

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 1104

SUSTAV ZA DETEKCIJU KARAKTERISTIČNIH TOČAKA LICA

Toni Polanec

Zagreb, lipanj 2023.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 1104

SUSTAV ZA DETEKCIJU KARAKTERISTIČNIH TOČAKA LICA

Toni Polanec

Zagreb, lipanj 2023.

Zagreb, 10. ožujka 2023.

ZAVRŠNI ZADATAK br. 1104

Pristupnik: **Toni Polanec (0036535771)**
Studij: Elektrotehnika i informacijska tehnologija i Računarstvo
Modul: Računarstvo
Mentor: izv. prof. dr. sc. Tomislav Hrkać

Zadatak: **Sustav za detekciju karakterističnih točaka lica**

Opis zadatka:

Postupci detekcije karakterističnih točaka lica u slikama lica posljednjih godina pronalaze brojne primjene. U okviru završnoga rada potrebno je proučiti postupke za detekciju karakterističnih točaka lica opisane u literaturi, s naglaskom na pristupe temeljene na dubokom učenju. Predložiti i programski ostvariti sustav koji će omogućiti automatsku detekciju karakterističnih točaka lica na zadanoj ulaznoj slici lica. Pripremiti bazu slika za učenje i ispitivanje sustava, analizirati ponašanje ostvarenog sustava te prikazati i ocijeniti ostvarene rezultate. Radu priložiti izvorni i izvršni kod razvijenih postupaka, ispitne sekvence i rezultate, uz potrebna objašnjenja i dokumentaciju te navesti korištenu literaturu.

Rok za predaju rada: 9. lipnja 2023.

Sadržaj

Uvod.....	1
1. Podaci	2
1.1. Skup podataka	2
1.2. Analiza podataka	3
1.2.1. Provjera nedostajućih vrijednosti	3
1.2.2. Imputacija nedostajućih vrijednosti	4
2. Model.....	7
2.1. Konvolucijska neuronska mreža	7
2.1.1. Konvolucijski sloj	8
2.1.2. Aktivacijski sloj	9
2.1.3. Sloj sažimanja.....	12
2.1.4. Potpuno povezani sloj	13
2.2. Arhitektura modela.....	14
3. Treniranje modela	17
3.1. Priprema podataka za treniranje	17
3.2. Treniranje modela.....	18
3.3. Evaluacija modela	19
3.4. Prikaz rezultata.....	20
Zaključak.....	22
Literatura	23
Slike	24
Sažetak	25
Summary	26
Privitak	27

Uvod

Problematika detekcije karakterističnih točaka lica seže desetak godina unazad s prvim sigurnosnim kamerama i osobnim mobitelima. Detekcijom lica i njegovih karakterističnih točaka mogu se automatski prepoznati dijelovi slike koji su najbitniji (često to bude baš lice osobe). To je potaknulo inženjere da pronađu učinkovito rješenje. To rješenje utjelovilo se u obliku konvolucijskih neuronskih mreža. Konvolucijske neuronske mreže prvi put su se pojavile u 80-ima kao rješenje za prepoznavanje ljudskog rukopisa. U ovom radu će se isto koristiti zbog njihove sposobnosti prepoznavanja i izdvajanja značajki u slikovnim podacima.

Opisat će se cijeli put od prvog pogleda na skup podataka do prikaza rezultata treniranja. Fokus će biti na analizi i manipulaciji početnog skupa podataka. Opisani će biti glavni dijelovi konvolucijske neuronske mreže i prikazat će se arhitektura korištenog modela. Za kraj će biti opisano treniranje modela i prikazivanje rezultata.

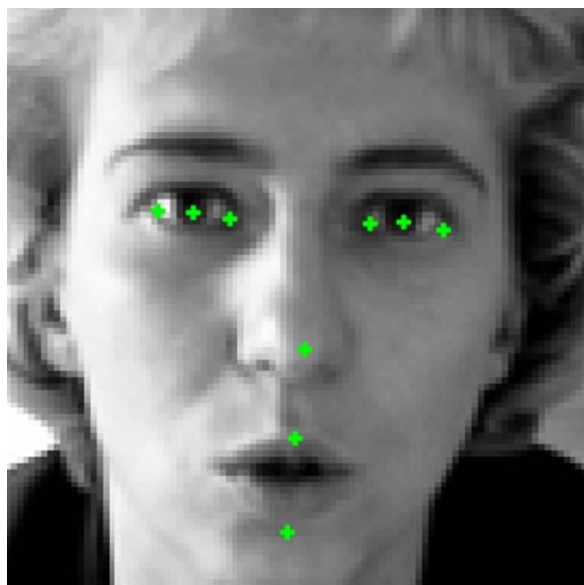
1. Podaci

1.1. Skup podataka

Podaci na kojima će se učiti model je preuzet s Kaggle stranice. *Face Images with Marked Landmark Points* [1] od dr. Yoshua Bengio sa sveučilišta u Montrealu. Skup podataka (engl. *dataset*) sastoji se od 7049 slika lica s do 15 karakterističnih točaka označenim na njima. Koordinate točaka spremljene su u csv formatu. Zbog samog skupa podataka (više o nedostacima skupa u nastavku) i zbog jednostavnosti rješenja koristit će se manji broj karakterističnih točaka, i to njih 9 (Slika 1.1):

- *left_eye_center*
- *left_eye_inner_corner*
- *left_eye_outer_corner*
- *right_eye_center*
- *right_eye_inner_corner*
- *right_eye_outer_corner*
- *nose_tip*
- *mouth_center_top_lip*
- *mouth_center_bottom_lip*

S ovih 9 točaka za bilo koju sliku dobiva se dovoljno precizna predodžba gdje su najbitnije točke lica (oči, nos, usta).



Slika 1.1 - Prikaz odabranih karakterističnih točaka na licu

1.2. Analiza podataka

Da se razumije zadatak i riješi na najbolji mogući način moraju se detaljno proučiti podaci na kojima će se trenirati model. Trebaju se pronaći mogući nedostaci u samim podacima (nedostajuće (engl. *null*) vrijednosti, krivi podaci, itd.) i adekvatno riješiti ti problemi.

1.2.1. Provjera nedostajućih vrijednosti

Ako se kod treniranja koristi slika koja ima nedostajuću (*null*) vrijednost na poziciji npr. *left_eye_center* može dovesti do iznimke u programu koji prekida cijelo dugotrajno treniranje ili može navesti model na potpuno krivo zaključivanje, pogotovo ako ima puno *null* vrijednosti. Za početak pročita se csv datoteka i sprema se u *Dataframe* (struktura podataka Pandas knjižnice).

Ako u određenim stupcima nedostaje samo par vrijednosti, takvi zapisi se brišu iz skupa podataka zbog jednostavnosti. Nekoliko slika manje u skupu nije presudno kod treniranju modela. Često je produktivnije maknuti problematične dijelove skupa podataka nego se mučiti s njihovim popravljanjem. Ako je broj *null* vrijednosti veći, mora se pronaći način kako iz poznatih podataka generirati te vrijednosti.


```
# provjera koliko ima null vrijednosti u svakom stupcu
landmarks.isnull().sum()
```

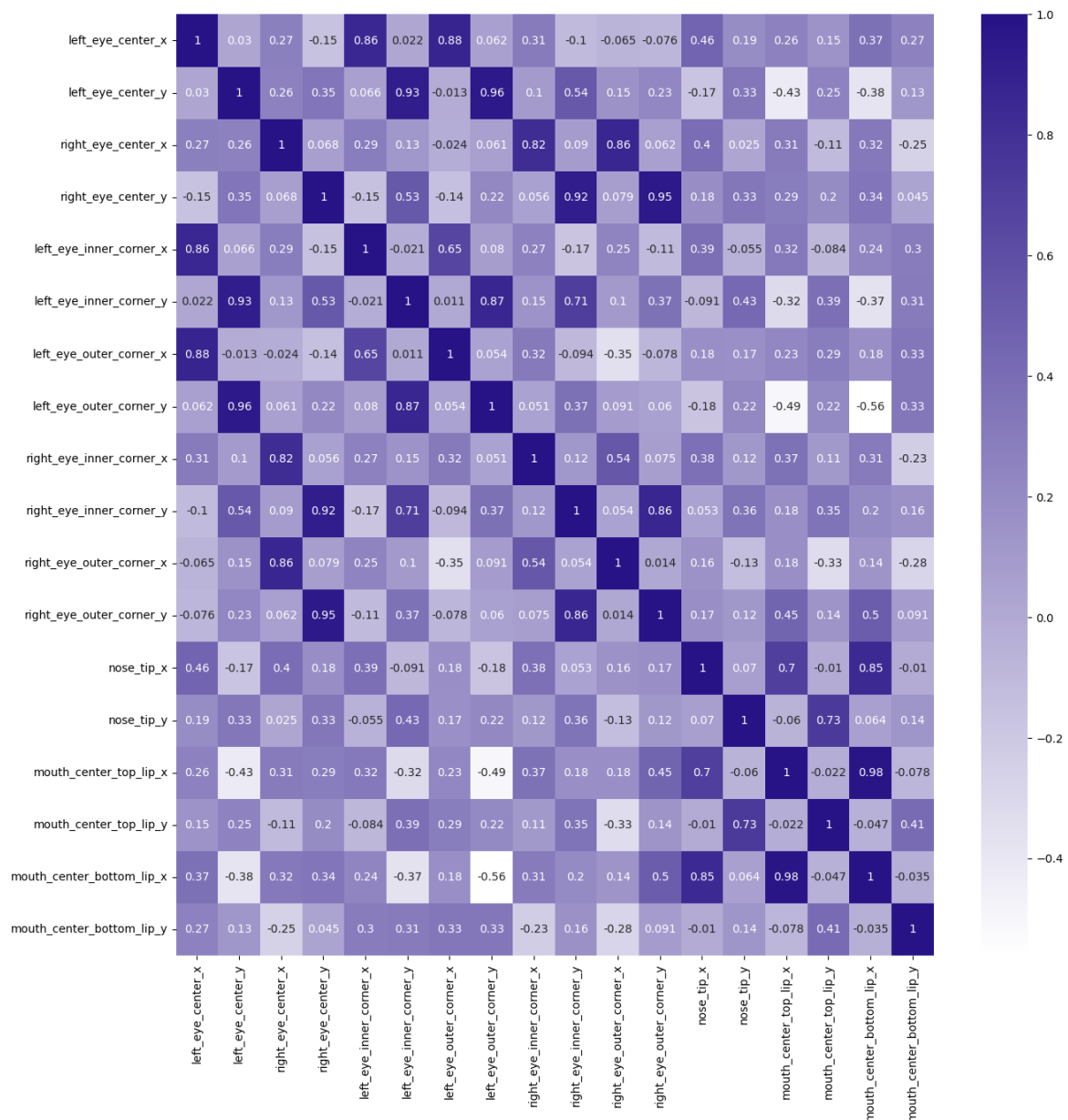
left_eye_center_x	10
left_eye_center_y	10
right_eye_center_x	13
right_eye_center_y	13
left_eye_inner_corner_x	4778
left_eye_inner_corner_y	4778
left_eye_outer_corner_x	4782
left_eye_outer_corner_y	4782
right_eye_inner_corner_x	4781
right_eye_inner_corner_y	4781
right_eye_outer_corner_x	4781
right_eye_outer_corner_y	4781
nose_tip_x	0
nose_tip_y	0
mouth_center_top_lip_x	4774
mouth_center_top_lip_y	4774
mouth_center_bottom_lip_x	33
mouth_center_bottom_lip_y	33

Iz priloženog ispisa je vidljivo da nedostaje nekoliko vrijednosti za točke *left_eye_center* i *right_eye_center*. Ovi zapisi se brišu iz skupa podataka. Brišu se i preostali zapisi kojima nedostaje vrijednost koordinata za *mouth_center_bottom_lip*.

1.2.2. Imputacija nedostajućih vrijednosti

Nedostaje jako puno vrijednosti za *right/left_eye_inner/outer_corner* i za *mouth_center_top_lip*. Za ove slučajeve koriste se metode imputacije. Imputacija je proces popunjavanja nepostojećih podataka s na novo izračunatim podacima od već poznatih podataka [2]. Ima puno različitih načina popunjavanja nepostojećih podataka i odabir metode znatno ovisi o konkretnom problemu. U korištenim podacima vidljivo je da nedostaju točke oko oka, ali točka centra oka je u svim preostalim zapisima prisutna. Na isti način postoji vrijednost koordinata točke centra donje usne, a nedostaju podaci za centar gornje usne. Pomoću intuicije prepoznaje se da bi te točke mogle biti povezane i da

bi pomoću jedne mogli pretpostaviti koordinate druge. U tehničkom aspektu spomenuta intuicija može se prikazati korelacijom točaka (Slika 1.2).



Slika 1.2 - Korelacija koordinata karakterističnih točaka

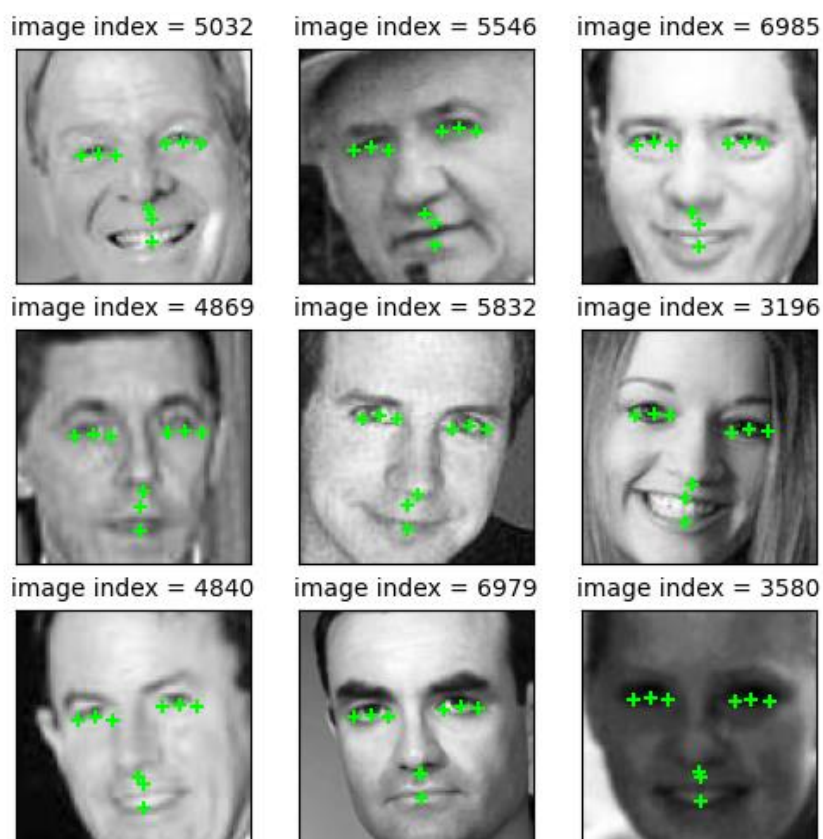
Intuitivno je da će y-koordinata unutarnje točke oka biti jako slična y-koordinati centra oka i to se vidi na tablici korelacije. Korelacija *left_eye_center_y* i *left_inner_corner_y* iznosi 0.93. Ista situacija je s usnama. Napraviti će se parovi točaka s visokom korelacijom s obzirom na to da li imaju nedostajuće vrijednosti ili ne, tj. trebaju se spojiti točka koja nema null vrijednosti s točkom koja ih ima tako da imaju čim veću međusobnu korelaciju.

Za nepoznate točke oko lijevog oka koristit će se pozicija točke *left_eye_center*. Isto tako za desno oko. Za centar gornje usne koristi se poznata točka centra donje usne.

Iteriranjem po svim spomenutim parovima s poznatim vrijednostima i jedne i druge točke izračunava se prosjek udaljenosti jedne točke od druge. Nedostajuće vrijednosti se popunjavaju tako da se zbroje koordinate poznate točke s izračunatim prosjekom. Ovim načinom se ne dobiju najpreciznije točke (Slika 1.3), ali je dobar kompromis s obzirom na nepotpuni skup podataka (Slika 1.4).



Slika 1.3 - Neprecizna imputacija točaka



Slika 1.4 - Imputirani skup podataka

2. Model

2.1. Konvolucijska neuronska mreža

Za izgradnju i treniranje modela koristit će se konvolucijska neuronska mreža (engl. *CNN – Convolutional Neural Network*). Konvolucijska neuronska mreža je vrsta neuronske mreže koja se često koristi u području obrade slika i prepoznavanja uzoraka [3].

Konvolucijske neuronske mreže koriste posebne slojeve nazvane konvolucijski slojevi, koji se sastoje od filtera koji obrađuju ulazne podatke. Svaki filter primjenjuje konvoluciju na podatke kako bi izdvojio određene značajke ili uzorke iz ulazne slike, te značajke su često rubovi, nagle promjene boja slike i slično. CNN se pokazala jako uspješna u domeni računalnog vida zbog sposobnosti automatskog izvlačenja značajki iz sirovih podataka.

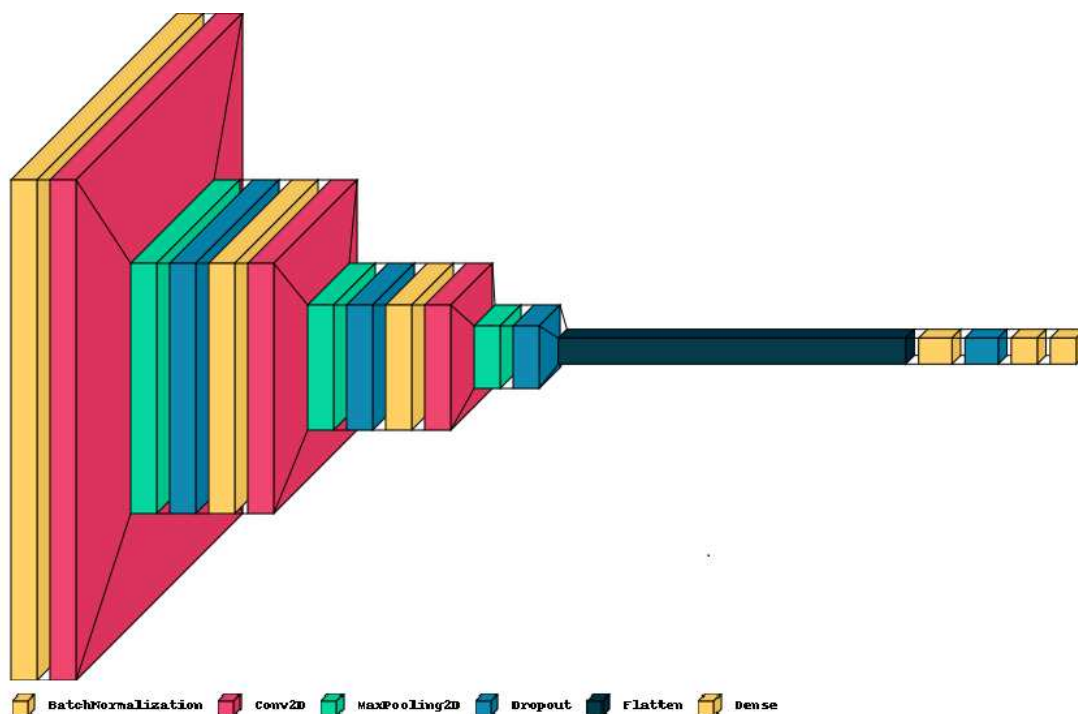
CNN je izgrađena s nekoliko glavnih slojeva:

- Konvolucijski sloj
- Aktivacijski sloj (ReLU sloj)
- Sloj sažimanja (engl. *Pooling layer*)
- Potpuno povezani sloj

Osim spomenutih koriste se i drugi slojevi, npr. Sloj normalizacije grupa (engl. *Batch normalization layer*) i sloj izostavljanja (engl. *Dropout layer*) koji s posebnim postupcima poboljšavaju učenje i model.

Sloj normalizacije grupa normalizira izlazne aktivacije prethodnog sloja [4]. To pomaže u održavanju stabilnosti tijekom učenja tako da sprječava ekstremne vrijednosti u ulaznim podacima sloja. Sloj izostavljanja pak nasumično odabire vrijednosti u mreži i ignorira ih kod učenja modela [5]. Time regulira prenaučenosť (engl. *overfitting*) i poboljšava generalizaciju modela. Postoji mnogo specifičnih slojeva koji se koriste za različite svrhe. U nastavku ćemo se fokusirati na 4 glavna sloja.

Kod planiranja arhitekture modela kombiniraju se prethodno spomenuti slojevi da se izgradi cijela konvolucijska neuronska mreža (Slika 2.1).



Slika 2.1 – Vizualizacija arhitekture modela

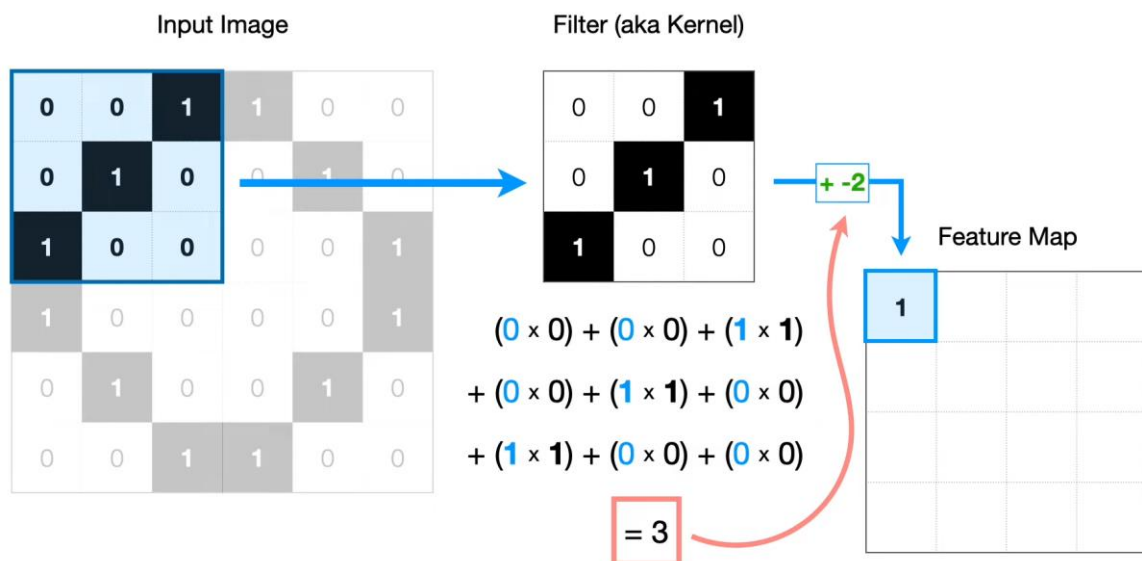
2.1.1. Konvolucijski sloj

Konvolucija je proces kojem uz pomoću filtera (jezgri, engl. *kernel*) smanjujemo veličinu ulaznih podataka. Ulazni podaci su strukturirani kao 2D polje. Kao primjer ćemo koristiti ulazno polje veličine 6x6 i jezgru veličine 3x3 (Slika 2.2)

Input Image						Filter (aka Kernel)		
0	0	1	1	0	0	0	0	1
0	1	0	0	1	0	0	1	0
1	0	0	0	0	1	1	0	0
1	0	0	0	0	1			
0	1	0	0	1	0			
0	0	1	1	0	0			

Slika 2.2 - Ulazno polje i jezgra filtra [6]

Jezgra svakim korakom prekriva dio ulazne slike i izračunava rezultat na temelju polja koje se podudaraju. Izračunava se dot produkt matrice koji se zbraja s težinom i upisuje se vrijednost u polje značajki (engl. *feature map*) (Slika 2.3).



Slika 2.3 - Dot produkt jezgre i ulaza, upis u polje značajki [6]

Parametar kojim se određuje kako se jezgra pomiče naziva se korak (engl. *stride*). On definira brzinu prolaska jezgre po ulaznom polju i određuje detaljnost uzorkovanja. Korak neka bude 1. Prolaskom jezgre po svim dijelovima ulaznog polja dobiva se popunjeno polje značajki (Slika 2.4).

Ovi koraci zaključuju postupak konvolucije. Iz ulaznog polja veličine 6x6 dobije se polje značajki veličine 4x4. Dodatnim koracima količina podataka se još dodatno smanjuje.

Feature Map


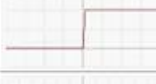







1	-1	-2	-1
-1	-2	-1	-2
-2	-1	-2	-1
-1	-2	-1	1

Slika 2.4 - Polje značajki [6]

2.1.2. Aktivacijski sloj

Aktivacijski sloj sastoji se od aktivacijskih funkcija. U jednom modelu najčešće se koristi samo jedna i primjenjuje se na sve ulaze sloja. Aktivacijske funkcije standardiziraju izlaze slojeva neuronskih mreža na od prije definirane domene. Svaka aktivacijska funkcija ima prednosti i nedostatke te se ovisno o problematici zadatka odabire aktivacijska

funkcija koja se pokazala najboljom u tom području. Primjeri aktivacijskih funkcija prikazane su na Slici 2.5.

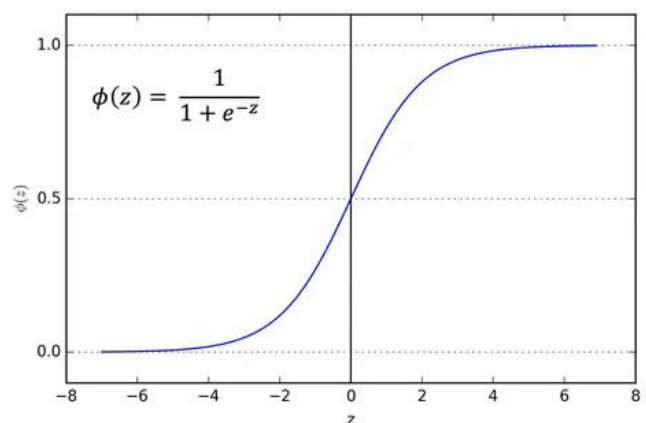
Name	Plot	Equation	Derivative
Identity		$f(x) = x$	$f'(x) = 1$
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x \neq 0 \\ ? & \text{for } x = 0 \end{cases}$
Logistic (a.k.a Soft step)		$f(x) = \frac{1}{1 + e^{-x}}$	$f'(x) = f(x)(1 - f(x))$
Tanh		$f(x) = \tanh(x) = \frac{2}{1 + e^{-2x}} - 1$	$f'(x) = 1 - f(x)^2$
ArcTan		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2 + 1}$
Rectified Linear Unit (ReLU)		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Parameteric Rectified Linear Unit (PReLU) [2]		$f(x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Exponential Linear Unit (ELU) [3]		$f(x) = \begin{cases} \alpha(e^x - 1) & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} f(x) + \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
SoftPlus		$f(x) = \log_e(1 + e^x)$	$f'(x) = \frac{1}{1 + e^{-x}}$

Slika 2.5 - Aktivacijske funkcije [7]

Sigmoidna aktivacijska funkcija

Sigmoidna funkcija mapira izlaze na kodomenu $[0, 1]$. Korisna je kod modela koji imaju vrijednosti podataka u rasponu $[0, 1]$ (Slika 2.6).

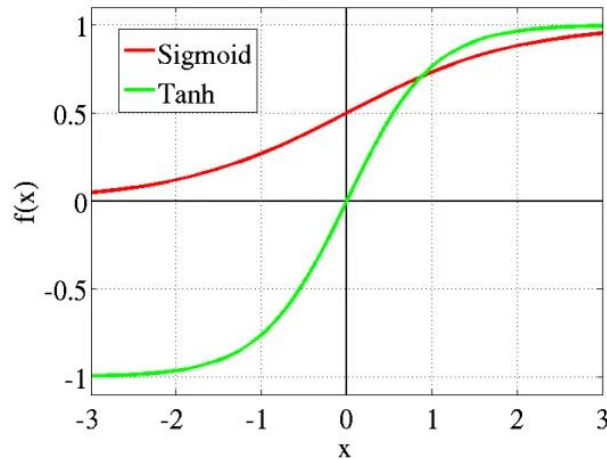
Ima nedostatke poput mogućnosti da neuronsku mrežu „zaglavi“ u treniranju [7]. Poopćena verzija sigmoidne funkcije je softmax aktivacijska funkcija.



Slika 2.6 - Sigmoidna aktivacijska funkcija

Hiperbolna aktivacijska funkcija

Mapira izlaz funkcije na raspon od $[-1, 1]$. Često se koristi u modelima koji klasificiraju podatke u dva razreda.

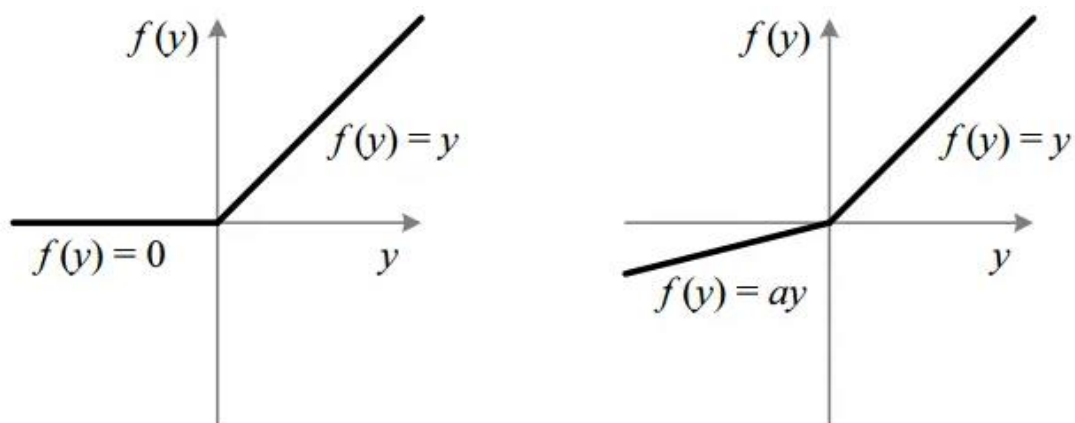


Slika 2.7 - Hiperbolna i sigmoidna aktivacijska funkcija

ReLU aktivacijska funkcija

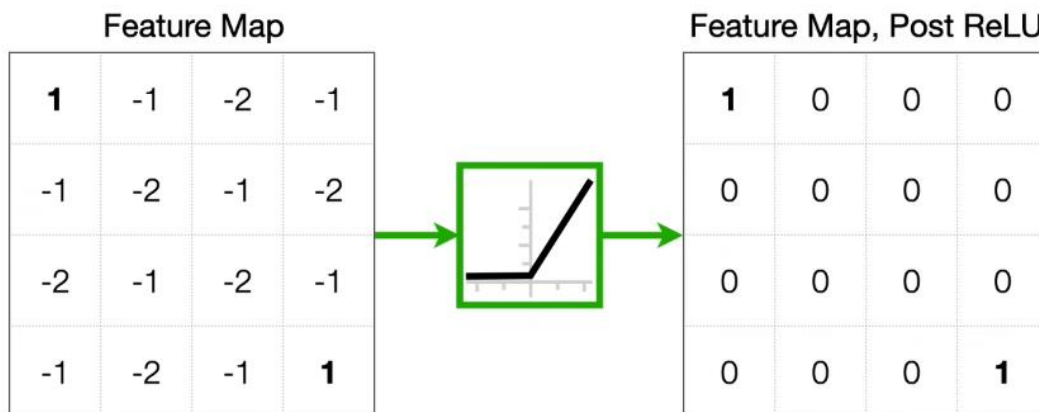
ReLU (engl. *Rectified Linear Unit*). Trenutno najkorištenija aktivacijska funkcija [7]. Najčešće korištena u konvolucijskim neuronskim mrežama i dubokom učenju. Svaka vrijednost koja je negativna se mapira na nulu. Pozitivne vrijednosti ostaju iste.

Mana ReLU aktivacijske funkcije je baš to naglo postavljanje negativnih vrijednosti na nulu. To nekad negativno utječe na treniranje modela pa se u tim slučajevima koristi Leaky ReLU (Slika 2.8). Leaky ReLU aktivacijska funkcija koristi se kod modela koji se obrađuje u ovom radu.



Slika 2.8 - ReLU i Leaky ReLU aktivacijske funkcije

Kada se prethodni primjer polja značajki *aktivira* aktivacijskom funkcijom ReLU dobije se sljedeća matrica (Slika 2.9).



Slika 2.9 - Aktivacija polja značajki [6]

2.1.3. Sloj sažimanja

Nakon aktivacijskog sloja slijedi sloj sažimanja (engl. *Pooling layer*). On dodatno smanjuje količinu podataka koje moramo obrađivati. Postoje više vrsta sažimanja: Prosječno sažimanje (engl. *Average Pooling*), Maksimalno sažimanje (engl. *Max Pooling*). Max pooling češće daje bolje rezultate zbog većih vrijednosti nakon sažimanja (Average pooling previše umanjuje vrijednosti i gube se specifičnosti modela).

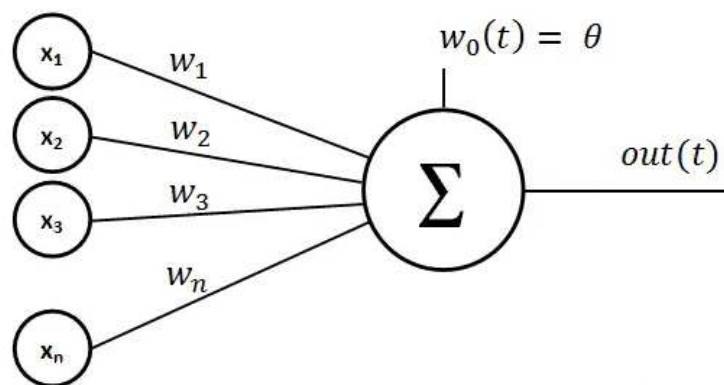
Pooling slično funkcionira kao i konvolucija. U poolingu imamo prozor koji pomičemo po ulaznoj matrici i uzimamo najveću vrijednost koja je prekrivena tim prozorom. Razlika od konvolucije je da se ne prekrivaju ista polja ulazne matrice više puta nego samo jednom (Slika 2.10).



Slika 2.10 - Sažimanje polja značajki [6]

2.1.4. Potpuno povezani sloj

Prethodni slojevi se mogu ponavljati po potrebi. Ako je problem zadatka kompliciraniji preporučuje se slijedno ponavljanje konvolucijskog, aktivacijskog i pooling sloja. Nakon tih slojeva mora slijediti neki način finalne klasifikacije i predikcije modela. To se implementira pomoću potpuno povezane neuronske mreže. Dakle, svi izlazi posljednjeg pooling sloja se „rastežu“ (pretvaraju u jednodimenzionalno polje) i to nam čini ulaz potpuno povezane neuronske mreže. Izlaze čine perceptroni (Slika 2.11) i svaki perceptron predstavlja jedan razred podataka.



Slika 2.11 - Perceptron [8]

Perceptron je najmanja jedinica neuronske mreže. On sumira sve ulaze pomnožene s njihovim težinama i zbraja k tome svoju težinu. Izlaz se često aktivira nekom od aktivacijskih funkcija.

2.2. Arhitektura modela

Korišteni model sastoji se od više ponavljajućih konvolucija. Veličina ulaza je 96x96 što se podudara s veličinom slika. Sljedeća grupa slojeva se ponavlja 3 puta:

```
BatchNormalization()  
Conv2D(32, (3,3), padding="same", kernel_initializer=he_uniform())  
LeakyReLU(alpha=0.1)  
MaxPooling2D(pool_size=(2, 2))  
Dropout(0.2)
```

Između grupa se mijenja broj filtera u konvoluciji. U prvoj iteraciji je broj filtera 32, u drugoj 64, trećoj 128. S tim se pokušava izvući čim više detalja iz ulazne slike.

Veličina jezgre je konstantna i iznosi 3x3.

Kernel initializer definira način odabira nasumičnih težina jezgre. Odabran je he_uniform koji je stabilniji od drugih i pruža dobre rezultate [9].

Korištena aktivacijska funkcija je Leaky ReLU.

Sloj sažimanja je implementiran pomoću Max Poolinga s jezgrom veličine 2x2. 4 vrijednosti se evaluiraju i uzima se najveća vrijednost. Četverostruko smanjuje broj parametara.

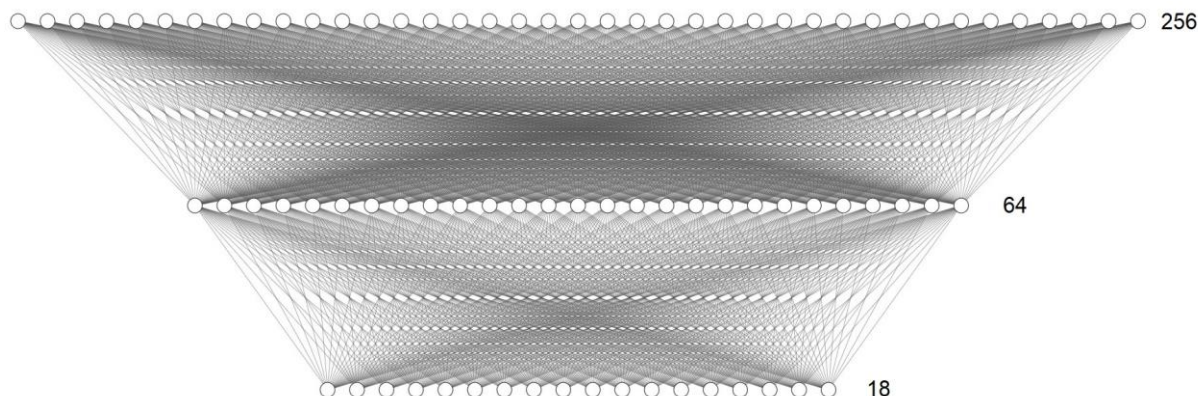
Za kraj je postavljen Dropout sloj koji nasumično odabire čvorove mreže koje ignorira [5]. Sprječava prenaučenos modela.

Nakon spomenutih slojeva slijede 3 potpuno povezana slojeva (Dense slojevi).

```
Flatten()  
Dense(256, kernel_initializer=he_uniform(), activation=LeakyReLU(0.1))  
Dropout(0.5)  
Dense(64, kernel_initializer=he_uniform(), activation=LeakyReLU(0.1))  
Dense(18)
```

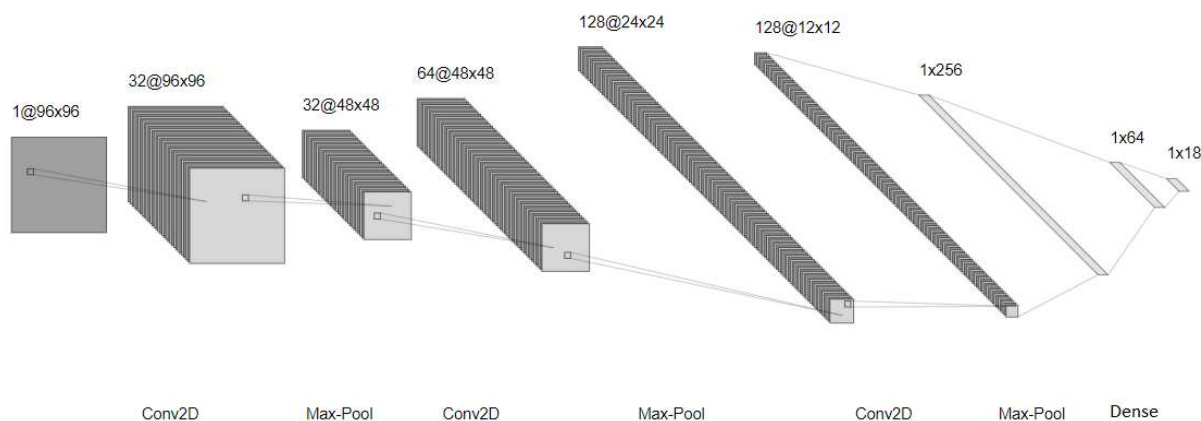
Sloj Flatten rasteže sve multidimenzionalne podatke u jednodimenzionalno polje. Potpuno povezani slojevi prikazani su na Slici 2.12. (broj čvorova smanjen zbog vidljivosti). Broj

izlaznih čvorova iznosi 18, svaki od tih izlaza predstavlja jednu koordinatu od 9 karakterističnih točaka lica.



Slika 2.12 - Potpuno povezani slojevi

Vizualizacija cijele arhitekture modela prikazana je na Slici 2.13.



Slika 2.13 - Arhitektura modela

Vizualizacije napravljene pomoću online alata [10].

Model se nakon strukturiranja mora kompajlirati. Kod kompilacije zadaju se dodatne značajke modela:

```
model.compile(loss= 'mean_squared_error',
              optimizer= Adam(),
              metrics= ['mean_squared_error'])
```

`loss` i `metrics` parametri određuju po koji metrici će se mjeriti uspješnost modela u predikciji. Koristi se `mean_square_error` zbog prirode problema koji se rješava. Model će predviđati koordinate točaka pa ima smisla da koristimo prosječnu udaljenost od prave točke za izračunavanje greške predikcije. Postavljanjem tog načina moći će se poslije treniranja grafički prikazati postupno smanjenje greške predikcije modela.

Za `optimizer` je postavljen Adam optimizer koji koristi stohastički gradijentni spust za optimizaciju modela [11].

3. Treniranje modela

Treniranje modela uključuje nekoliko koraka:

- Priprema podataka za treniranje
- Treniranje modela
- Evaluacija modela

3.1. Priprema podataka za treniranje

Popravljeni skup podataka od prije sada se može finalno pripremiti za ulaz u model. Postoji naivna opcija u kojoj se cijeli skup koristi za treniranje, ali tu postoji problem. Nakon treniranja nije moguće evaluirati model i njegovu naučenost.

Skup podataka se dijeli na 2 seta. Set za treniranje i set za testiranje.

```
train_x, test_x, train_y, test_y =  
    train_test_split(faces, landmarks, test_size=0.2)
```

80% skupa je namijenjeno za treniranje, dok je 20% namijenjeno za testiranje i validaciju tijekom treniranja. Točan broj slika i oblik skupova podataka vidljiv je na slijedećem isječku ispisa:

```
faces.shape      -> (7000, 96, 96, 1)  
landmarks.shape -> (7000, 18)  
  
train_x.shape -> (5600, 96, 96, 1)  
train_y.shape -> (5600, 18)  
test_x.shape  -> (1400, 96, 96, 1)  
test_y.shape  -> (1400, 18)
```

3.2. Treniranje modela

Treniranje modela izvodi se Keras-ovom (Keras Python knjižnica) metodom `fit()`. Ima nekoliko bitnih parametara koji se trebaju postaviti.

```
model.fit(  
    x = train_x,  
    y = train_y,  
    batch_size = 128,  
    epochs = 60,  
    validation_data = (test_x, test_y))
```

`x` su ulazni podaci trening seta, dok su `y` izlazni podaci trening seta. Znači `x` su slike, a `y` koordinate karakterističnih točaka lica. `batch_size` određuje koliko ulaznih podataka model analizira prije ažuriranja (optimizacije) modela [12]. `epochs` određuje koliko puta će model proći kroz sve ulazne podatke. Ponovno prolazeći kroz ulazne podatke model uči i poboljšava predviđanje. `validation_data` služi za dinamičko provjeravanje uspješnosti učenja modela. Nakon svake epohe model pokušava predvidjeti koordinate karakterističnih točaka na skupu slika koje još nije vidio. Model ne uči na tim podacima!

Primjer ispisa `model.fit()` metode:

```
...  
Epoch 5/60  
44/44 [=====] - 145s 3s/step  
- loss: 191.4231 - mean_squared_error: 191.4231  
- val_loss: 369.4090 - val_mean_squared_error: 369.4090  
  
Epoch 6/60  
44/44 [=====] - 144s 3s/step  
- loss: 169.5139 - mean_squared_error: 169.5139  
- val_loss: 274.2195 - val_mean_squared_error: 274.2195  
  
Epoch 7/60  
44/44 [=====] - 145s 3s/step  
- loss: 154.3697 - mean_squared_error: 154.3697  
- val_loss: 211.8781 - val_mean_squared_error: 211.8781  
...
```

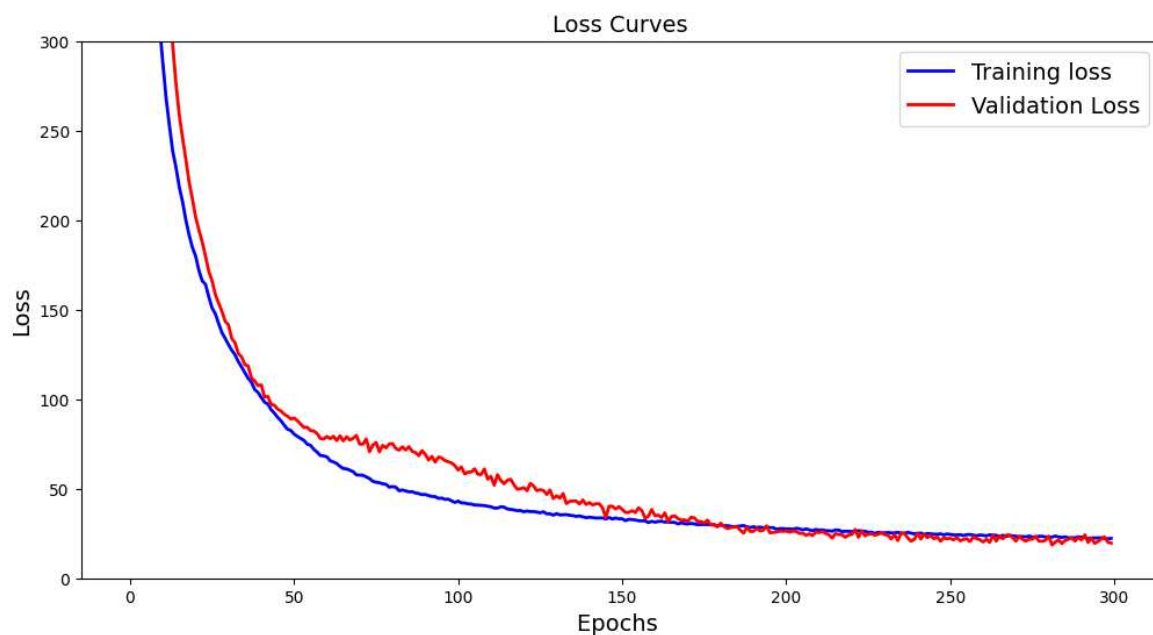
`loss` predstavlja grešku na trening podacima. `val_loss` predstavlja grešku na testnim podacima. Poželjno ponašanje je da kroz epohe pada `val_loss` jer to znači da se model

poboljšava na podacima na kojima nije učio i koji su mu nepoznati. Na isto takvim podacima će se kasnije model koristiti pa je to najbitnija metrika uspješnosti modela.

3.3. Evaluacija modela

Evaluacijom modela tek saznajemo je li model uspješan ili ne. Imamo više načina evaluacije. Kod mnogih problema koji se rješavaju učenjem evaluacija se vrši matricom zabune, preciznošću, itd. Kod regresijskih problema ne možemo pomoću tih klasičnih metrika odrediti uspješnost modela. Najjednostavnije je provjeriti kako model predviđa točke, iscrtati ih na slici i provjeriti ručno. Takav način je dobar za površnu evaluaciju modela, ali nam ne garantira uspješnost jer se pregledava samo mali dio skupa podataka. Točnija evaluacija dobiva se izračunavanjem greške modela na nepoznatim podacima.

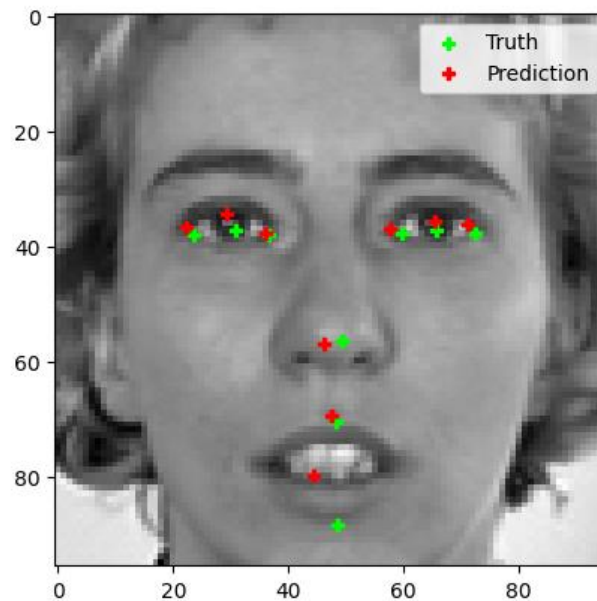
Dinamičku evaluaciju nudi Keras knjižnica koja tijekom treniranja modela bilježi grešku modela na validacijskim podacima nakon svake epohe (Slika 3.1)



Slika 3.1 - Graf greške modela tijekom učenja

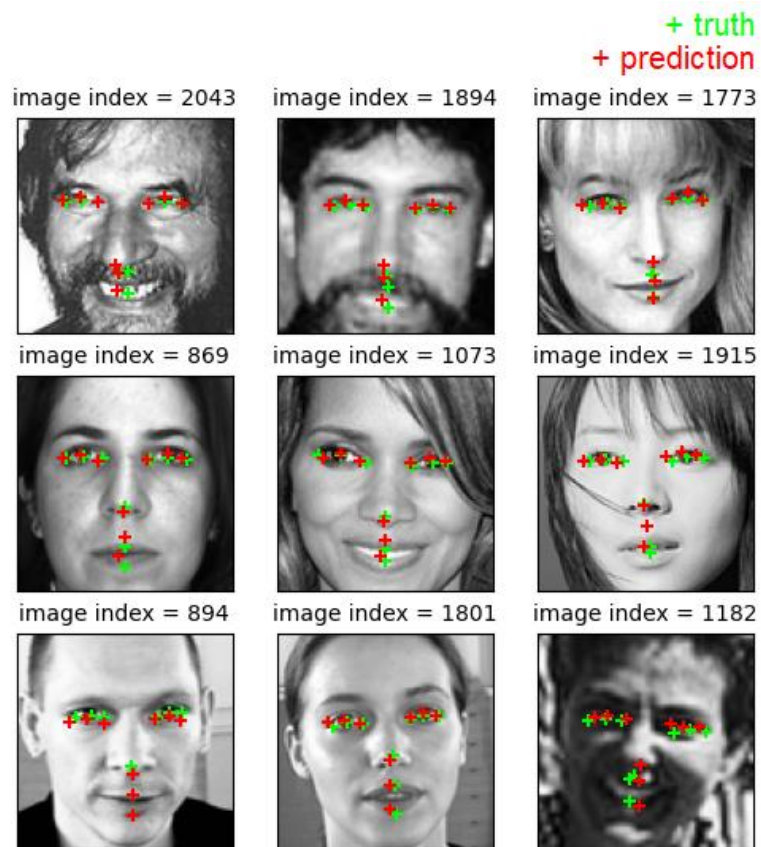
3.4. Prikaz rezultata

Nakon treniranja testira se model s nepoznatim podacima. Daju mu se nasumične slike i on pogađa koordinate karakterističnih točaka. Primjer rezultata prikazan je na Slici (3.2). Vidljivo je da model raspoznaje i točno određuje karakteristične točke lica. Minimalne greške postoje, ali to je i očekivano.

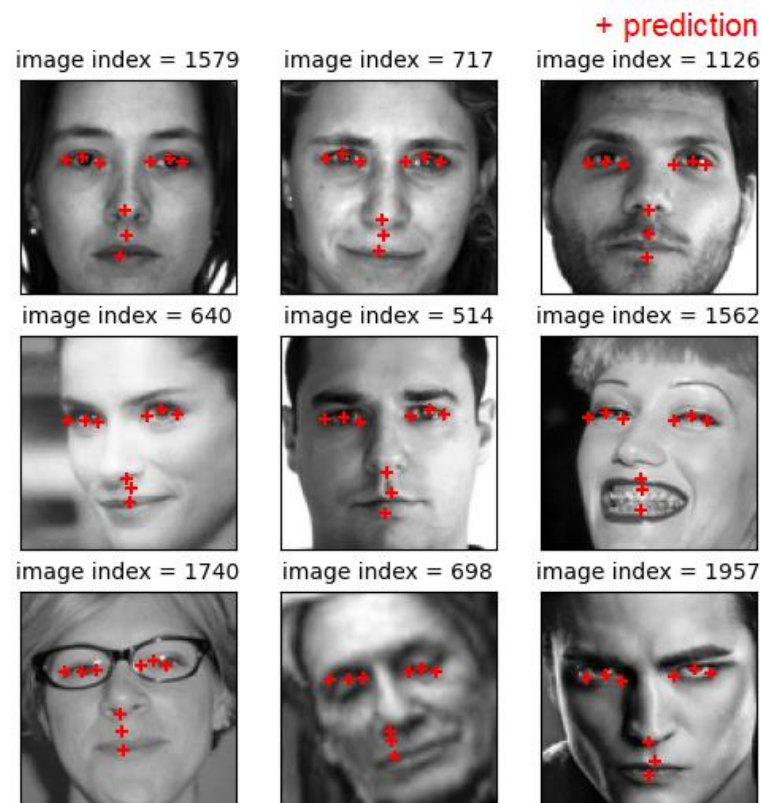


Slika 3.2 - Rezultat modela

Na Slici 3.3 prikazano je više primjera predikcije modela. Zanimljivo je prepoznati da na nekim slikama, ono što je model predvidio, izgleda preciznije nego stvarni podaci. Primjer su oči slike s indeksom 1182 (Slika 3.3). Model se ne nosi najbolje s licima koje su malo zarotirane (Slika 3.4 indeks 698). Bilo ih je premalo takvih u skupu za treniranje. Rješenje za to je uvođenje afinih transformacija (refleksija, translacija, rotacija) u pretprocesiranje podataka. Slika 3.4 prikazuje samo predikciju modela bez stvarnih točaka zbog bolje preglednosti.



Slika 3.3 - Dodatni primjeri predikcije modela



Slika 3.4 - Dodatni primjer predikcije modela bez stvarnih točaka

Zaključak

Konvolucijske neuronske mreže odlično odrađuju posao izvlačenja značajki iz slika i za to se jako često i koriste. Pametnom analizom i manipulacijom skupa podataka za treniranje moguće je dobiti vrlo dobro temelje za treniranje modela. Strukturiranje samog modela i podešavanje njegovih parametara može izgledati kao igra na sreću. No, uz poznavanje što koji sloj radi i koji im je zadatak može se, uz par pretpostavaka koje se kasnije detaljnije namjeste, dobiti vrlo uspješan model.

U daljnjem radu neke stvari koje bi se mogle poboljšati bile bi povećanje inicijalnog skupa podataka ili biranje drugačijeg skupa podataka koji nema toliko standardne slike. Bolje rezultate bi dao skup sa više varijacija u slikama. Gdje bi lica bila manja, veća, više lica u istoj slici, u nekim prirodnim pozama, itd. Dodatno poboljšanje nudilo bi dodavanje afinih transformacija u pretprocesiranje podataka. Preporučuje se i dodatno testiranje arhitekture modela sa više slojeva i promijenjenim parametrima. Dobiveni model nije savršen, i nije najkorisniji u stvarnim problemima, ali daje jako dobar uvod u svijet računalnog vida.

Literatura

- [1] Omri Goldstein, *Face Images with Marked Landmark Points*, <https://www.kaggle.com/datasets/drgilermo/face-images-with-marked-landmark-points> (pristupljeno 20. travnja 2023.)
- [2] Simplilearn, *Introduction to Data Imputation*, <https://www.simplilearn.com/data-imputation-article> (pristupljeno 24. travnja 2023.)
- [3] *What Is a Convolutional Neural Network?*, <https://www.mathworks.com/discovery/convolutional-neural-network-matlab.html> (pristupljeno: 27. travnja 2023.)
- [4] Jason Brownlee, *A Gentle Introduction to Batch Normalization for Deep Neural Networks*, <https://machinelearningmastery.com/batch-normalization-for-training-of-deep-neural-networks/> (pristupljeno 25. svibnja 2023)
- [5] Jason Brownlee, *A Gentle Introduction to Dropout for Regularizing Deep Neural Networks*, <https://machinelearningmastery.com/dropout-for-regularizing-deep-neural-networks/> (pristupljeno 25. svibnja 2023)
- [6] Joshua Starmer, *Neural Networks Part 8: Image Classification with Convolutional Neural Networks (CNNs)*, <https://www.youtube.com/watch?v=HGwBXDKFk9I> (pristupljeno 29. svibnja 2023.)
- [7] Sagar Sharma, *Activation Functions in Neural Networks*, <https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6> (pristupljeno 1. lipnja 2023.)
- [8] *Perceptron: Concept, function, and applications*, <https://datascientest.com/en/perceptron-definition-and-use-cases> (pristupljeno 2. lipnja 2023.)
- [9] Arnab Das, *Weight Initialization for Neural Networks — Does it matter?*, <https://towardsdatascience.com/weight-initialization-for-neural-networks-does-it-matter-e2fd99b3e91f> (pristupljeno 3. lipnja 2023.)
- [10] Alexander Lenail, <https://alexlenail.me/NN-SVG/> (pristupljeno 24. svibnja 2023.)
- [11] *Adam*, <https://keras.io/api/optimizers/adam/> (pristupljeno 3. lipnja 2023.)
- [12] Jason Brownlee, *Difference Between a Batch and an Epoch in a Neural Network*, <https://machinelearningmastery.com/difference-between-a-batch-and-an-epoch/> (pristupljeno 4. lipnja 2023.)

Slike

Slika 1.1 - Prikaz odabranih karakterističnih točaka na licu	3
Slika 1.2 - Korelacija koordinata karakterističnih točaka	5
Slika 1.3 - Neprecizna imputacija točaka.....	6
Slika 1.4 - Imputirani skup podataka	6
Slika 2.1 – Vizualizacija arhitekture modela.....	8
Slika 2.2 - Ulazno polje i jezgra filtra	8
Slika 2.3 - Dot produkt jezgre i ulaza, upis u polje značajki.....	9
Slika 2.4 - Polje značajki.....	9
Slika 2.5 - Aktivacijske funkcije	10
Slika 2.6 - Sigmoidna aktivacijska funkcija.....	10
Slika 2.7 - Hiperbolna i sigmoidna aktivacijska funkcija	11
Slika 2.8 - ReLU i Leaky ReLU aktivacijske funkcije	11
Slika 2.9 - Aktivacija polja značajki.....	12
Slika 2.10 - Sažimanje polja značajki	12
Slika 2.11 - Perceptron.....	13
Slika 2.12 - Potpuno povezani slojevi.....	15
Slika 2.13 - Arhitektura modela.....	15
Slika 3.1 - Graf greške modela tijekom učenja	19
Slika 3.2 - Rezultat modela	20
Slika 3.3 - Dodatni primjeri predikcije modela	21
Slika 3.4 - Dodatni primjer predikcije modela bez stvarnih točaka.....	21

Sažetak

Sustav za detekciju karakterističnih točaka lica

U ovom radu opisan je cijeli proces treniranja modela za detekciju karakterističnih točaka lica. Opisani su procesi manipulacije ulaznog skupa podataka. Opisana je arhitektura i najbitniji dijelovi konvolucijske neuronske mreže. Prikazano je treniranje modela i objašnjeni parametri treniranja. Uz rad izrađen je i programski kod koji prolazi kroz svaki korak opisan u radu. Na samom kraju prikazani su dobiveni rezultati treniranja.

Ključne riječi: računalni vid, točke lica, konvolucijske neuronske mreže, pretprocesiranje podataka, arhitektura neuronskih mreža, duboko učenje

Summary

System for facial keypoints detection

This work describes the entire process of training a model for facial landmark detection. It describes the processes of manipulating the input dataset. Explains the architecture and the most important components of the convolutional neural network. The training of the model is demonstrated, and the training parameters are explained. In addition to the paper, a source code is provided and goes through each step described in the paper. The obtained training results are presented at the end.

Keywords: computer vision, facial landmarks, convolutional neural networks, preprocessing data, neural network architecture, deep learning

Privitak

Na sljedećoj poveznici: <https://github.com/tonipolanec/Facial-landmarks-detection-system> može se pronaći izvorni kod i datoteka za demonstraciju u Jupyter bilježnici.

Demonstracijska datoteka pokreće se na Python 3.8+ verziji i potrebno je imati instalirane sljedeće module:

- Pandas
- Numpy
- Keras
- Scikit-learn
- Matplotlib
- Pickle

Demonstracijska datoteka nalazi se pod nazivom: *project_demo.ipynb*. Za rad je potrebno preuzeti skup podataka *face_images.npz* sa poveznice [1]. Provjeriti prisutnost demo modela *model_demo.h5* i *history_demo* datoteke za prikaz grafa učenja spomenutog modela.

Otvoriti *project_demo.ipynb* i postupiti po uputama.