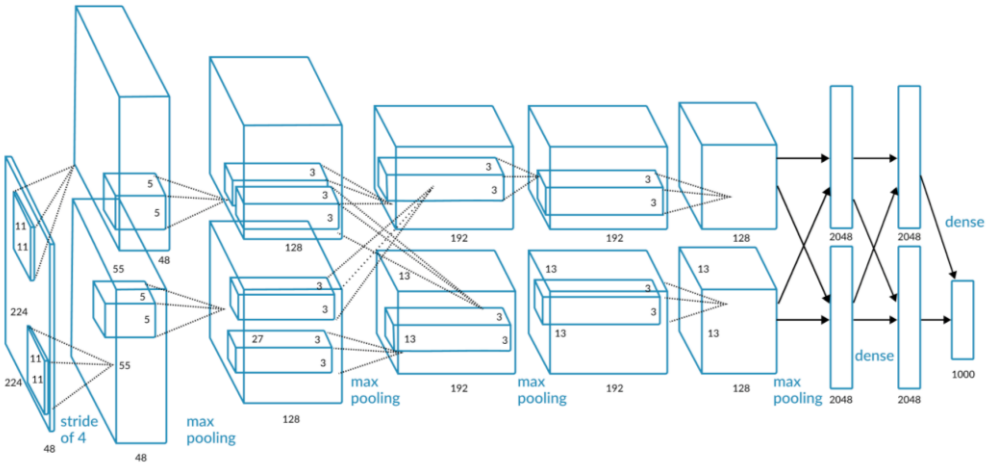
3.2.1. Arhitektura konvolucijskih neuronskih mreža

Arhitektura se programira na način da mreža bude jako dobra u prepoznavanju obrazaca i uzoraka. Pod obrasce i uzorke se misli na rubove, oblike, teksture i objekte. Kako bi se prepoznali obrasci koriste se filtri, a što mreža ide dublje u analizu slike to lakše prepoznaje detalje poput lica, ramena, listova, grana i sl, a to će se vidjeti na primjerima u poglavlju 5. Filtar je matrica kvadratnih dimenzija koja postepeno prolazi kroz sliku u koracima (eng. *stride*) i na taj način detektira razne objekte. Primjer takvog filtra se može vidjeti na [Slika 4]. U osnovi arhitektura ove vrste mreže se sastoji od ulaznog sloja, konvolucijskog sloja, sloja sažimanja, potpuno povezanog sloja i aktivacijske funkcije.

S obzirom na načine slaganja slojeva postoje različiti pretrenirani modeli, a neki poznatiji su:

- AlexNet

  Jedna od značajnijih razlika između ove arhitekture i ostalih, korištenje je ReLU aktivacijske funkcije [10]. To čini ovu vrstu algoritma znatno bržom. Arhitektura ove mreže prikazana je na [Slika 1].



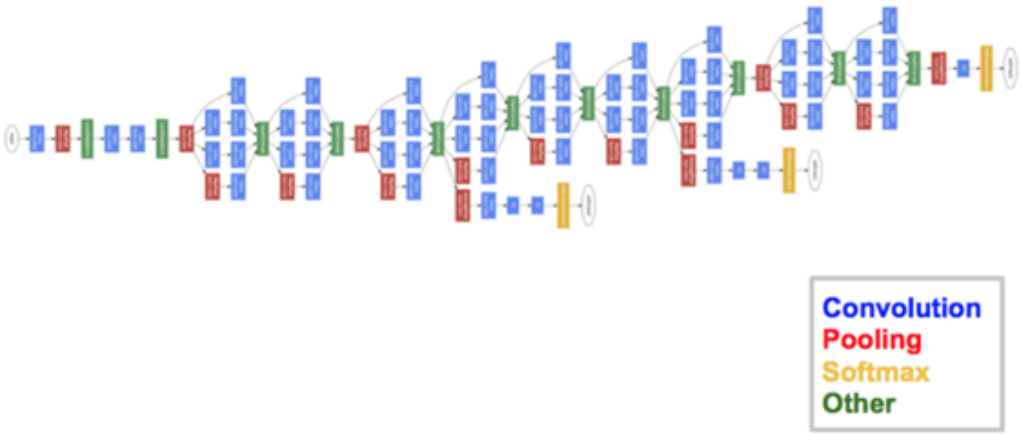
Slika 1 AlexNet arhitektura

- VGGnet

  Objašnjen u poglavlju 3.2.2.

- GoogleNet

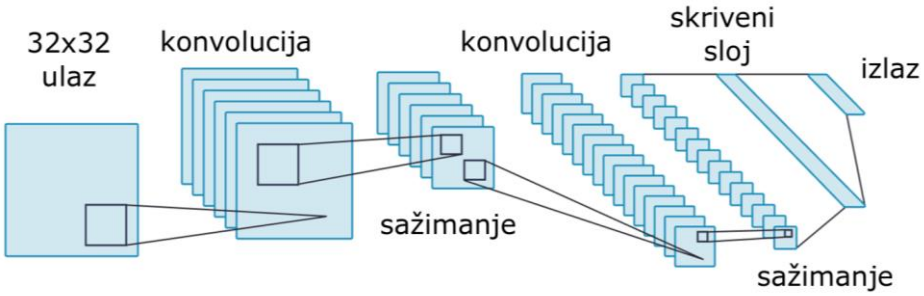
  Napravljena od strane Google-a, GoogleNet ima 22 sloja duboku mrežu baziranu na malim konvolucijama zvanim „*inception module*“ od kud je i dobila drugi naziv *InceptionV1* (V1 stoji za verziju 1).



Slika 2 Arhitektura GoogleNet-a

- LeNet-5

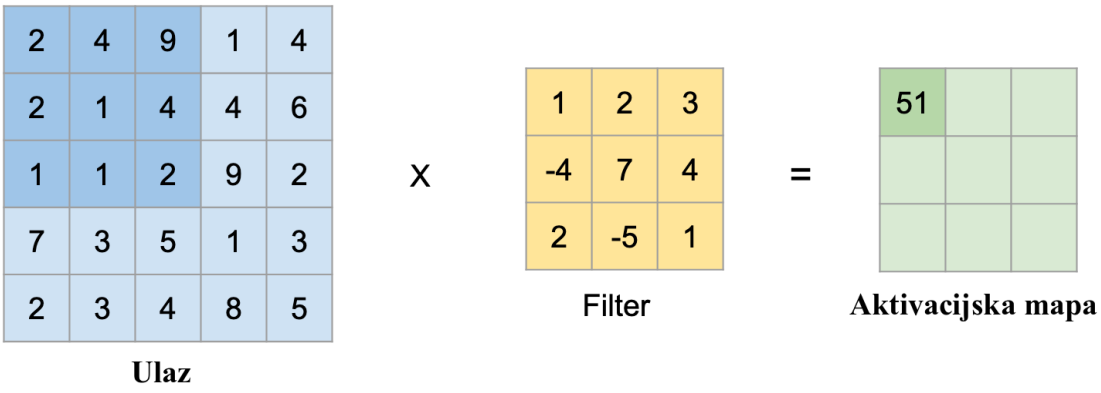
  Radi se o konvolucijskoj neuronskoj mreži sa 7 slojeva koju su posloženi kao na [Slika 3]. Koristi se u bankama kako bi se prepoznali rukom pisani brojevi na čekovima. Ulaz u mrežu je bila 32x32 piksela slika u sivim tonovima.



Slika 3 LeNet-5 arhitektura

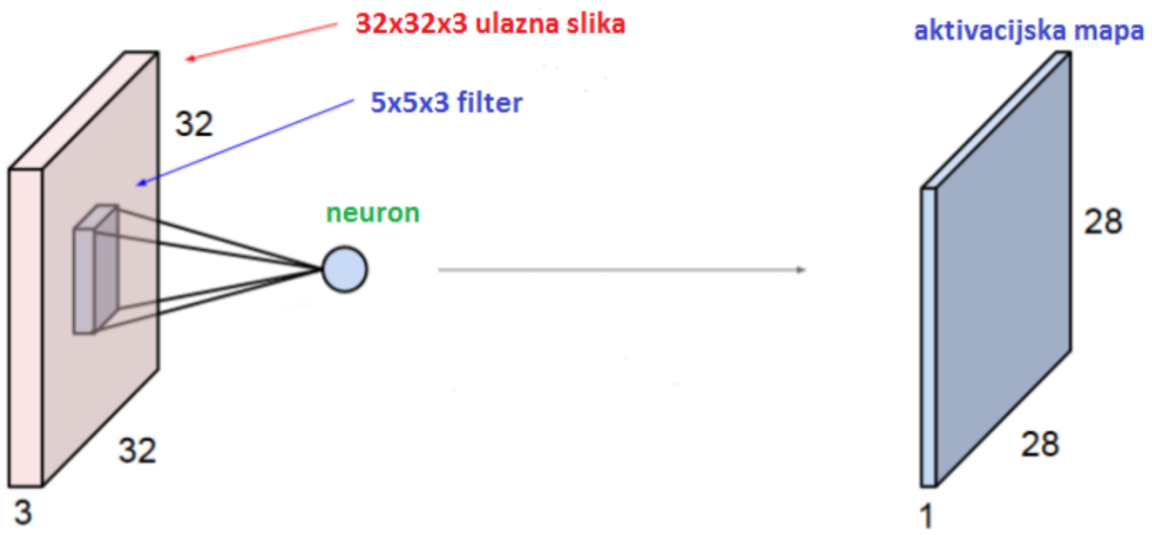
3.2.1.1. Konvolucijski sloj

Konvolucijski sloj je jezgra konvolucijskih neuronskih mreža koja obavlja većinu računanja. Sastoji se od kompleta filtera sa mogućnošću učenja. Svaki filter ima malu površinu, ali se proteže kroz cijelu dubinu ulaznog volumena. Prilikom unaprijednog prolaza, svaki filter prolazi po dužini i širini ulaznog volumena i izračunava skalarni umnožak između podataka unutar filtera i ulaza na određenoj poziciji što se može vidjeti na [Slika 4].

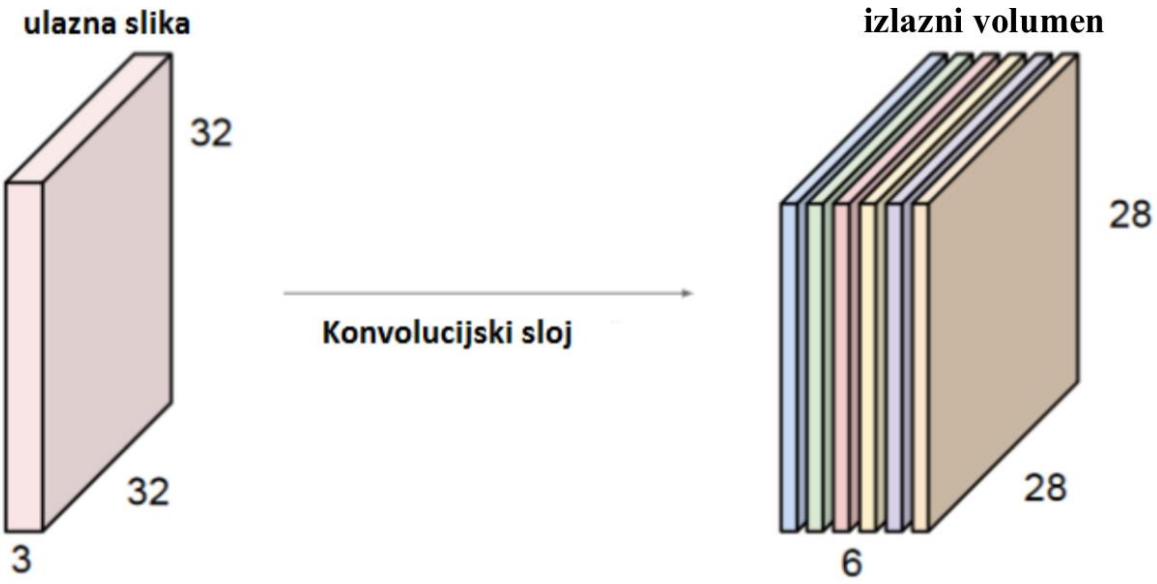


Slika 4 Rad filtera

Tim postupkom se dobiva 2-D mapa aktivacija koja daje reakcije filtra za svaku poziciju na kojoj se on nalazio. Mreža u prvom sloju uči filtre da se aktiviraju kada vide rub, boju, promjenu boje i slično, a postepeno kroz slojeve počinje učenje cijelih uzoraka kao što su krug, kocka itd. Unutar svakog konvolucijskog sloja postoji cijeli set takvih filtera i svaki od njih će imati kao produkt 2-D aktivacijsku mapu [Slika 5]. Te mape će se poslagati po dubini i kao produkt će se dobiti izlazni volumen [Slika 6].



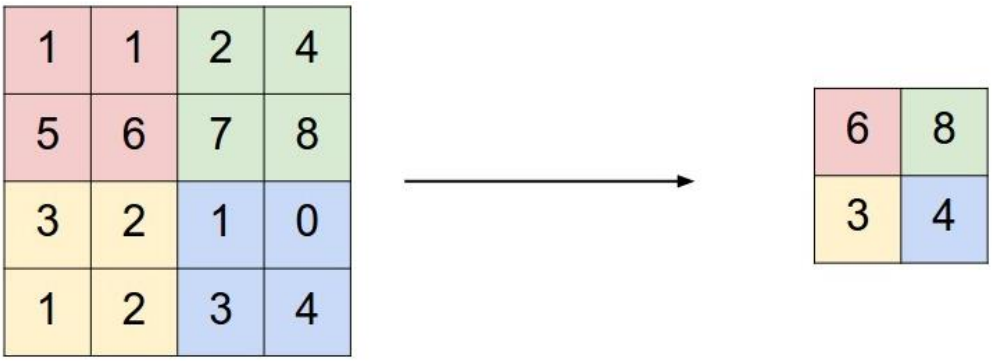
Slika 5 Rad konvolucijskog sloja



Slika 6 Izlazni volumen

3.2.1.2. Sloj sažimanja

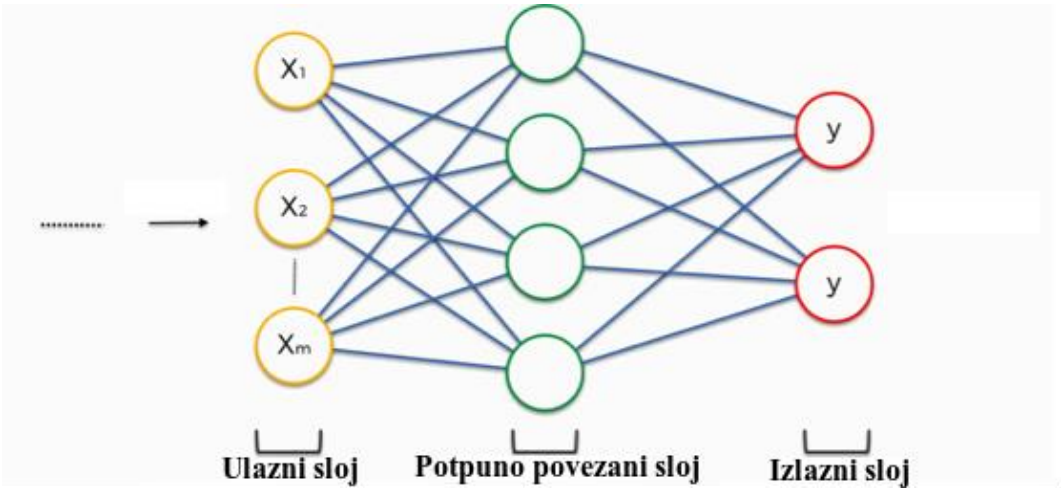
Često se može uočiti da se ovaj sloj nalazi između konvolucijskih slojeva. Funkcija mu je smanjiti prostornu veličinu izlaznog volumena kako bi se smanjila količina parametara i računanja sljedećeg konvolucijskog sloja. Sloj djeluje neovisno na svakom komadiću dubine i mijenja prostornu veličinu koristeći MAX operaciju [9]. Najčešća forma filtera u sloju je 2x2 sa korakom 2. Na taj način se odbacuje 75% podataka. Rezultat takvog postupka se vidi na [Slika 7].



Slika 7 Rezultat MAX operacije

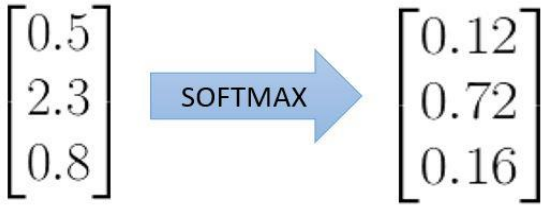
3.2.1.3. Potpuno povezani slojevi

Potpuno povezani slojevi (skraćeno FC što stoji za *Fully connected*) sadrže neurone koji imaju potpune veze sa svim aktivacijama u prethodnom sloju kao i kod običnih neuronskih mreža [Slika 8]. Ovi slojevi vrše klasifikaciju slika, a formirani su kao vektor dimenzija 1x1x1000 za primjer od 1000 klasa.



Slika 8 Potpuno povezani sloj

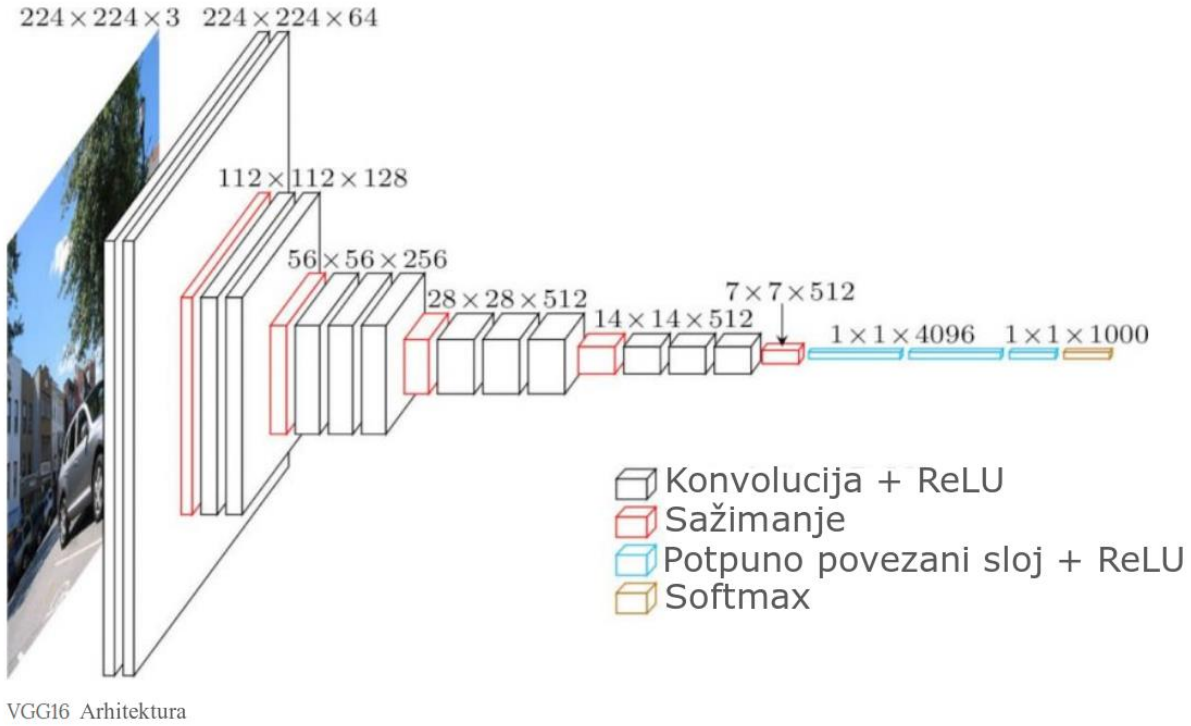
FC slojevima dosta često prethodi spljošteni sloj koji prethodni volumenski ulaz pretvaraju u vektorski izlaz. Izlazni sloj FC funkcije se provodi kroz Softmax funkciju [11] koja izlazne parametre svodi na vjerojatnosti [Slika 9] čiji je zbroj 1.



Slika 9 Softmax funkcija

3.2.2. VGG16

VGG16 je model konvolucijske neuronske mreže kojeg su predstavili K. Simonyan i A. Zisserman u radu *Very Deep Convolutional Networks for Large-Scale Image Recognition* [5]. VGG dolazi od *Visual Geometry Group*, što je naziv grupe na Oxfordu koja je razvila model, a broj 16 označava broj slojeva koji sadržavaju težine. Model postiže točnost od 92,7% u bazi podataka ImageNet koja sadrži preko 14 milijuna slika koje se razvrstavaju u preko 1000 klasa.



Slika 10 VGG16 arhitektura

Na [Slika 10] se vidi da je ulaz u 1. konvolucijski sloj RGB slika veličine 244 x 244. Slika se provlači kroz skupinu konvolucijskih slojeva gdje se koriste filtri sa najmanjim mogućim receptivnim poljem 3x3. Korak konvolucije je 1 piksel, a tako je napravljen kako bi se nakon konvolucije sačuvala prostorna rezolucija. Prostorno sažimanje je provedeno s 5 slojeva sažimanja, koji dolaze nakon nekih konvolucijskih slojeva, koristeći filtre dimenzija 2x2 piksela s korakom 2. Nakon konvolucijskih slojeva i slojeva sažimanja dolaze tri potpuno povezana sloja, prva dva sa 4096 kanala te treći sa tisuću kanala koji rade klasifikaciju u 1000 klasa. Svi skriveni slojevi sadrže ReLU aktivacijsku funkciju.

3.3. Vizualizacija konvolucijskih neuronskih mreža

Vizualizacija mreža se radi preko metoda za razumijevanje i vizualizaciju. Prema [1] postoje metode za razumijevanje i vizualizaciju:

- aktivacije slojeva
- težina
- pomoću preuzimanja slika koje maksimalno aktiviraju neuron
- preko ugradnje slika u dvije dimenzije
- tehnikom zamućivanja dijela slike

U ovom radu će se koristiti metoda za razumijevanje i vizualizaciju aktivacije slojeva. Uz ovu metodu, aktivacija dubljih slojeva će se prikazati pomoću toplinske skale koja će pokazati na kojem dijelu su najaktivniji neuroni koji su sudjelovali u kategoriziranju slike u kategoriju koja je prevladala.

3.3.1. Vizualizacija aktivacije slojeva

Radi se o najjasnijem načinu vizualizacije koji prikazuje aktivacije mreže prilikom unaprijednog prolaza. Za *ReLU* mreže, aktivacije, uobičajeno, pri samom početku izgledaju mrvičasto i gusto, ali kroz trening one postaju sve rjeđe i lokaliziranije. Vizualizirani izlazi su izlazni volumen sloja posložen u 2D prikazu. Zbog visokog stupnja učenja, ova vrsta vizualizacije može prikazivati „mrtve“ filtre, tj. neke aktivacije mogu biti jednake nuli. Sve navedeno će biti prikazano u poglavlju 5.

3.3.2. Vizualizacija težina

Težine se najčešće najbolje interpretiraju na prvom konvolucijskom sloju koji direktno gleda sirove podatke piksela, ali je moguće prikazati težine filtra koji se nalazi i dublje unutar mreže. Korištenje ovog načina je korisno jer dobro istrenirane mreže mogu lijepo prikazati filtre bez šumova. Šumovi mogu biti indikator da mreža nije dovoljno dugo trenirana ili da ima nisku jačinu regulacije.

3.3.3. Preuzimanje slika koje maksimalno aktiviraju neuron

Metoda se zasniva na velikoj bazi podataka popunjenoj slikama, koje se propuštaju kroz mrežu nakon čega pratimo koje slike maksimalno aktiviraju pojedini neuron. Tada se slike vizualiziraju da bi se razumjelo na što se neuron fokusira u svom ulaznom polju.

Problem se javlja jer ReLU neuroni nemaju nužno semantičko značenje sami za sebe. Bolje je razmišljati na način da više ReLU neurona možemo gledati kao osnovne vektore nekog prostora koji reprezentira sliku u dijelovima. Drugim riječima, vizualizacija pokazuje dijelove na rubovima reprezentacija, uz osi koje odgovaraju filtriranim težinama.

3.3.4. Ugradnja slika u dvije dimenzije

Zasniva se na interpretaciji konvolucijskih neuronskih mreža kao postepenoj transformaciji slika u reprezentaciju čiji su razredi linearno razdvojeni. Ugradnjom slika u 2D-u na način da njihova nisko dimenzionalna reprezentacija ima približno iste udaljenosti za razliku od visoko dimenzionalne reprezentacije.

Postoji veliki broj metoda ugradnje od kojih je najpoznatija t-SNE.

3.3.5. Zamućivanje dijela slike

Ova metoda se zasniva na pretvaranju dijela slike u nule. Prolaženjem nula kroz sliku uzimaju se podaci o vjerojatnosti pripadanja slike u kategoriju kojoj je prethodno dodijeljena. Pomoću ove metode mogu se vidjeti koji dio slike najviše utječe na pravilnu kategorizaciju mreže.

4. IZVEDBA ZADATKA

Vizualizacija konvolucijskih neuronskih mreža svakim danom postaje sve aktualnija tema. Pomoću nje napokon se mogu vidjeti aktivacije neurona duboko unutar mreže, gdje izlazi postaju već apstraktni i počinju reagirati na lice, kosu, uši i slično. U ovom poglavlju će biti objašnjena izrada modularne aplikacije u programu *Python* koja je zadužena za vizualizaciju. Prije samog početka programiranja potrebna je priprema paketa koji će se koristiti [Slika 11].

```
1   from PIL import Image as Im
2   from keras.applications.vgg16 import VGG16
3   from keras.applications.vgg16 import decode_predictions
4   from keras.applications.vgg16 import preprocess_input
5   from keras.preprocessing.image import img_to_array
6   import matplotlib.pyplot as plt
7   from tkinter import *
8   from tkinter import ttk
9   import os
10  import functools
11  import keract
```

Slika 11 Uvezeni paketi

4.1. Keras

Keras [6] je aplikacijsko programsko sučelje koje se nalazi unutar paketa *TensorFlow2.0*. Nastao je kako bi se smanjila kompleksnost programiranja neuronskih mreža i omogućila lakša i šira upotreba istih. U zadatku je korišten za pozivanje pretrenirane neuronske mreže koju sadrži unutar sebe VGG16 (isto tako je mogla biti pozvana bilo koja druga). VGG16 je korištena jer ima manji broj slojeva, pa će se lakše objasniti u poglavlju 5.

```
61          image = preprocess_input(image)
62          yhat = model.predict(image)
63          label = decode_predictions(yhat)
64          label = label[0][0]
65          print('{} ({})'.format(label[1], label[2] * 100))
```

Slika 12 Keract funkcije

Na liniji 54 [Slika 12] se ulaznom sloju dodjeljuje slika, dok linija 55 zove pretrenirani model da obradi sliku. Unutar linije 55 *model* označava korišteni model kojeg prethodno definiramo na način: *model=VGG16* [II]. Ostatak koda iz liste predviđanja vuče predviđanje sa najvećim postotkom te ga prikazuje.

4.2. Priprema ulaznih podataka

Kako bi VGG16 model mogao funkcionirati potrebno mu je na ulazu dati sliku dimenzija 224x224x3. [Slika 13] prikazuje dio koda koji će sliku pripremiti za ulaz u model.

```
40
41          # Obrađivanje slike na pravilne dimenzije
42          image = Im.open(im)
43          vel = image.size
44          vel_max = max((vel[0], vel[1]))
45          if vel_max >= 224:
46              vel_x = vel_max / 224
47              img_x = int(vel[0] / vel_x)
48              img_y = int(vel[1] / vel_x)
49              image = image.resize((img_x, img_y))
50              image = image.crop((0, 0, 224, 224))
51          else:
52              vel_x = 224 / vel_max
53              img_x = int(vel[0] * vel_x)
54              img_y = int(vel[1] * vel_x)
55              image = image.resize((img_x, img_y))
56              image = image.crop((0, 0, 224, 224))
57
58          # Obrada slike za korištenje VGG16 mreže
59          image = img_to_array(image)
60          image = image.reshape((1, image.shape[0], image.shape[1], image.shape[2]))
```

Slika 13 Obrada slike

Između linija 40 i 57 [Slika 13] nalazi se kod koji učitava sliku te joj podešava dimenzije tako da budu 224x224. U liniji 59 slika se pretvara u matricu podataka kako bi se mogla dobiti dubina slike. Linija 60 kao rezultat daje sliku volumena 224x224x3.

4.3. Keract

Keract je novonastali paket nastao zbog velike potražnje za vizualizacijom slojeva različitih mreža. Paket se može preuzeti na [7]. Funkcije koje će se koristiti ovdje su prikazane na [Slika 14].

```
75          # Pozivanje funkcije za povlačenje aktivacije slojeva
76          activations = keract.get_activations(model, image, layer_name=layer_name)
77          # Pozivanje funkcije za prikaz slojeva
78          keract.display_activations(activations, cmap=None,
79                                     save=False, directory=r'Slojevi/',
80                                     data_format='channels_last')
81          # keract.display_heatmaps(activations, image, directory=r'Heatmaps/',
82          #                         save=False)
```

Slika 14 Vizualizacija aktivacija

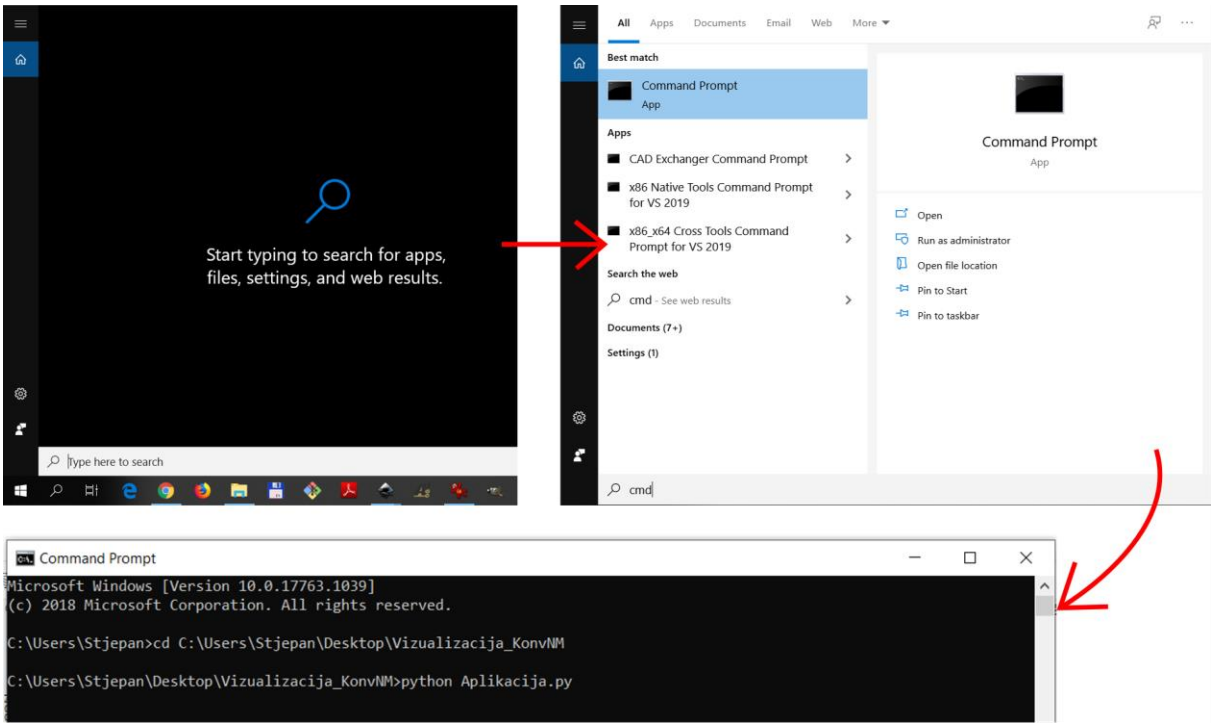
4.4. Tkinter

Tkinter [13] je paket koji se koristi za izradu grafičkih sučelja i aplikacija. Ovdje je korišten kako bi aplikacija bila lakša za korištenje bilo kojem korisniku. [Slika 15] prikazuje interaktivne gumbe i prozore napravljene pomoću paketa (više u [II]). S lijeve strane je prikazan početni prozor u kojem se klikom na gumb odabire slika koja se nalazi u direktoriju slike, nakon čega se otvara prozor sa gumbima pomoću kojih je moguće odabrati pojedini sloj za vizualizaciju.

Slika 15 Interaktivni prozori aplikacije

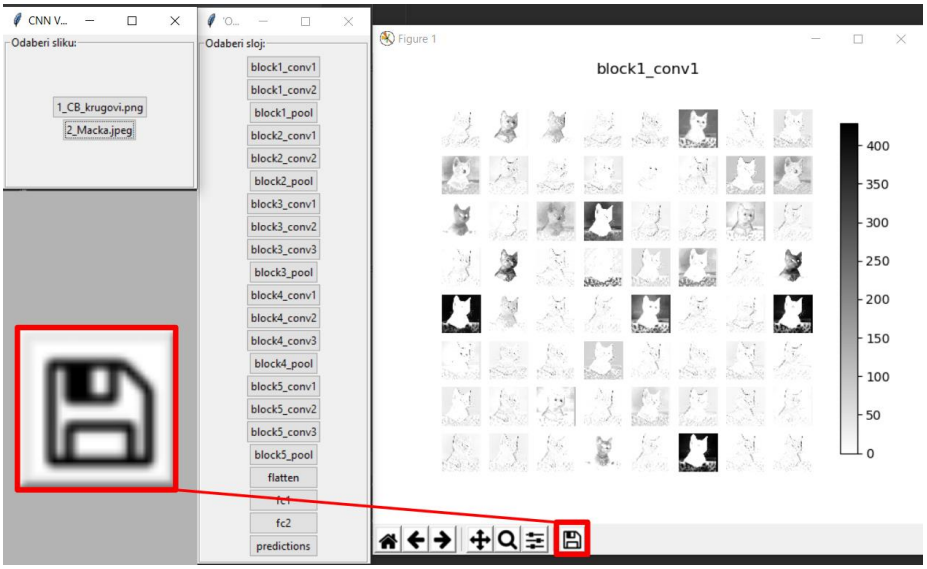
4.5. Opis rada aplikacije

Kako bi se koristila aplikacija potrebno je skinuti datoteku Vizualizacija_KonvNM. Unutar datoteke se nalazi datoteka „slike“ u koju je potrebno staviti slike koje se žele testirati u mreži. Preko *Command promta* se je potrebno pozicionirati u datoteku Vizualizacija_KonvNM pomoću naredbe „cd C:/Users/User/Desktop/Vizualizacija_konvNM“ ukoliko se datoteka nalazi na u „C:/Users/User/Desktop/“. Kako bi aplikacija bila pokrenuta potrebno je upisati „python Aplikacija.py“ [Slika 16] nakon čega se otvara interaktivni prozor u kojem se odabire te sloj koji želimo vidjeti [Slika 15].



Slika 16 Upute za aplikaciju

Klikom na željeni sloj otvara se prozor sa prikazanim vizualiziranim izlazom iz sloja. Vizualizaciju sloja je moguće spremiti klikom na gumb za spremanje [Slika 17]. Predlaže se spremanje u .svg formatu radi lakše naknadne manipulacije slikom.



Slika 17 Gumb za spremanje

4.6. Mogući problemi

Ukoliko se unutar *Command prompta* javi greška da ne postoji neki od paketa potrebno je upisati „pip install -r requirements.txt“ kako bi se instalirali svi potrebni paketi. Ukoliko nedostaje mali broj paketa moguće ih je instalirati jedan po jedan tako da se upiše „pip install ime_paketa“. Kako bi se vidjeli postojeći paketi potrebno je upisati „pip freeze“.

Ako se aplikacija preuzima iz ovog rada potrebno je napraviti sljedeće. Prvo se izradi datoteka sa nazivom „Vizualizacija_KonvNM“. Nakon toga se unutar datoteke napravi datoteka „slike“, te dokument „requirements.txt“. U dokument je potrebno kopirati sve što se nalazi unutar III, a u datoteku slike ubaciti željene slike. Za dalje upute se vratiti na 4.5.

5. REZULTATI

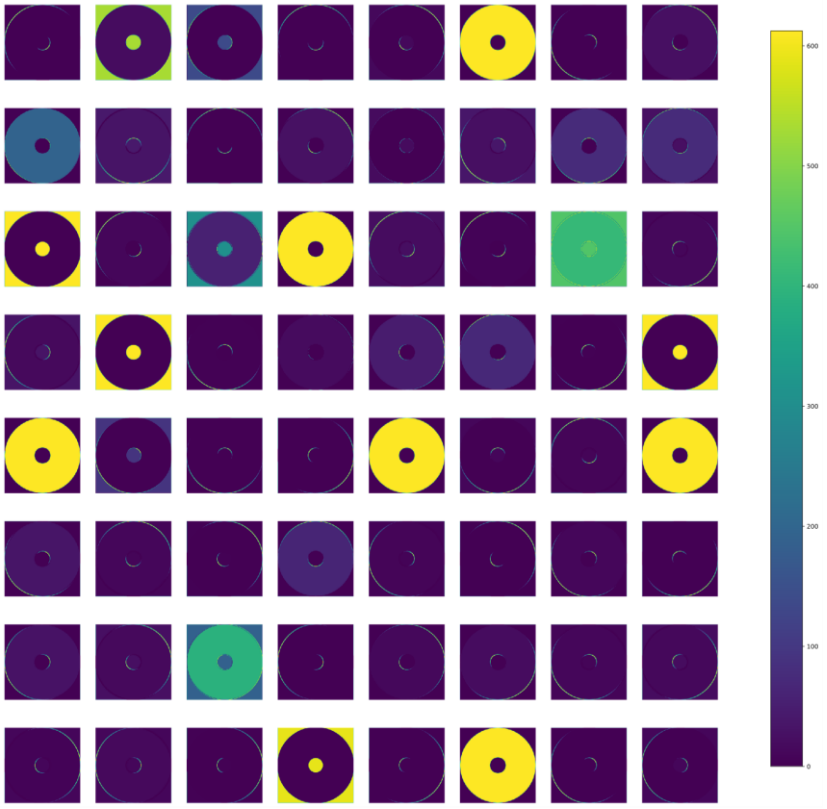
Rezultati ovog zadatka će biti vizualizirani izlazi slojeva VGG16 neuronske mreže. Prvo će biti prikazani slojevi [Slika 18] i [Slika 24] koje se nalazi u [Tablica 1]. Biti će prikazan cijeli put od prepoznavanja rubova do prepoznavanja karakteristika. Prva slika je odabrana zbog svoje jednostavnosti, a druga je primjer stvarnog korištenja vizualizacije u svrhu razumijevanja rada konvolucijskih neuronskih mreža. Nakon toga će se prikazati tablica sa svim slikama iz [Tablica 1] s vizualiziranim rezultatima.

5.1. Objašnjenje vizualizacije izlaza

5.1.1. Objašnjenje vizualizacije 2D aktivacijskih mapa na primjeru kruga

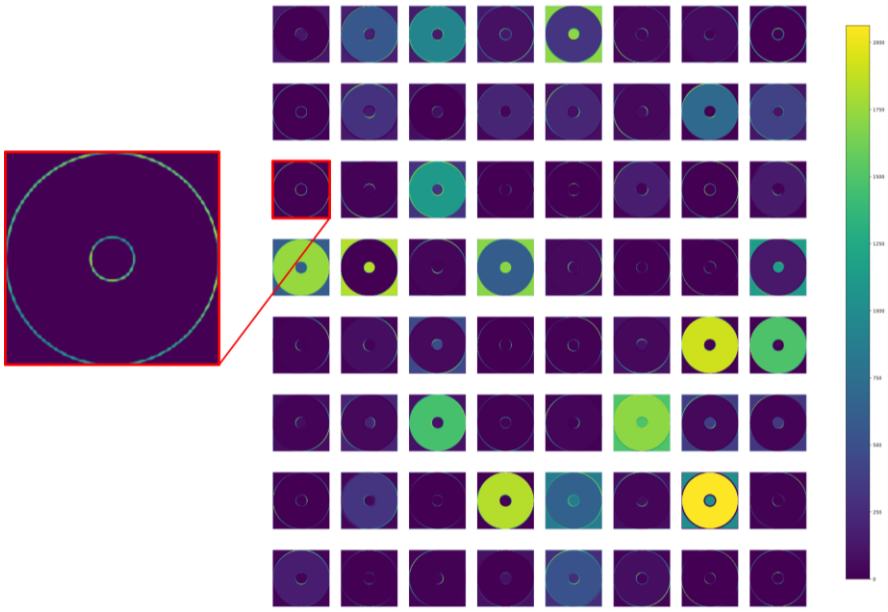
Slika 18 1. Slika iz tablice baze podataka

[Slika 18] ulazi u VGG16 konvolucijsku neuronsku mrežu. Prolaženjem kroz prvi konvolucijski sloj daje vizualizirani izlaz [Slika 19] s kojeg je vidljivo da neuroni detektiraju različite varijacije u prijelazima između boja što za rezultat daje obrise rubova u pojedinim smjerovima.



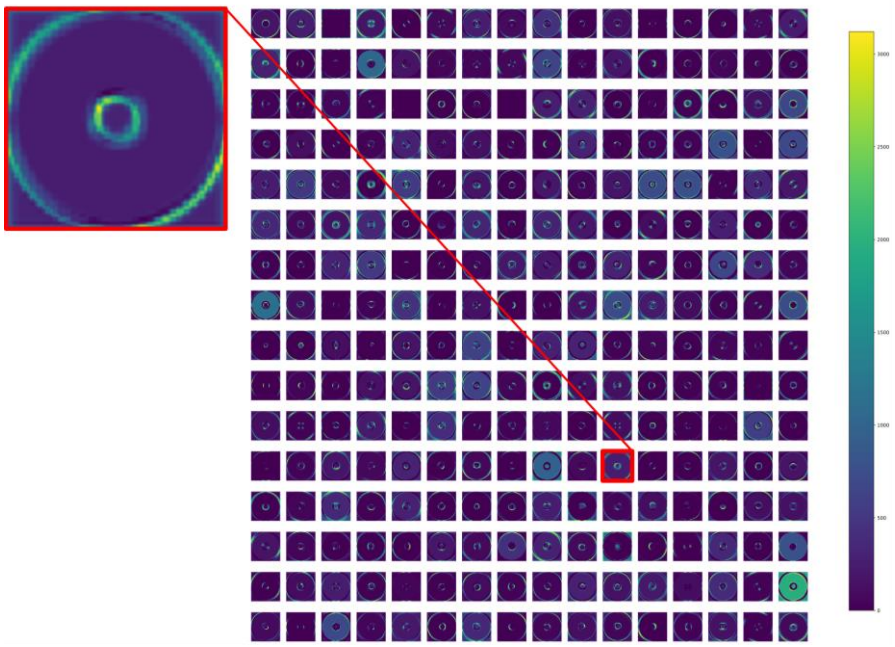
Slika 19 Prvi konvolucijski sloj

Izlaz drugog konvolucijskog sloja već daje prikaz kompletnog ruba [Slika 20], što se vidi na istaknutim neuronima koji prikazuju rubove dvaju krugova (dvije kružnice).



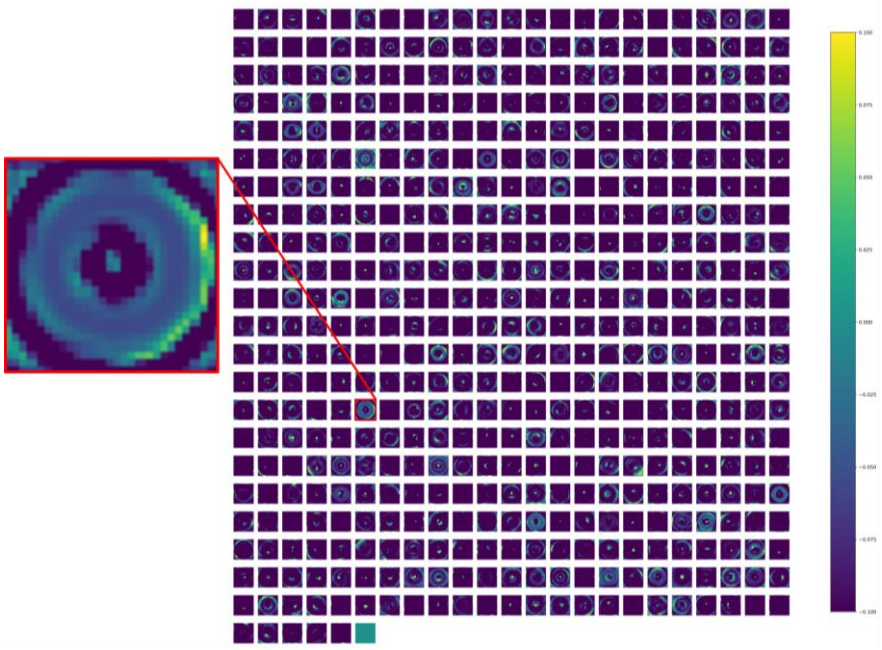
Slika 20 Drugi konvolucijski sloj

Izlaz petog konvolucijskog sloja [Slika 21] već počinju biti oblici. Istaknuti neuroni prikazuju prostor između tri objekta: okoline, velikog kruga i unutarnjeg kruga.



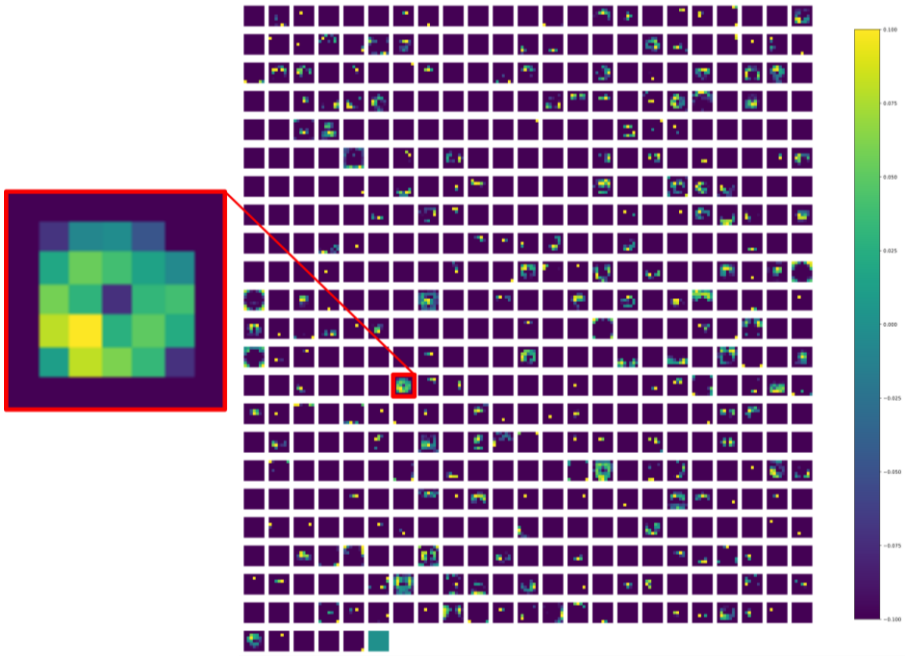
Slika 21 Peti konvolucijski sloj

U devetom konvolucijskom sloju [Slika 22] mreža se fokusira na objekte. Iz istaknutih neurona se vidi da je sav fokus na velikom krugu i da ga se definira kao objekt iz čega se može zaključiti da je to vjerojatno razlog zbog kojeg VGG16 obični crni krug detektira kao CD (predviđanje mreže).



Slika 22 Deveti konvolucijski sloj

Iz četvrtog sloja sažimanja [Slika 23] se može vidjeti kako su neuroni jako aktivni u području gdje se nalazi crni krug. Pretpostavljeno je da su navedeni neuroni najviše zaduženi za pretpostavku VGG16 mreže da se radi o CD-u.



Slika 23 Četvrti sloj sažimanja

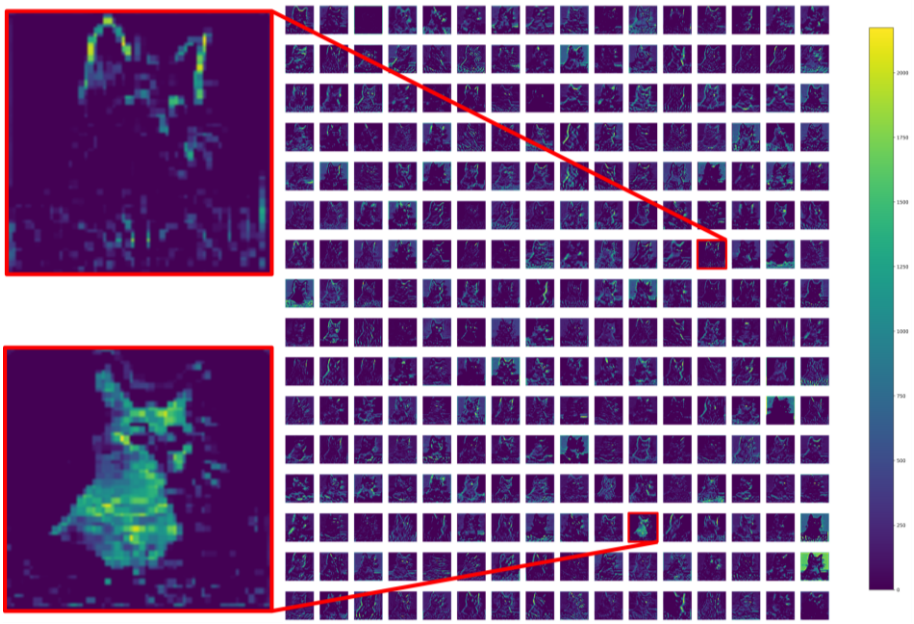
Dalje će biti prikazani i objašnjeni vizualizirani izlazi za [Slika 24].

5.1.2. Objašnjenje vizualizacije 2D aktivacijskih mapa na primjeru mačke

Slika 24 2. slika iz tablice

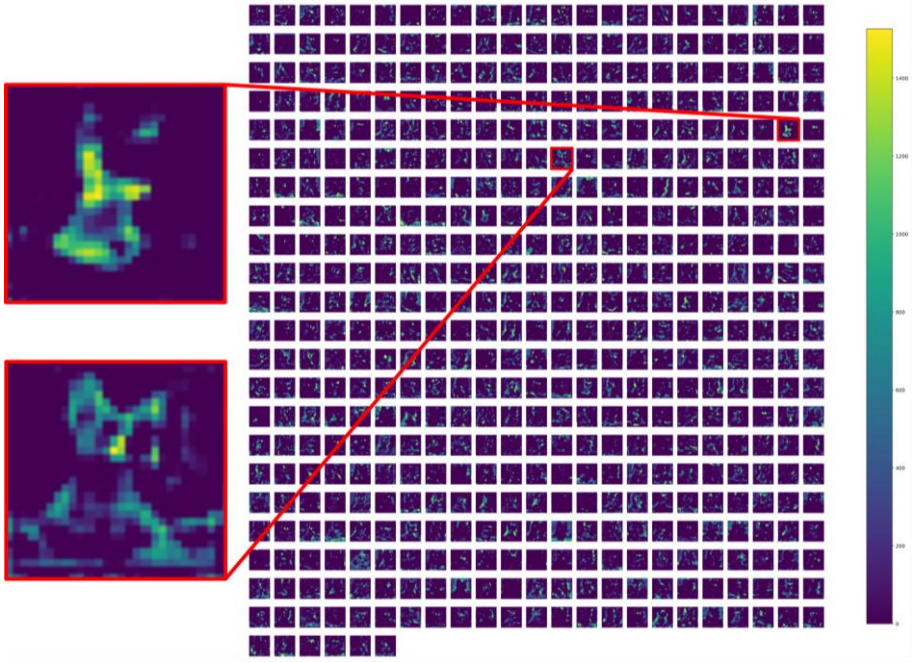
[Slika 24] ulazi u VGG16 konvolucijsku neuronsku mrežu. Kao što je već navedeno, prva dva konvolucijska sloja detektiraju rubove tako da će ovdje fokus biti na ostalima.

Peti konvolucijski sloj počinje detektirati oblike, ali na [Slika 25] se može vidjeti da se tu ne detektiraju samo oblici već i neke karakteristike. Istaknuti neuroni na 2D aktivacijskoj mapi 1 prikazuju aktivnosti u području ruba uši mačke, dok oni na 2. su aktivni na području cijele mačke.



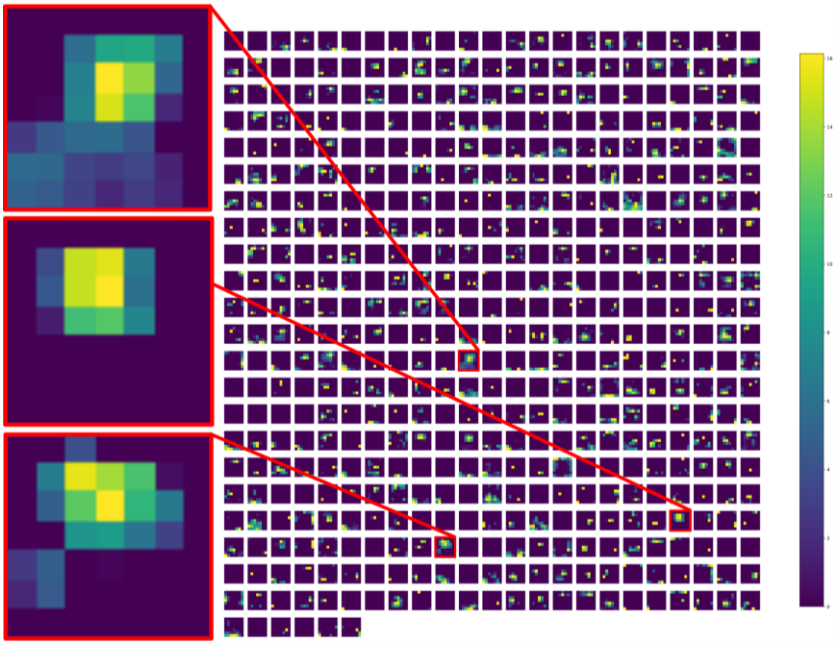
Slika 25 Peti konvolucijski sloj sa istaknutim neuronima na 2D aktivacijskim mapama 1 i 2 (odozgo prema dolje)

Iz vizualizacije izlaza devetog konvolucijskog sloja [Slika 26] vidi se velika aktivacija neurona 2D aktivacijske mape 1 na području vrata mačke te na području glave kod mape 2. U ovom dijelu se može uočiti kako su vizualizirani izlazi sve apstraktniji. Ono što mape detektiraju sve je više vezano uz elemente kao što su uši, nos, usta i slično, no zbog prevelike količine podataka teško je odrediti što se točno na šta odnosi.



Slika 26 Deveti konvolucijski sloj sa istaknutim neuronima 2D aktivacijskih mapa 1 i 2 (označene odozgo prema dolje)

Zadnji sloj [Slika 27] prije spljoštenog sloja i početka klasifikacije prikazuje 2D aktivacijske mape na kojima je moguće vidjeti aktivacije neurona na apstraktne pojmove kao što su lice, tijelo i slično. Sve navedeno je pretpostavka pošto je teško zaključiti što na slici točno aktivira neurone.

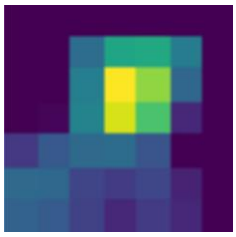
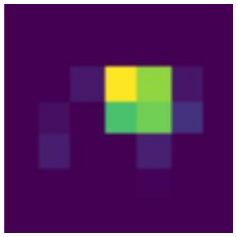
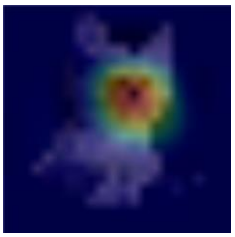
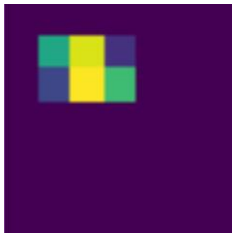
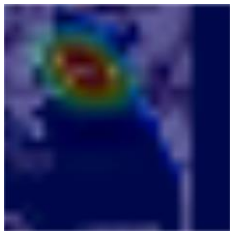
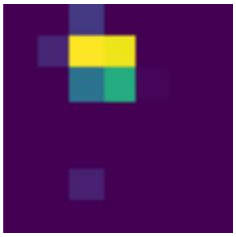
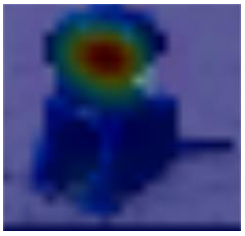
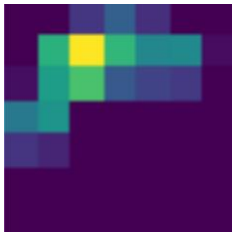
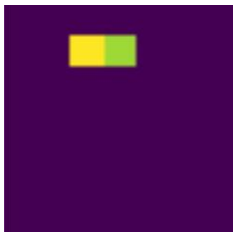
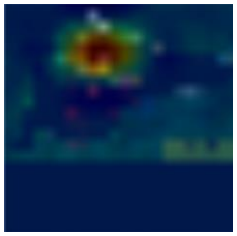
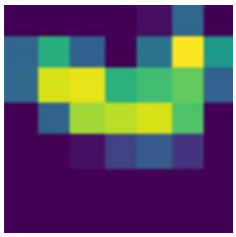
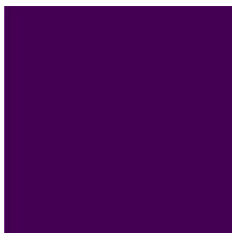


Slika 27 Četvrti sloj sažimanja

5.2. Tablica izlaza

Sljedeća tablica će sadržavati originalnu sliku sa 2D aktivacijskom mapom koja ima najaktivnije neurone jer se pretpostavlja da ta najviše utječe u kategoriziranju slike. Također će sadržavati karakterističnu 2D aktivacijsku mapu za sve slike životinja unutar tablice, kao i toplinsku skala iste 2D aktivacijske mape kako bi se bolje vidjelo što se točno detektira.

Tablica 2 Tablica 2D aktivacijskih mapa

Br.	Originalna slika životinje	2D aktivacijska mapa najaktivnijih neurona	Karakteristična 2D aktivacijska mapa	Toplinska skala karakteristične 2D aktivacijske mape
1				
2				
3				
4				
5				

Iz [Tablica 2] se može vidjeti kako jedna 2D aktivacijska mapa može biti zadužena za prepoznavanje lica, što se vidi iz stupca u kojem se nalazi prikaz karakteristične 2D aktivacijske mape s različitim aktivacijama neurona. Neuroni navedene mape se aktiviraju na području očiju i nosa, tj. lica što se može vidjeti i iz toplinske skale.

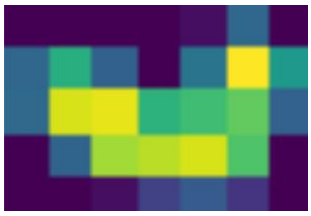
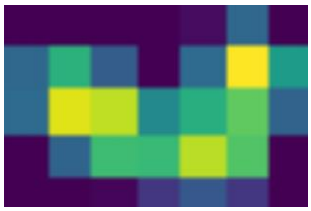
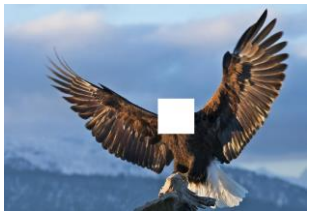
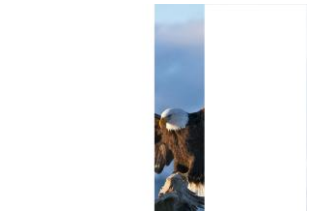
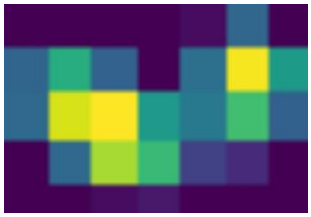
Može se uočiti kako 5. originalna slika unutar tablice, na kojoj se nalazi orao, nema aktivne neurone u karakterističnoj 2D aktivacijskoj mapi koja prepoznaje lice, ali bez obzira na to mreža preko mape za prepoznavanje krila radi pretpostavku da se radi o bjeloglavom orlu (pretpostavka mreže: „bald_eagle (54.9530565738678)“). Nedostatak prepoznatog lica je vjerojatno razlog sigurnosti mreže od 55%, dok za ostale slike je rezultat:

- 1. slika tablice predstavlja mačku, a mreža vraća da se tu radi o egipatskoj mački sa sigurnošću 56% („Egyptian_cat (56.6297709941864)“). Razlog tome je nedostatak ostatka tijela na slici mačke.
- 2. slika tablice predstavlja crvenu pandu, za koju kaže da se radi o istoj sa sigurnošću od 100% („lesser_panda (99.99641180038452)“). Može se uočiti kako bez obzira što navedena životinja liči i na mačku, mreža je sto posto sigurna da se radi o crvenoj pandi. Razlog tome je (za razliku od slike mačke) što se panda cijela na slici, pa je mreža mogla analizirati sve dijelove tijela i pritom se ne zbuniti i krenuti u krivom smjeru.
- 3. slika prikazuje psa, što je mreža i predvidjela sa postotkom 72% („bull_mastiff (72.38357067108154)“).
- 4. slika prikazuje papagaja u letu, što mreža prepoznaje kao kolibrića sa 76% sigurnosti („hummingbird (76.25269293785095)“). Daljnjom analizom vizualiziranih slojeva dodatnih slika kolibrića i papige bi se moglo vidjeti zašto je mreža napravila takvu grešku.

5.2.1. Analiza slike orla

Iz rezultata u [Tablica 2] moguće je zaključiti kako lice orla ne igra ulogu u njegovoj kategorizaciji. U ovom dijelu će se napraviti analiza segmenata slike koji igraju ulogu u njegovoj kategorizaciji. U svrhu toga će biti napravljena tablica manipulirane slike orla [Tablica 3].

Tablica 3 Manipulirane slike orla

Br.	Manipulirana slika orla (prva je original)	2D aktivacijska mapa iz prošle tablice	2D aktivacijska mapa z	Pretp. mreže:
1				bald_eagle (54.9530565738678)
2				bald_eagle (46.917736530303955)
3				bald_eagle (52.36409306526184)
4				bald_eagle (57.90954232215881)
5				web_site (18.00656169652939)
6				vulture (44.09656524658203)
7				vulture (47.47912585735321)

Br.	Manipulirana slika orla (prva je original)	2D aktivacijska mapa iz prošle tablice	2D aktivacijska mapa z	Pretp. mreže:
8				bald_eagle (21.34156972169876)

Treći stupac [Tablica 3] prikazuje 2D aktivacijsku mapu sa puno aktivnih neurona. Može se vidjeti kako sve manipulirane slike tablice, osim one pod Br. 5, imaju aktivirane neurone u toj mapi. Iz petog stupca se vidi kako su sva predviđanja osim za Br. 5 sliku vezana za vrstu ptice. S obzirom da slika Br. 8 ima prekrivena krila, a i dalje aktivne neurone u toj mapi pretpostavlja se da 2D aktivacijska mapa u trećem stupcu posjeduje neurone koji se aktiviraju kada se na slici nalazi ptica.

Iz manipuliranih slika se može zaključiti kako maksimalni značaj pri kategorizaciji imaju rep, kandže i krila, a ne glava i kljun orla. To je vidljivo iz slike pod Br. 4, gdje je prekrivanjem orlove glave mreža sigurnija da se na slici radi o orlu.

U četvrtom stupcu se može uočiti 2D aktivacijska mapa čiji su neuroni aktivni na istim mjestima za pretpostavke da se radi o orlu, a ne strvinaru. Također se može uočiti aktivacija neurona na modificiranoj slici Br. 5 za koju mreža pretpostavlja da se radi o web stranici. Analizom tih slika može se zaključiti kako se neuroni aktiviraju na mjestima gdje se nalazi noga i kandže orla. Iz toga se pretpostavlja da ta mapa ima neurone koji se aktiviraju kada se na slici nalazi noga.

6. ZAKLJUČAK

Kroz ovaj završni rad može se vidjeti teorijski opis konvolucijskih neuronskih mreža u svrhu boljeg razumijevanja i poboljšanja njihove arhitekture. Kako teorija nije dovoljna za razumijevanje mreža koje uče same sebe, stvorila se potreba za drugačijim pristupom tom modelu. U tu svrhu je unutar ovog rada proučavan i jedan od načina vizualizacije konvolucijskih mreža, vizualizacija aktivacija slojeva mreže. Kroz proučavanje te teme u radu može se vidjeti kako se tu radi o vizualiziranim 2D aktivacijskim mapama. Vizualizacija zahtjeva aplikaciju koja će to raditi, stoga je u ovome radu osmišljena osnovna modularna aplikacija. Na temelju rezultata koje je dala aplikacija se može zaključiti koji sloj neuronske mreže posjeduje neurone koji reagiraju na neke apstraktne objekte, kao što su oči, uši, lice i sl. Tako se u radu može vidjeti 2D aktivacijska mapa VGG16 konvolucijske neuronske mreže čiji neuroni reagiraju na lice na slici.

Kako bi se još bolje interpretirale 2D aktivacijske mape potrebno je izraditi bolju aplikaciju, koja će moći uspoređivati različite ulaze, slojeve, tražiti sama određene karakteristike i vraćati mape na kojima se ona nalazi.

LITERATURA

[1] Razumijevanje konvolucijskih neuronskih mreža vizualizacijom: https://cs231n.github.io/understanding-cnn/, Pristupljeno:16.02.2020.

[2] Konvolucijske neuronske mreže: https://cs231n.github.io/convolutional-networks/, Pristupljeno: 16.02.2020.

[3] Compressed Residual-VGG16 CNN Model for Big Data Places Image Recognition Hussam Qassim Abhishek Verma David Feinzimer Dept. of Computer Science Dept. of Computer Science Dept. of Computer Science California State University New Jersey City University California State University Fullerton, California 92831 Jersey City, NJ 07305 Fullerton, California 92831

[4] VGG16 konvolucijska neuronska mreža: https://neurohive.io/en/popular-networks/vgg16/, Pristupljeno: 16.02.2020.

[5] VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION Karen Simonyan & Andrew Zisserman + Visual Geometry Group, Department of Engineering Science, University of Oxford

[6] Keras: https://www.infoworld.com/article/3336192/what-is-keras-the-deep-neural-network-api-explained.html, Pristupljeno: 16.02.2020.

[7] Keract: https://github.com/philipperemy/keract#get-gradients-of-activations, Pristupljeno: 16.02.2020.

[8] Novaković, Branko ; Majetić, Dubravko ; Široki, Mladen (1998) *Umjetne neuronske mreže. = Artificial neural networks.* Sveučilište u Zagrebu, Fakultet strojarstva i brodogradnje, Zagreb, -. ISBN 953-6313-17-0

[9] Evaluation of Pooling Operations in Convolutional Architectures for Object Recognition Dominik Scherer, Andreas M¨uller, and Sven Behnke University of Bonn, Institute of Computer Science VI, Autonomous Intelligent Systems Group, R¨omerstr. 164, 53117 Bonn, Germany

[10] Deep Learning using Rectified Linear Units (ReLU) Abien Fred M. Agarap

[11] Large-Margin Softmax Loss for Convolutional Neural Networks Weiyang Liu, Yandong Wen, Zhiding Yu, Meng Yang

[12] Deep CNNs for microscopic image classification by exploiting transfer learning and feature concatenation Long D. Nguyen, Dongyun Lin, Zhiping Lin School of Electrical and Electronic Engineering, Nanyang Technological University, 639798, Singapore Jiuwen Cao Key Lab for IOT and Information Fusion Technology of Zhejiang, Hangzhou Dianzi University, 310018, China

[13] Tkinter: https://www.geeksforgeeks.org/python-gui-tkinter/, Pristupljeno: 16.02.2020.

PRILOZI

I. CD-R disc

II. Python kod

III. Zahtjevi

II. Python kod

```
from PIL import Image as Im
from keras.applications.vgg16 import VGG16
from keras.applications.vgg16 import decode_predictions
from keras.applications.vgg16 import preprocess_input
from keras.preprocessing.image import img_to_array
import matplotlib.pyplot as plt
from tkinter import *
from tkinter import ttk
import os
import functools
import keract

if __name__ == '__main__':

    model = VGG16()
    i = 0


    # Funckija koja se poziva prilikom klika na ime slike
    def klik(slika):
        j = 0
        im = 'slike/' + slika
        global image
        global top

        # Definiranje potprozora aplikacije
        top = Toplevel()
        top.title("'" + 'Odabir sloja od slike:' + slika + "'")
        frame_2 = LabelFrame(top, text="Odaberi sloj:", padx=50,
pady=5)
        frame_2.grid(row=1, column=1, padx=2)

        # Definiranje liste sa imenima slojeva
        layer_names = [layer.name for layer in VGG16().layers[1:]]
        # Definiranje aktivnih gumba za svaki sloj
        for layer in layer_names:
            button = ttk.Button(frame_2, text=layer,
                                command=functools.partial(klik2,
layer))
            button.grid(row=j + len(slike_list))
            j += 1



        # Obrađivanje slike na pravilne dimenzije
        image = Im.open(im)
        vel = image.size
```

```
        vel_max = max((vel[0], vel[1]))
        if vel_max >= 224:
            vel_x = vel_max / 224
            img_x = int(vel[0] / vel_x)
            img_y = int(vel[1] / vel_x)
            image = image.resize((img_x, img_y))
            image = image.crop((0, 0, 224, 224))
        else:
            vel_x = 224 / vel_max
            img_x = int(vel[0] * vel_x)
            img_y = int(vel[1] * vel_x)
            image = image.resize((img_x, img_y))
            image = image.crop((0, 0, 224, 224))

        # Obrada slike za korištenje VGG16 mreže
        image = img_to_array(image)
        image = image.reshape((1, image.shape[0], image.shape[1],
image.shape[2]))
        image = preprocess_input(image)
        yhat = model.predict(image)
        label = decode_predictions(yhat)
        label = label[0][0]
        print('{} ({})'.format(label[1], label[2] * 100))

        model.compile(optimizer='adam',
                      loss='categorical_crossentropy',
                      metrics=['accuracy'])


    # Funkcija koja se poziva prilikom klika na ime sloja
    def klik2(layer):
        layer_name = layer
        # Pozivanje funkcije za povlačenje aktivacije slojeva
        activations = keract.get_activations(model, image,
layer_name=layer_name)
        # Pozivanje funkcije za prikaz slojeva
        keract.display_activations(activations, cmap=None,
                                   save=False,
directory=r'Slojevi/',
                                   data_format='channels_last')
        # keract.display_heatmaps(activations, image,
directory=r'Heatmaps/', save=False)



    # Definiranje prozora aplikacije
    window = Tk()

    window.title('CNN Visualization')
```

```
    # Definiranje liste slika iz direktorija slike
    slike_list = os.listdir(r'slike\\')

    # Definiranje okvira unutar kojega će se nalaziti gumbi za slike
    frame_1 = LabelFrame(window, text="Odaberi sliku:", padx=50,
pady=50)
    frame_1.grid(row=1, column=1, padx=2)

    # Definiranje gumba za svaku sliku
    for slika in slike_list:
        button = ttk.Button(frame_1, text=slika,
                            command=functools.partial(klik, slika))
        button.grid(row=i)
        i += 1

    window.mainloop()
```

III. Zahtjevi

absl-py==0.9.0

astor==0.8.1

cachetools==4.0.0

certifi==2019.11.28

chardet==3.0.4

Click==7.0

cycler==0.10.0

gast==0.2.2

google-auth==1.11.0

google-auth-oauthlib==0.4.1

google-pasta==0.1.8

grpcio==1.26.0

h5py==2.10.0

idna==2.8

joblib==0.14.1

keract==3.0.1

Keras==2.3.1

Keras-Applications==1.0.8

Keras-Preprocessing==1.1.0

kiwisolver==1.1.0

Markdown==3.1.1

matplotlib==3.1.3

numpy==1.18.1

oauthlib==3.1.0

opt-einsum==3.1.0

Pillow==7.0.0

protobuf==3.11.3

pyasn1==0.4.8

pyasn1-modules==0.2.8

pyparsing==2.4.6

python-dateutil==2.8.1

PyYAML==5.3

requests==2.22.0

requests-oauthlib==1.3.0

rsa==4.0

scikit-learn==0.22.1

scipy==1.4.1

six==1.14.0

tensorboard==2.1.0

tensorflow==2.1.0

tensorflow-estimator==2.1.0

termcolor==1.1.0

urllib3==1.25.8

Werkzeug==0.16.1

wrapt==1.11.2