

SVEUČILIŠTE U ZAGREBU  
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 000

# **Prepoznavanje emocija iz izraza lica pomoću strojnog učenja**

Matej Ciglencečki

Zagreb, lipanj 2020.

*Umjesto ove stranice umetnite izvornik Vašeg rada.  
Da bi ste uklonili ovu stranicu obrišite naredbu \izvornik.*

*Zahvaljujem se mentoru na izdvojenom vremenu kojim mi je pomogao sa korisnim savjetima i uvidima tijekom pisanja ovog završnog rada*

# SADRŽAJ

<b>1. Uvod</b>	<b>1</b>
<b>2. Podatkovni skup</b>	<b>2</b>
2.1. Uvod . . . . .	2
2.2. Extended Cohn-Kanade podatkovni skup . . . . .	2
2.3. Ručno generirani podatkovni skup . . . . .	3
2.4. Priprema podatkovnih skupova . . . . .	4
2.4.1. Priprema Cohn-Kanade podatkovnog skupa . . . . .	4
2.4.2. Priprema Google podatkovnog skupa . . . . .	7
2.4.3. Objedinjavanje podatkovnih skupova . . . . .	10
<b>3. Treniranje</b>	<b>12</b>
3.1. Duboke neuronske mreže za analizu slika . . . . .	13
3.1.1. Neuron . . . . .	13
3.1.2. Aktivacijska funkcija . . . . .	14
3.1.3. Sloj neurona . . . . .	15
3.1.4. Neuronska mreža . . . . .	15
3.1.5. Funkcija gubitka . . . . .	17
3.1.6. Treniranje neuronske mreže . . . . .	18
3.1.7. Rezidualna neuronska mreža . . . . .	19
3.1.8. Prijenosno učenje . . . . .	23
3.2. Implementacija treniranja u PyTorch-u . . . . .	24
3.2.1. Priprema podataka . . . . .	24
3.2.2. ResNet 50 . . . . .	28
3.2.3. Hiperparametri . . . . .	28
<b>4. Evaluacija</b>	<b>30</b>
4.0.1. Konfuzijska matrica . . . . .	31

<b>5. Zaključak</b>	<b>32</b>
<b>Literatura</b>	<b>33</b>

# 1. Uvod

Većina komunikacije među ljudima je neverbalna. Ton glasa, geste, govor tijela i izrazi lica često vjernije prenose informacije od riječi. Običaji, način govora i kontekst trenutačne situacije govor čine podložnim mnoštvu različitih interpretacija dok je neverbalna komunikacija intuitivnija i univerzalnija. Ljudski dojmovi o drugim ljudima stvoreni su iz naoko suptilnih ali itekako važnih oblika neverbalne komunikacije. Saznanja i objašnjenja kognitivnih procesa iza ovih krajnje uobičajenih pojava još uvijek se aktivno istražuju.

Strojno učenje se zadnjih godina počelo primjenjivati na probleme poput razumijevanje govora, raspoznavanje objekata na slikama i ostalim područjima koje su za čovjeka intuitivne i lake. Računalni vid kao novija grana znanosti u nekoliko godina naišla je na mnoštvo otkrića i inovacija. Susret sa računalnim vidom događa se gotovo svakodnevno, od kamera i sustava sposobnih za prepoznavanje lica koji su se počeli implementirati od strane mnogih država do personaliziranih filtara za lica popularne na društvenim mrežama. Jedan od takvih izazova računalnog vida je raspoznavanje ljudskih emocija na temelju izraza lica. Iako uporaba ove tehnologije još nije izašla u širu primjenu trenutno već postoje slučajevi gdje se ova tehnologija pokazala korisnom. Automobil s mogućnošću prepoznavanjem emocija može upozoriti vozača kad je umoran ili pod većim nesvjesnim stresom kako ne bi dovodio sebe i druge sudionike u promet u opasnost. Raspoznavanje emocija nalazi primjeni i u razgovorima za posao. Kvalitetno treniran model objektivniji je od ljudskog ispitivača koji može imati subjektivne predrasude zbog čega u nekim slučajevima model može točnije procijeniti emocije osobe koju se ispituje, rezultirajući u bolji uvid u ponašanje i karakteristike osobe. Nadalje, ovakva se tehnologija može koristiti i u istraživanju tržišta. Mišljenjem korisnika o proizvodima dobivena putem razgovora i anketa često su netočna ili iskrivljena zbog nedostatka spomenute neverbalne komunikacije. Analiza emocija korisnika za vrijeme interakcije s proizvodom puno je reprezentativniji pokazatelj korisnikovog mišljenja.

## **2. Podatkovni skup**

### **2.1. Uvod**

Podatkovni skup sastavni je dio u izvedbi treniranja modela. Treniranje se svodi na ulazne i izlazne podatke gdje su ulazni podaci podskup podatkovnog skupa. Korišteni podatkovni skupovi su Cohn-Kanade (CK) i skup slika preuzetih sa Google-a na temelju ključne riječi. Podatkovni skup dijelimo na dva djela. Skup za treniranje i skup za testiranje. Svi podaci koji su u skupu za treniranje iskorištavaju se za treniranje i optimiziranje odabranog modela dok se skup za testiranje koristi samo za evaluaciju točnosti treniranog modela. 80% nasumičnih slika odabrane su za treniranje a ostalih 20% koristi se za evaluaciju.

### **2.2. Extended Cohn-Kanade podatkovni skup**

Cohn-Kanade podatkovni skup sastoji se od 593 sekvenci slika od 123 subjekta (osoba). Pojedina sekvenca sastoji se od 10 do 60 slika. Početna slika u sekvenci je neutralna emocija dok je zadnja slika vrhunac izraza emocije. Subjekti na slikama imaju od 18 do 50 godina, 69% su žene, 81% euro-Amerikanci i 6% su subjekti ostalih rase. Rezolucija pojedine slike iznosi 640x480 ili 640x490 piksela u 8-bitnom crno-bijelom ili 24 bitnom puno-bojnom formatu[6].

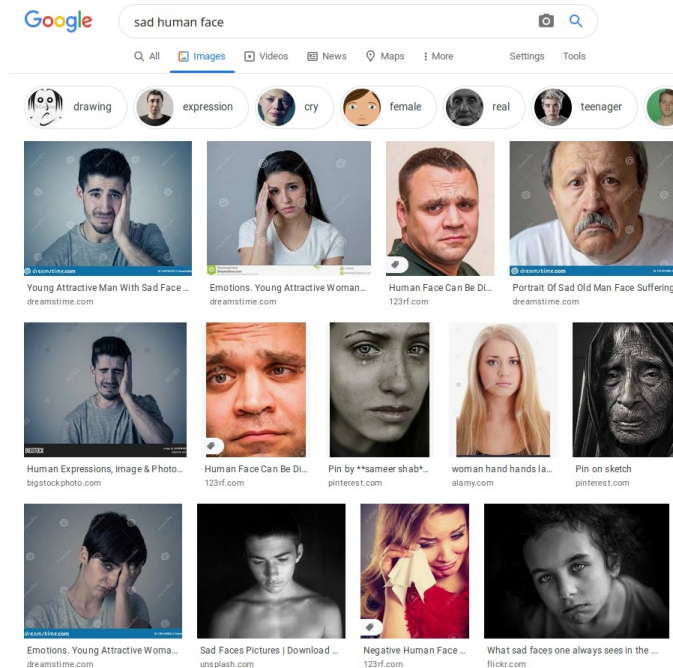


**Slika 2.1:** Primjer slike iz CK+ podatkovnog skupa

### **2.3. Ručno generirani podatkovni skup**

Slike CK+ podatkovnog skupa slikane su u istom okruženju (ista prostorija u kojoj se slikaju subjekti, ista kamera, slična svjetlina slike...). Nedostatak raznolikosti među slikama stvara potrebu za uvođenjem apstraktnijih slika ljudskih lica ne bi li model klasificirao emocije ljudskih lica koja nisu slična samo CK+ podatkovnom skupu. Zbog toga se uvodi podatkovni skup Google slika. Google omogućuje pretraživanje slika po zadanom upitu te dodatnim parametrima koji olakšavaju pronalaženje ljudskog lica koji predstavlja određenu emociju. Npr. pronalazak ljudskih lica koji iskazuju tužnu emociju mogao bi biti "sad human face" sa tipom Google slike "lice". Rezultati tog upita prikazani su na slici 2.2.





**Slika 2.2:** Rezultati Google upita "sad human face"

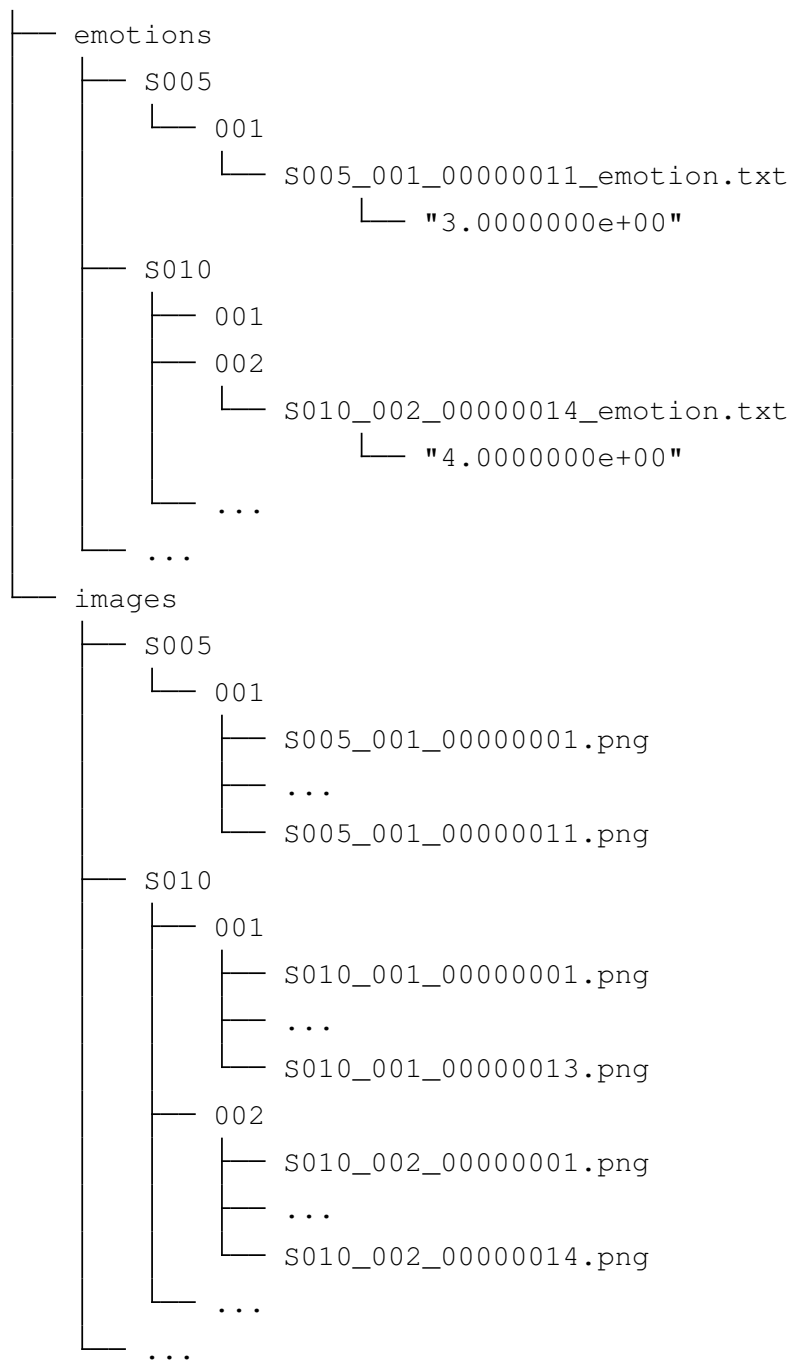
Rezultati ovog upita su slike veće raznolikosti (različiti scenarij i drugačiji kut gledanja) što pridonosi apstrakciji emocije u ukupnom podatkovnom skupu. Dobivene slike ne odgovaraju uvijek nužno zadanom upitu zbog čega je potrebno dodatno provjeriti valjanost pojedine slike. Proces filtriranja objašnjen je u poglavlju 2.4.2

## 2.4. Priprema podatkovnih skupova

### 2.4.1. Priprema Cohn-Kanade podatkovnog skupa

#### Struktura podataka

Dijelovi podatkovnog skupa značajni za treniranje dijele se na direktorij sekvence i emocije. Svaki subjekt (npr. S005) ima svoj direktorij u kojem se nalaze pod direktoriji za emociju koju je subjekt odglumio (npr. 001, 002...) a u njemu se nalaze sekvence. Za 327 sekvenca postoji odgovarajuća emocija koja je definirana samo za krajnju sliku sekvence i njezina putanja je definirana jednako kao i za sekvencu.



**Slika 2.3:** Struktura podataka CK+ podatkovnog skupa

### Obrada podataka

Jedina slika u sekvenci za koju je definirana emocija je zadnja slika, što znači da je potrebno je potrebno odrediti vektor emocije ostalih slika u sekvenci na temelju krajnje vrijednosti emocije. Emocija za pojedinu sliku definirana je kao red emocija. Svaki in-

deks reda predstavlja emociju kojih ima kojih ima 8. Indeks pojedine emocije definiran je na slici 2.4 a vrijednost na indeksu predstavlja intenzitet emocije

```
emocije = {
    1: "neutral",
    2: "anger",
    3: "contempt",
    4: "disgust",
    5: "fear",
    6: "happy",
    7: "sadness",
    8: "surprise",
}
```

**Slika 2.4:** Deklaracija emocija

Znajući da je emocija za početnu sliku neutralna a za krajnju maksimalna sekvencijska emocija, emocije za ostale slike dodijeljene su linearno na način da se odredi intenzitet neutralne emocije 2.1 i sekvencijske emocije 2.2 gdje je  $n$  ukupan broj slika u sekvenci a  $i$  slika kojoj se određuje vektor emocije.

$$p_n = \frac{i - 1}{n - 1} \quad (2.1)$$

$$p_s = 1 - p_n \quad (2.2)$$

$$p_s + p_n = 1$$

**Primjer vektora emocije za sekvencu "disgust" koja sadrži 10 slika:**

broj slike	vektor emocije
$i = 1$	[1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
...	...
$i = 6$	[0.4, 0.0, 0.0, 0.6, 0.0, 0.0, 0.0, 0.0]
...	...
$i = n = 10$	[0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0]
$i$	$[p_n, 0.0, 0.0, p_s, 0.0, 0.0, 0.0, 0.0]$

**Slika 2.5:** Vektor emocije za pojedinu pojedinu sliku u sekvenci u CK podatkovnom skupu

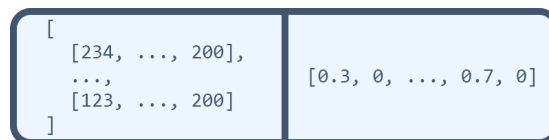
## Spremanje slike i vektora emocije u .npy podatak

Za daljnje korištenje svaka slika i njezin vektor emocije će pretvorena u numpy red [2] i spremljen kao .npy podatak. Prvi element numpy reda je slika a drugi je odgovarajući vektor emocije. Ovime je osigurano da izračun emocija za pojedinu sliku je izračunat samo jedanput što će smanjiti vrijeme potrebno za treniranje. Nakon provođenja transformacije podataka u .npy podatak ukupan broj slika i pripadajućih vektora emocija iznosi 5703.

**.npy**



(a) Struktura .npy podatka



(b) Sadržaj .npy podatka

**Slika 2.6:** .npy podatak

## 2.4.2. Priprema Google podatkovnog skupa

### Prikupljeni podaci

Umjesto preuzimanja jedne po jedne slike korištena je google-images-download skripta [8] koja preuzima sve moguće slike na temelju zadanog upita. Upiti korišteni za pronalaženje odgovarajućih emocija nalaze se na slici 2.7 a svi su popraćeni dodatnim "Google search" parametrom koji pretražuje slike samo ljudskih lica. Svaka slika spremljena je u direktoriji čije je ime upit koji je bio korišten prilikom preuzimanje te slike.

"sad human face"  
"neutral human face"  
"neutral expression"  
"angry human face"  
"angry expression"  
"contempt "  
"fear human face"  
"fear expression"  
"surprise human face"  
"surprise expression"  
"disgusted human face"  
"disgust expression"

**Slika 2.7:** Upiti za preuzimanje ljudskih lica sa Google-a

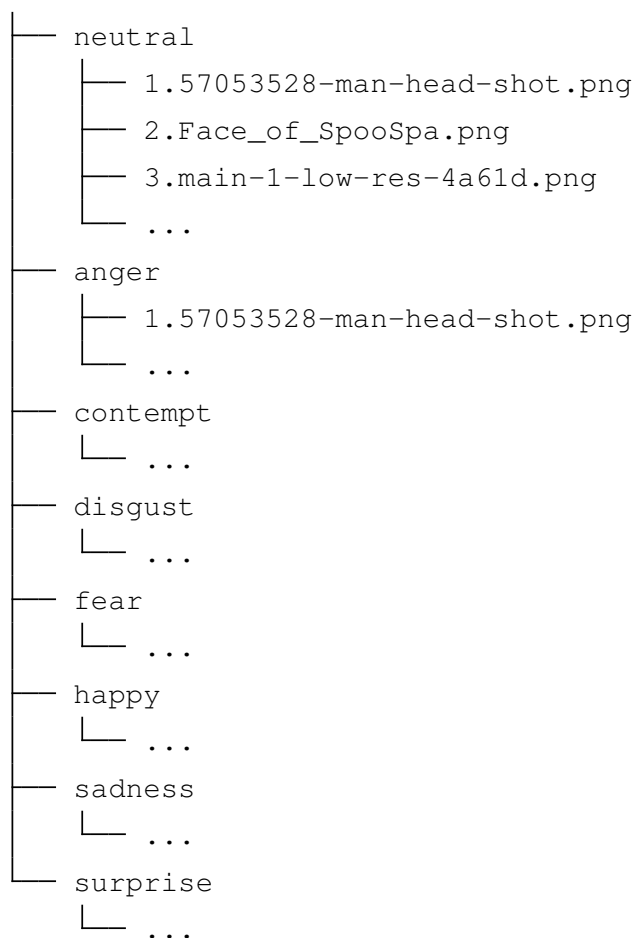
Nakon preuzimanja svih mogućih slika potrebno je ručno proći kroz svaki direktorij svakog upita i izbaciti slike koje ne zadovoljavaju sljedeće uvjete

- Na slici se nalazi samo jedno ljudsko lice
- Na slici se nalazi ljudsko lice čija emocija odgovara upitu pomoću kojeg je slika preuzeta
- Veličina slike je manja od 20MB
- Rezolucija slike je veća od 20px po duljini i visini
- Slika nije duplikat prethodno viđene slike
- Slika nije dio CK+ podatkovnog skupa

Uklanjanjem slika koje ne zadovoljavaju bilo koje od navedenih uvjeta dobiven ukupan broj slika povoljnih za treniranje iznosi 3160 a njihova ukupna veličina je 2,3GB.

### **Struktura podataka**

Nakon filtriranja slika nepogodnih za treniranje potrebno je objediniti upite čiji su rezultati ljudska lica efektivno istih emocija. Primjer takva dva upita su "neutral human face" i "neutral expression". Nakon objedinjavanja rezultata upita i preimenovanja direktorija stvorena je struktura podataka dana na slici 2.8



**Slika 2.8:** Struktura podataka Google podatkovnog skupa

## Obrada podataka

Obrada Google podatkovnog skupa bit će manje zahtjevna od CK+ podatkovnog skupa zbog toga što će se vektor emocije za pojedinu sliku odrediti samo na temelju upita korištenog za preuzimanje slike. Rezultat svakog vektora emocije bit će vektor dobiven metodom "One hot encoding". "One hot encoding" je metoda dodjele binarne vrijednosti za kategoriju u koju neki uzorak pripada ili ne pripada[7]. Ovom metodom uzorak (slika) može pripadati samo jednoj kategoriji (emocija). Slika koja pripada određenoj emociji za tu će emociju imati vrijednost 1 a za sve ostale 0. Na slici 2.9 prva kolumna označava ime direktorija u kojem se slike nalaze, druga kolumna predstavlja vektor emocije koje će slike u direktoriju poprimiti.

emocija (ime direktorija)	vektor emocije
neutral	[1,0,0,0,0,0,0,0]
anger	[0,1,0,0,0,0,0,0]
contempt	[0,0,1,0,0,0,0,0]
disgust	[0,0,0,1,0,0,0,0]
fear	[0,0,0,0,1,0,0,0]
happy	[0,0,0,0,0,1,0,0]
sadness	[0,0,0,0,0,0,1,0]
surprise	[0,0,0,0,0,0,0,1]

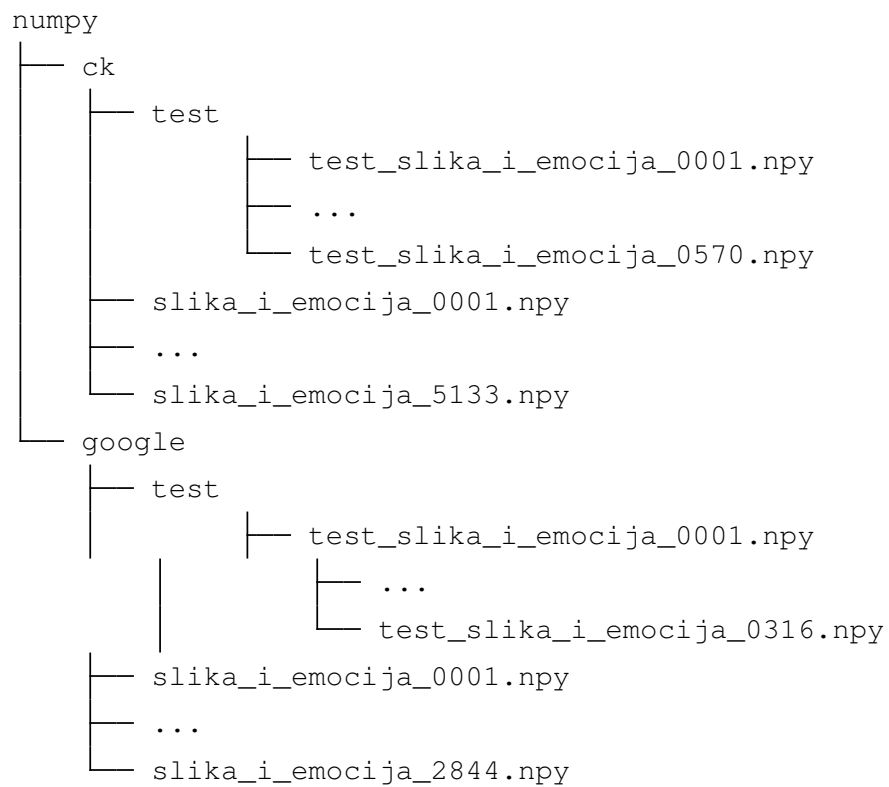
**Slika 2.9:** Vektor emocije za pojedinu emociju u Google podatkovnom skupu

### **Spremanje slike i vektora emocije u .npy podatak**

Nakon dodjele vektora emocije za pojedinu sliku potrebno je pretvoriti sliku i vektor emocije u .npy podatak. Kako se radi o slici i vektoru emocije, postupak pretvorbe slike i vektora emocije u pojedinačni .npy podatak jednak je kao i kod CK+ podatkovnog skupa definiranom u poglavlju 2.4.1.

### **2.4.3. Objedinjavanje podatkovnih skupova**

Nakon obrade CK+ i Google podatkovnog skupa svi .npy podaci bit će spremljeni u direktoriji "ck" ili "Google" ovisno o tome iz kojeg je podatkovnog skupa slika dobivena. Struktura svih .npy podataka prikazana je na slici 2.10. Ovom strukturom moguće je definirati udio svakog podatkovnog skupa koji će biti korišten za treniranje modela. Poslije podjele u zasebne direktorije potrebno je odvojiti dio .npy podataka u test pod-direktoriji. U njemu će se nalaziti 10% svih .npy podataka koji će modelu biti neviđeni prilikom treniranja. Tih 10% .npy podataka koristit će se pri evaluaciji točnosti modela.



**Slika 2.10:** Struktura .npy podataka

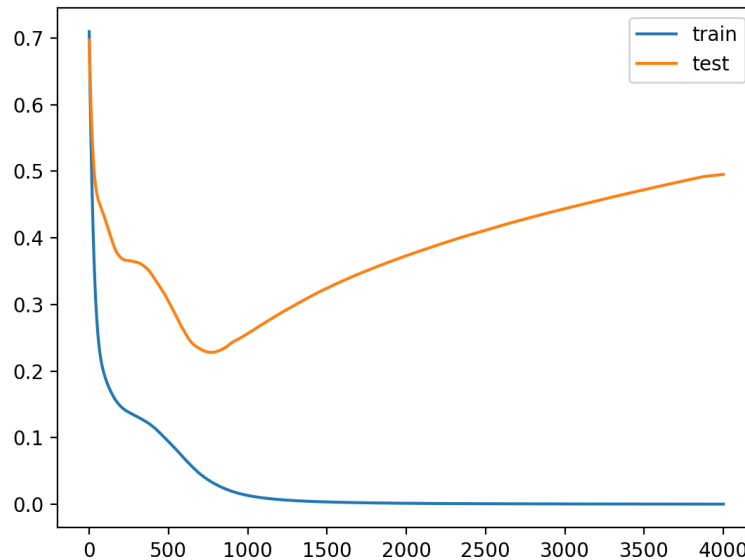


### 3. Treniranje

Treniranje je proces u kojem model postupno mijenja svoje parametre ne bi li došao do idealnih parametara, čime bi model optimalno služio onome čemu je namijenjen. U slučaju klasifikacije za treniranje je potrebno imati skup podataka nad kojim će se provoditi treniranje i labelle tih podataka koje govore klasu podataka. U slučaju klasifikacije emocija, skup podataka nad kojim se trenira su slike ljudskih lica dok je klasa emocija koja je prikazana na pojedinoj slici. Kad treniranje započne, u model se pošalju podaci za koje model pokušava predvidjeti koje su klase ubačeni podaci. Nakon toga potrebno je usporediti predviđanja klase koje je stvorio model sa pravim klasom. Pogrešku koju je model napravio prilikom predviđanja potrebno je javiti modelu ne bi li ispravio parametre. Mijenjanjem modela parametara model postaje sve bolji za predviđanje predviđanje rezultata na skupu za treniranje. Bitno je naglasiti da daljnjim treniranjem model ne postaje bolji za općeniti slučaj predviđanja. Treniranjem model se prilagođava podatkovnom skupu za treniranje. Podatkovni skup za treniranje je podskup svih podataka za koje je model namijenjen a to znači da skup za treniranje nije nužno najbolja mjera općenitog slučaja za podatke.

Treniranje modela nakon određenog trenutka može pogoršati sposobnost modela da predviđa klase na općenitom slučaju. Uzrok toga je što treniranjem nakon određenog trenutka model ima veću točnost na skupu za treniranje ali točnost na općenitim, neviđenim slučajevima postaje sve manja. Zbog toga je potrebno pronaći optimalan trenutak u kojem treba prekinuti treniranje modela. Prekidanje treniranja modela nije moguće odrediti pomoću točnosti modela na skupu za treniranje jer ta točnost neprestano raste daljnjim treniranjem. Zbog toga je potrebno uvesti validacijski skup koji neće biti korišten prilikom treniranja. Validacijski skup o tom slučaju igra ulogu neviđenih podataka nad kojima se mjeri točnost. Prilikom treniranja točnost nad validacijskim skupom će rasti do optimalnog trenutka gdje je greška predviđanja najmanja. U tom trenutku točnost nad validacijskim skupom će biti maksimalna a daljnjim treniranjem točnost će se smanjivati jer model postaje previše prilagođen skupu za treniranje (engl. *overfitting*). Overfitting prikazan je na slici 3.1. Slika prikazuje overfitting zbog

toga što prilikom treniranja greška validacija poslije optimalne točke greška nad validacijskim skupom postaje sve veća i veća.



**Slika 3.1:** Primjer overfitting-a

## 3.1. Duboke neuronske mreže za analizu slika

### 3.1.1. Neuron

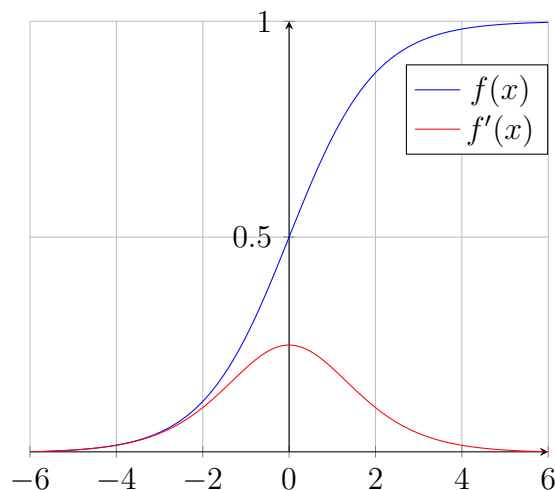
Neuron je osnovni dio neuronske mreže. Svaki neuron na svoj ulaz prima vektor  $x$  varijabli za koje neuron predviđa  $\hat{y}$  izlaz. Skup  $x$  vrijednosti pomnožen je sa vektorom  $w$  koji predstavlja težine (engl. *weights*) i zbrojen sa vrijednošću  $b$  koja predstavlja sklonost (engl. *bias*). Težine neurona određuju povezanost između trenutnog i prethodnog neurona. Težištima neuroni utječu jedne na druge i određuju koliko su oni zavisni na prethodnom neuronu i za koje  $x_i$  varijable. Parametri  $w$  i  $b$  su promjenjivi i mijenjaju se prilikom treniranja a  $g$  je aktivacijska funkcija.

$$z = w_1x_1 + w_2x_2 + \dots + w_nx_n = \mathbf{w}^T \cdot \mathbf{x} \quad (3.1)$$

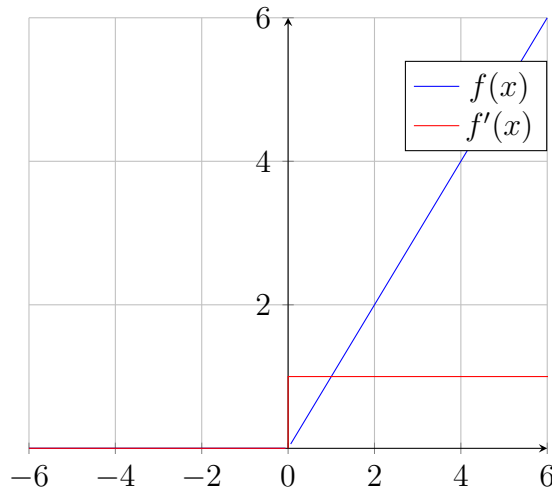
$$\hat{y} = g(z) \quad (3.2)$$

### 3.1.2. Aktivacijska funkcija

Nakon zbrajanja unutar neurona, izlaz neurona potrebno je provesti kroz aktivacijsku funkciju koja određuje koliko je signal tog neurona bio zastupljen u ukupnom izlazu, tj. koliko je neuron "aktiviran". Aktivnost predstavlja vezu između dva neurona a vrijednost aktivnosti govori koliko će rezultat jednog neurona utjecati na rezultat drugog. Cilj koji se postiže tom funkcijom je izlaz neurona bude postane nelinearan čime cijela neuronska mreža postaje nelinearna. Kad bi neuronska mreža uistinu bila linearan model njezina bi bila jednako dobra kao i regresijski model. Uvođenjem aktivacijske funkcije neuronska mreža može aproksimirati funkcije više reda što znači da su granice klasifikacija te neuronske mreže podijeljene granicom odluke (engl. *decision boundary*) koja nije linearna. Primjer takvih funkcija je sigmoid prikazan na slici 3.2 i rektifikacijska funkcija ReLU (engl. *Rectifier*) prikazana na slici 3.3. Prilikom aktivacije neurona koristiti će se rektifikacijska funkcija zbog toga što je opće primjenjiva, daje dobre rezultate i računanje je znatno zbog jednostavnosti funkcije.



Slika 3.2: Sigmoid



**Slika 3.3:** Linearna rektifikacijska funkcija (engl. *ReLU*)

### 3.1.3. Sloj neurona

Sloj neurona sastoji se od skupa neurona. Matematički, sloj neurona sastoji se od vektora  $\mathbf{a}_l - 1$  koji označava aktivaciju sloja neurona i vektora  $\mathbf{w}$  koji pred. U slučaju prvog sloja vrijedi  $\mathbf{a} = \mathbf{x}$ . Račun izlaza sloja neurona računa se slično kao i kod pojedinog neurona. Za svaki sloj neurona  $l$  izlaz  $z$  jednak je umnošku prethodne aktivacije  $\mathbf{a}_{l-1}$  sa trenutnim težinama sloja neurona  $\mathbf{w}_l$  zbrojen sa sklonošću  $\mathbf{b}_l$

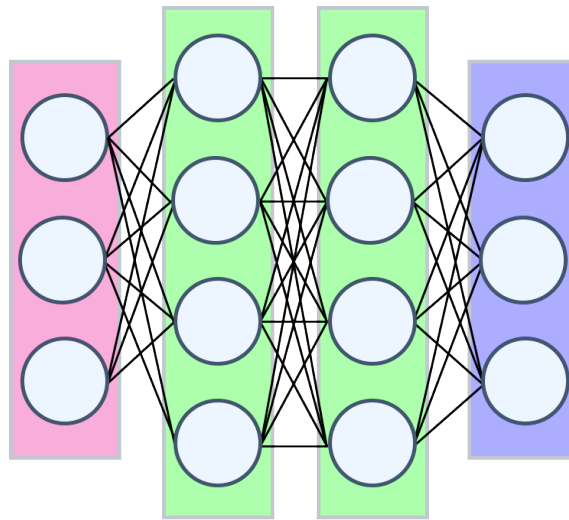
$$z_l = \mathbf{w}_l \cdot \mathbf{a}_{l-1} + \mathbf{b}_l \quad (3.3)$$

$$\hat{y} = g(z) \quad (3.4)$$

### 3.1.4. Neuronska mreža

Neuronska mreža sastoji se od više slojeva neurona koji su međusobno povezani. Slojevi su povezani tako što je svaki neuronski sloj razine  $l$  povezan sa svakim neuronskim slojem razine  $l + 1$ . Povezani su na način da se izlaz neuronskog sloja  $l$  dovodi na ulaz neuronskog sloja  $l + 1$ . Prije dovođenja na ulaz  $l_{i+1}$  izlaz sloja  $l_i$  proveden je kroz aktivacijsku funkciju čime je izlaz aktiviran. Ovaj algoritam prijenosa podataka s jednog sloja na drugi zove se *feedforward* algoritam. Slojevi se dijele u tri različite grupe: ulazni, skriveni i izlazni sloj. Ulazni sloj je prvi sloj na koji se dovodi ulazni podatak, slika. Izlazni sloj je sloj koji na izlazu daje klasifikaciju ulazne slike, u ovom slučaju vektor predviđenih emocija. Skriveni slojevi su među-slojevi koji obrađuju po-

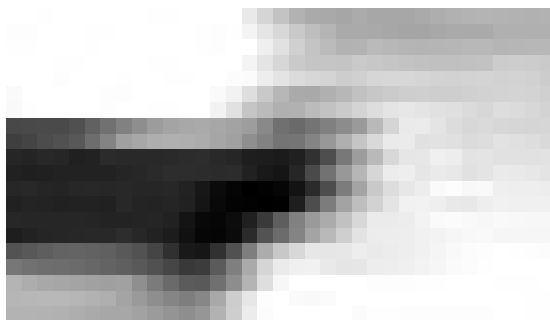
datke prethodnog izlaznog sloja neurona. U sredinom skrivenog područja obrađuje se najviše informacija



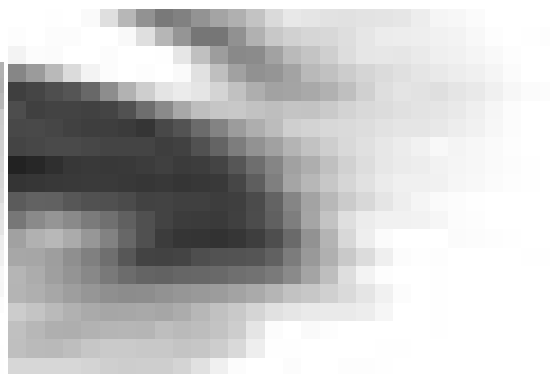
ulazni sloj   skriveni slojevi   izlazni sloj

**Slika 3.4:** Različiti neuronski slojevi

Što je neuronski sloj dublje razine to je apstrakcija podataka koju on računa veća. Izlaz skrivenog  $l_i$  sloja poslat će se na ulaz sljedećeg skrivenog sloja  $l_{i+1}$  a tim prijelazom apstrakcija podataka će biti uvećana. Uloga svakog neuronskog sloja je da obrađuje podatke na razini apstrakcije definiranju njegovom dubinom. Većim brojem slojeva neurona stvara se dublja neuronska mreža koja može računati na većoj razini apstrakcije. Interpretacija razine apstrakcije u slučaju slika svodi se na promatranje različitih dijelova slike različite veličine. Iako se promatraju različiti dijelovi slike, međuzavisnost neuronskih slojeva omogućava neuronskoj mreži da zna koje je svojstvo slike proizašlo iz prethodnog svojstva slike. Primjer bi bio pronalaženje ruba usnice i ruba oka na slici.



(a) Manji dio usnice, rub usnice



(b) Manji dio oka, rub oka

**Slika 3.5:** Slični rubovi djelova lica

Iako ta dva rubovi izgledom mogu biti slični bitna spoznaja za neuronsku mrežu je da jedan rub proizlazi iz slike usnice a drugi iz slike oka. Tako na primjer rub usnice usmjeren prema gore ukazuje na emociju sreće, dok to za rub oka ne mora biti slučaj. Ovakvo širenje konteksta bitno je za ispravno treniranje neuronske mreže.



(a) Veći dio usnice



(b) Veći dio oka

**Slika 3.6:** Dijelovi lica

### 3.1.5. Funkcija gubitka

Funkcija gubitka (engl. *Loss function*) je funkcija koja govori koliko je neuronska mreža daleko ili blizu rezultata koji se želi postići. Rezultat funkcije gubitka bit će pogreška koju je mreža napravila prilikom predviđanja vektora emocije određene slike. Taj rezultat bit će korišten za ispravljanje parametara neuronske mreže. Funkcija gu-

bitka treba ukazati na kojim je predviđanjima neuronska mreža pogriješila i za koliko. Križni entropijski gubitak (engl. *Cross Entropy Loss*) je funkcija gubitka dana sa 3.5 koja će biti korištena prilikom određivanja gubitka. Funkcija prima dva parametra, ispravna vrijednost označenu kao vektor  $\mathbf{x}$  je i predviđenu vrijednost označenu kao vektor  $\mathbf{y}$ . U slučaju predviđanja emocija  $\mathbf{x}$  je stvaran vektor emocije za pojedinu sliku dok je  $\mathbf{y}$  previđen vektor emocije stvoren na izlazu neuronske mreže. Vektor  $\mathbf{w}$  predstavlja težišta za pojedini element vektora  $\mathbf{x}$  i  $\mathbf{y}$ . Elementi vektora  $\mathbf{w}$  su  $w_1, w_2, \dots, w_n$  gdje svaki  $w_i$  predstavlja koeficijent koji će biti pomnožen sa rezultatom gubitka. Razlog uvođenja težišta je neravnomjerna zastupljenost klasa u podatkovnom skupu za treniranje. Klase s manjim brojem uzoraka imat će veće težište prilikom računanja gubitka i samim time će biti "bitnije" prilikom treniranja. Kazna pogreške predviđanja za klasu emocije manjeg brojem uzoraka bit će veća nego u slučaju ne korištenja težišta.

$$CrossEntropyWeighted(\mathbf{x}, \mathbf{y}, \mathbf{w}) = - \sum_i w_i (x_i \log(y_i)) \quad (3.5)$$

### 3.1.6. Treniranje neuronske mreže

Neuronska mreža predstavlja nelinearnu funkciju koja se sastoji od  $n$  varijabli a parametri su spomenuti parametri pojedinog neurona spomenuti na formuli ??, težišta neurona  $\mathbf{w}_2$  i sklonost neurona  $b$ . Mijenjanje tih parametara parametara na način da se rezultat funkcije gubitka smanjuje naziva se treniranje. Cilj treniranja je minimizirati gubitak tako da se pronađe lokalni minimum u  $n$  dimenzionalnom prostoru, gdje je broj  $n$  broj varijabli u neuronskoj mreži. Metoda pronalaska minimuma zove se spuštanje gradijentom (engl. *gradient descent*). Računanje spusta svodi se na izračun derivacije parametra neuronske mreže prema 3.6 gdje je greška neuronskog sloja  $E$ ,  $a$  aktivacijski izlaz i  $w$  težina sloja.

$$\frac{\partial E}{\partial w} = \frac{\partial E}{\partial a} \cdot \frac{\partial a}{\partial w} \quad (3.6)$$

Ovom jednadžbom dobiven je smjer negativnog gradijenta koji vodi ka minimumu  $n$  dimenzionalne plohe. Brzina kojom neuronska mreža konvergira prema minimumu označena sa  $lr$  zove se stopa učenja (engl. *learning rate*). Gradijent se računa prilikom svake iteracije treniranja, tj. nakon svakog dovođenja slike na ulaz mreže. Izračunati gradijent na sloju  $l_i$  pomnožen sa stopom učenja  $lr$  povratnom vezom se šalje na prethodni sloj  $l_{i-1}$ . Sloj  $l_{i-1}$  koristiti će dobivenu vrijednost da ažurira svoja težišta sljedećim izračunom:

$$w_{l_{i-1}} - lr \frac{\partial E_{l_i}}{\partial w_{l_i}} \rightarrow w_{l_{i-1}} \quad (3.7)$$

Ovaj postupak naziva se backpropagation i primjenjuje se na sve slojeve neurona u neuronskoj mreži prilikom svake iteracije treniranja mreže.

### 3.1.7. Rezidualna neuronska mreža

#### Motivacija

Problem prilikom treniranja duboke neuronske mreže je mogućnost nestajućeg gradijenta (engl. *vanishing gradient problem*) pri ranim neuronskim slojevima mreže. Prilikom backpropagation algoritma sloj  $l_i$  ažurirat će svoja težišta danom formulom 3.7. Bitno je naglasiti da vrijednost koja će ažurirati njegovo težište  $lr \frac{\partial E_{l_n}}{\partial w_{l_n}}$  je gradijent izračunat od svih prethodnih slojeva  $l_{i+1}, \dots, l_n$ . Ukoliko iznos gradijenta u više slučajeva bude manji od 1 težište neurona će s vremenom početi konvergirati prema 0. Što težište više konvergira prema 0 to je teže ažurirati njegovu vrijednost.

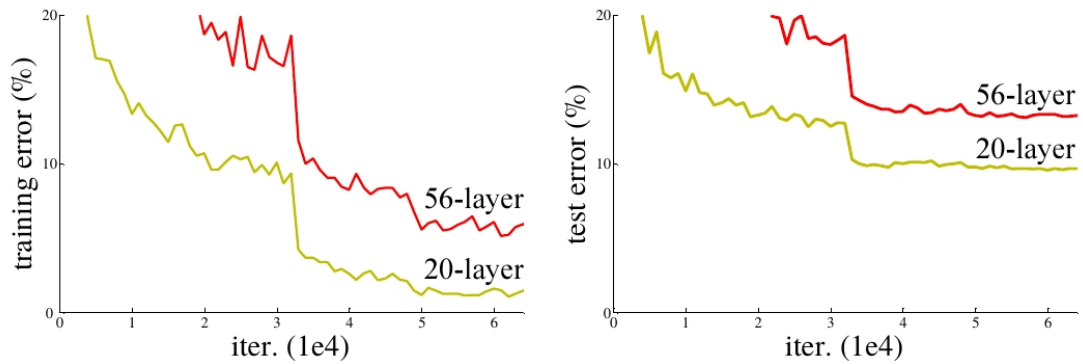
Prilikom treniranja neuronske mreže ulaz  $x$  koji prolazi kroz jedan neuronski sloj  $l$  bit će izračunat formulom 3.1. U nekim iteracijama treniranja poželjno je i izostaviti utjecaj nekog sloja  $l$  jer njegov utjecaj ne mora nužno biti potreban za iteraciju treniranja. Drugim riječima ulaz  $x$  prolazom kroz sloj  $l$  treba ostati nepromijenjen, čime bi izlaz  $y$  sloja  $l$  bio jednak  $x$ . Funkcija čija je vrijednost za argument  $x$  jednaka  $x$  zove se funkcija identiteta dana na 3.8.

$$f(x) = x \quad (3.8)$$

U praksi neuronski sloj ne pronalazi funkcija identiteta s lakoćom jer metoda pronalaska funkcije identiteta zahtjeva računalnu snagu koja je aproksimacija funkcije identiteta.

Opisan postupak je posebno problematičan kad je neuronska mreža velike dubine. Suprotno intuiciji, dodavanjem većeg broja slojeva na neuronsku mrežu gubitak neće uvijek nužno biti smanjen. Nakon određenog broja slojeva dodavanjem novih slojeva gubitak prilikom treniranja postaje sve veći a time točnost predviđanja manja. Primjer ovog problema dan je na slici 3.7.





**Slika 3.7:** Pogreška prilikom treniranja obične duboke neuronske mreže ovisno o dubini mreže [5]

### Rezidualni blok

Rješenje problema spomenutog u motivaciji može se izbjeći korištenjem rezidualnog bloka. Taj prijedlog dan je prema [5]. Umjesto aproksimiranja identifikacijske funkcije  $f(x) = x$  uvodi se izraz pomoću kojeg se funkcija identiteta lako izračuna.

$$f(x) := h(x) + x \quad (3.9)$$

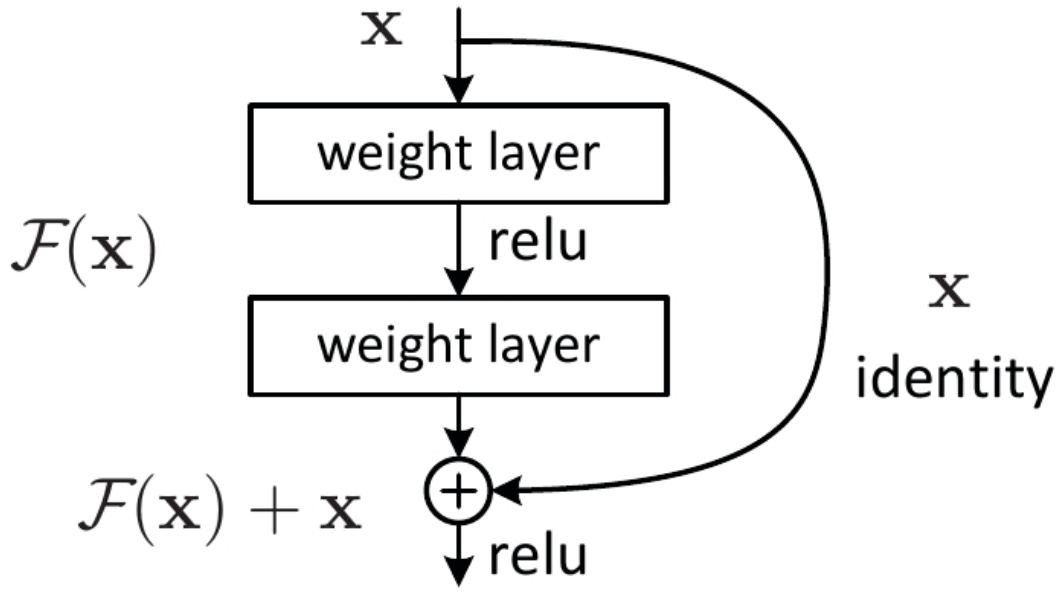
Sad je za neuronsku mrežu lako pronaći funkciju identiteta na način da se postavi  $f(x) = 0$  čime se dobiva:

$$h(x) = f(x) + x$$

$$h(x) = x$$

Ovime je osigurano da je originalan podatak  $x$  moguće proslijediti sljedećem neuronskom sloju  $l_{i+2}$  kroz neuronski sloj čak i ukoliko težišta  $w$  sloja  $l_i$  budu jednaka 0. Čak i ako neuronski sloj ne djeluje na podatak  $x$  on će ostati nepromijenjen čime je doprinos na buduće slojeve uvijek jednak ili bolji, što nije slučaj za obične duboke neuronske mreže. Ova činjenica omogućava dodavanje slojeva bez ikakvog gubitka točnosti.

Implementacija rezidualnog bloka prikazan na 3.8 ostvaruje se dodavanjem prečaca (engl. *skip connection*) između bloka  $l_i$  i bloka  $l_{i_2}$  koji služi da se podatak  $x$  koji se zbraja sa izlazom  $f(x)$ .



**Slika 3.8:** Pogreška prilikom treniranja obične duboke neuronske mreže ovisno o dubini mreže [5]

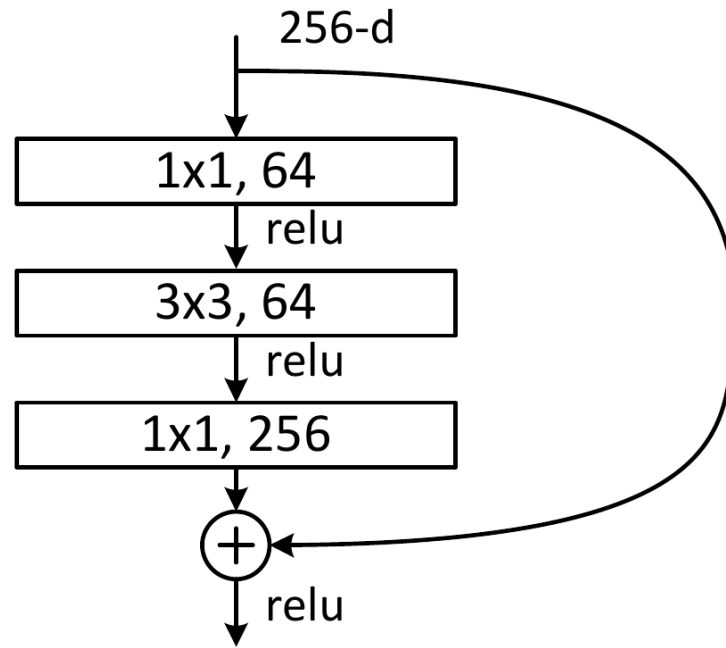
Primjer izračuna izlaza korištenjem rezidualnog bloka za aktiviran ulaz  $\mathbf{a}_{l_i}$ , ReLU aktivacijsku funkciju  $g$ , težište  $\mathbf{w}_{l_i}$  te prinos  $\mathbf{b}_{l_i}$  gdje  $l_i$  redni broj neuronskog sloja.

$$\begin{aligned} \mathbf{a}_{l_{i+2}} &= g(\mathbf{z}_{l_{i+2}} + \mathbf{a}_{l_i}) \\ \mathbf{a}_{l_{i+2}} &= g(\underbrace{\mathbf{w}_{l_{i+2}} \mathbf{a}_{l_{i+1}} + \mathbf{b}_{l_{i+2}}}_0 + \mathbf{a}_{l_i}) \\ \mathbf{a}_{l_{i+2}} &= g(\mathbf{a}_{l_i}), \quad g \text{ je aktivacijska funkcija ReLU} \\ \mathbf{a}_{l_{i+2}} &= \mathbf{a}_{l_i} \end{aligned}$$

Ovime je pokazano da se po potrebi ulazni podatak lako može dobiti i na izlazu bez kompliciranih računskih operacija čime se izbjegava problema nestajućih gradijenta uz smanjenje korištenje računalne snage.

### Arhitektura mreže ResNet50

Za stvaranje rezidualne neuronske mreže potrebno je ulančati više rezidualnih blokova. Konkretni rezidualni blok koji je korišten u ResNet50 arhitekturi prikazan je na slici 3.9. Cijela arhitektura ResNet50 mreže prikazana je na slici 3.10 označena sa stupcem "50-layer".



**Slika 3.9:** Rezidualni blok korišten u ResNet50 arhitekturi [5]

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
		3×3 max pool, stride 2				
conv2_x	56×56	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$
conv3_x	28×28	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 8$
conv4_x	14×14	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 23$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 36$
conv5_x	7×7	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		$1.8 \times 10^9$	$3.6 \times 10^9$	$3.8 \times 10^9$	$7.6 \times 10^9$	$11.3 \times 10^9$

**Slika 3.10:** ResNet50 arhitektura [5]

Podaci dovedeni na ulaz ResNet50 mreže prolaze kroz nju sljedećim redoslijedom:

- 1 sloj - Konvolucijski filter veličine  $7 \times 7$  sa veličinom pomaka 2 koji slijedi konvolucijskim filterom veličine  $3 \times 3$  uz maxpool metodu sa veličinom pomaka 2
- 9 slojeva
  - Konvolucijski filter  $1 \times 1$  ulaza veličine 64

- Konvolucijski filter  $3 \times 3$  ulaza veličine 64
  - Konvolucijski filter  $1 \times 1$  ulaza veličine 256
  - slojevi su ponovljeni 3 puta
- 12 slojeva
- Konvolucijski filter  $1 \times 1$  ulaza veličine 128
  - Konvolucijski filter  $3 \times 3$  ulaza veličine 128
  - Konvolucijski filter  $1 \times 1$  ulaza veličine 512
  - slojevi su ponovljeni 4 puta
- 18 slojeva
- Konvolucijski filter  $1 \times 1$  ulaza veličine 256
  - Konvolucijski filter  $3 \times 3$  ulaza veličine 256
  - Konvolucijski filter  $1 \times 1$  ulaza veličine 1024
  - slojevi su ponovljeni 6 puta
- 9 slojeva
- Konvolucijski filter  $1 \times 1$  ulaza veličine 512
  - Konvolucijski filter  $3 \times 3$  ulaza veličine 512
  - Konvolucijski filter  $1 \times 1$  ulaza veličine 2048
  - slojevi su ponovljeni 3 puta
- 1 sloj - zadnji sloj je sloj prosječnog sažimanja čiji je izlaz veličine 1000 na koji djeluje funkcija "softmax" koja izlazne veličine pojedine klase pretvara u vjerojatnost prisutnosti klase.

Ukupan broj navedenih slojeva je 50

### 3.1.8. Prijenosno učenje

Prijenosno učenje je tehnika u području strojnog gdje je naučenost modela za predviđanje jednog cilja prenesena na predviđanje drugog cilja. Potreba za ovom tehnikom javlja se iz nedovoljno velikog skupa podataka korištenog za treniranje. Zbog toga što podatkovni skup nije dovoljno velik, neuronska mreža neće biti dovoljno trenirana da predviđa emocija za neviđene slike. Unaprijed trenirana neuronska mreža ResNet 50, koja je trenirana na ImageNet podatkovnom skupu [4], omogućava mreži da od

početka prepoznaje bitne značajke neviđene slike. Time je izbjegnuta potreba za treniranjem modela od početka jer su početni neuronski slojevi mreže već dobro naučeni za prepoznavanje bitnih značajki u slici, poput istaknutih objekta i uočljivijih oblika.

Zadnji slojevi unaprijed trenirane ResNet 50 mreže mogu predviđati 1000 različitih značajki. Broj mogućih klasifikacijskih emocija je 8 zbog čega će biti potrebno dodati zadnji neuronski sloj na postojeću neuronsku mrežu tako da izlaz mreže bude dimenzije 8, a ne 1000 što odgovara broju različitih klasa korištenih u ImageNet podatkovnom skupu. Korištenjem većeg i apstraktnijeg podatkovnog skupa ova se tehnika može izostaviti.

## 3.2. Implementacija treniranja u PyTorch-u

U poglavlju 2.4 i 2.4.2 objašnjeno su iz oba podatkovna skupa dobiveni .numpy podaci koji sadrže podatke slike i odgovarajućeg vektora emocije te slike. Sljedeći korak je iskoristiti te podatke za treniranje neuronske mreže koja će klasificirati emocije neviđenih slika ljudskih lica.

### 3.2.1. Priprema podataka

.numpy podatke učitani su iz strukture definirane na 2.10. Parametrima GOOGLE\_TRAIN\_SPLIT i CK\_TRAIN\_SPLIT moguće je kontrolirati koji će udio slika iz pojedinog podatkovnog skupa biti zastupljen u ukupnom skupu za treniranje.

```
GOOGLE_TRAIN_SPLIT = 1
CK_TRAIN_SPLIT = 1
```

**Slika 3.11:** Udio .numpy podataka pojedinog podatkovnog skupa

Sljedeći korak je implementacija klase Dataset iz PyTorch biblioteke. Stvaranjem klase FERDataset iz klase Dataset poslana su tri argumenta: lokacija .numpy podataka i funkcija za transformaciju slika. FERDataset sad ima pristup lokaciji .numpy podataka i preko te lokacije će dohvaćati .numpy podatke prilikom treniranja.

```
class FERDataset(Dataset):
    def __init__(self, filepaths_numpy, transform_image=None):
        self.filepaths_numpy = filepaths_numpy
        self.transform_image = transform_image
```

**Slika 3.12:** Definiranje FERDataset inicijalizacije

Nakon dohvata .numpy podataka preko funkcije `__getitem__` potrebno ga je raspakirati u sliku lica i vektor emocije te slike. Ukoliko je zadana transformacija slike ona će se primijeniti nakon raspakiravanja .numpy podataka. Više o transformaciji slika bit će rečeno kasnije.

```
def __getitem__(self, idx):
    name_numpy = filepaths_numpy[idx]
    image, emotion = load_numpy(name_numpy)
    image = transform(image)
    return image, emotion
```

**Slika 3.13:** Pojednostavljeni dohvat i raspakiravanje .numpy podataka

## Podjela skupa

Nakon stvaranja objekta tipa `Dataset`, u ovom slučaju `FERDataset` potrebno je taj podatkovni skup podijeliti u skup za treniranje i skup za evaluaciju. Kako je taj posao već napravljen kao što je prikazano na slici 2.10 jedino što je potrebno je inicijalizirati zaseban skup za treniranje i evaluaciju kao što je prikazano na sljedećoj slici:

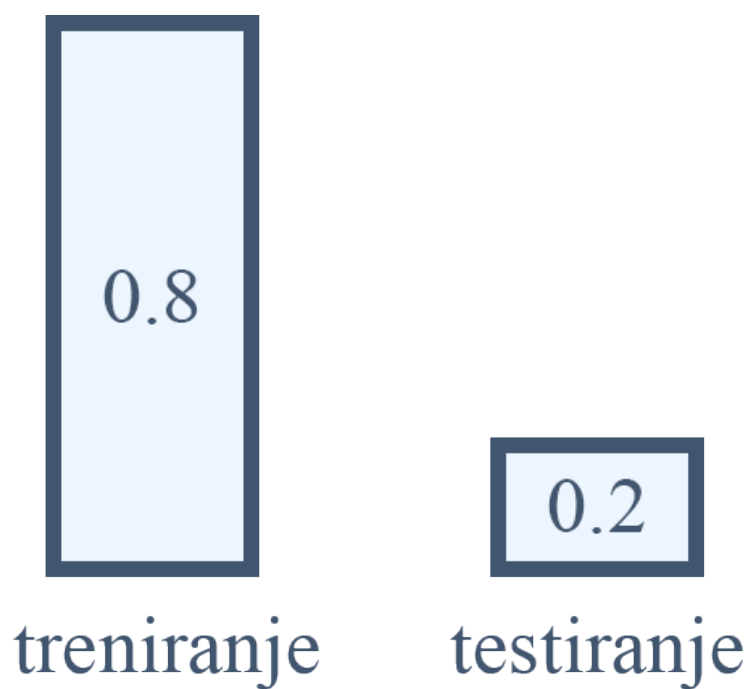
```
train_dataset = FERDataset(filepaths_numpy=FILEPATHS_NUMPY,
                           transform_image=transform_image_train)
test_dataset = FERDataset(filepaths_numpy=FILEPATHS_NUMPY_TEST,
                          transform_image=transform_image_val)
```

**Slika 3.14:** Implementacija FERDataset klasa



podatkovni skup

**Slika 3.15:** Vizualni prikaz cijelog podatkovnog skupa



**Slika 3.16:** Vizualni prikaz podijeljenog podatkovnog skupa

### Transformacija slike

Prije nego što se podaci dobiveni iz "train\_datseta" mogu slobodno koristiti za treniranje neuronske mreže potrebno ih je transformirati u pravilan oblik. Sliku ljudskog

lica potrebno je pretvoriti u PyTorch Tensor [3]. Pretvorbom podatka u ovaj tip omogućava PyTorch-u nastavljati koristiti ovaj podatak u svojoj domeni uz moguće korištenje PyTorch funkcija. Korak prije slanja podataka slike u model je normalizacija. Kako se koristi unaprijed trenirana neuronska mreža ResNet 50 potrebno je provesti normalizaciju koja je matematički ispravna nad skupom koji je korišten prilikom treniranja. Ta normalizacija dana je od strane ImageNet-a čiji je podatkovni skup korišten prilikom treniranja ResNet 50 neuronske mreže [1] a ona glasi:

```
transforms.Normalize(  
    mean = [0.485, 0.456, 0.406],  
    std = [0.229, 0.224, 0.225])
```

**Slika 3.17:** Implementacija FERDataset klasa

### Augmentacija slike

Dodatne i opcionalne transformacije su transformacije koje povećavaju ukupnu apstraktnost podataka (engl. *data augmentation*). Ove transformacije igraju ulogu simuliranja raznolikijeg podatkovnog skupa na način da rade manje promjene na pojedinoj slici. Primjerice, nasumično povećavanje svjetline slike ili nasumično okretanje slike za kut  $\alpha$ . Na ovaj način model će naučiti klasificirati slike i u okolnostima stvorenim okolnostima gdje je slika svjetlija ili blago zarotirana čime se proširuje primjena modela na veći skup različitih slika.

Augmentacija slika primjenjivati će se nasumično za svaku sliku, čime se osigurava da model ne bude pretreniran za slike iste svjetline ili slike na kojima je ljudsko lice jednakog nagiba. Također, sve slike ćemo postaviti na fiksnu veličinu iznosa 224x224 pixela zbog lakšeg računanja konvolucije.

Razlog broja `IMG_SIZE = 224` je uzastopna mogućnost djeljivosti sa brojem 2, pogodna za ResNet 50 neuronsku mrežu.

Sliku ćemo pretvoriti u crno-bijelu iz razloga što ne želimo da boje igraju ulogu u predviđanju emocija. Oblik, kontrast i rubovi na slici su dovoljno značajni za prepoznavanje ekspresije ljudskog lica.

Nasumično horizontalno zrcaljenje je još jedna od korisnih augmentacijskih transformacija koja će povećati apstrakciju. Model će bolje raditi za općenite slučajeve, neovisno o zrcaljenju kamere koja slika lice.

Transformacija i augmentacija slika u PyTorch-u implementira se na način da funkcija `"transform.Compose"` primi listu željenih transformacija koja se šalje u `FERData-`



set pod imenom "transform\_image". Augmentacija slike mora se obaviti prije normalizacije slike. Kod za potpunu transformaciju je sljedeći:

```
transform_image_train = transforms.Compose([
    transforms.Resize(IMG_SIZE),
    transforms.Grayscale(num_output_channels=3),
    transforms.RandomHorizontalFlip(),
    # transforms.RandomAffine(degrees=(-AUG_DEGREE, AUG_DEGREE)),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[
        0.229, 0.224, 0.225])
])
```

**Slika 3.18:** Implementacija FERDataset klasa

### 3.2.2. ResNet 50

Spomenutu unaprijed treniranu mrežu ResNet 50 potrebno je uvesti u projekt. Srećom PyTorch već ima ugrađenu unaprijed treniranu ResNet 50 mrežu koja se poziva na sljedeći način:

```
resnet50 = models.resnet50(pretrained=True, progress=True)
```

**Slika 3.19:** Inicijalizacija ResNet 50

### 3.2.3. Hiperparametri

#### Epoha

Broj epoha (engl. *epoch*) je broj koji predstavlja koji će se puta treniranje izvršiti nad cijelim skupom za treniranje. Odabrana vrijednost je 100.

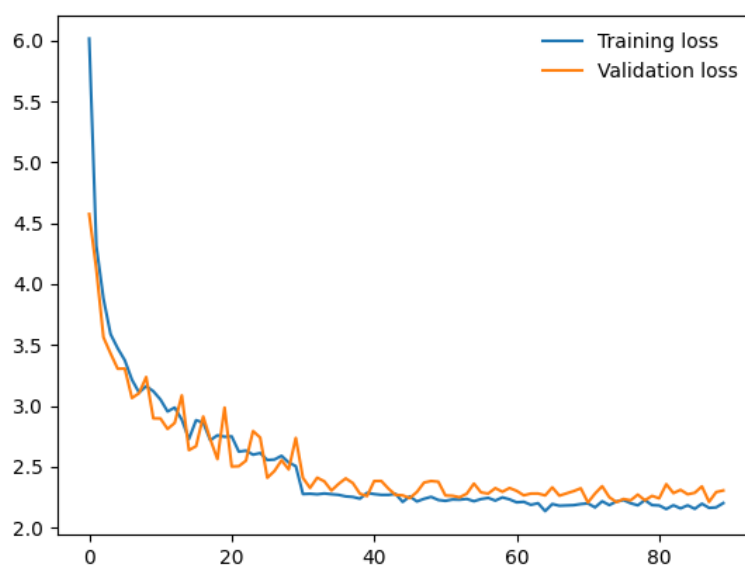
#### Veličina serija

Veličina serije (engl. *batch size*) je veličina koja predstavlja koliko se od jednom slika šalje neuronskoj mreži. Odabrana vrijednost iznositi će 16. To znači da će model u jednoj seriji predviđati emocije za 16 slika.

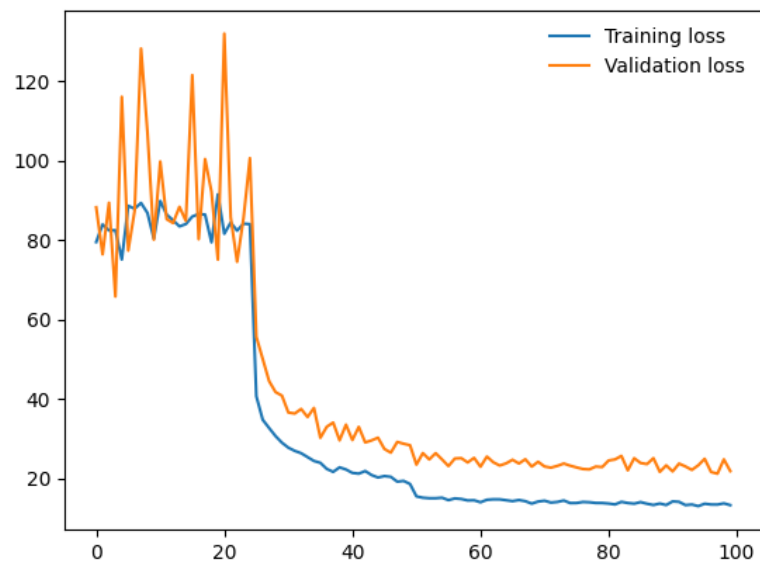
## **Optimizer**

Optimizer koji će utjecati na mijenjanje težišta neurona prilikom backpropagation algoritma je Adam algoritam. Koristi stohastički gradijentni spust.

## 4. Evaluacija



**Slika 4.1:** Greška prilikom treniranja nad CK+ podatkovnim skupom



**Slika 4.2:** Greška prilikom treniranja nad Google i CK+ podatkovnim skupom

#### 4.0.1. Konfuzijska matrica

```
[17, 1 , 4 , 4 , 3 , 0 , 11, 3]
[4 , 19, 2 , 5 , 1 , 1 , 5 , 2]
[3 , 0 , 6 , 4 , 3 , 0 , 3 , 0]
[4 , 6 , 1 , 22, 4 , 2 , 4 , 1]
[0 , 0 , 1 , 5 , 12, 1 , 5 , 6]
[1 , 0 , 2 , 3 , 2 , 23, 1 , 3]
[11, 5 , 2 , 3 , 3 , 4 , 31, 3]
[1 , 2 , 1 , 2 , 2 , 1 , 0 , 26]
```

**Slika 4.3:** Konfuzijska matrica

## 5. Zaključak

Korištenjem dva podatkovna skupa i alata PyTorch istrenirani model može do određene točnosti predviđati koju emociju izražava slika ljudskog lica. Uvođenjem dodatnog ručno generiranog Google podatkovnog skupa poboljšana je apstrakcija prepoznavanja emocija koja dopušta modelu da bolje predviđa emocije sa neviđenih slika lica koje nisu slične slikama iz podatkovnog skupa CK+. Točnost predviđanja emocija može biti poboljšana uvođenjem većeg i apstraktnijeg podatkovnog skupa koji sadrži raznolike slike ljudskih lica.

# LITERATURA

- [1] Imagenet normalization. URL <https://github.com/pytorch/examples/blob/97304e232807082c2e7b54c597615dc0ad8f6173/imagenet/main.py#L197-L198>.
- [2] Numpy array. URL <https://numpy.org/doc/stable/reference/generated/numpy.array.html>.
- [3] Pytorch tensor. URL <https://pytorch.org/docs/stable/tensors.html>.
- [4] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, i L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. U *CVPR09*, 2009.
- [5] K. He, X. Zhang, S. Ren, i J. Sun. Deep residual learning for image recognition. U *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, stranice 770–778, 2016.
- [6] Patrick Lucey, Jeffrey Cohn, Takeo Kanade, Jason Saragih, Zara Ambadar, i Iain Matthews. The extended cohn-kanade dataset (ck+): A complete dataset for action unit and emotion-specified expression. stranice 94 – 101, 07 2010. doi: 10.1109/CVPRW.2010.5543262.
- [7] rakshithvasudev. One hot encoding. URL <https://hackernoon.com/what-is-one-hot-encoding-why-and-when-do-you-have-to-use-it-e3c618>
- [8] Hardik Vasa. google-images-download. URL <https://github.com/hardikvasa/google-images-download>.

## **Prepoznavanje emocija iz izraza lica pomoću strojnog učenja**

### **Sažetak**

Ljudske emocije temeljni dio svih mogućih ljudskih radnji. Treniranjem modela pomoću strojnog učenja koji može klasificirati emocije na temelju slika ljudskih lica otvaraju se mogućnosti za objektivno analiziranje ljudskog ponašanja i razmišljanja u široko primjenjivim situacijama. Uz dva podatkovna skupa ljudskih lica, ResNet 50 mrežu i prijenosnim učenjem ponovno istrenirani model može klasificirati neviđene slike ljudskog lica sa 60% točnosti.

**Ključne riječi:** strojno učenje, duboko učenje, rezidualne neuronske mreže, emocija, lice

## **Recognizing Emotions From Facial Expressions Using Machine Learning**

### **Abstract**

Human emotions are the base of all human activities. Training a model by using a machine learning method which can later classify emotions given an image of a human facial expression opens up a possibility for an objective analysis of human behaviour and thought process in diverse situations. With two datasets containing images of human facial expressions, ResNet 50 network and transferred learning method the trained model can classify emotions given an unseen image of human facial expression with an accuracy of 60%

**Keywords:** machine learning, deep learning, residual neural network, emotion, face