SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 1202

Raspoznavanje lica

Neven Pičuljan

Zagreb, lipanj 2016.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA
ODBOR ZA DIPLOMSKI RAD PROFILA

Zagreb, 29. veljače 2016.

DIPLOMSKI ZADATAK br. 1202

Pristupnik: **Neven Pičuljan (0036464897)**
Studij: Računarstvo
Profil: Računarska znanost

Zadatak: **Raspoznavanje lica**

Opis zadatka:

Automatsko raspoznavanje lica zasniva se na tehnikama računalnog vida a služi za raspoznavanje identiteta čovjeka na osnovu slike lica, pri čemu se koristi usporedba s ranije registriranim slikama lica za koje je poznat identitet. Skup ranije registriranih lica naziva se galerija. Jedna od osnovnih tehnologija potrebih za raspoznavanje lica je detekcija značajki lica, kojoj je cilj pronaći karakteristične točke ljudskog lica (npr. vrh nosa, kutevi usana, zjenica itd.) u digitaliziranoj slici lica.
Vaša je zadaća proučiti tehnologije automatskog raspoznavanja lica i sustav za detekciju značajki lica dostupan na Zavodu za telekomunikacije te predložiti i implementirati metodu za automatsko raspoznavanje lica na bazi detektiranih značajki i teksture lica.
Svu potrebnu literaturu i uvjete za rad osigurat će Vam Zavod za telekomunikacije.

Zadatak uručen pristupniku: 18. ožujka 2016.
Rok za predaju rada: 1. srpnja 2016.

Mentor:

Prof. dr. sc. Igor Sunday Pandžić

Djelovođa:

Doc. dr. sc. Tomislav Hrkać

Predsjednik odbora za diplomski rad profila:

Prof. dr. sc. Siniša Srbljić

Zahvaljujem se svom mentoru, prof. dr. sc. Igoru S. Pandžiću, na savjetima i stručnoj pomoći na diplomskom i preddiplomskom studiju.

Posebno se zahvaljujem Nenadu Markušu, mag. ing. na korisnim i konstruktivnim savjetima oko problema vezanih uz izradu diplomskog rada.

Hvala kolegama iz Visage Technologiesa na pomoći oko izrade diplomskog rada.

Veliko hvala svim kolegama s FER-a koji su na bilo koji način uljepšali i olakšali studiranje.

Hvala svim prijateljima koji su cijelo vrijeme uz mene.

Zahvaljujem se roditeljima, Katarini i Nedjeljku, te braći, Bojanu i Filipu, bez čije bi podrške ovaj rad teško ugledao svjetlo dana.

Na kraju, zahvaljujem se Margareti Kušan na lijepim trenutcima bez kojih studentski dio života ni približno ne bi bio toliko dobar i zanimljiv.

SADRŽAJ

1. Uvod

U ovom radu opisan je razvoj sustava za automatsko raspoznavanje lica. Automatsko raspoznavanje lica zasniva se na tehnikama računalnog vida, a služi za raspoznavanje identiteta čovjeka na osnovu slike lica, pri čemu se koristi usporedba s ranije registriranim slikama lica za koje je poznat identitet. Skup ranije registriranih lica naziva se galerija.

Sustav opisan u radu koristi sustav za detekciju značajki lica dostupan na Zavodu za telekomunikacije Fakulteta elektrotehnike i računarstva u Zagrebu.

Cjelokupni sustav temelji se na strojnom učenju. Strojno učenje sastoji se od tri bitna dijela. Prvo je potrebno prikupiti podatke za učenje. Nakon toga se model strojnog učenja uči na prikupljenim podacima i za svaki ulazni podatak daje neki izlaz. Izlaz može biti vjerojatnost da neki podatak pripada određenoj kategoriji, a može biti i deskriptor ulaznog podatka, npr. ako je ulazni podatak slika, model može sažeti sliku u prostor niže dimenzionalnosti, ali tako da Euklidska udaljenost deskriptora slika na kojima je isti objekt bude što manja, a Euklidska udaljenost deskriptora slika na kojima su različiti objekti bude što veća. Procesom učenja model postaje precizniji i daje točnije izlaze. Na kraju se naučeni model testira na podacima koji nisu sudjelovali u učenju modela.

Metoda korištena u radu temelji se na učenju izlučivanja deskriptora slike lica korištenjem duboke konvolucijske umjetne neuronske mreže, jednog od najmoćnijih algoritama strojnog učenja za računalni vid. Mreža je učena tako da kvadrat L2 udaljenosti u prostoru deskriptora izravno odgovara sličnosti lica. Udaljenost između deskriptora lica iste osobe manja je od udaljenosti između deskriptora lica različitih osoba. Jednom kada je deskriptor izlučen, raspoznavanje postaje trivijalan zadatak - svodi se na *k-NN* klasifikacijski problem.

2. Raspoznavanje lica

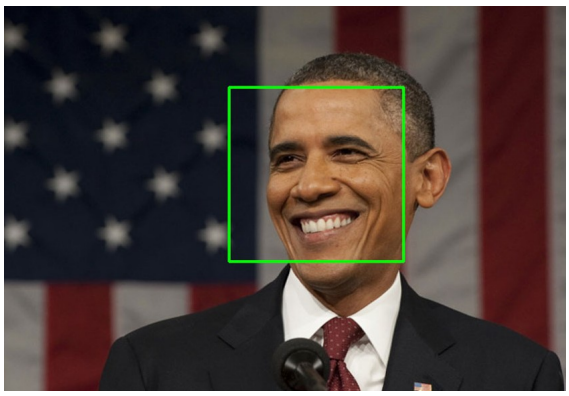
Lice je jedinstvena oznaka osobe. Svi ljudi imaju međusobno različita lica, ono je diskriminatorna značajka ljudskih bića. Iz lica možemo dobiti puno bitnih informacija o osobi. Na primjer, možemo odrediti njeno raspoloženje i emocije i koliko je osoba stara.

Raspoznavanje lica ljudi obavljaju s nevjerojatnom lakoćom, brzinom i spontanošću čak i u nepovoljnijim vizualnim uvjetima kao što su npr. razlike u osvijetljenosti prostora. Ljudi vrlo lako raspoznaju lica i kada postoji promjena poze i izraza lica, promjena frizure ili kada npr. osoba čije se lice promatra nosi naočale. Starenje lica obično ne utječe značajno na ljudsku moć raspoznavanja lica. Raspoznavanje je vrlo lagan zadatak za ljude i kada je lice različito centrirano, u različitim položajima, kada je kut gledanja u lice različit i slično. Ipak, za računala je taj zadatak izuzetno težak baš zbog tog velikog broja različitih oblika u kojima se isto lice može pojaviti, a pravila koja bi pokrila sve moguće slučajeve nemoguće je eksplicitno isprogramirati.

Slika 2.1: Različiti oblici istog lica

Prepoznavanje lica ima veliku potencijalnu primjenu u sigurnosnim sustavima. Budući da svi ljudi na svijetu imaju različita lica, ono može poslužiti kao ključ za pristup različitim sustavima.

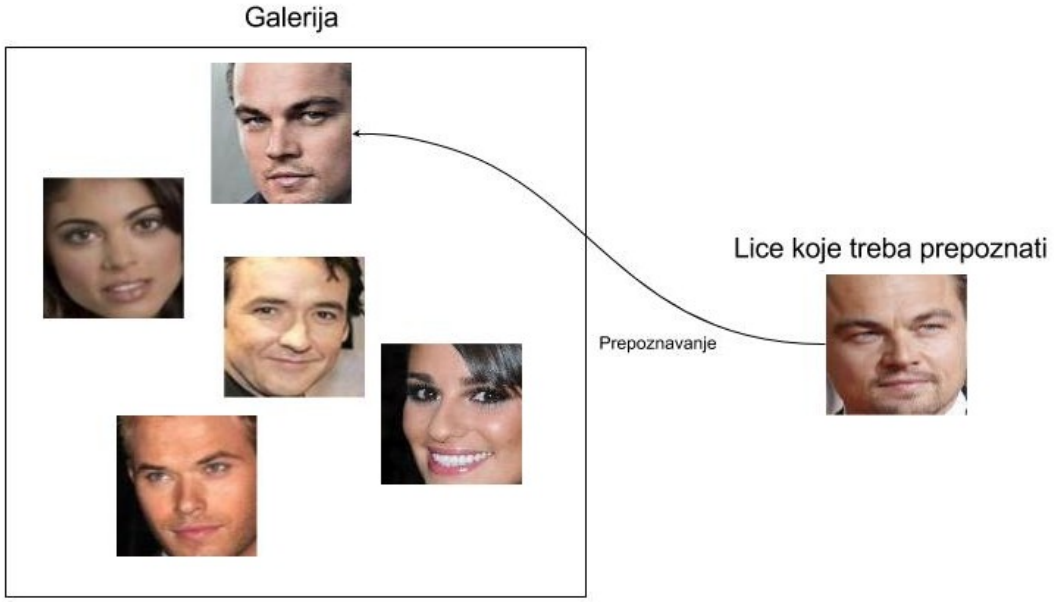
Ulaz u sustav za raspoznavanje lica je slika. Prvi korak koji sustav mora napraviti je detektirati lice na slici, odnosno pronaći koordinate pravokutnika koji uokviruju lice (engl. *bounding box*). Primjer je prikazan na slici 2.2.



Slika 2.2: *Bounding box* oko detektiranog lica

Nakon što je lice detektirano, potrebno je komprimirati, sažeti informacije s lica i zadržati samo njegove najbitnije značajke. To se postiže izlučivanjem deskriptora lica, preslikavanjem u prostor niže dimenzionalnosti, ali tako da deskriptor bude diskriminirajući, da za svaka dva deskriptora dva lica s dvije različite slike nastale u različitim uvjetima Euklidska udaljenost bude što manja, a za svaka dva deskriptora dva različita lica Euklidska udaljenost bude što veća.

U zadnjem koraku raspoznavanja identiteta osobe koristi se usporedba s ranije registriranim deskriptorima slika lica za koje je poznat identitet. Registrirani deskriptori slika lica kojima je poznat identitet nalaze se u galeriji lica. Ilustracija zadnjeg koraka u kojem se slika lica na kojoj je nepoznati identitet uspoređuje s galerijom prikazana je na slici 2.3. Važno je još jednom napomenuti da se ne uspoređuje ulazna slika sa svim slikama lica iz galerije, kako je zbog bolje vizualizacije prikazano na slici, nego se uspoređuje deskriptor ulaznog lica sa svim deskriptorima lica iz galerije.



Slika 2.3: Usporedba slike na kojoj je nepoznati identitet s galerijom

U ovom radu detekcija lica promatrat će se kao crna kutija, a izlučivanje deskriptora će se obavljati dubokim konvolucijskim neuronskim mrežama.

3. Umjetne neuronske mreže

U strojnom učenju, umjetne neuronske mreže skupina su modela inspiriranih biološkim neuronskim mrežama koje se koriste kako bi se procijenila ili aproksimirala funkcija koja ovisi o velikom broju ulaza. Umjetne neuronske mreže općenito su prikazane kao sustavi međusobno povezanih neurona koji izmjenjuju poruke. Vezama između neurona pridružene su težine koje se podešavaju ovisno o ulazima u mrežu. Učenje neuronskih mreža je podešavanje težina između neurona i ponavlja se dok neuronska mreža za većinu ulaznih podataka ne daje željene izlaze.

Kao i druge metode koje uče iz podataka, neuronske mreže koriste se za rješavanje širokog spektra zadataka kao što su računalni vid i raspoznavanje zvuka koji se teško rješavaju ili ih je nekad čak i nemoguće riješiti klasičnim programiranjem koje se temelji na eksplicitnom pisanju pravila koja bi pokrila sve moguće slučajeve u kojima se uzorak koji se želi raspoznati može nalaziti.

Neuronska mreža obrađuje podatke distribuiranim paralelnim radom neurona. Neuroni u neuronskoj mreži obično se slažu slojevito, pa ako mreža ima više slojeva između ulaznog i izlaznog sloja zove se višeslojna ili duboka umjetna neuronska mreža. Slojevi između ulaznog i izlaznog sloja nazivaju se skriveni slojevi. Signal se u klasi neuronskih mreža koja se koristi u ovom radu propagira od ulaznih prema izlaznim slojevima, dakle nema povratnih veza, pa se takve neuronske mreže zovu i unaprijedne neuronske mreže.

3.1. Umjetni neuron

Osnovna jedinica višeslojne unaprijedne umjetne neuronske mreže je umjetni neuron. Općeniti model umjetnog neurona prikazan je na slici 3.1. (prof. dr. sc. Bojana Dalbelo Bašić et al., 2008)

Ulazni signali označeni su s $x_1, x_2, ..., x_n$. Težine su označene s $w_1 w_2, ..., w_n$. Ulazni signali općenito su realni brojevi. Težinska suma net ulaznih signala i težina

je:

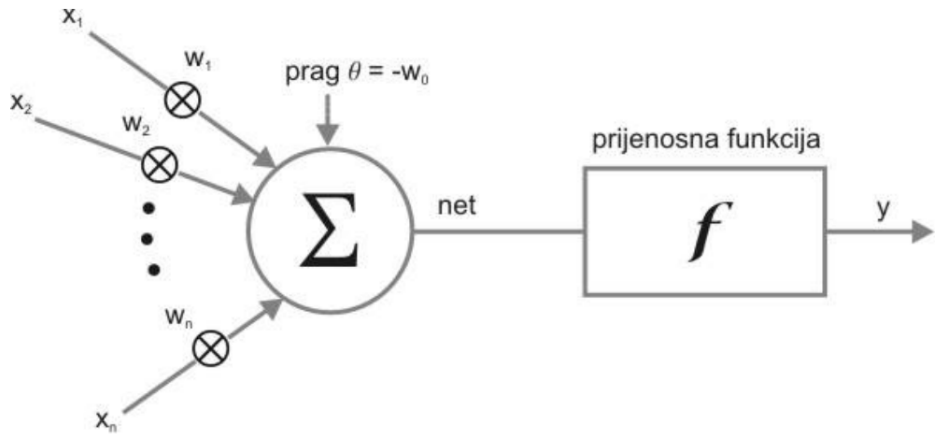
$$net = w_1x_1 + w_2x_2 + ... + w_nx_n - \theta \tag{3.1}$$

ali se često uzima da je vrijednost praga $\theta = -w_0$ i dodaje se ulazni signal x_0 s konstantnom vrijednošću 1, pa se gornji izraz može zapisati kao:

$$net = w_0x_0 + w_1x_1 + w_2x_2 + ... + w_nx_n = \sum_{i=0}^{n} w_ix_i \tag{3.2}$$

Izlaz neurona y rezultat je prijenosne funkcije:

$$y = f(\sum_{i=0}^{n} w_ix_i) = f(net) \tag{3.3}$$



Slika 3.1: Umjetni neuron

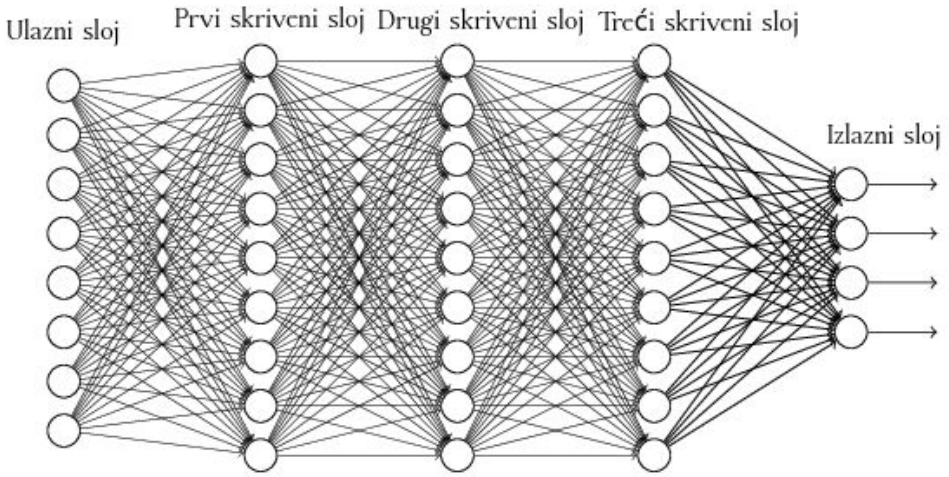
3.2. Višeslojna unaprijedna neuronska mreža

Na slici 3.2 prikazana je višeslojna unaprijedna neuronska mreža s jednim ulaznim slojem, tri skrivena sloja i jednim izlaznim slojem koja se koristi za klasifikacijski problem. Broj neurona na ulazu jednak je dimenzionalnosti uzoraka, a broj neurona na izlazu jednak je broju klasa. Broj skrivenih slojeva i broj neurona u svakome od njih je proizvoljan s time da struktura skrivenih slojeva utječe na ekspresivnost same neuronske mreže te je optimalan odabir parametara strukture mreže važan za postizanje dobrih rezultata klasifikacije.

Zadnji, klasifikacijski sloj najčešće je potpuno povezani sloj neurona kojima je izlaz linearan: $y = \sum_{i=1}^{n} w_ix_i$, a nakon tog sloja slijedi *softmax* funkcija koja djeluje na izlaze neurona zadnjeg sloja. *Softmax* funkcija definirana je na sljedeći način:

$$P(y = j|\boldsymbol{x}) = \frac{e^{\boldsymbol{x}^T \boldsymbol{w}_j}}{\sum_{k=1}^{K} e^{\boldsymbol{x}^T \boldsymbol{w}_k}} \tag{3.4}$$

Svaki izlazni neuron zadnjeg sloja mreže daje vrijednost koja predstavlja pripadnost jednoj od klasa, a *softmax* funkcija daje probabilističku interpretaciju toj pripadnosti. Jednadžba 3.4 daje vjerojatnost pripadnosti ulaznog primjera danog vektorom $\boldsymbol{x}$ klasi j.



Slika 3.2: Višeslojna neuronska mreža za rješavanje problema klasifikacije

Ako se makne klasifikacijski sloj, tada izlaz novonastale neuronske mreže predstavlja deskriptor ulaznog podatka, odnosno sažeti prikaz ulaznog podatka u kojem su pohranjene bitne značajke ulaznog podatka. U tom slučaju neuronska mreža postaje izlučivač značajki.

3.3. Algoritam unazadne propagacije (engl. *Backpropagation algorithm*)

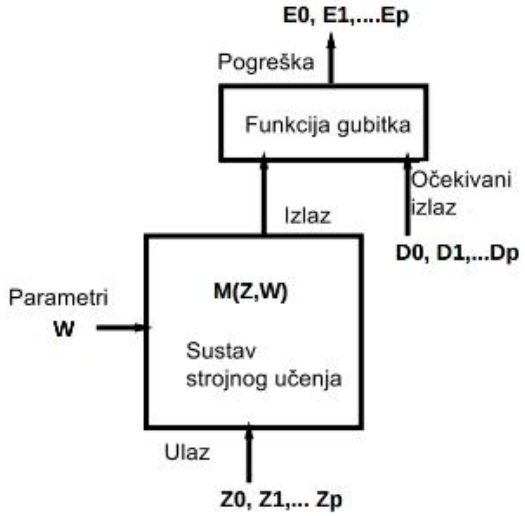
Backpropagation algoritam popularan je algoritam za učenje neuronskih mreža. Konceptualno je jednostavan, računalno efikasan i najčešće radi. Međutim, podešavanje *backpropagation* algoritma da dobro radi, a ponekad i da uopće radi, može se činiti više umjetnošću nego znanošću. Dizajniranje i učenje mreže korištenjem *backpropagation* algoritma zahtijeva odabir velikog broja parametara. Potrebno je odabrati broj i vrstu neurona, broj i vrstu slojeva, stopu učenja, skup podataka za učenje i testiranje,

itd. Ti izbori mogu biti kritični i teški jer ne postoji teoretska podloga koja govori koliko koji od tih parametara mora iznositi, a parametri jako ovise o problemu koji se rješava i podacima.

U nastavku slijedi detaljan opis standardnog *backpropagation* algoritma s pripadajućim matematičkim izvodima. (LeCun et al., 1998)

Sustav koji uči, prikazan na slici 3.3 računa funkciju $M(Z^p, W)$ gdje je Z^p p-ti uzorak, a W predstavlja skup parametara sustava koji se mogu podešavati. Funkcija gubitka $E^p = C(D^p, M(Z^p, W))$ mjeri razliku između D^p, željenog izlaza za uzorak Z^p, i izlaza sustava. Prosječna funkcija gubitka $E_{train}(W)$ je prosjek grešaka E^p skupa parova ulaz/izlaz koji se zovu skup podataka za učenje: $(Z^1, D^1), ..., (Z^P, D^P)$. U najjednostavnijoj formulaciji, problem učenja svodi se na nalaženje vrijednosti vektora W koji minimizira $E_{train}(W)$. U praksi, performanse sustava na skupu podataka za učenje nisu toliko zanimljive. Relevantnija mjera je greška sustava na podacima koji nisu sudjelovali u učenju. Te performanse su procijenjene mjerenjem preciznosti na skupu uzoraka koji su odvojeni od skupa uzoraka za učenje i taj skup se zove skup uzoraka za testiranje. Najčešće korištena funkcija gubitka je srednja kvadratna pogreška:

$$E^p = \frac{1}{2}(D^p - M(Z^p, W))^2, E_{train} = \frac{1}{P}\sum_{p=1} E^p \tag{3.5}$$



Slika 3.3: Sustav strojnog učenja

Backpropagation algoritam počinje sa skupom uzoraka, od kojih se svaki uzorak sastoji od parova ulaz/izlaz funkcije koja se treba naučiti. Parovi ulaz/izlaz funkcije najčešće sadrže šum. Ako se sakupi nekoliko skupova uzoraka iste funkcije tada će svaki skup izgledati malo drugačije zbog šuma i zbog različitih točaka koje su uzor-

kovane. Svaki od tih skupova podataka će rezultirati mrežama s minimumima koje su različite jedna od druge, a isto tako imaju minimume koji su različiti i od minimuma prave funkcije.

Postoji nekoliko tehnika (rano zaustavljanje, regularizacija) kako bi se maksimizirala sposobnost davanja točnih izlaza za ulazne podatke koji nisu sudjelovali u učenju. Ta sposobost neuronske mreže zove se generalizacija. Intuitivno je jasno i da je, osim izbora skupa uzoraka, izbor modela, arhitekture i funkcije gubitka presudan za dobivanje mreže koja dobro generalizira.

Najjednostavniji oblik višeslojnog sustava za učenje koji se uči učenjem zasnovanom na gradijentima je skup modula poslaganih jedan na drugi, od kojih svaki implementira funkciju $X_n = F_n(W_n, X_{n-1})$ gdje je X_n vektor koji predstavlja izlaz modula, W_n je vektor parametara koji se mogu podešavati u modulu (podskup W-a), a X_{n-1} je ulazni vektor modula (isto što i izlazni vektor prethodnog modula). Ulaz X_0 prvog modula je ulazni uzorak Z^p. Ako je poznata parcijalna derivacije E^p po X_n, tada mogu biti izračunate parcijalne derivacije E^p s obzirom na W_n i X_{n-1}:

$$\begin{aligned} \frac{\partial E^p}{\partial W_n} &= \frac{\partial F}{\partial W_n}(W_n, X_{n-1})\frac{\partial E^p}{\partial X_n} \\ \frac{\partial E^p}{\partial X_{n-1}} &= \frac{\partial F}{\partial X}(W_n, X_{n-1})\frac{\partial E^p}{\partial X_n} \end{aligned} \tag{3.6}$$

gdje je $\frac{\partial F}{\partial W_n}(W_n, X_{n-1})$ Jakobijeva matrica od F s obzirom na W evaluirana u točci (W_n, X_{n-1}), a $\frac{\partial F}{\partial X}(W_n, X_{n-1})$ je Jakobijeva matrica od F s obzirom na X. Jakobijeva matrica vektorske funkcije je matrica koja sadrži parcijalne derivacije svih izlaza s obzirom na sve ulaze. Kada se gornje jednadžbe primijene na module u obrnutom redoslijedu, od sloja N do sloja 1, mogu biti izračunate sve parcijalne derivacije funkcije gubitka s obzirom na sve parametre. Takav način računanja gradijenata naziva se *backpropagation*.

$$Y_n = W_n X_{n-1} \tag{3.7}$$

$$X_n = F(Y_n) \tag{3.8}$$

gdje je W_n matrica čiji je broj stupaca jednak dimenziji vektora X_{n-1} i broj redaka je jednak dimenziji vektora X_n. F je vektorska funkcija koja primjenjuje aktivacijsku funkciju na svaku komponentu svog ulaza. Y_n je vektor ponderiranih suma.

Korištenjem pravila ulančavanja za gornje jednadžbe, izvode se klasične jednadžbe za *backpropagation*.

$$\frac{\partial E^p}{\partial y_n^i} = f'(y_n^i)\frac{\partial E^p}{\partial x_n^i} \tag{3.9}$$

$$\frac{\partial E^p}{\partial w_n^{ij}} = x_{n-1}^j \frac{\partial E^p}{\partial y_n^i} \tag{3.10}$$

$$\frac{\partial E^p}{\partial x_{n-1}^k} = \sum_i w_n^{ik} \frac{\partial E^p}{\partial y_n^i} \tag{3.11}$$

Gornje jednadžbe mogu se zapisati i u matričnom obliku:

$$\frac{\partial E^p}{\partial Y_n} = F'(Y_n)\frac{\partial E^p}{\partial X_n} \tag{3.12}$$

$$\frac{\partial E^p}{\partial W_n} = X_{n-1}^j \frac{\partial E^p}{\partial Y_n} \tag{3.13}$$

$$\frac{\partial E^p}{\partial X_{n-1}} = W_n^T \frac{\partial E^p}{\partial Y_n} \tag{3.14}$$

Najjednostavnije učenje (minimizacija) je gradijentni spust gdje se W iterativno podešava na sljedeći način:

$$W(t) = W(t-1) - \eta \frac{\partial E}{\partial W} \tag{3.15}$$

U najjednostavnijem slučaju stopa učenja η je skalarna konstanta. Neke sofisticiranije metode koriste varijabilnu stopu učenja η. U nekim metodama η je oblika dijagonalne matrice ili je procjena inverza Hesseove matrice funkcije gubitka (matrica drugih derivacija).

Backpropagation algoritam može biti jako spor za višeslojne mreže ako je funkcija gubitka nekvadratna, nekonveksna i visokodimenzionalna s mnogo lokalnih minimuma i/ili ravnih regija. Ne postoji formula koja će garantirati da će 1) mreža konvergirati do dobrog rješenja, ili da je 2) konvergencija brza ili čak da uopće postoji.

U svakoj iteraciji, jednadžba 3.15 zahtijeva potpuni prolaz kroz cijeli skup podataka kako bi se izračunao prosječan ili pravi gradijent. To se naziva skupnim (engl. *batch*) učenjem, budući da cijeli skup podataka mora biti viđen prije nego su težine podešene. Alternativno, može se koristiti i stohastičko učenje gdje je jedan primjer $\{Z^t, D^t\}$ odabran (nasumično) iz skupa podataka za učenje u svakoj iteraciji t. Procjena pravog gradijenta je onda računata s obzirom na grešku E^t tog primjera i tada se težine podešavaju na sljedeći način:

$$W(t+1) = W(t) - \eta \frac{\partial E^t}{\partial W} \tag{3.16}$$

Stohastičko učenje je općenito preferirana metoda za osnovni *backpropagation* algoritam jer je puno brža od *batch* učenja i rezultira boljim rješenjima.

Na kraju, objasnit će se još dva bitna pojma vezana koliko za neuronske mreže, toliko i za strojno učenje općenito - prenaučenost i podnaučenost.

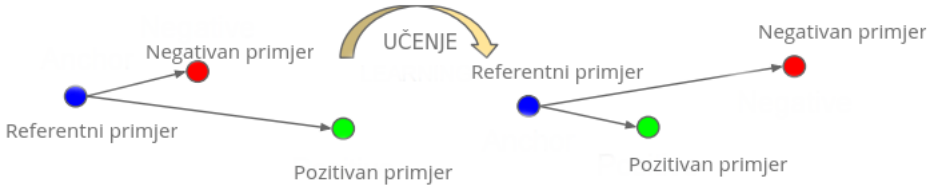
Prenaučenost (engl. *overfitting*) – ako je model previše složen u odnosu na stvarnu funkciju, model će naučiti i šum iz skupa podataka za učenje i tada će jako loše generalizirati na skupu podataka za testiranje jer će previše ovisiti o skupu podataka za učenje.

Podnaučenost (engl. *underfitting*) – ako je model prejednostavan u odnosu na stvarnu funkciju, tada model loše opisuje podatke iz skupa za učenje. Izgledno je da će još lošije performanse biti i na skupu podataka za testiranje.

3.4. Triplet loss

Funkcija gubitka korištena u ovom radu je tzv. *triplet loss*. U ovom potpoglavlju detaljno je opisana i matematički formulirana ta funkcija gubitka. (Schroff et al., 2015)

Deskriptor slike lica matematički se može zapisati kao $f(x) \in R^d$. Deskriptor preslikava sliku x u d-dimenzionalni Euklidski prostor. Cilj je osigurati da je slika x_i^a (referentni primjer (engl. *anchor*)) određene osobe bliža svim slikama x_i^p (pozitivni primjer (engl. *positive*)) iste te osobe, a dalje od svih ostalih slika osoba x_i^n (negativan primjer (engl. *negative*)). To je prikazano na slici 3.4.



Slika 3.4: *Triplet loss*

Matematički se gornja tvrdnja može zapisati kao:

$$||f(x_i^a) - f(x_i^p)||_2^2 + \alpha < ||f(x_i^a) - f(x_i^n)||_2^2, \ \forall (f(x_i^a), f(x_i^p), f(x_i^n)) \in \tau \tag{3.17}$$

gdje je α margina između pozitivnih i negativnih parova. τ je skup kardinaliteta N svih mogućih trojki u skupu za učenje. Gubitak koji je potrebno minimizirati je:

$$L = \sum_{i}^{N} [||f(x_i^a) - f(x_i^p)||_2^2 - ||f(x_i^a) - f(x_i^n)||_2^2 + \alpha] \tag{3.18}$$

Trojke se biraju nasumično. Nasumično se odaberu dva identiteta. Iz prvog identiteta odaberu se dvije slike - referentni primjer i pozitivan primjer, a iz drugog identiteta odabere se jedna slika - negativan primjer. Dakle, svaki identitet u bazi mora imati barem dvije slike. Odabere se n trojki i one se šalju kroz neuronsku mrežu. Na izlazu mreže računa se gubitak i gradijenti se šalju nazad kroz mrežu kako bi se minimizirala greška.

3.5. *AdaDelta*

U potpoglavlju 3.3 opisan je klasični *backpropagation* algoritam koji koristi gradijentni spust za podešavanje težina. U ovom radu nije korišten običan gradijentni spust za optimizaciju nego se koristi sofisticiranija metoda - *AdaDelta*.

Metoda ne zahtijeva ručno podešavanje stope učenja i robusna je na šumoviti gradijent, odabir arhitekture, različite podatke i selekciju hiperparametara (oni parametri koji se podešavaju učenjem).

Cilj mnogih metoda strojnog učenja je ažurirati skup parametara x kako bi se optimizirala ciljna funkcija $f(x)$. To obično uključuje iterativne postupke koji mijenjaju parametre Δx u svakoj iteraciji algoritma. U iteraciji t, najjednostavnije pravilo za ažuriranje parametara je:

$$x_{t+1} = x_t + \Delta x_t \tag{3.19}$$

Taj opći pristup može se primijeniti za ažuriranje bilo kojih parametara koji se mogu derivirati.

$$\Delta x_t = -\eta g_t \tag{3.20}$$

gdje je g_t gradijent parametara u t-toj iteraciji $\frac{\partial f(x_t)}{\partial x_t}$ i η je stopa učenja koja kontrolira iznos pomaka u smjeru negativnog gradijenta.

Algoritam *AdaDelta* poboljšana je verzija klasičnog algoritma gradijentnog spusta koji optimizira ciljnu funkciju slijedeći smjer negativnog gradijenta g_t. Algoritam *AdaDelta* akumulira w prethodnih gradijenata. Budući da je spremanje w prethodnih gradijenata neefikasno, akumulacija se implementira kao eksponencijalno padajući prosjek

kvadriranih gradijenata (engl. *exponentially decaying average of the squared gradients*). Pretpostavka je da je u vremenu t trenutni prosjek $E[g^2]_t$ i tada se računa:

$$E[g^2]_t = \rho E[g^2]_{t-1} + (1-\rho)g_t^2 \tag{3.21}$$

gdje je ρ konstanta i označava stopu propadanja. Budući da se u algoritmu koristi korijen ove vrijednosti kod ažuriranja parametara, gornji izraz, kada se kvadrira, postaje srednja kvadratna greška (engl. *Root Mean Square*) prethodnih kvadriranih gradijenata do vremena t:

$$RMS[g]_t = \sqrt{E[g^2]_t + \epsilon} \tag{3.22}$$

gdje je ϵ konstanta.

Osim toga u izrazu za ažuriranje parametara kod *AdaDelta* metode pojavljuje se i izraz $RMS[\Delta x]_{t-1}$ koji se definira analogno:

$$RMS[\Delta x]_{t-1} = \sqrt{E[\Delta x^2]_{t-1} + \epsilon} \tag{3.23}$$

Izraz za ažuriranje parametara kod *AdaDelta* metode iz kojeg se vidi da je procijenjena η koja se u originalnoj verziji gradijentnog spusta mora ručno namještati aproksimirana varijabilnim izrazom $\frac{RMS[\Delta x]_{t-1}}{RMS[g]_t}$ glasi:

$$\Delta x_t = -\frac{RMS[\Delta x]_{t-1}}{RMS[g]_t} g_t \tag{3.24}$$

gdje je ista konstanta ϵ dodana ***RMS***-u u brojniku i nazivniku. Konstanta je dodana kako bi postupak mogao početi kada je u prvoj iteraciji $\Delta x_0 = 0$ i $g_0 = 0$ i kako bi se osigurao napredak čak i ako prethodna ažuriranja postanu jako mala.

U nastavku je dan algoritam za računanje *AdaDelta* ažuriranja u trenutku t, a detaljan izvod i pojašnjenja dana su u radu (Zeiler, 2012).

Algoritam 1 Računanje ADADELTA ažuriranja u trenutku t

Require: brzina propadanja (engl. *rate decay*) ρ, konstanta ϵ

Require: početni parametar x_1

1: Inicijalizirati akumulacijske varijable $E[g^2]_0 = 0, E[\Delta x^2]_0 = 0$

2: **for** `t=1:T` **do**

3: Izračunati gradijent g_t

4: Akumulirati gradijent: $E[g^2]_t = \rho E[g^2]_{t-1} + (1-\rho)g_t^2$

5: Izračunati ažuriranje: $\Delta x_t = -\frac{RMS[\Delta x]_{t-1}}{RMS[g_t]} g_t$

6: Akumulirati ažuriranja: $E[\Delta x^2]_t = \rho E[\Delta x^2]_{t-1} + (1-\rho)\Delta x_t^2$

7: Ažurirati: $x_{t+1} = x_t + \Delta x_t$

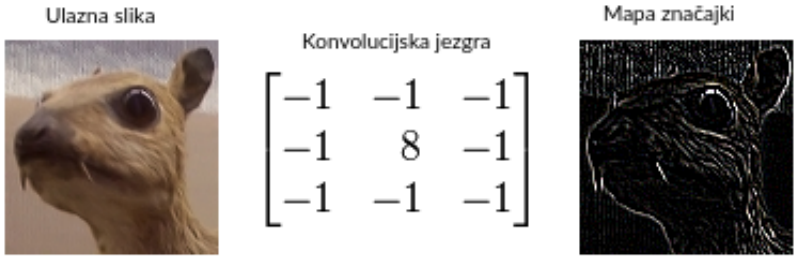
3.6. Konvolucijska neuronska mreža

U do sada opisanim neuronskim mrežama izlaz svakog neurona bio je skalar. U konvolucijskim neuronskim mrežama uvodi se proširenje na $\boldsymbol{R}^2$. Takvi elementi s 2D izlazom nazivaju se mapama značajki (engl. *feature maps*). Ulaz je također dvodimenzionalan, a umjesto težina koriste se jezgre (engl. *kernels*). (Vukotić, 2014)

Na ulazu same konvolucijske neuronske mreže može biti monokromatska slika ili višekanalna slika u boji. Nakon toga slijede naizmjenično konvolucijski slojevi i slojevi sažimanja (engl. *pooling layers*).

3.6.1. Konvolucijski slojevi

Konvolucija je matematički način kombiniranja dvije funkcije kako bi se formirala nova, treća funkcija. Mapa značajki i konvolucijska jezgra kombiniraju se kako bi se dobila transformirana mapa značajki. Konvolucija se obično interpretira kao filter. Konvolucijska jezgra filtrira mapu značajki kako bi se izlučila određena značajka (na primjer, jezgra može izlučiti rubove i odbaciti ostale informacije). Na slici 3.5 prikazana je konvolucija slike s konvolucijskom jezgrom za detektiranje rubova. (Dettmers, 2015)



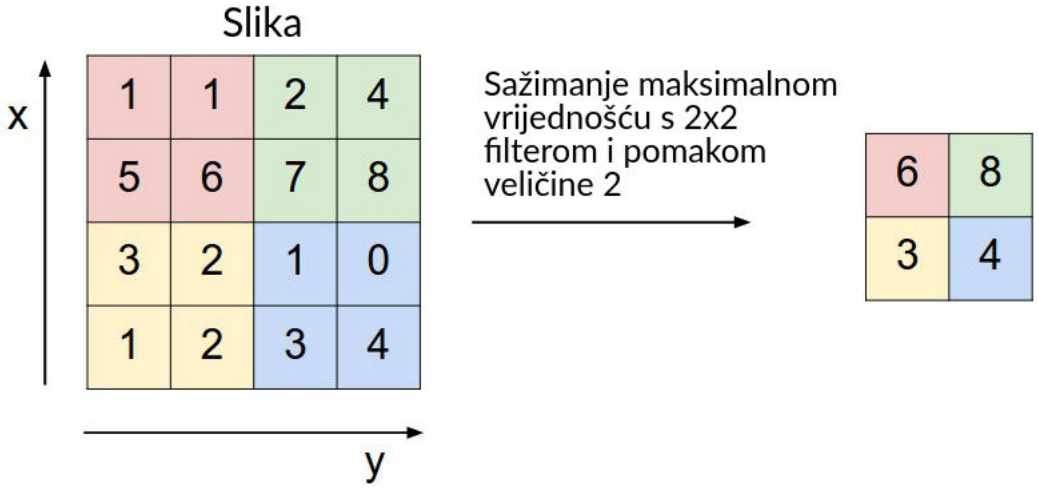
Slika 3.5: Konvolucija

Konvolucijska neuronska mreža koristi konvolucijske slojeve kako bi iz ulaza izlučila korisne informacije. Konvolucijski slojevi imaju parametre koji su učeni tako da izluče najkorisnije informacije za dani zadatak. Na primjer, u općenitom raspoznavanju objekata može biti korisno izlučiti informacije o obliku objekta (različiti objekti obično imaju različite oblike) dok za raspoznavanje ptica može biti korisnije izlučiti informacije o boji ptice (mnoge ptice imaju sličan oblik, ali različitih su boja i u tom slučaju je boja puno korisnija značajka za raspoznavanje različitih kategorija ptica). Konvolucijske neuronske mreže same pronalaze najbolje značajke za određeni zadatak procesom učenja. Obično se koristi više konvolucijskih slojeva kako bi se došlo do što apstraktnijih razina značajki.

3.6.2. Slojevi sažimanja

Osim konvolucijskih slojeva, postoje i slojevi sažimanja. Slojevi sažimanja reduciraju određeni dio ulaza na jednu vrijednost. Na taj način izlazne veze obično primaju slične informacije (informacije su usmjerene na pravo mjesto izlaznoj mapi značajki sljedećeg konvolucijskog sloja). To omogućava invarijantnost na blage rotacije i translacije. Na primjer, ako lice na slici nije u centru nego blago translatirano, postupak će svejedno raditi jer je informacija usmjerena na pravo mjesto operacijom sažimanja pa konvolucijski filteri mogu detektirati lice. Što je veća veličina područja sažimanja, smanjuju se memorijski zahtjevi umjetne neuronske mreže. Ipak, ako je područje sažimanja preveliko, previše informacija se gubi i performanse opadaju. Postoje različiti načini sažimanja, npr. sažimanje maksimalnom vrijednošću (engl. *Max-pooling*) ili sažimanje usrednjavanjem (engl. *Mean-pooling*). U ovom radu korišteno je sažimanje maksimalnom vrijednošću koje je prikazano na slici 3.6. Iz slike se može zaključiti da ako lice blago translatiramo ili rotiramo, dakle ako se svi elementi ulazne mape značajki zajedno blago translatiraju ili rotiraju, i dalje će izlazna mapa značajki sadr-

žavati iste vrijednosti. Isto tako, izlazna mapa značajki je manje dimenzionalnosti, pa se smanjuju memorijski zahtjevi.



Slika 3.6: Sažimanje maksimalnom vrijednošću

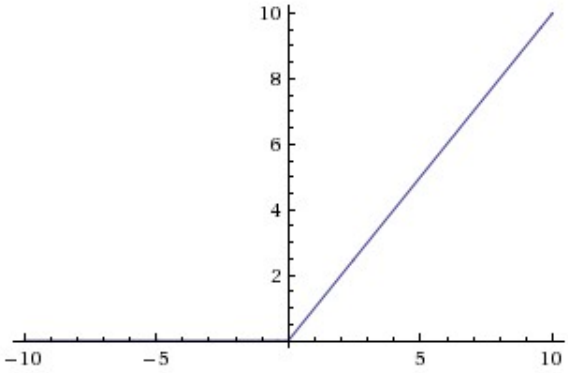
Spomenuto sažimanje usrednjavanjem umjesto maksimalne vrijednosti u svakom području uzima prosjek svih vrijednosti u svakom od tih područja.

3.6.3. *ReLU* aktivacijska funkcija

U konvolucijskim dubokim umjetnim neuronskim mrežama trenutno je najpopularnija *ReLU* (engl. *rectified linear unit*) aktivacijska funkcija. Aktivacijska funkcija unosi nelinearnost u model i samim time model postaje ekspresivniji. *ReLU* se definira na sljedeći način:

$$f(x) = \max(0, x) \tag{3.25}$$

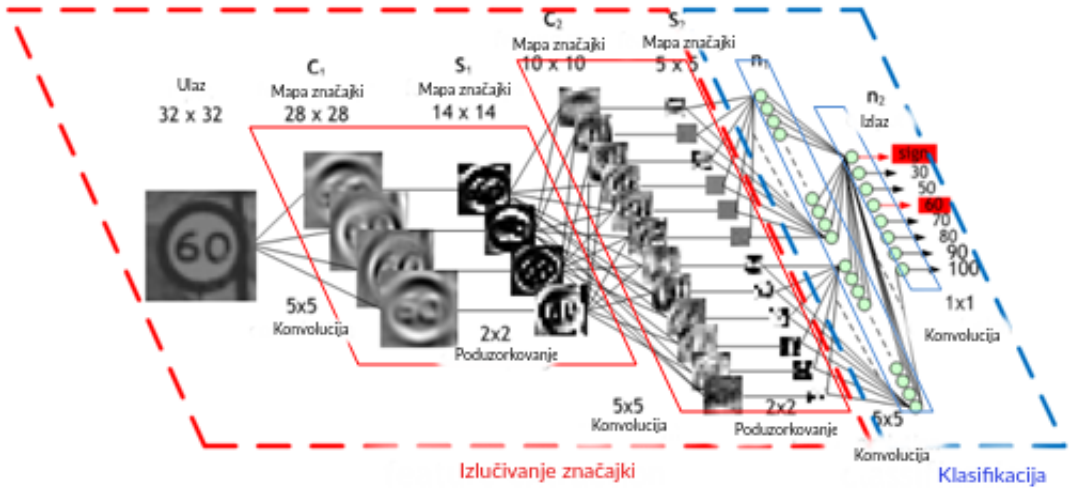
gdje je x ulaz u neuron. Ova aktivacijska funkcija pokazala se vjerodostojnijom u opisivanju procesa koji se događaju u pravoj biološkoj neuronskoj mreži od široko korištenog logističkog sigmoida (koji je inspiriran teorijom vjerojatnosti) ili hiperboličkog tangensa. *ReLU*, u usporedbi sa sigmoidalnom funkcijom ili sličnim aktivacijskim funkcijama, omogućava brže i djelotvornije učenje dubokih neuronskih arhitektura na velikim i složenim bazama podataka. Na slici 3.7 prikazan je graf *ReLU* aktivacijske funkcije.



Slika 3.7: *ReLU* aktivacijska funkcija

3.6.4. Primjer konvolucijske neuronske mreže

Na slici 3.8 prikazan je primjer konvolucijske duboke neuronske mreže. Sastoji se od dva dijela. U prvom dijelu se izlučuju značajke, a u drugom dijelu neuronska mreža radi klasifikaciju na temelju tih značajki. U ovom radu potrebno je samo izlučiti značajke kako bi se dobio deskriptor ulazne slike.



Slika 3.8: Konvolucijska duboka neuronska mreža

3.7. Proširenje algoritma unaprijedne propagacije za konvolucijske neuronske mreže

Backpropagation algoritam (algoritam unazadne propagacije) najčešće je korišten algoritam za učenje neuronskih mreža. U potpoglavlju 3.3 opisan je klasičan algoritam unazadne propagacije koji se vrlo lako može proširiti za rad s konvolucijskim neuronskim mrežama.

Budući da je konvolucijske neuronske mreže teško učiti, korisno je inicijalizirati težine prije početka učenja.

3.7.1. Inicijaliziranje težina

Težine konvolucijskih slojeva inicijalizirane su tako da su pristranosti (engl. *bias*) svih neurona postavljene na nulu, a težina svakog neurona je broj iz normalne razdiobe čija je srednja vrijednost nula, a varijanca je $\sqrt{\frac{2}{n}}$, gdje je n jednak umnošku širine i visine jezgre te izlazne mape značajki trenutnog sloja. (He et al., 2015)

Torch [1] kod za opisanu inicijalizaciju prikazan je u nastavku:

```
local function MSRinit(net)
  local function init(name)
    for k,v in pairs(net:findModules(name)) do
      local n = v.kW*v.kH*v.nOutputPlane
      v.weight:normal(0,math.sqrt(2/n))
      v.bias:zero()
    end
  end
  init'nn.SpatialConvolution'
end
```

3.7.2. Konvolucijski slojevi

Unaprijedna propagacija

Neka je ulaz u konvolucijski sloj dimenzionalnosti $N \times N$. (Gibiansky, 2014) Ako se koristi $m \times m$ filter w, izlaz konvolucijskog sloja bit će veličine $(N-m+1) \times (N-m+1)$. Kako bi se izračunao izlaz, moraju se pozbrajati svi doprinosi (ponderirani s komponentama filtera) prethodnog sloja:

[1] radni okvir (engl. *framework*) koji se koristi za dizajniranje neuronskih mreža, opisan u 4.1.1

$$x_{ij}^{l} = \sum_{a=0}^{m-1} \sum_{b=0}^{m-1} w_{ab} y_{(i+a)(j+b)}^{l-1} \tag{3.26}$$

Nakon toga se još primijeni nelinearnost, dodaje se aktivacijska funkcija, koja djeluje na izlaz:

$$y_{ij}^{l} = \sigma(x_{ij}^{l}) \tag{3.27}$$

Unazadna propagacija

Neka je E funkcija gubitka i neka su poznate vrijednosti $\frac{\partial E}{\partial y_{ij}^{l}}$. Prvo će se odrediti komponente gradijenta za svaku težinu korištenjem pravila ulančavanja.

$$\frac{\partial E}{\partial w_{ab}} = \sum_{i=0}^{N-m} \sum_{j=0}^{N-m} \frac{\partial E}{\partial x_{ij}^{l}} \frac{\partial x_{ij}^{l}}{\partial w_{ab}} = \sum_{i=0}^{N-m} \sum_{j=0}^{N-m} \frac{\partial E}{\partial x_{ij}^{l}} y_{(i+a)(j+b)^{l-1}} \tag{3.28}$$

Moraju se sumirati svi x_{ij}^{l} izrazi u kojima se pojavljuje w_{ab}. Iz jednadžbe 3.26 može se vidjeti da je $\frac{\partial x_{ij}^{l}}{\partial w_{ab}} = y_{(i+a)(j+b)}^{l-1}$.

Kako bi se izračunao gradijent, moraju se znati vrijednosti $\frac{\partial E}{\partial x_{ij}^{l}}$. Te vrijednosti se lagano računaju pravilom ulančavanja:

$$\frac{\partial E}{\partial x_{ij}^{l}} = \frac{\partial E}{\partial y_{ij}^{l}} \frac{\partial y_{ij}^{l}}{\partial x_{ij}^{l}} = \frac{\partial E}{\partial y_{ij}^{l}} \frac{\partial \sigma(x_{ij}^{l})}{\partial x_{ij}^{l}} = \frac{\partial E}{\partial y_{ij}^{l}} \sigma'(x_{ij}^{l}) \tag{3.29}$$

Na kraju je potrebno propagirati greške u prethodni sloj:

$$\frac{\partial E}{\partial y_{ij}^{l-1}} = \sum_{a=0}^{m-1} \sum_{b=0}^{m-1} \frac{\partial E}{\partial x_{(i-a)(j-b)}^{l}} \frac{\partial x_{(i-a)(j-b)}^{l}}{\partial y_{ij}^{l-1}} = \sum_{a=0}^{m-1} \sum_{b=0}^{m-1} \frac{\partial E}{\partial x_{(i-a)(j-b)}^{l}} w_{ab} \tag{3.30}$$

3.7.3. Slojevi maksimalnog sažimanja

Unaprijedna propagacija

Slojevi maksimalnog sažimanja su prilično jednostavni i ne rade nikakvo učenje. Jednostavno uzimaju regiju $k \times k$ i kao izlaz daju jednu vrijednost koja je maksimum te regije. Na primjer, ako je ulazni sloj dimenzija $N \times N$, sloj maksimalnog sažimanja će dati $\frac{N}{k} \times \frac{N}{k}$ dimenzionalni sloj jer je $k \times k$ regija reducirana na jednu vrijednost max funkcijom.

Unazadna propagacija

Slojevi sažimanja kod unazadne propagacije propagiraju grešku manje mape trenutnog sloja u veću mapu prethodnog sloja. U ovom slučaju nema učenja parametara.

Tijekom unaprijedne propagacije maksimalnog sažimanja propagiraju se samo maksimalne vrijednosti prethodne mape značajki, a lokacije pojedinih maksimuma se zapisuju da bi se greška mogla propagirati unazad.

Za primjer sa slike 3.6 neka je greška trenutnog sloja npr.: $\begin{bmatrix} -0.29 & 0.32 \\ -0.4 & 0.6 \end{bmatrix}$. Unazadnom propagacijom dobiva se greška prethodnog sloja koja iznosi: $\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & -0.29 & 0 & 0.32 \\ -0.4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.6 \end{bmatrix}$.

4. Implementacija

U ovom poglavlju, opisani su korišteni programski alati i algoritam koji je implementiran za učenje i testiranje duboke konvolucijske neuronske mreže.

4.1. Korišteni alati

Korišteni su sljedeći programski alati: radni okvir *Torch* baziran na programskom jeziku *Lua* i programski jezik *Python*. U nastavku je ukratko opisan svaki od njih.

4.1.1. *Torch*

Torch je *open source* knjižnica za strojno učenje, radni okvir (engl. *framework*) za znanstveno računanje i skriptni jezik baziran na programskom jeziku *Lua*. (Collobert et al., 2011) Pruža širok spektar algoritama za duboko učenje i koristi jako brz skriptni jezik *LuaJIT*, s *C* implementacijom u pozadini.

Jezgra radnog okvira *Torch* je *torch*. *torch* pruža fleksibilno N-dimenzionalno polje, tzv. tenzor koji je implementiran u klasi *Tensor* i podržava osnovne rutine za indeksiranje, rezanje, transponiranje matrica, za pretvorbu između tipova, promjenu veličine internog spremnika podataka, dijeljenje internog spremnika i kloniranje. Taj objekt je korišten u većini ostalih *Torch* paketa i osnovni je objekt knjižnice. *Tensor* isto tako podržava vrlo lagan rad s matematičkim operacijama kao što su *max*, *min*, *sum*, rad sa statističkim distribucijama kao što su uniformna, normalna i multinomijalna distribucija i podržava *BLAS* (engl. *Basic Linear Algebra Subprograms*) operacije kao što su skalarni produkt, množenje matrice i vektora, množenje dvije matrice, produkt matrice i vektora i matrični produkt. *Torch* podržava objektno orijentirano programiranje i serijalizaciju.

Sljedeći primjer pokazuje korištenje *Torch*-a kroz *REPL* (engl. *Read–eval–print loop*) interpreter.

```
> a = torch.randn(3,4)
```

```
> =a
-0.2381 -0.3401 -1.7844 -0.2615
 0.1411  1.6249  0.1708  0.8299
-1.0434  2.2291  1.0525  0.8465
[torch.DoubleTensor of dimension 3x4]

> a[1][2]
-0.34010116549482

> a:narrow(1,1,2)
-0.2381 -0.3401 -1.7844 -0.2615
 0.1411  1.6249  0.1708  0.8299
[torch.DoubleTensor of dimension 2x4]

> a:index(1, torch.LongTensor{1,2})
-0.2381 -0.3401 -1.7844 -0.2615
 0.1411  1.6249  0.1708  0.8299
[torch.DoubleTensor of dimension 2x4]

> a:min()
-1.7844365427828
```

nn paket koristi se za kreiranje neuronskih mreža. Podijeljen je u modularne objekte koji dijele zajedničko *Module* sučelje. Moduli imaju *forward()* i *backward()* metode koje omogućavaju unaprijednu i unazadnu propagaciju. Moduli mogu biti spojeni korištenjem kompozita, kao što su *Sequential*, *Parallel* i *Concat*. Jednostavniji moduli su *Linear*, *Tanh* i *Max*. Modularno sučelje omogućava automatsko diferenciranje prvog reda.

Funkcije gubitaka implementirane su kao podklase *Criterion*-a, koji ima slično sučelje kao i *Module*. Isto ima *forward()* i *backward()* metode za računanje vrijednosti funkcije gubitka i unazadno propagiranje gradijenata. Poznate funkcije gubitaka su srednja kvadratna pogreška implementirana u *MSECriterion*-u i pogreška unakrsne entropije implementirana u *ClassNLLCriterion*-u.

Postoje paketi koji pružaju širok spektar korisnih alata za paralelizaciju, asinkroni ulaz/izlaz, procesiranje slike itd.

Torch se koristi u *Google DeepMind*-u, *Facebook AI* istraživačkoj grupi, *IBM*-u, *Yandex*-u i *Indiap* istraživačkom institutu. Proširen je za korištenje na *Android* operacijskom sustavu i *iOS*-u. Koristi se i za izradu hardverskih implementacija za protok podataka kakvi postoje u neuronskim mrežama.

4.1.2. *Python*

Python je široko korišten skriptni jezik visoke razine i opće namjene. (Python Software Foundation, 2016) Njegov dizajn naglašava čitljivost koda, a sintaksa omogućava programerima da izraze koncepte u puno manje linija koda nego što bi to bilo moguće u drugim jezicima kao što su *C++* ili *Java*.

Python podržava više programskih paradigmi, uključujući objektno orijentirano i funkcijsko programiranje. Podržava dinamičko tipiziranje [1] i automatsko upravljanje memorijom te ima veliku i opsežnu standardnu knjižnicu.

Python koristi uvlačenje kao metodu razlikovanja programskih blokova, tj. ne koristi vitičaste zagrade ili ključne riječi kao većina programskih jezika. Povećanje uvlačenja znači da dolazi novi, ugniježđeni blok, dok smanjenje označava kraj trenutnog bloka.

U nastavku slijedi primjer *Python* koda:

```
#komentar u Pythonu
#ispise se "Hello World"
print "Hello World"

#primjer liste:
brojevi = [1, 8, 9, 3, 10, 44, 78, 33, 100]

#ispis prvog elementa u listi brojevi, u ovom slucaju "1"
print brojevi[0]

#primjer ispitivanja uvjeta:
a = 5
b = 3
if a > b:
    print "a je vece od b"
else:
    print "b je vece od a"

#primjer iteriranja kroz listu i ispis njenih elemenata:
for broj in brojevi:
    print broj
```

[1] provjera tipa podataka obavlja se tijekom izvedbe programa

```
#primjer funkcije koja ispisuje "Hello World"
def Hello():
    print "Hello World"
```

4.2. Algoritam

U ovom radu korištena je duboka konvolucijska neuronska mreža koja je općenito opisana u 3.6. Slike koje ulaze u mrežu pretprocesirane su detektorom lica tvrtke *Visage Technologies*. (Visage Technologies, 2016) Napravljeno je nekoliko eksperimenata s različitim arhitekturama mreže, a način evaluacije i rezultati prikazani su u sljedećim poglavljima.

4.2.1. Pretprocesiranje

Pretprocesiranje je obavljeno pomoću detektora lica tvrtke *Visage Technologies*. *Visage Technologies* detektor lica napisan je u programskom jeziku *C++*, a *Python* je korišten za povezivanje programskih komponenti (engl. *glue*), za manipulaciju datoteka i za pretprocesiranje datoteka.

Pretprocesiranjem se dobiva slika jednog lica detektiranog detektorom. Lice je izrezano iz originalne slike i veličina mu je promijenjena na 96×96 piksela. Primjeri pretprocesiranih slika prikazani su ispod na slici 4.1.

Slika 4.1: Primjeri pretprocesiranih slika

4.2.2. Učenje

Pretprocesirane slike podijeljene su po identitetima. Svaki identitet ima svoju mapu u koju su spremljene slike tog identiteta. U svakoj mapi su barem dvije slike kako bi bilo koji identitet koji je izabran kao pozitivan identitet mogao imati svoj referentni primjer i pozitivan primjer.

Skup podataka podijeljen je u skup podataka za učenje, skup podataka za validaciju i skup podataka za testiranje. Učenje se obavlja na skupu podataka za učenje *nEpoch* epoha. Svaka epoha traje *nIter* iteracija. Nakon svake epohe, obavlja se validacija na skupu podataka za validaciju, a na kraju čitavog učenja obavi se konačno testiranje na skupu podataka za testiranje.

Neuronska mreža implementirana je u *Torch* radnom okviru. Dio za učenje sastoji se od nekoliko *Lua* skripti. Prva je *main.lua* u kojoj su definirani parametri - putanje do direktorija u kojima se nalaze skupovi podataka za učenje, validaciju i testiranje, broj epoha *nEpoch*, broj iteracija *nIter*, veličina *mini batch*-a (broj trojki (*anchor*, *positive* i *negative*) koji se u istom trenutku šalju u mrežu). Skripta *load.lua* učitava *mini batch*-eve u radnu memoriju, *train.lua* obavlja učenje, a *test_*.lua* skriptama obavlja se testiranje na validacijskim bazama i bazama za testiranje.

U svakoj se iteraciji u mrežu šalje *mini_batch_size* $\times$ 3 slika. Nakon toga, računa se *triplet loss* koji je opisan u 3.4, računaju se gradijenti i šalju se nazad kroz mrežu. Optimizacija se obavlja *AdaDelta* algoritmom opisanom u 3.5. Cilj je postići da deskriptori pozitivnih primjera, onih primjera koji pripadaju istom identitetu budu bliži jedni drugima od deskriptora negativnih primjera, onih koji ne pripadaju istom identitetu.

Izlaz iz mreže je 128-dimenzionalni vektor koji je deskriptor ulaznog lica. Nakon što je mreža naučena, kada se pronađe optimalan model na validacijskom skupu podataka, mreža se testira na skupu podataka za testiranje.

4.2.3. Arhitektura konvolucijske neuronske mreže

U nastavku, u donjoj tablici, prikazana je najbolja isprobana arhitektura neuronske mreže.

n	Sloj	Broj parametara
1	SpatialConvolution(3, 16, 3, 3, 1, 1, 1, 1)	448
2	SpatialBatchNormalization(16, 1e-3)	32
3	ReLU	0

4	SpatialConvolution(16, 16, 3, 3, 1, 1, 1, 1)	2320
5	SpatialBatchNormalization(16, 1e-3)	32
6	ReLU	0
7	MaxPooling(2, 2, 2, 2)	0
8	SpatialConvolution(16, 32, 3, 3, 1, 1, 1, 1)	4640
9	SpatialBatchNormalization(32, 1e-3)	64
10	ReLU	0
11	SpatialConvolution(32, 32, 3, 3, 1, 1, 1, 1)	9248
12	SpatialBatchNormalization(32, 1e-3)	64
13	ReLU	0
14	SpatialConvolution(32, 32, 3, 3, 1, 1, 1, 1)	9248
15	SpatialBatchNormalization(32, 1e-3)	64
16	ReLU	0
17	SpatialMaxPooling(2, 2, 2, 2)	0
18	SpatialConvolution(32, 64, 3, 3, 1, 1, 1, 1)	18496
19	SpatialBatchNormalization(64, 1e-3)	128
20	ReLU	0
21	SpatialConvolution(64, 64, 3, 3, 1, 1, 1, 1)	36928
22	SpatialBatchNormalization(64, 1e-3)	128
23	ReLU	0
24	SpatialConvolution(64, 64, 3, 3, 1, 1, 1, 1)	36928
25	SpatialBatchNormalization(64, 1e-3)	128
26	ReLU	0
27	SpatialConvolution(64, 64, 3, 3, 1, 1, 1, 1)	36928
28	SpatialBatchNormalization(64, 1e-3)	128
29	ReLU	0
30	SpatialMaxPooling(2, 2, 2, 2)	0
31	SpatialConvolution(64, 128, 3, 3, 1, 1, 1, 1)	73856
32	SpatialBatchNormalization(128, 1e-3)	256
33	ReLU	0
34	SpatialConvolution(128, 128, 3, 3, 1, 1, 1, 1)	147584
35	SpatialBatchNormalization(128, 1e-3)	256
36	ReLU	0
37	SpatialConvolution(128, 128, 3, 3, 1, 1, 1, 1)	147584

38	SpatialBatchNormalization(128, 1e-3)	256
39	ReLU	0
40	SpatialConvolution(128, 128, 3, 3, 1, 1, 1, 1)	147584
41	SpatialBatchNormalization(128, 1e-3)	256
42	ReLU	0
43	SpatialMaxPooling(2, 2, 2, 2)	0
44	SpatialConvolution(128, 256, 3, 3, 1, 1, 1, 1)	295168
45	SpatialBatchNormalization(256, 1e-3)	512
46	ReLU	0
47	SpatialConvolution(256, 256, 3, 3, 1, 1, 1, 1)	590080
48	SpatialBatchNormalization(256, 1e-3)	512
49	ReLU	0
50	SpatialConvolution(256, 256, 3, 3, 1, 1, 1, 1)	590080
51	SpatialBatchNormalization(256, 1e-3)	512
52	ReLU	0
53	SpatialConvolution(256, 256, 3, 3, 1, 1, 1, 1)	590080
54	SpatialBatchNormalization(256, 1e-3)	512
55	ReLU	0
56	SpatialConvolution(256, 256, 3, 3, 1, 1, 1, 1)	590080
57	SpatialBatchNormalization(256, 1e-3)	512
58	ReLU	0
59	SpatialMaxPooling(2, 2, 2, 2)	0
60	View(2304)	0
61	Linear(2304, 128)	295040
62	Normalize(2)	0

Ukupan broj parametara prikazane arhitekture je 3626672.

Slijedi opis slojeva navedenih u gornjoj tablici.

SpatialConvolution(nInputPlane, nOutputPlane, kW, kH, [dW], [dH], [padW], [padH]) konvolucijski je sloj opisan u 3.6.1.

U nastavku su objašnjeni svi parametri jednog konvolucijskog sloja koji je implementiran u *Torch*-u.

nInputPlane - broj očekivanih ulaznih ravnina (kanala ili mapi značajki)

nOutputPlane - broj izlaznih ravnina (kanala ili mapi značajki)

kW - širina jezgre

kH - visina jezgre

dW - korak konvolucije u širinu, zadana vrijednost je 1

dH - korak konvolucije u visinu, zadana vrijednost je 1

padW - broj dodatnih nula dodanih ulaznoj ravnini po širini, (*kW*-1)/2 je dobar odabir jer se tada izlaznoj ravnini ne smanjuje dimenzionalnost koji nastaje operacijom konvolucije, zadana vrijednost je 0

padH - broj dodatnih nula dodanih ulaznoj ravnini po visini, (*kH*-1)/2 je dobar odabir jer se tada izlaznoj ravnini ne smanjuje dimenzionalnost koji nastaje operacijom konvolucije, zadana vrijednost je *padW*

SpatialBatchNormalization(N [,eps] [, momentum] [,affine]) implementira normalizaciju *mini batch*-a. (Ioffe i Szegedy, 2015) Neka je $\boldsymbol{x}$ d-dimenzionalan vektor $\boldsymbol{x} = (x^{(1)}, x^{(2)}, ..., x^{(d)})$. Implementirana je sljedeća operacija koja normalizira svaku dimenziju vektora $\boldsymbol{x}$, $y^{(k)} = \frac{(x^{(k)} - \mathbb{E}(x^{(k)}))}{\sqrt{Var(x^{(k)})}} * \gamma + \epsilon$, gdje su očekivanje i varijanca izračunati na cijelom skupu podataka za učenje, a γ i ϵ su parametri čije se vrijednosti određuju učenjem.

Parametri znače sljedeće:

N - broj ulaznih mapi značajki

eps - malena vrijednost dodana standardnoj devijaciji kako bi se izbjeglo dijeljenje s nulom, zadana vrijednost je $1e^{-5}$

affine - boolean varijabla, kada je postavljena na *false* onemogućena je afina transformacija koja se može učiti, zadana vrijednost je *true*

ReLU() - primjenjuje *ReLU* funkciju na svaki element ulaznog tenzora.

SpatialMaxPooling(kW, kH [, dW, dH, padW, padH]) - primjenjuje 2D operaciju sažimanja maksimalnom vrijednošću na $kW \times kH$ regije s korakom $dW \times dH$. Broj izlaznih značajki jednak je broju ulaznih ravnina.

Ako je ulazna slika 3D tenzor veličine $nInputPlane \times height \times width$, izlazna slika će biti veličine $nOutputPlane \times oheight \times owidth$ gdje je

$$\begin{aligned} owidth &= op((width + 2 * padW - kW)/dW + 1) \\ oheight &= op((height + 2 * padH - kH)/dH + 1) \end{aligned} \tag{4.1}$$

op je operator zaokruživanja. Zadani op je $floor$ operacija. Može biti promijenjen pozivima : $ceil()$ ili : $floor()$ metoda.

View(sizes) - mijenja oblik tenzora u oblik koji je definiran parametrom $sizes$.

Linear(inputDimension, outputDimension, [bias = true]) - obavlja linearnu transformaciju na ulaznim podacima $y = Ax + b$. Ulazni vektor mora biti ili vektor (1D tenzor) ili matrica (2D tenzor). Ako je ulaz matrica, tada je svaki redak jedan primjer

iz skupa podataka. Sloj se može koristiti i bez pristranosti (engl. *bias*) ako se postavi $bias = false$.

Normalize(p, [eps]) - normalizira ulazni tenzor (koristi se jedinična L^p norma). Parametar eps spriječava dijeljenje s nulom kada su svi elementi ulaza nule (zadana vrijednost je $eps = 1e^{-10}$).

4.2.4. Odabir optimalnog modela

Zbog izuzetne vremenske zahtjevnosti učenja modela jedne arhitekture (po nekoliko dana), isprobano je svega nekoliko arhitektura. Ne postoje teoretske osnove koje bi kazale koja arhitektura je najbolja ili koja je arhitektura bolja od druge, ali ipak postoje neke smjernice koje kažu da je do neke razine korisno povećavati dubinu mreže. Dobro je, do neke mjere, koristiti što više konvolucijskih slojeva za izlučivanje što apstraktnijih značajki. Veličina jezgre ne smije biti ni prevelika ni premala. Veličina regije koja se obrađuje u sloju sažimanja maksimalnom vrijednošću ne smije biti prevelika da se ne izgubi previše informacija, ali ni premala jer onda nema nikakvog efekta. Zadovoljavajuće vrijednosti određuju se eksperimentalno.

Unutar jedne arhitekture, odabir optimalnih parametara mreže postiže se učenjem, a učenje se u eksperimentima najčešće zaustavlja u slučajevima kada greška prestane konvergirati i na skupu podataka za učenje i na skupu podataka za validaciju što znači da model više ne može apsorbirati nove informacije ili se zaustavlja kada se greška na skupu podataka za učenje smanjuje, a greška na skupu podataka za validaciju počinje rasti što znači da je model previše složen i da se prenaučio.

5. Baze podataka i evaluacija

U ovom poglavlju opisane su baze podataka korištene u radu. Prikazan je i način evaluacije modela naučenih na podacima iz opisanih baza.

5.1. Baze podataka

Velike baze slika lica bitne su za razvoj automatskog raspoznavanja lica, ali ih je jako teško napraviti. Potrebno je uložiti dosta vremena i truda za prikupljanje i pročišćavanje podataka za takvu bazu.

5.1.1. *FaceScrub*

FaceScrub je baza podataka koja se sastoji od 106 863[1] slika lica 530 muških i ženskih poznatih osoba, s oko 200 slika po osobi. (Ng i Winkler., 2014)

Dostupni su samo linkovi na slike s anotacijama, a budući da ima nevažećih linkova, *FaceScrub* baza podataka korištena u ovom radu sadrži manje slika nego što piše na službenoj web stranici baze. Na nekim slikama nalazi se više osoba, ali je za svaku sliku anotiran jedan *bounding box* s pripadajućim identitetom.

Tablica 5.1: *FaceScrub*

	Muške osobe	Ženske osobe	Ukupno
Broj osoba	265	265	530
Broj slika	55306	51557	106863

[1]Originalno izdanje *FaceScrub* baze podataka sadržavalo je nešto više slika (107818) ali je uključivalo duplikate, nekoliko krivo označenih lica kao i nekoliko izobličenih lica. One su izbačene iz baze 9. 3. 2016.

5.1.2. *YouTube Faces*

YouTube Faces je baza video isječaka na kojima su u prvom planu lica osoba. (Wolf et al., 2011) Baza sadrži 3425 video isječaka 1595 različitih osoba. Svi video isječci preuzeti su s *YouTube*-a. Prosječno je dostupno 2.15 video isječaka za svaki identitet. Najkraći isječak traje 48 sličica, najduži 6070 sličica, a prosječna duljina je 181.3 sličica.

Tablica 5.2: *YouTube Faces*

broj video isječaka	1	2	3	4	5	6
broj identiteta	591	471	307	167	51	8

5.1.3. *CASIA-WebFace*

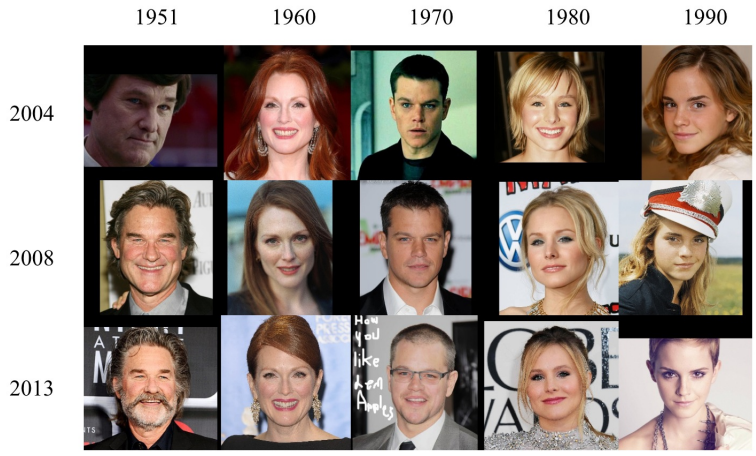
CASIA-WebFace baza podataka sadrži 10575 različitih identiteta na 494414 slika. (Yi et al., 2014) Na svakoj slici nalazi se samo jedna osoba.

5.1.4. *VGG-Face Dataset*

VGG-Face Dataset sadrži slike 2622 različitih identiteta. (Parkhi et al., 2015) Svakom identitetu pridružena je *.txt* datoteka koja sadrži linkove na slike i pripadajuće anotacije *bounding box*-ova. Postoje nevažeći linkovi, tako da *VGG-Face Dataset* baza podataka korištena u ovom radu sadrži manje slika od ukupnog broja linkova. Neke slike prikazuju više osoba, ali svaka slika ima anotiran jedan *bounding box* s pripadajućim identitetom.

5.1.5. *Cross-Age Celebrity Dataset* (*CACD*)

Cross-Age Celebrity Dataset (*CACD*) sadrži 163446 slika 2000 poznatih osoba skinutih s Interneta. (Chen et al., 2014) (Chen et al., 2015) Za ovu bazu postoje i anotacije procjena godina u kojima su slike nastale.



Slika 5.1: Primjeri slika iz *CACD* baze slika nastalih u različitim periodima života poznatih osoba

5.1.6. *Labeled Faces in the Wild* (*LFW*)

Baza podataka sadrži više od 13000 slika lica skinutih s Interneta. (Huang et al., 2007) Svako lice označeno je imenom osobe koja je na slici. 1680 osoba ima dvije ili više različitih slika.

Tablica 5.3: *Labeled Faces in the Wild* (*LFW*)

broj slika	13233
broj osoba	5749
broj osoba s dvije ili više različitih slika	1680

Slika 5.2: Primjeri slika iz *LFW* baze slika

5.1.7. *Face Recognition Grand Challenge Database v2.0* (*FRGC*)

Face Recognition Grand Challenge Database v2.0 je baza lica koja sadrži 44278 slika lica 568 različitih osoba. (Phillips et al., 2005) Prvotni cilj izrade ove baze bio je unaprijeđenje tehnologije za raspoznavanje lica za potrebe Vlade Sjedinjenih Američkih Država.

5.2. Evaluacija

Model je učen na *FaceScrub* bazi, *YouTube Faces* bazi, *CASIA-WebFace* bazi i *VGG-Face Dataset* bazi podataka. Ukupno je učen na 2465175 slika, na 14884 različitih identiteta. Ako je isti identitet prisutan u više baza, ti identiteti se uzimaju kao različiti identiteti jer je zbog velikog broja identiteta mala vjerojatnost da će se za pozitivan, odnosno referentni primjer i negativan primjer uzeti dva ista identiteta iz dvije različite baze.

Validacijska baza je *CACD*, a baze za testiranja su *LFW* i *FRGC*. Prethodno su definirani parovi u *CACD* i *LFW* bazama na kojima se vrši testiranje. Za *FRGC* bazu podataka nasumično su odabrani parovi za testiranje. U uvodu je rečeno da se raspoznavanje svodi na *k-NN* klasifikacijski problem. Za općeniti k, svaki identitet u galeriji bi trebao imati nekoliko slika i tada se gleda k prvih slika iz galerije koje s ulaznom slikom imaju najmanju Euklidsku udaljenost. Od k slika iz galerije koje su najbliže ulaznoj slici, identitet osobe s ulazne slike se odredi tako da se razmotri koji identitet se pojavljuje najviše puta u tih k identiteta. U ovom slučaju, kada postoje definirani parovi, k je jednak 1 pa se gleda podudaranje jedan-na-jedan.

Definirani parovi su ili pozitivni ili negativni. Pozitivan par sadrži dvije slike iste osobe, a negativan par dvije slike različitih osoba. Računa se Euklidska udaljenost deskriptora slika iz parova i potrebno je odrediti diskriminacijsku granicu koja će što bolje razdvajati pozitivne i negativne parove. Problem se svodi na binarnu klasifikaciju, tj. na raspoređivanje primjera u dvije klase - pozitivnu koja se sastoji od pozitivnih parova ili negativnu koja se sastoji od negativnih parova.

Na donjoj slici prikazana je tzv. matrica zabune koja prikazuje sve moguće kombinacije odnosa stvarne klase primjera i predviđene klase primjera koju daje klasifikator. Pozitivna klasa primjera označena je s *1*, a negativna klasa s *0*.

Slika 5.3: Matrica zabune

Ako je predviđena klasa primjera *1* i stvarna klasa primjera je *1*, tada se kaže da je taj par *true positive*.

Ako je predviđena klasa primjera *1*, a stvarna klasa primjera je *0*, tada se kaže da je taj par *false positive*.

Ako je predviđena klasa primjera *0*, a stvarna klasa primjera je *1*, tada se kaže da je taj par *false negative*.

Ako je predviđena klasa primjera *0* i stvarna klasa primjera je *0*, tada se kaže da je taj par *true negative*.

Iz matrice zabune izvode se sljedeće mjere koje su općenito korištene u strojnom učenju za evaluaciju algoritama:

Preciznost (engl. *precision*) = $\frac{TP}{TP+FP}$

Odziv (engl. *recall*) = $\frac{TP}{TP+FN}$

Točnost (engl. *accuracy*) = $\frac{TP+TN}{TP+TN+FP+FN}$

F1 mjera (engl. *F1 score*) = $\frac{2 \cdot TP}{2 \cdot TP+FP+FN}$

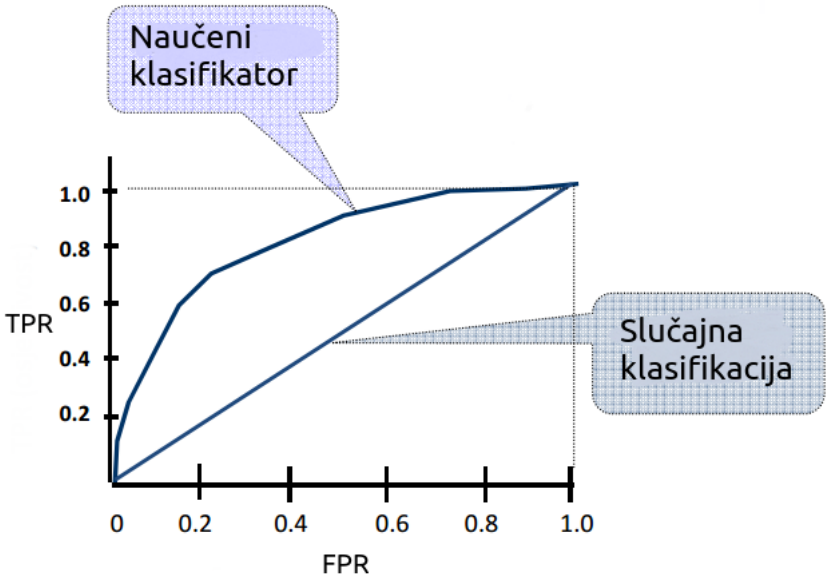
Za određivanje najbolje moguće diskriminacijske granice i za određivanje koliko je ta najbolja granica zapravo dobra koristi se *ROC* (engl. *receiver operating characteristic*) krivulja. To je krivulja koja prikazuje performanse binarnog klasifikatora ako se mijenja diskriminacijska granica. Krivulja je nastala iscrtavanjem omjera stope stvarno točnih primjera (engl. *true positive rate (TPR)*) prema stopi krivo pozitivnih primjera (engl. *false positive rate (FPR)*) korištenjem različitih granica.

TPR - broj točnih klasifikacija u (pozitivnoj) klasi u odnosu na ukupan broj pozitivnih primjera:

$$TPR = \frac{TP}{TP + FN} \tag{5.1}$$

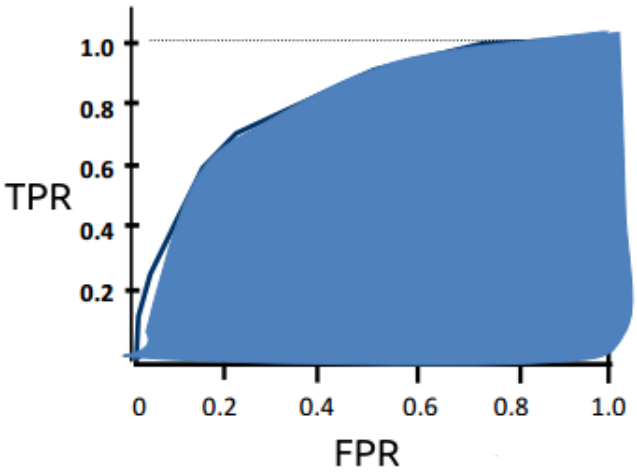
FPR - broj krivih klasifikacija u (pozitivnoj) klasi u odnosu na ukupan broj negativnih primjera:

$$FPR = \frac{FP}{FP + TN} \tag{5.2}$$



Slika 5.4: *ROC* krivulja

Površina ispod *ROC* krivulje - *AUC* (engl. *area under the curve*) je mjera koja govori o kvaliteti klasifikatora. Ako je *AUC* 0.5 klasifikator pogađa slučajno, a ako je maksimalna: 1, znači da se radi o savršenom klasifikatoru.



Slika 5.5: Površina ispod *ROC* krivulje - *AUC*

6. Eksperimenti

Neuronske mreže učene su na računalu s grafičkom karticom NVIDIA GeForce GTX 970 12GB, procesorom Intel(R) Core(TM) i7-2600 CPU @ 3.40GHz s 8 jezgri. Računalo je imalo 16 GB RAM-a.

U tablici 6.1 prikazani su parametri algoritma za učenje.

Tablica 6.1: Parametri algoritma za učenje

parametar	Vrijednost
broj iteracija (*nIter*)	100
broj epoha (*nEpoch*)	100000
veličina *mini batch*-a (*mini_batch_size*)	250

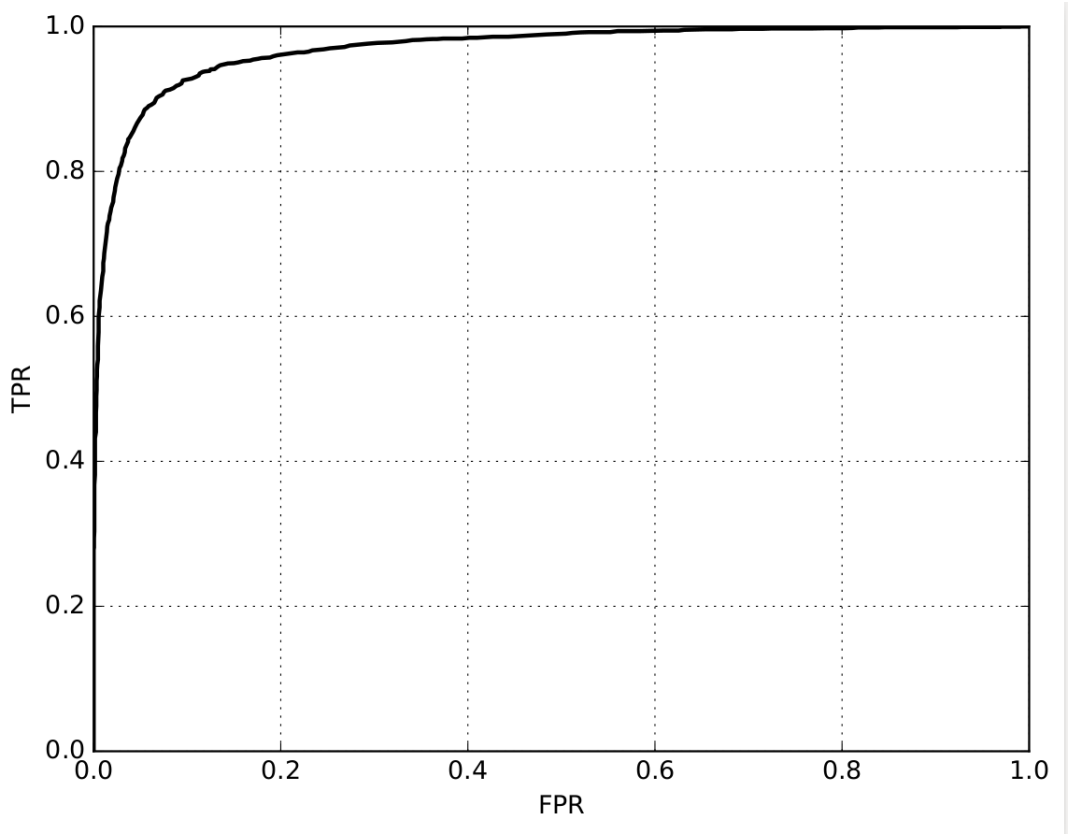
Slike koje ulaze u mrežu imaju tri kanala, R (crveni kanal), G (zeleni kanal) i B (plavi kanal). Veličina slike je 96×96 piksela.

Učenje s brzom kovergencijom (porast *AUC*-a od 0.5 do preko 0.9) traje oko 48 sati, s time da se nakon tog vremena model nije prenaučio. Greška na skupu podataka za učenje, na skupu podataka za validaciju i skupu podataka za testiranje i dalje pada i nakon 48 sati, ali je konvergencija jako spora i gotovo beznačajna pa je učenje prekinuto zbog vremenskih ograničenja.

U nastavku su prikazani rezultati na skupovima podataka za testiranje, skupu podataka za validaciju i skupu podataka za učenje. Za skupove podataka za testiranje i skup podataka za validaciju prikazane su *ROC* krivulje i vrijednosti *AUC*-a, točnost uz diskriminacijsku granicu koja maksimizira točnost te su prikazani preciznost, odziv i F1 mjera uz fiksiranu granicu koja maksimizira točnost. Dane su i matrice zabune. Za skup podataka za učenje prikazana je samo pogreška na tom skupu.

Skup podataka za testiranje

Na slici 6.1 prikazana je dobivena *ROC* krivulja za najbolji model na skupu podataka za testiranje (*LFW*). Definirano je 6000 parova, 3000 pozitivnih parova i 3000 negativnih parova i na temelju njih je određena *ROC* krivulja. Neka od lica na slikama nije moguće detektirati detektorom lica koji se u ovom radu koristi kao crna kutija, pa je na kraju od 6000 parova ostalo njih 5647, od čega 2824 pozitivnih parova i 2823 negativnih parova.



Slika 6.1: *ROC* krivulja za najbolji primjer na skupu podataka za testiranje (*LFW*)

AUC iznosi 0.969494874157. Točnost tog modela za diskriminacijsku granicu koja maksimizira točnost (0.94 je iznos diskriminacijske granice) je 0.917301221888. U tom slučaju preciznost iznosi 0.923463887891, odziv je 0.910056657224, a F1 mjera iznosi 0.91671125379.

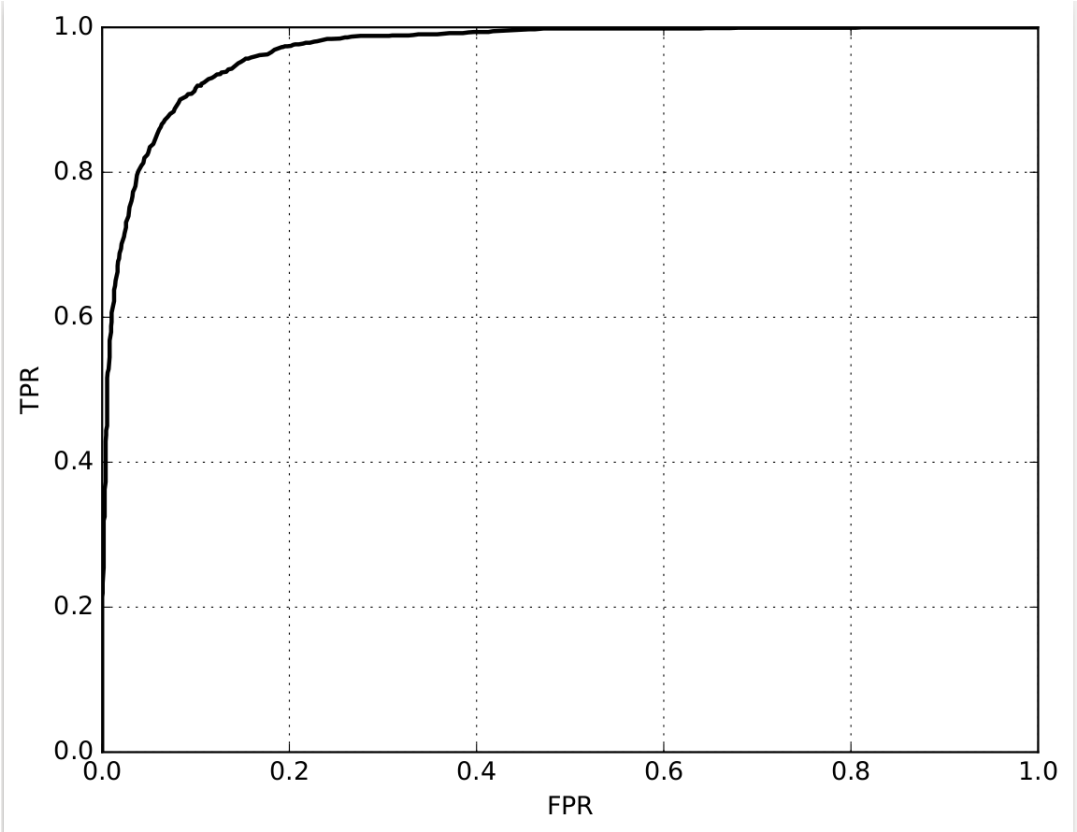
Prikazana je matrica zabune za gore navedene rezultate. *1* označava pozitivnu klasu primjera, a *0* negativnu klasu primjera.

Tablica 6.2: Matrica zabune za skup podataka za testiranje (*LFW*)

		Stvarna klasa		
		1	*0*	Ukupno
Predviđena klasa	*1*	2570	213	2783
	0	254	2610	2864
	Ukupno	2824	2823	5647

Skup podataka za validaciju

Na slici 6.2 prikazana je *ROC* krivulja za isti gornji najbolji model, ali na skupu podataka za validaciju (*CACD*). Definirano je 4000 parova, 2000 pozitivnih parova i 2000 negativnih parova. Isto kao i kod skupa podataka za testiranje, neka od lica na slikama nije moguće detektirati detektorom lica, pa je na kraju od 4000 parova ostalo njih 3762, od čega 1877 pozitivnih parova i 1885 negativnih parova.



Slika 6.2: *ROC* krivulja za najbolji primjer na skupu podataka za validaciju (*CACD*)

AUC iznosi 0.969898350689. Točnost modela na validacijskoj bazi za diskriminacijsku granicu koja maksimizira točnost (0.9516 je iznos diskriminacijske granice)

je 0.90855927698, preciznost je 0.896533885153, odziv 0.923281832712, a F1 mjera iznosi 0.909711286089.

Kao i za skup podataka za testiranje, prikazana je matrica zabune za gore navedene rezultate.

Tablica 6.3: Matrica zabune za skup podataka za validaciju (*CACD*)

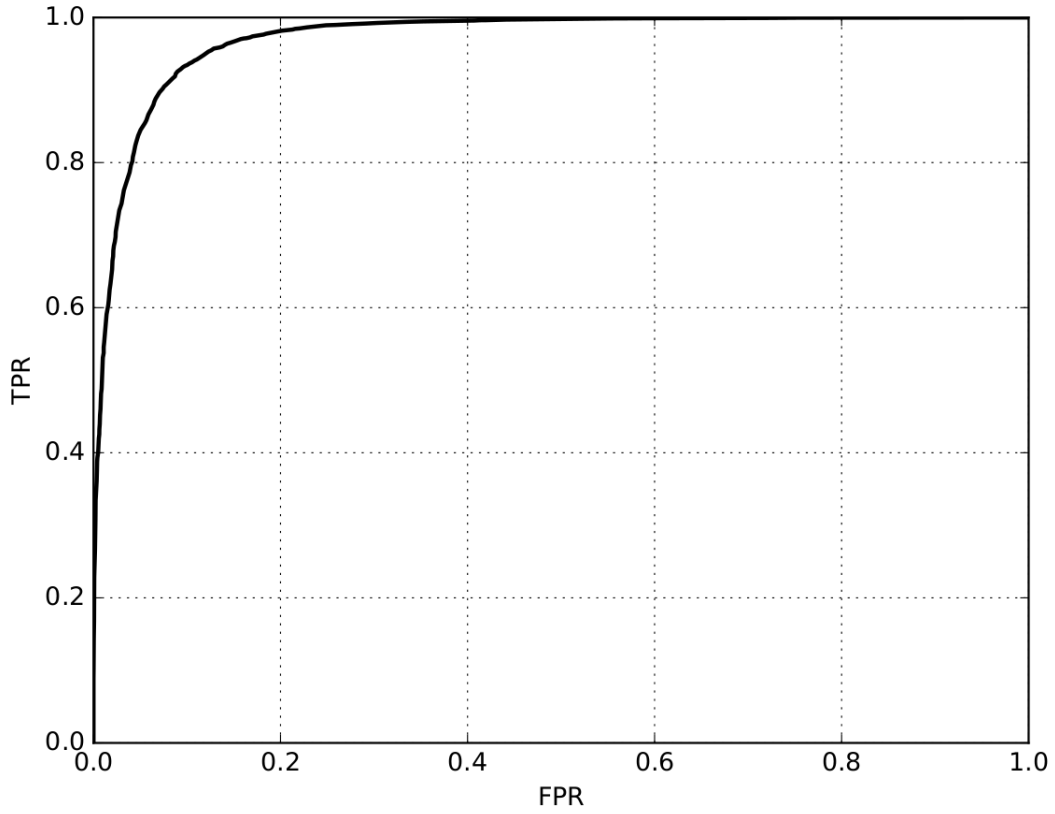
		Stvarna klasa		
		1	*0*	Ukupno
Predviđena klasa	*1*	1733	200	1933
	0	144	1685	1829
	Ukupno	1877	1885	3762

Skup podataka za učenje

Pogreška (*triplet loss*) koja se minimizira učenjem na skupu podataka za učenje (*FaceScrub*, *YouTube Faces*, *CASIA-WebFace*, *VGG-Face Dataset*) za najbolji model iznosi ≈ 0.01. Za skup podataka za učenje nisu definirani parovi pomoću kojih se može odrediti *ROC* krivulja.

Testiranje na *FRGC* bazi podataka

Na slici 6.3 prikazana je *ROC* krivulja za najbolji model na drugom skupu podataka za testiranje (*FRGC*). Neka lica na slikama nije bilo moguće detektirati detektorom lica pa je od ukupnog broja slika lica koji se nalazi u bazi ostalo njih 43947. Nasumično je odabrano 20000 različitih parova slika, 10000 pozitivnih parova i 10000 negativnih parova te je izvršeno testiranje.



Slika 6.3: *ROC* krivulja za najbolji primjer na drugom skupu podataka za testiranje (*FRGC*)

AUC iznosi 0.97179892. Točnost tog modela za diskriminacijsku granicu koja maksimizira točnost (0.8653 je iznos diskriminacijske granice) je 0.9182. U tom slučaju preciznost iznosi 0.910886225192, odziv je 0.9271, a F1 mjera iznosi 0.91892159778.

Prikazana je matrica zabune za gore navedene rezultate.

Tablica 6.4: Matrica zabune za drugi skup podataka za testiranje (*FRGC*)

		Stvarna klasa		
		1	*0*	Ukupno
Predviđena klasa	*1*	9271	907	10178
	0	729	9093	9822
	Ukupno	10000	10000	20000

7. Zaključak

U ovom radu prikazan je razvoj sustava za automatsko raspoznavanje lica. Korištene su duboke konvolucijske neuronske mreže koje su se, gledajući eksperimentalne rezultate, pokazale kao dobar algoritam za rješavanje problema raspoznavanja lica. Duboke konvolucijske neuronske mreže trenutno su najbolji (*state of the art*) algoritmi za rješavanje mnogih problema povezanih s računalnim vidom.

Sustav je sposoban sam, uz dobro podešene parametre, iz slika ljudskih lica izvući bitne značajke koje međusobno razlikuju ljudska lica. To je problem koji je izuzetno teško i kompleksno računalno riješiti.

Sam proces učenja traje relativno dugo (više od dva dana), ali jednom kada je naučen, sustav u vrlo kratkom roku, u svega nekoliko milisekundi, može klasificirati parove slika u pozitivnu (dvije osobe na dvije različite slike su iste) ili negativnu klasu (dvije osobe na dvije različite slike nisu iste). Zbog brzine, sustav je potencijalno pogodan i za korištenje u obradi video datoteka uživo.

Opisani sustav s velikom točnošću (0.917301221888 na *LFW* bazi) generalizira i raspoznaje neviđene pozitivne i negativne parove slika. Ipak, ima prostora za poboljšavanje performansi. Performanse bi se daljnjim razvojem mogle povećati povećanjem baze podataka za učenje, korištenjem nekih drugih arhitektura neuronskih mreža, ali isto tako i različitim izborom parametara samog algoritma učenja kao što su npr. veličina *mini batch*-a i veličina slika koje ulaze u mrežu ili izborom različitih mehanizama koji se koriste u algoritmu, npr. neki drugačiji način odabira slika koje ulaze u neuronsku mrežu (u ovom radu slike se biraju slučajnim izborom).

Duboke konvolucijske neuronske mreže imaju veliki potencijal u rješavanju mnogobrojnih problema, a njihovim daljnjim razvojem i usavršavanjem, u što se danas ulažu velika sredstva, i automatsko raspoznavanje lica će postajati sve preciznije, pouzdanije i bolje.

LITERATURA

Bor-Chun Chen, Chu-Song Chen, i Winston H. Hsu. *Cross-Age Reference Coding for Age-Invariant Face Recognition and Retrieval*. ECCV, 2014.

Bor-Chun Chen, Chu-Song Chen, i Winston H. Hsu. *Face Recognition using Cross-Age Reference Coding with Cross-Age Celebrity Dataset*. IEEE Transactions on Multimedia, 2015.

Ronan Collobert, Koray Kavukcuoglu, i Clement Farabét. *Torch7: A Matlab-like Environment for Machine Learning*. BigLearn, NIPS Workshop, 2011.

Tim Dettmers. *Deep Learning in a Nutshell: Core Concepts*. 2015. URL `https://devblogs.nvidia.com/parallelforall/deep-learning-nutshell-core-concepts/`.

Andrew Gibiansky. *Convolutional Neural Networks*. 2014. URL `http://andrew.gibiansky.com/blog/machine-learning/convolutional-neural-networks/`.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, i Jian Sun. *Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification*. arXiv, arXiv:1502.01852v1, 2015.

Gary B. Huang, Manu Ramesh, Tamara Berg, i Erik Learned-Miller. *Labeled Faces in the Wild: A Database for Studying Face Recognition in Unconstrained Environments*. 2007.

Sergey Ioffe i Christian Szegedy. *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*. arXiv, arXiv:1502.03167v3, 2015.

Yann LeCun, Leon Bottou, Genevieve B. Orr, i Klaus-Robert Müller. *Efficient BackProp*. Springer, 1998.

H.-W. Ng i S. Winkler. *A data-driven approach to cleaning large face datasets*. Proc. IEEE International Conference on Image Processing (ICIP), Paris, France, 2014.

O. M. Parkhi, A. Vedaldi, i A. Zisserman. *Deep Face Recognition*. British Machine Vision Conference, 2015.

P. Jonathon Phillips, Patrick J. Flynn, Todd Scruggs, Kevin W. Bowyer, Jin Chang, Kevin Hoffman, Joe Marques, Jaesik Min, i William Worek. *Overview of the Face Recognition Grand Challenge*. 2005.

prof. dr. sc. Bojana Dalbelo Bašić, mr.sc. Marko Čupić, i mr.sc. Jan Šnajder. *Umjetne neuronske mreže*. Zavod za elektroniku, mikroelektroniku i inteligentne sustave, Fakultet elektrotehnike i računarstva, Zagreb, 2008.

Python Software Foundation. 2016. URL `https://www.python.org/`.

Florian Schroff, Dmitry Kalenichenko, i James Philbin. *FaceNet: A Unified Embedding for Face Recognition and Clustering*. arXiv, arXiv:1503.03832v3, 2015.

Visage Technologies. 2016. URL `http://visagetechnologies.com/`.

Vedran Vukotić. *Raspoznavanje objekata dubokim neuronskim mrežama*. DIPLOMSKI RAD br. 696, Sveučilište u Zagrebu, Fakultet elektrotehnike i računarstva, Zagreb, 2014.

Lior Wolf, Tal Hassner, i Itay Maoz. *Face Recognition in Unconstrained Videos with Matched Background Similarity*. 2011.

Dong Yi, Zhen Lei, Shengcai Liao, i Stan Z. Li. *Learning Face Representation from Scratch*. 2014.

Matthew Zeiler. *ADADELTA: An Adaptive Learning Rate Method*. arXiv, arXiv:1212.5701v1, 2012.

Raspoznavanje lica

Sažetak

Opisan je razvoj sustava za automatsko raspoznavanje lica. Automatsko raspoznavanje lica zasniva se na tehnikama računalnog vida, a služi za raspoznavanje identiteta čovjeka na osnovu slike lica, pri čemu se koristi usporedba s ranije registriranim slikama lica za koje je poznat identitet. Skup ranije registriranih lica naziva se galerija. Korištene su duboke konvolucijske neuronske mreže. Duboke konvolucijske neuronske mreže sažimaju slike u 128-dimenzionalni deskriptor, a Euklidske udaljenosti deskriptora lica izravno odgovaraju sličnostima lica.

Ključne riječi: raspoznavanje lica, duboka konvolucijska neuronska mreža, strojno učenje, triplet loss, AdaDelta, algoritam unazadne propagacije, stohastički gradijentni spust, Torch framework

Face Recognition

Abstract

This paper describes the development of a system for automatic face recognition. Automatic face recognition is based on the techniques for computer vision and is used to identify the identity of the person based on the facial image. It uses a comparison with previously registered facial images with identities that are known. A set of previously registered faces is called the gallery. Deep convolutional neural network is used for face recognition. It compresses the image into 128-dimensional descriptor. Euclidean distance between descriptors directly corresponds to the similarities between the faces.

Keywords: face recognition, deep convolutional neural network, machine learning, triplet loss, AdaDelta, backpropagation algorithm, stochastic gradient descent, Torch framework