FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Lokalizacija karakterističnih točaka lica u videu

Generalić Boris
Gulan Filip
Kopljar Damir
Miličević Andrija
Nuić Hrvoje
Šarić Fredi
Zadro Tvrtko

Zagreb, siječanj 2017.

SADRŽAJ

1. Projektni zadatak

1.1. Opis projektnog zadatka

Lokalizacija karakterističnih točaka lica u videu ili fotografiji je tehnika koja se danas koristi u mnogim sustavima i uređajima. Susrećemo je na raznim društvenim servisima, poput *Facebook-a*, koji ju koriste za automatsko označavanje ljudi na fotografijama. Većina algoritama lokalizacije točaka lica su iznimno kompleksni i zahtijevaju veliku količinu procesorske snage i memorije, pa je težnja usmjerena na poboljšavanje tih algoritama. No razvojem i napretkom tehnologije algoritmi lokalizacije točaka lica se danas uspješno, bez velikih problema, izvode i na mobilnim uređajima koji ih koriste u raznoraznim aplikacijama poput alata za šminkanje gdje osoba može uz pomoć praćenja lica vidjeti kako bi izgledali s određenim bojama na svom licu.

Kroz ovaj projekt će se pokušati dani problem lokalizacije karakterističnih točaka lica riješiti uporabom dubokih neuronskih mreža.

1.2. Pregled i opis srodnih rješenja

1.2.1. Xavier P. Burgos-Artizzu - Robust Cascaded Pose Regression

RCPR metoda za detekciju karakterističnih točaka lica je poboljšanje metode CPR (*Cascaded pose regression*). CPR metoda je vrlo efikasna i precizna u određivanju karakterističnih točaka lica, ali preciznost joj drastično opada kada se određuju karakteristične točke na licima koje su prekrivene s nekom preprekom.

CPR uči kaskadu regresora $\mathrm{R}^{1\cdots\mathrm{T}}$ koji progresivno transformiraju inicijalni skup karakterističnih točaka S^0 u finalni odabira karakterističnih točaka S^{T}. Skup točaka S^{i} je definiran kao $S_p^i = [x_p, y_p], p \in 1 \dots P$. Svaki od regresora R^t producira pomak osnovne konfiguracije točaka δS^t koji se kombinira s ulazom tog regresora i stvara se ulaz u slijedeći regresor $S^t = S^{t-1} + \delta S^t$.

RCPR metoda zahtijeva da označene karakteristične točke imaju informaciju o tome je li karakteristična točka prekrivena nekom zaprekom (naočale, ruka ...) ili ne. Time je definirana karakteristična točka kao $S_p^i = [x_p, y_p, v_p]$, gdje je v_p realna vrijednost iz intervala $[0, 1]$.

Slika se podijeli u 3×3 polje. Svako polje ima informaciju o postotku prepreka unutar polja dobivenu kao procjena temeljena na S^{t-1} karakterističnih točaka. U svakom koraku t se uči S_tot regresora koji se treniraju samo na 1 od 9 predefiniranih polja (svakom regresoru R_i^t je nasumično dodijeljeno polje). Regresori generiraju pomake $\delta S_{1...tot}$ iz kojih se računa težinski prosjek, gdje su težine obrnuto proporcionalne količini zapreka u polju na kojem je treniran regresor(za dobre rezultate dovoljno je koristiti $S_tot = 3$ regresora). Na kraju kaskade se dobiva finalni skup točaka $S_p^T = [x_p, y_p, v_p]$ te je potrebno odrediti prag τ koji označava je li točka S_p prekrivena ili ne.

Metoda je testirana na tri različita skupa podataka (**LFPW**, **HELEN**, **LFW**) u kojima lica nisu prekrivena i postigla je bolje rezultate u odnosu na prijašnje metode, te je testirana na skupu podataka **COFW** u kojem su karakteristične točke lica djelomično prekrivene te je postigla zadovoljavajuće rezultate.

1.3. Konceptualno rješenje zadatka

Sam sustav za lokalizaciju karakterističnih točaka lica je podijeljen u više segmenata, tj. podsustava. Prvi segment sustava na ulaz prima sliku ili jedan vremenski okvir video isječka. Dana slika ili isječak se zatim pretvaraju u sliku sivih nijansi. Tako obrađena slika se dovodi na ulaz podsustava za izlučivanje položaja svih lica na slici te kao rezultat vraća listu u obliku: koordinate gornjeg lijevog ugla, širina i visina lica.

Tako dobivena lista se zatim iskoristi na način da se iz slike sivih nijansi izrežu prepoznata lica i skaliraju. Pojedina skalirana lica dovede se na ulaze duboke neuronske mreže koja kao izlaze daje koordinate odabranih karakterističnih točaka lica. Tako dobivene točke skaliraju se u prostor početne slike ili isječka te se iscrtavaju i prikazuju korisniku sustava.

2. Postupak rješavanja zadatka

2.1. Pretvorba boje u nijanse sive

Prvi korak koji je potrebno napraviti na ulaznoj slici je pretvoriti ju u sliku sivih nijansi. Kod prikaza boja slike korištenjem aditivnog RGB (engl. *Red Green Blue*) modela postoje tri komponente: crvena, zelena i plava. Kombinacijom te tri komponente u različitim omjerima dobivamo ostale boje. Da se uočiti da podjednakom raspodjelom svih triju komponenti dobivamo boje iz sivog spektra, pa se algoritam pretvorbe u sliku sivih nijansi temelji na odabiru jedne vrijednosti iz dane tri komponente kako bi se dobila siva nijansa.

2.1.1. Prvi algoritam

Prvi algoritam je vrlo jednostavan i intuitivan. Siva nijansa pojedinog slikovnog elementa se dobiva tako da se boja slikovnog elementa rastavi na tri navedene komponente. Omjer pojedinih komponenti se zbraja te se uzima srednja vrijednost, kako je prikazano izrazom (2.1).

$$E_y = \frac{E_R + E_G + E_B}{3} \tag{2.1}$$

2.1.2. Drugi algoritam

Prvi algoritam je intuitivan i vrlo jednostavan, no u praksi ne daje najbolje rezultate. Problem leži u ljudskom oku i načinu na koji opaža boje. Ljudsko oko zelenu boju opaža puno jače nego crvenu, te crvenu opaža jače nego plavu. Stoga intuitivno slijedi da bi zelena trebala biti najzastupljenija, zatim crvena i plava. Organizacija ITU (engl. *International Telecommunication Union*) u svojoj normi ITU-R BT.709-6 [2] predlaže drugi algoritam u kojem bi crvena komponenta imala udio s 22.16%, zelena s 71.56% te plava s 7.22%, što je prikazano izrazom (2.2).

$$E_y = 0.2126E_R + 0.7152E_G + 0.0722E_B \tag{2.2}$$

Na slici 2.1 mogu se usporediti rezultati obadva izraza, gdje je jasno vidljiva prednost korištenja izraza (2.2).

Slika 2.1: Usporedba prvog i drugog algoritma pretvorbe boja u nijanse sive

2.2. Izjednačavanje histograma

Nakon što je slika u boji pretvorena u sliku sivih nijansi vrlo lako je moguće da kontrast na danoj slici nije najbolji. U namjeri poboljšanja kontrasta na slici te izoštravanja pojedinih elemenata koristi se izjednačavanje histograma.

Izjednačavanje histograma slike je operacija pri kojoj se slika mijenja tako da broj točaka za pojedinu nijansu sive bude približno jednoliko raspoređen. Matematički

rečeno izjednačavanje histograma implicira preslikavanje početne distribucije na širu i "uniformniju" distribuciju.

Kako je za potrebe ovog projekta korištena biblioteka *OpenCV*, za izjednačavanje histograma korištena je metoda *equalizeHist* iz navedene biblioteke. Primjer rada algoritma vidljiv je na slici 2.2.

Slika 2.2: Izjednačavanje histograma

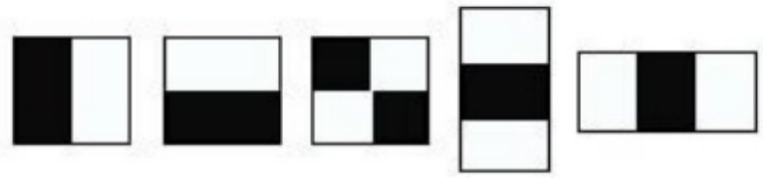
2.3. Detekcija i izlučivanje lica

Nakon što je slika u potpunosti obrađena slijedi detekcija, tj. izlučivanje pozicija lica na slici. Kako je za potrebe ovog projekta korištena biblioteka *OpenCV*, za izlučivanje pozicije lica korištene korištene su kaskade boostanih Haarovih klasifikatora.

Metodu korištenja boostanih Haarovih klasifikatora predložili su Paul Viola i Michael Jones ne tako davne 2002. godine. To je ujedno bila prva metoda iz strojnog učenja koja postiže dobre rezultate u realnom vremenu. Iako je primarna svrha te metode prepoznavanje lica, može se iskoristiti za učenje prepoznavanja i ostalih objekata [6].

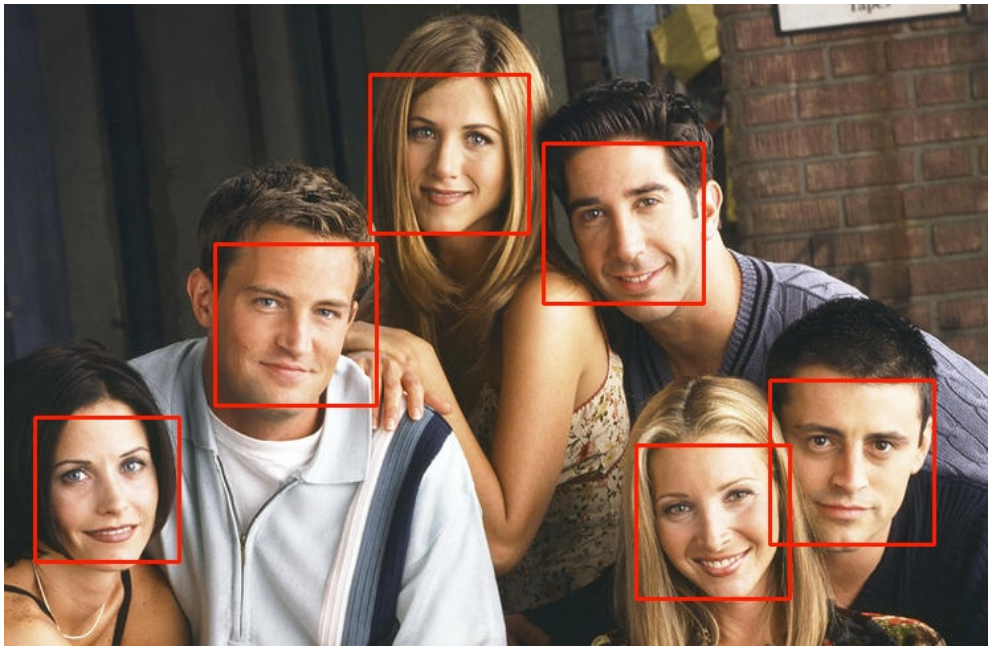
Metoda Viola-Jones umjesto direktnog rada s slikovnim elementima ulazne slike, što se pokazao kao proces s velikom složenošću, radi s određenim skupom reduciranih značajki te slike, to jest Haarovim značajkama. Haarove značajke su pravokutne značajke i računaju se kao razlika nekoliko suma slikovnih elemenata unutar različitih pravokutnih područja slike. Vrijednosti dobivene takvim računom predstavljaju prisustvo ili odsustvo nekih karakteristika slike (kao što su rubovi, kutevi i sl.) [6].

Na slici 2.3 prikazane su osnovne Haarove značajke. Skaliranjem, rotacijom i translacijom osnovnih značajki nastaju ostale Haarove značajke. Svaka značajka je konvolucijska maska, a vrijednost značajke, kako je već navedeno, dobiva se oduzimanjem



Slika 2.3: Osnovne Haarove značajke

sume vrijednosti slikovnih elemenata ispod bijelog područja od sume crnog područja pomnoženog s omjerom crne i bijele površine. Upravo zbog toga su te značajke vrlo osjetljive na promjenu kontrasta pa ih se interpretira kao značajke ruba ili linije [1].



Slika 2.4: Primjer rada kaskadnog Haarovog klasifikatora

CascadeClassifier razred iz biblioteke *OpenCV* u konstruktoru prima putanju do *xml* datoteke koja sadrži naučenu kaskadu Haarovih klasifikatora. Za potrebe ovog rada korištena je datoteka *haarcascade frontalface default* koja dolazi skupa s bibliotekom *OpenCV*. Metoda *detectMultiScale* prima na ulaz sliku sivih nijansi te vraća listu pozicija lica na slici. Primjer rada algoritma je vidljiv na slici 2.4.

2.4. Lokalizacija karakterističnih točaka lica

Nakon izlučivanja pojedinih lica iz slike, dana lica se režu te skaliraju na uniformne vrijednosti, u ovom slučaju na dimenziju 96×96 slikovnih elemenata. Tako skalirana

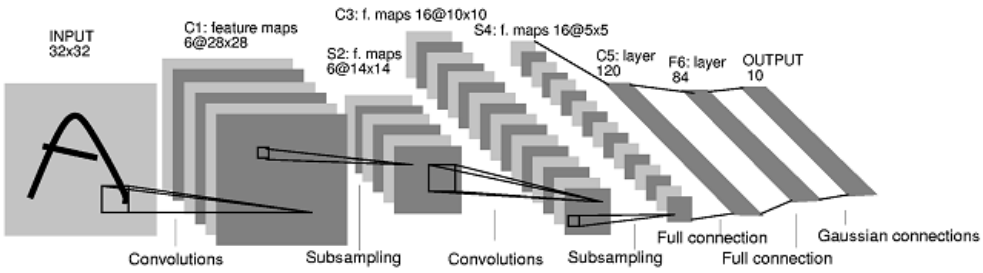
lica, tj. njihovi slikovni elementi se dovode na ulaze konvolucijske neuronske mreže koja na svojim izlazima daje koordinate petnaest karakterističnih točaka lica: vrh nosa, lijevi i desni rub usta, sredina gornje i donje usne, centar oba oka, lijevi i desni kraj oba oka te lijevi i desni kraj obje obrve.

Primjer rada konvolucijske neuronske mreže na lokalizaciji karakterističnih točaka lica vidljiv je na slici 2.5.

Slika 2.5: Lokalizacija karakterističnih točaka lica

2.4.1. Konvolucijska neuronska mreža

Konvolucijske neuronske mreže mogu se opisati kao nadogradnja nad običnim višeslojnim unaprijednom mrežama. Konvolucijska, kao i obična, neuronska mreža sastoji se od jednog ulaznog, jednog izlaznog i jednog ili više skrivenih slojeva. Kod konvolucijskih neuronskih mreža specifični su konvolucijski slojevi i slojevi sažimanja. Osim njih često se koriste i potpuno povezani slojevi. Konvolucijske neuronske mreže najčešće kreću s jednim ili više konvolucijskih slojeva, zatim slijedi sloj sažimanja, pa ponovo konvolucijski sloj i tako nekoliko puta. Mreža najčešće završava s jednim ili više potpuno povezanih slojeva koji služe za klasifikaciju. Arhitektura konvolucijskih neuronskih mreža pokazala se izrazito dobra u radu sa slikama i prepoznavanju značajki s istih. Arhitektura jedne takve mreže prikazana je na slici 2.6.



Slika 2.6: Konvolucijske neuronske mreže - preuzeto: *https://goo.gl/a8baL9*

2.4.2. Slojevi konvolucijske neuronske mreže

Konvolucijski sloj jedan je od glavnih dijelova svake konvolucijske neuronske mreže. Svaki konvolucijski sloj sastoji se od filtara koje sadrže težine koje je potrebno naučiti kako bi mreža davala dobre rezultate. Filtri su najčešće manjih prostornih dimenzija od ulaza, no uvijek su jednake dubine kao i ulaz. Izlaz sloja bit će sve aktivacijske mape generirane od strane filtara, odnosno izlaz će biti više dvodimenzionalnih matrica gdje svaka predstavlja jednu dubinu izlaza [4]. U konvolucijskom sloju svaki izlazni neuron povezan je s pravokutnim podskupom ulazne rešetke neurona pri čemu korenspondirajuće veze imaju iste težine. Lokalna povezanost neurona i dijeljenje parametara uvelike smanjuje broj parametara koje je potrebno naučiti treniranjem mreže. Upravo zbog navedenih svojstava konvolucijskog sloja konvolucijske mreže su idealne za računalni vid gdje pretpostavljamo da se objekti na slikama sastoje od jednostavniji elemenata tj. značajki.

U većini slučajeva nakon kovolucijskog sloja dolazi sloj sažimanja s ciljem smanjivanja rezolucije izlaznih mapi iz konvolucijskog sloja. No osim smanjivanja rezolucije, slojevi sažimanja povećavaju prostornu invarijantnost. Najčešće korišteni način sažimanja je sažimanje maksimalnom vrijednosti gdje se grupirane vrijednosti zamjenjuju maksimalnom vrijednošću.

Kao regularizacijski mehanizam kod dubokih neuronskih mreža koriste se *dropout* slojevi. *Dropout* sloj dolazi nakon kombinacije konvolucijskog sloja i sloja sažimanja ili potpuno povezanoga sloja. Njegova je svrha da tokom učenja prekida pojedine veze između neurona i na taj način ne dolazi do promjena težina neurona niti on utječe na druge neurone.

Za potrebe ovog projekta korištena su sva tri navedena sloja te je arhitektura mreže bila sljedeća:

- ulazni sloj dimenzija $96 \times 96 = 9216$,
- kombinacija konvlucijskog sloja, sloja sažimanja te droput sloja - tri slijeda,

– potpuno povezani sloj,
– dropout sloj te
– izlazni sloj dimenzija 30 - koordinate 15 karakterističnih točaka lica.

3. Ispitivanje rješenja

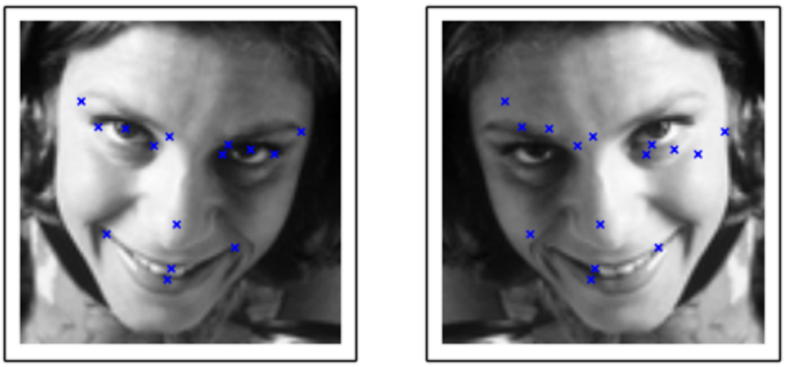
3.1. Ispitna baza

Baza lica preuzeta je s jednog od natjecanja sa servisa *Kaggle* [3]. Točnije, preuzeta je s natjecanja *Facial Keypoints Detection* koje je otvoreno 2013. godine i zatvorilo se nedavno, početkom 2017. godine, za vrijeme pisanja ovog rada.

Dana baza se sastoji od 7049 označenih slika lica u skupu za učenje te 1783 neoznačenih slika u skupu za testiranje, odnosno u skupu koji se poglavito koristio za samu evaluaciju natjecanja. Svaka slika lica je dimenzija 96×96. Slike lica su označene sa 15 karakterističnih točaka:

- vrh nosa,
- lijevi i desni rub usta,
- sredina gornje i donje usne,
- centar oba oka,
- lijevi i desni kraj oba oka te
- lijevi i desni kraj obje obrve.

Bitno je napomenuti da nisu svih 7049 slika lica iz skupa za učenje u potpunosti označene, već nekima nedostaju pojedine značajke. Stoga se broj iskoristivih slika za učenje smanjio na brojku nešto manju od 2200. No, za rješavanje tog problema iskorištena je metoda koja je predložena u članku [5]. Dakle, svaka dostupna slika lica sa svim označenim značajkama je zrcaljena, kako je prikazano na slici 3.1, te na taj način se skup za učenje udvostručio i osigurana je bolja generalizacija neuronske mreže.

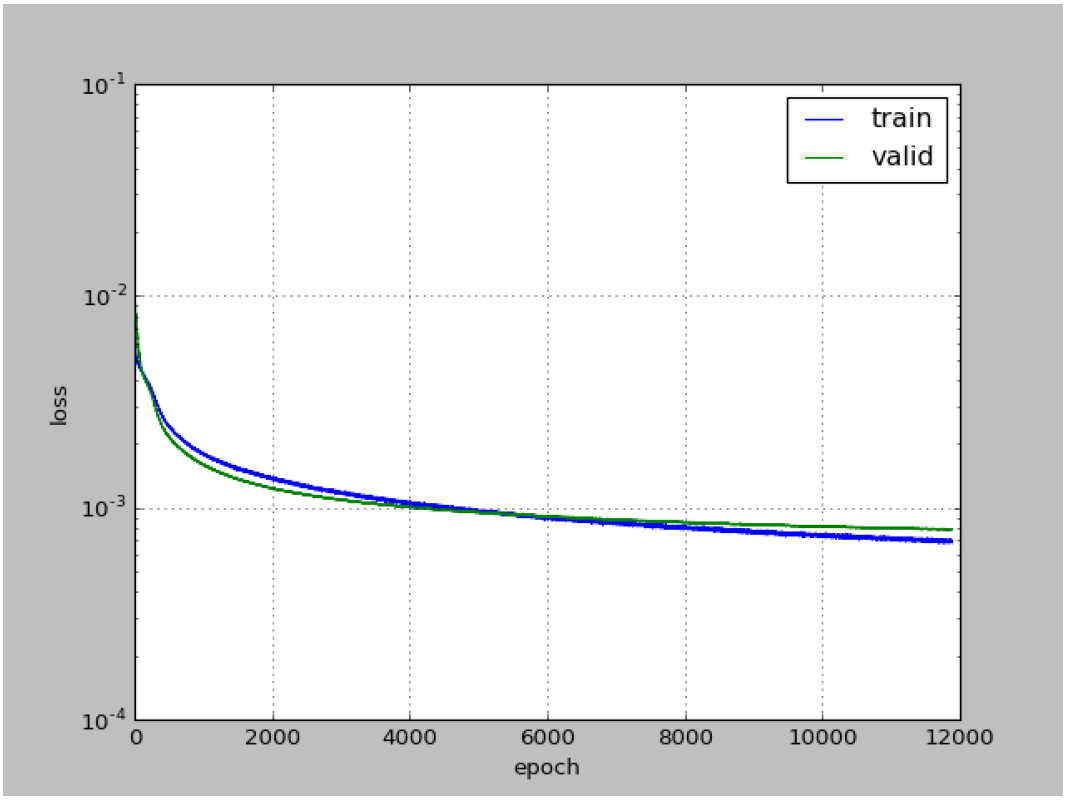


Slika 3.1: Zrcaljeno lice

3.2. Rezultati učenja i ispitivanja

Što se tiče samog učenja duboke neuronske mreže, ono je provedeno uporabom grafičke kartice. Kako je učenje duboke neuronske mreže uporabom običnog stolnoga računala, bez korištenja grafičke kartice, vrlo sporo bilo je nužno posegnuti za nešto kompliciranijim i skupljim metodama. Za potrebe ovog projekta na servisu *Microsoft Azure* podignut je virtualni stroj s *nVidia Tesla K80 24GB CUDA* grafičkom karticom koji je učenje mreže obavljao bez većih problema. Usporedbe radi, na stolnom računalu jedna epoha učenja duboke neuronske mreže je trajala oko 63 sekunde, dok je na virtualnom stroju to iznosilo oko 1.33 sekunde. Kako je mreža u ovom projektu učena i na više od 18.000 epoha, razlika je bila više nego osjetna.

Prilikom učenja duboke neuronske mreže 30% skupa za učenje je izuzeto kao skup za validaciju na kojem se provjeravala uspješnost generalizacija. Na grafu 3.2 je prikazan odnos srednje kvadratne pogreške između skupa za validaciju i skupa za testiranje prilikom učenja. Tako naučena mreža je ostvarila minimalnu srednju pogrešku na skupu za učenje iznosa 0.00068, te na skupu za validaciju iznosa 0.00072.



Slika 3.2: Srednja kvadratna pogreška tokom učenja

4. Opis programske implementacije rješenja

Programski kod korišten u ovome projektu napisan je u programskom jeziku Python (verzija 2.7.13). U projektu je korišteno nekoliko vanjskih biblioteka koje su potrebne za rad programa:

- NumPy - rad s matricama,
- Scikit-learn - za učenje mreže,
- Theano - za učenje mreže kako na grafičkoj kartici tako i na procesoru,
- Matplotlib - prikaz grafova,
- Pandas - olakšan rad s korištenim strukturama,
- CUDA toolkit - za pristup i rad na grafičkoj kartici i
- OpenCV - biblioteka za prepoznavanje lica i rad s video zapisima.

Program se sastoji od 7 datoteka koje su potrebne za učenje i izvođenje programa. Datoteke *common.py*, *tst_scene_render.py* i *video.py* koriste se za podršku rada s kamerom.

Unutar datoteke *learn.py* nalazi se implementacija koda koja uči mrežu, odnosno donosi predikcije za dani ulaz. Iako je većina parametara za trenutni problem dobro podešena, korisnik može mijenjati parametre mreže ili parametre algoritma učenja. Neki od parametara koje je moguće lako podesiti su broj filtara, veličinu filtra, *dropout* stopu, broj neurona u skrivenom sloju, moment i stopu učenja ili broj epoha.

Parametre koje je potrebno podesiti su varijable *FTRAIN* i *FTEST*. Vrijednost varijable *FTRAIN* potrebno je postaviti kao putanju do datoteke u kojoj se nalaze podaci za učenje. Vrijednost varijable *FTEST* treba postaviti kao putanju do datoteke u kojoj se nalaze podaci za testiranje.

Unutar datoteke *face_points_detector.py* nalazi se kod koji najprije poziva algoritam za prepoznavanje lica te zatim koristi sliku lica kao ulaz za implementirani algori-

tam koji prepoznaje karakteristične točke u licu.

Unutar datoteke *facedetect.py* nalazi se glavna metoda koja pokreće sam program. Unutar nje potrebno je postaviti putanju do mreže koja će se koristi.

Program je moguće pokretati na dva načina. Prvi način poziva se kao: *python facedetect.py* i na taj način program će koristiti kameru računala te će nad videozapisom kojeg dobiva od kamere pokretati predikciju karakterističnih točaka. Drugi način poziva se kao: *python facedetect.py putanjaDoSlike*, gdje *putanjaDoSlike* predstavlja stvarnu putanju do slike, koja će se koristi kao ulaz za predikciju karakterističnih točaka lica na slici.

5. Zaključak

Duboka neuronska mreža se pokazala kao dobar model za rješavanje problema lokalizacije karakterističnih točaka lica. Unatoč lošijim performanse na običnim stolnim računalima prilikom samog učenja, sama mreža nije predstavljala usko grlo sustava prilikom samog iskorištavanja za lokalizaciju točaka.

Daljnja poboljšanja se mogu ostvariti u vidu nešto boljeg i većeg skupa podataka. Uočeno je nekoliko problema sa samo mrežom za koje je većinom odgovoran skup podataka za učenje. Na primjer, mreža teško prati jako otvorena usta, čisto iz razloga jer takvih primjera slika nema u skupu za učenje. Također ima problema s praćenjem visoko podignutih obrva, iz istog razloga kao i s ustima.

Što se tiče samog sustava kao cjeline problem predstavlja nerobusnost Haarovih kaskadnih klasifikatora prilikom lokalizacije lica te bi se u budućim radovima trebalo posvetiti i tom problemu. Navedeni kaskadni klasifikatori jako teško detektiraju lice ukoliko je osoba malo nakrivila ili zakrenula glavu.

LITERATURA

[1] Dino Franić. Opencv biblioteka i primjene, 2016. URL http://digre.pmf.unizg.hr/5189/1/Dino_Franic_diplomski.pdf. Pristupano: 15.01.2017.

[2] ITU. Parameter values for the *HDTV* standards for production and international programme exchange, 2015. URL https://www.itu.int/dms_pubrec/itu-r/rec/bt/R-REC-BT.709-6-201506-I!!PDF-E.pdf. Pristupano: 15.01.2017.

[3] Kaggle. Facial keypoints detection, 2017. URL https://www.kaggle.com/c/facial-keypoints-detection. Pristupano: 15.01.2017.

[4] Damir Kopljar. Konvolucijske neuronske mreže, 2016.

[5] Daniel Nouri. Using convolutional neural nets to detect facial keypoints tutorial, 2014. URL http://danielnouri.org/notes/2014/12/17/using-convolutional-\
\neural-nets-to-detect-facial-keypoints-tutorial/. Pristupano: 15.01.2017.

[6] Mateja Čuljak. Programska implementacija učenja kaskade boostanih haarovih klasifikatora, 2010. URL http://www.zemris.fer.hr/~ssegvic/project/pubs/culjak10bs.pdf. Pristupano: 15.01.2017.