Prepoznavanje oblika na fotografijama pomoću konvolucijske neuronske mreže

Vuković, Filip

Sveučilište J.J. Strossmayera u Osijeku
Odjel za matematiku
Sveučilišni preddiplomski studij matematike i računarstva

Filip Vuković

Prepoznavanje oblika na fotografijama pomoću konvolucijske neuronske mreže

Završni rad

Osijek, 2021.

Sveučilište J.J. Strossmayera u Osijeku
Odjel za matematiku
Sveučilišni preddiplomski studij matematike i računarstva

Filip Vuković

Prepoznavanje oblika na fotografijama pomoću konvolucijske neuronske mreže

Završni rad

Mentor: izv. prof. dr. sc. Domagoj Matijević

Osijek, 2021.

Sažetak

Tema ovog završnog rada je problem prepoznavanja objekata na fotografijama. U radu će biti detaljno predstavljene, opisane i komentirane tehnologije strojnog učenja i računalne vizije (end. Computer Vision) koje se koriste pri rješavanju problema klasifikacije objekata na fotografijama. Najvažnije korištene tehnologije su upravo neuronske mreže, odnosno, za ovakav problem, konvolucijske neuronske mreže bez kojih je nemoguće brzo i optimalno riješiti iskazani problem. Cilj ovog rada je predstaviti problem, pružiti nekoliko ideja rješavanja te pronaći rješenja koja rade dovoljno dobro u praksi. Zatim takva rješenje detaljno istražiti i opisati. Na kraju rada će biti prokomentirana već postojeća, sofisticiranija rješenja.

Ključne riječi

Prepoznavanje fotografija, strojno učenje, računalna vizija, neuronska mreža, konvolucijska neuronska mreža

Image classification using convolutional neural networks

Summary

This bachelor thesis deals with the problem of image classification. It introduces, describes and comments on technologies of Machine Learning and Computer Vision that are used to solve the problem of image classification. The most commonly used technologies for solving this problem are neural networks, in particular convolutional neural networks, without which it would be impossible to provide practically good solutions. The aim of this thesis is to introduce the problem, provide a few solution ideas, find and further describe the practical solutions and lastly discuss already existing, more sophisticated solutions.

Key words

Image classification, neural networks, convolutional neural network, machine learning, computer vision.

Sadržaj

Uvod

Za ljude je problem prepoznavanja objekata na fotografijama trivijalan. Naime, ljudi svoje vizualne sposobnosti poput binokularne integracije, percepcije dubine, prepoznavanja uzoraka te multitasking uzimaju zdravo za gotovo. Jedna od ključnih sposobnosti ljudskog mozga je da nesvjesno identificira objekte, bez obzira razlikuju li se po veličini, rotaciji, osvjetljenju ili položaju. Ljudski mozak je u stanju otprilike prepoznati i zamućene slike.

Za računalo se takva klasifikacija temelji na formiranju algoritma strojnog učenja koji će za dani par slike te njene oznake (eng. label), "naučiti" ispravno klasificirati neku drugu sliku bez pripadne oznake. Odnosno, "naučiti" će klasificirati fotografije ovisno o predmetu koji se na njima nalazi. Prilikom konstruiranja takvog algoritma dolazi do nekoliko izazova poput varijacije točke gledišta, varijacije veličine, osvjetljenja i vidljivosti te deformacije objekata. Naravno, algoritam mora biti robustan na sve navedene izazove kako bi bio praktično upotrebljiv.

Današnja računalna rješenja nadmašuju ljudski vid u mnogim zadacima poput medicinske dijagnoze i klasifikacije objekata na fotografijama.

Ovaj rad podijeljen je u šest poglavlja. U prvom poglavlju opisat ću dva rješenja koja nemaju veze s konvolucijskim neuronskim mrežama, ali će poslužiti kao uvod u problem klasifikacije fotografija. Zatim ću objasniti što je to konvolucijska neuronska mreža te kako i zašto se koristi. Također, opisat ću razne tehnike optimizacije i regularizacije prilikom treniranja konvolucijskih neuronskih mreža. Nakon toga ću napraviti kratak pregled već postojećih klasifikacijskih konvolucijskih neuronskih mreža. U zadnjem poglavlju ću prezentirati svoj završni projekt u kojem sam implementirao konvolucijsku neuronsku mrežu koja prepoznaje 6 glavnih likova iz serije Simpsoni s točnošću od 94%.

1 Problem prepoznavanja objekata na fotografijama

Kako bi izgledao algoritam koji može klasificirati slike u neke kategorije?

Za razliku od pisanja algoritma za, na primjer, sortiranje brojeva, nije očito kako bi izgledao algoritam za identifikaciju, na primjer, mačaka na slikama. Stoga, umjesto da pokušavamo odrediti kako svaka od kategorija izgleda izravno u kodu, pristup kojim ćemo ići nije puno drugačiji od onog kojim biste išli s djetetom. Dakle pružit ćemo računalu puno slika svake kategorije, a zatim razviti algoritam koji promatra sve slike i uči o vizualnom izgledu svake kategorije. Ovaj pristup naziva se pristupom na temelju podataka jer se oslanja na velikom broju podataka za treniranje označenih slika.

Budući da je u klasifikaciji objekata na fotografijama zadatak uzeti niz piksela koji zajedno s oznakom predstavljaju jednu sliku, konkretni problem možemo podijeliti u tri koraka.

1. Ulaz
2. Učenje
3. Evaluacija

Ulaz se sastoji od skupa podataka za treniranje, odnosno, n fotografija, svaka označena s jednom od k kategorija. U drugom koraku je zadatak koristeći se skupom podataka za treniranje formirati algoritam, odnosno naučiti model kako izgleda pojedina kategorija. Pri evaluaciji se zapravo ocjenjuje kvaliteta istreniranog modela. Točnije ocjenjujemo ispravnost modela na novim, modelu "neviđenim", podacima, tj. podacima za testiranje. Zahtijevamo da model predvidi oznake danih slika te očekujemo da će se većina predviđenih oznaka podudarati s istinitima.

1.1 Klasifikator najbližih susjeda (eng. Nearest Neighbour Classifier)

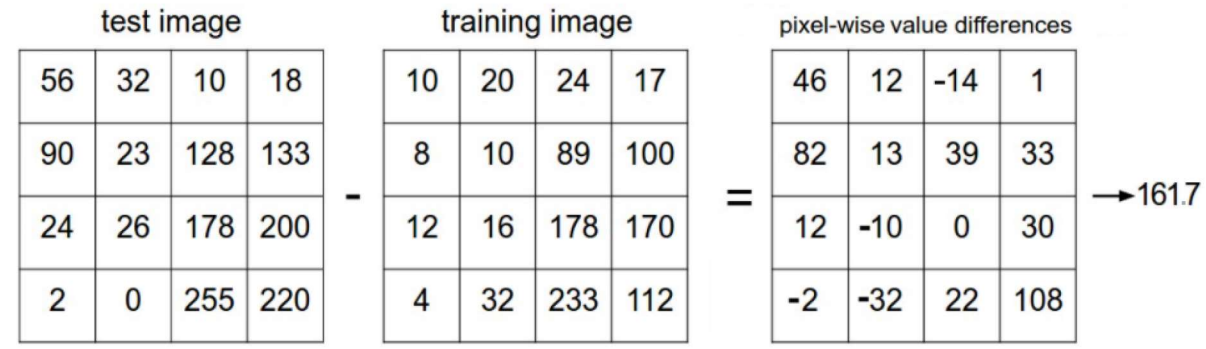
Kao prvi pristup, razvit ćemo nešto što nazivamo klasifikatorom najbližih susjeda [1](eng. Nearest neighbour). Ovaj klasifikator nema nikakve veze s konvolucijskim neuronskim mrežama i jako se rijetko koristi u praksi, ali će nam omogućiti da dobijemo ideju o osnovnom pristupu rješavanja problema klasifikacije fotografija.
Recimo da imamo izuzetno veliki set podataka za treniranje i znatno manji set podataka za testiranje. Ideja je uzeti fotografiju iz skupa podataka za testiranje, usporediti ju sa svim fotografijama iz skupa podataka za treniranje te vratiti oznaku najsličnije fotografije iz skupa podataka za treniranje. Kako bi usporedili fotografije najjednostavnije je gledati piksel po piksel i vratiti razliku.

[1]https://cs231n.github.io/classification/

Drugim riječima, dobivene slike prikazujemo kao vektore, I_1 i I_2, njih usporedimo npr. L2 normom na sljedeći način:

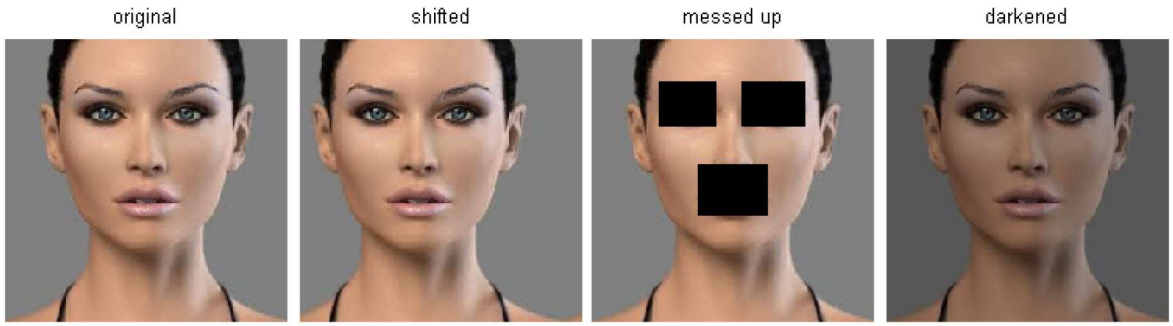
$$d_2(I_1, I_2) = \sqrt{\sum_p (I_1^p - I_2^p)^2}$$

gdje suma ide po svim pikselima. Ovako bi to izgledalo na konkretnom primjeru:



Slika 1: Računanje razlike u slikama

Klasifikator najbližeg susjeda ponekad može biti dobar izbor ako su podaci opisani malim brojem dimenzija, ali je rijetko prikladan za upotrebu u praktičnoj klasifikaciji fotografija. Problem je što su slike objekti s mnogo dimenzija, odnosno sadrže mnogo piksela, a udaljenosti u visoko-dimenzionalnim prostorima mogu biti vrlo kontra-intuitivne. Donja slika ilustrira činjenicu da se sličnosti, po L2 normi, temeljene na pikselima jako razlikuju od perceptivnih sličnosti. Sve četiri slike imaju jednaku L2 normu.



Slika 2

Izvor slike: https://cs231n.github.io/classification/

Ove slike su međusobno različite, ali su namjerno konstruirane kako bi imale jednaku L2 normu. Naravno, ovo je samo primjer u kojem je radi o istim, ali malo pobrkanim slikama. U praksi se može dogoditi da dvije potpuno različite slike imaju jednaku L2 normu što znači da će ovako definiranom klasifikatoru potpuno različite slike biti jednake samo zato što, slučajno, imaju jednaku L2 normu. Iz tog razloga ovakvim pristupom ne možemo, ni približno, razviti prihvatljiv model za klasifikaciju fotografija.

1.2 Linearni klasifikator

Kao drugi pristup rješavanju problema, razmotrimo linearni klasifikator[2]. Koncept linearnog klasifikatora je izuzetno važan, koristit ćemo ga kasnije u neuronskim mrežama.

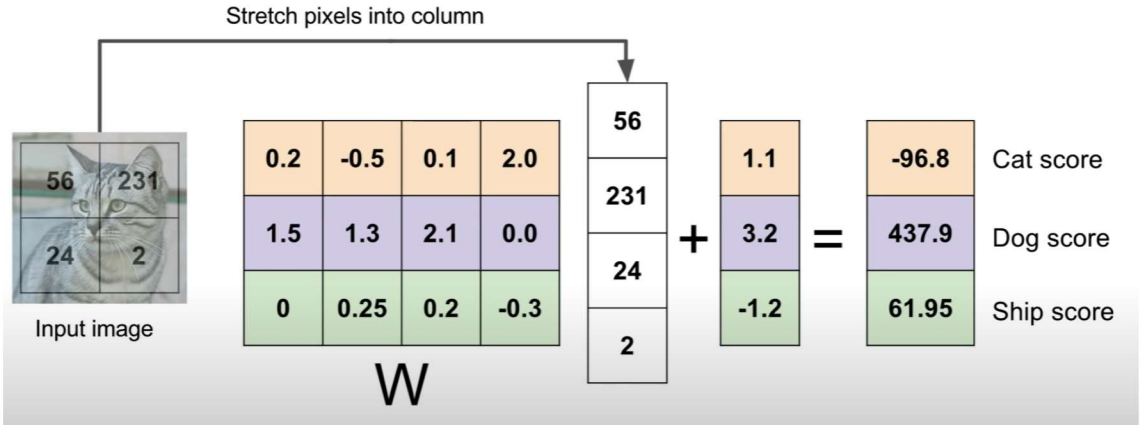
Linearni klasifikator je parametarski model, ima dvije komponente, jedna je slika (x_i), a druga je set parametara (W). Također može imati i treću komponentu pristranosti b (end. bias) koji se koristi kako bi prilagodili izlaz ovisno o parametrima.

$$\text{Slika} \rightarrow f(x_i, W, b) \rightarrow \text{k brojeva}$$

Slike i parametri se prosljeđuju model funkciji f. Model funkcija vraća k brojeva koji opisuju vjerojatnost da pripadna slika pripada kategoriji k.
Za razliku od klasifikatora najbližih susjeda koji nije imao parametre i morao je čuvati sve podatke, prednost ovog parametarskog pristupa je u tome što nakon što naučimo parametre možemo odbaciti podatke za treniranje. Nadalje, predviđanje za sliku iz podataka za testiranje je puno brže jer zahtijeva samo jedno množenje s matricom težina W, a ne iscrpnu usporedbu sa svakom slikom iz podataka za treniranje.
Ideja je pronaći pravu strukturu model funkcije, kombinirajući različite kompleksne kombinacije parametara i podataka. Najjednostavniji primjer bi bio pomnožiti podatke s parametrima.

$$f(x_i, W, b) = Wx_i + b$$

Recimo da imamo slike 32x32 rezolucije u RGB formatu te da imamo 10 kategorija. Promotrimo da je ovakvo množenje dobro definirano ako je x_i vektor a W matrica. Tada će x_i biti dimenzija 1x3072, W dimenzija 3072x10 i b dimenzija 1x10. Nakon množenja dobijemo vektor dimenzije 1x10 u kojem je svaki element rezultat za pojedinu klasu. Indeks elementa s najvećom vrijednošću je oznaka klase koju je model prepoznao.



Slika 3: Ilustracija linearnog klasifikatora u 3 kategorije na slici koja sadrži 4 pixela

[2]https://cs231n.github.io/linear-classify/

Iz slike 3 vidimo da će rezultat za sliku kategorije k biti zapravo skalarni umnožak k-tog reda matrice W i piksela slike reprezentirane kao vektor stupac. Dakle linearnu klasifikaciju možemo interpretirati kao podudaranje predloška (eng. template matching) gdje je svaki red matrice zapravo predložak po kojem linearni klasifikator klasificira slike iz skupa podataka za testiranje. Drugim riječima, ako uzmemo $k-$ti redak konačne, odnosno istrenirane matrice težina W te ga prebacimo u trodimenzionalno polje, tj. sliku, dobili bi predloške prikazane na slici 4.

Slika 4: Predložak za pojedine klase

Izvor slike: https://cs231n.github.io/linear-classify/

Dakle skalarni produkt reda matrice i vektora pixela slike zapravo govori kolika je sličnost testne slike u odnosu na predložak.
Budući da linearni klasifikator stvara jedan predložak po klasi, loš je kada klasificira ulazne slike istog objekta, ali različitih boja i pozadina.

2 Konvolucijska neuronska mreža (eng. CNN)

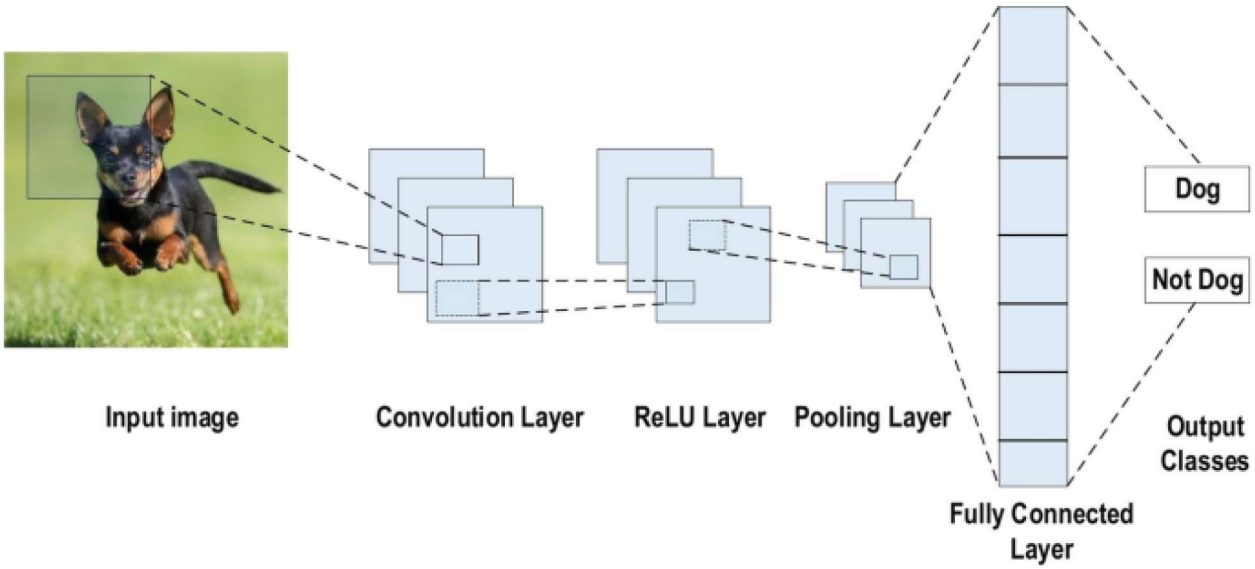
Konvolucijske neuronske mreže su vrlo slične običnima međutim najčešće se koriste pri rješavanju problema kojima su slike ulazni podaci. Naime za malu sliku u boji, dimenzija 200x200x3 jedan neuron u običnoj neuronskoj mreži imao bi 120 000 težina, dodavanjem neurona taj broj bi brzo porastao, a s time i mogućnost pretreniranja (eng. overfitting).

Arhitektura konvolucijske neuronske mreže je prilagođena slikama kao ulaznim podacima. Tako smanjuje broj parametara te ubrzava propagaciju unaprijed (eng. forward propagation).

Konvolucijska neuronska mreža se sastoji od više slojeva. Svaki sloj ima jednostavno sučelje. Ovakva mreža pretvara ulaznu sliku, odnosno trodimenzionalno polje u izlazno trodimenzionalno polje koristeći slojeve koji mogu, ali ne moraju imati dodatne parametre.

Slojevi konvolucijskih neuronskih mreža:

1. Konvolucijski
2. Pooling
3. Aktivacijski
4. Normalizacijski
5. Potpuno povezani



Slika 5: CNN - slojevi

2.1 Konvolucijski sloj

Konvolucijski slojevi glavni su građevni blokovi u konvolucijskim neuronskim mrežama. Pojam konvolucije se odnosni na matematičku operaciju koja kombinira dvije funkcije kako bi stvorila treću. Pod konvolucijom podrazumijevamo rezultirajuću funkciju kao i proces kojim se ona računa. Drugim riječima, konvolucija spaja dva seta informacija te računa treći.

U kontekstu konvolucijske neuronske mreže, konvolucija je linearna operacija koja uključuje množenje težina s ulazom, slično tradicionalnoj neuronskoj mreži. S obzirom na to da je tehnika osmišljena za višedimenzionalni ulaz, množenje se vrši između niza ulaznih podataka i višedimenzionalnog niza težina, koje nazivamo filterima ili jezgrama (eng. kernel). Rezultat filtriranja je aktivacija, odnosno mapa značajki (eng. feature map). Mapa značajki je izlaz svakog konvolucijskog sloja, ona sadrži detektirane značajke ulaza, u ovom slučaju slika. Mreža će naučiti filtere koji se aktiviraju kada vide neku vrstu vizualne značajke, poput ruba određene orijentacije ili mrlje neke boje na prvom sloju, ili uzorke poput kotača na višim slojevima mreže.

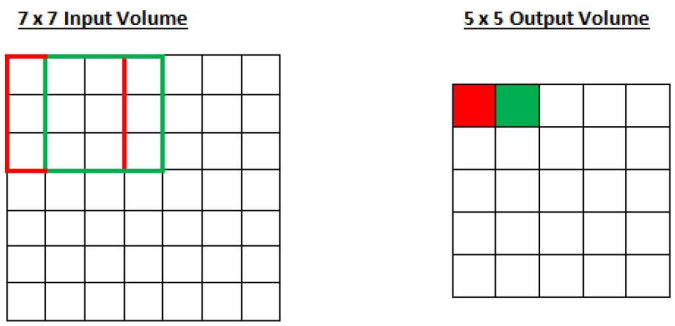
U konvolucijskim neuronskim mrežama postoji nekoliko konvolucijskih slojeva koji sadrže različite filtre kojima otkrivamo značajke. Raniji slojevi se pokušavaju usredotočiti na šire značajke, a kasniji slojevi pokušavaju otkriti vrlo specifične značajke.

Filteri različitih veličina detektirat će značajke različite veličine na ulaznoj slici, što će rezultirati mapama značajki različite veličine. Uobičajeno je koristiti filtere veličine 3×3, 5×5 ili čak 7×7, za veće ulazne slike.

2.1.1 Korak filtera

Korak (eng. stride) je parametar filtra konvolucijskog sloja koji određuje količinu kretanja preko slike. Na primjer, ako je korak filtera postavljen na 1, filter će se pomicati po jedan piksel.

Zamislimo sliku 7x7 kao ulazni volumen, filter koji je 3x3 (zanemarimo treću dimenziju radi jednostavnosti) i korak od 1. Sljedeća slika vizualizira kretanje filtera po slici.



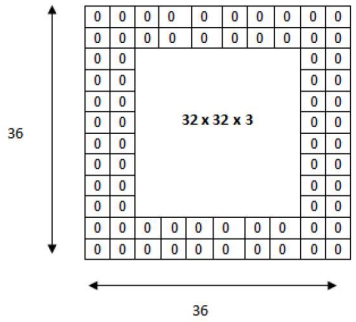
Slika 6: Vizualizacija koraka

Izvor slike: https://adeshpande3.github.io/

Na slici 6 vidimo kako se filter pomiče jedan korak u desno, intuitivno jedan korak prema dolje. Izlazni volumen je tada dimenzija 5x5.

2.1.2 Razmak filtera

Razmak (eng. padding) je parametar koji određuje količinu piksela dodanu slici prilikom filtriranja. Na primjer, ako je razmak postavljen na nulu, tada će svaka dodana vrijednost piksela biti nula. Ako je, međutim, nulti-razmak postavljen na jedan, slici će se dodati rub od jednog piksela s vrijednošću nula. Sljedeća slika vizualizira razmak pri filtriranju.



Slika 7: Vizualizacija razmaka

Izvor slike: https://adeshpande3.github.io/

Slika 7 ima razmak postavljen na 0, a nulti-razmak postavljen na 2. Zato ima dva dodatna rubna pixela postavljena na vrijednost nula.

2.1.3 Koncept zajedničkih težina (eng. Shared weights)

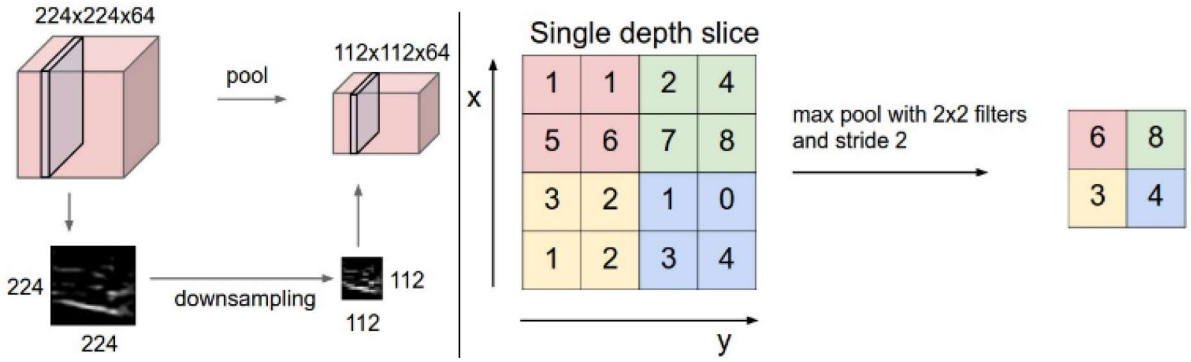
U jednom konvolucijskom sloju filter se primjenjuje na ulaznu sliku. Taj filter daje jednu parametrizaciju težina i pristranosti (eng. bias) koja tvori jednu mapu značajki. Za razliku od tradicionalnih neuronskih mreža, konvolucijske se svode na treniranje filtera koji sadrže zajedničke težine. Različiti filteri se tada, zbog nelinearnosti aktivacijskih funkcija, istreniraju na različitim značajkama. Rezultat ovog koncepta je puno manji broj ukupnih parametara mreže. Također, budući da u svakom sloju imamo jednu rezultirajuću matricu težina koja traži značajke preko cijele slike, nije bitno gdje se ta značajka na slici nalazi, bit ćemo u mogućnosti pronaći ju.

2.2 Pooling sloj

Uobičajeno je povremeno umetnuti pooling sloj između uzastopnih konvolucijskih slojeva zato što pooling sloj nastoji postupno smanjiti prostornu veličinu kako bi se smanjila količina parametara i računanja u mreži, a s time kontrolira i pretreniranje (eng. overfitting).

Najčešći oblik pooling sloja, prikazanog na slici dolje, je MaxPool s filterom veličine 2x2 primijenjenim s korakom od 2.

Ovakav pooling sloj će prepoloviti dimenzije ulaza duž širine i visine.

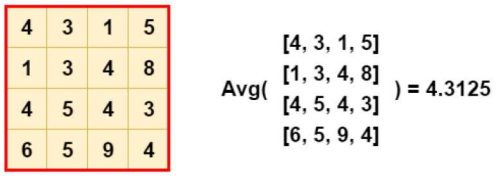


Slika 8: Vizualizacija pooling sloja

Izvor slike: https://cs231n.github.io/convolutional-networks/

Još jedan, bitan, pooling sloj je globalni pooling srednje vrijednosti (eng. global average pool). Naime, ideja je generirati jednu mapu značajki za svaku odgovarajuću klasu u posljednjem sloju. Global average pool služi, između ostalog, kao zamjena za potpuno povezane slojeve na kraju konvolucijske neuronske mreže. Umjesto dodavanja potpuno povezanih slojeva na mape značajki, uzimamo prosjek svake mape značajki, a rezultirajući vektor se unosi izravno u zadnji potpuno povezani sloj koji vraća izlaz neuronske mreže, tj. vektor rezultata po klasama.

Dodatno, manji broj potpuno povezanih slojeva znači manji broj parametara u mreži. Još jedna prednost je ta što u globalni pooling srednje vrijednosti ne postoji parametar za optimizaciju pa je, u ovom sloju, izbjegnuto pretreniranje. Nadalje, globalni pooling srednje vrijednosti sažima prostorne informacije, pa je robusniji prema prostornim manipulacijama ulaza.

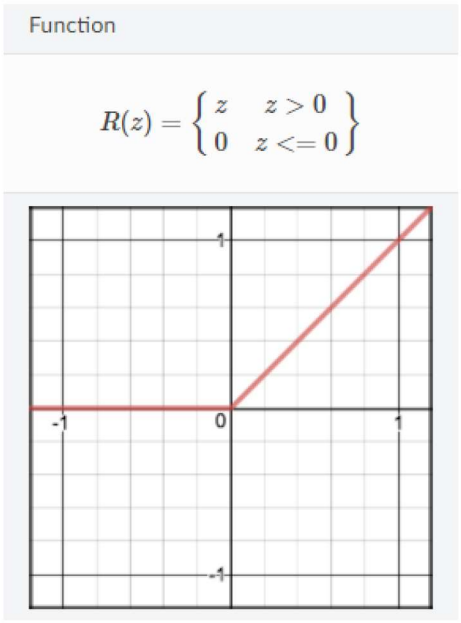


Slika 9: Global average pool

2.3 Aktivacijski sloj

Aktivacijski sloj u konvolucijskoj neuronskoj mreži sastoji se od aktivacijske funkcije koja uzima mapu značajki generiranu konvolucijskim slojem te stvara mapu aktivacije kao svoj izlaz. Funkcija aktivacije je operacija koja se računa element po element (eng. element-wise) nad ulaznim volumenom pa dimenzije ulaza i izlaza ostaju nepromijenjene.

Najčešće korištena aktivacijska funkcija u konvolucijskim neuronskim mrežama je ReLU (eng. Rectified Linear Unit). ReLU računa funkciju $f(x) = \max(0, x)$



Slika 10: ReLU

Izvor slike: https://ml-cheatsheet.readthedocs.io/en/latest/activation_functions.html

Prednosti korištenja ReLU aktivacijske funkcije:

1. vrlo je jednostavna za implementirati
2. ne koristi zahtjevne eksponencijalne operacije.
3. uvelike ubrzava konvergenciju stohastičke gradijentne metode.

Nažalost, problem ReLU funkcije je u tome što sve negativne vrijednosti odmah postaju nule, što smanjuje sposobnost modela da se pravilno nauči iz podataka. Drugim riječima, svaki negativan ulaz će "umrijeti", ovaj problem nazivamo problemom umirućeg ReLU-a (eng. dying ReLU). Propustni (eng. Leaky) ili parametrizirani ReLU jedno je od rješenja problema "umirućeg ReLU-a". Umjesto da je funkcija nula kada je $x < 0$, leaky ReLU će umjesto toga imati mali nagib (od 0,01 ili slično). Odnosno, funkcija izračunava $f(x) = \alpha x$ za $x < 0$ i $f(x) = x$ za $x >= 0$ gdje je α mala konstanta ili parametar neuronske mreže.

2.4 Normalizacijski sloj

Normalizacijski sloj u konvolucijskim neuronskim mrežama koristi razne normalizacijske tehnike koje mogu značajno smanjiti vrijeme treniranja mreže.

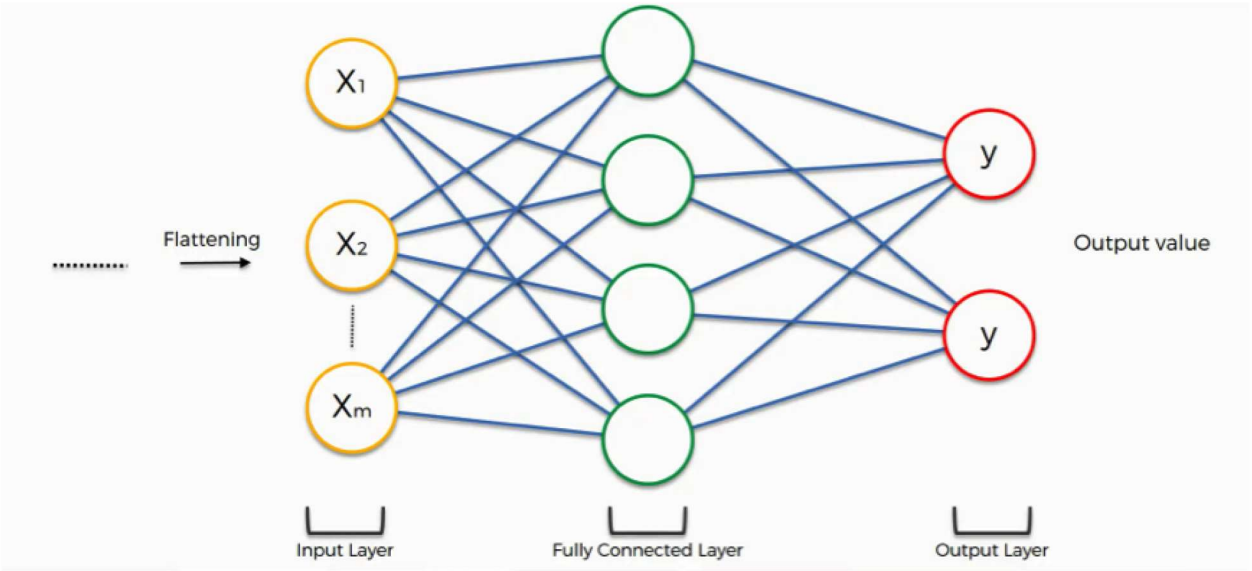
Ovaj sloj normalizira svaku značajku tako da im održava doprinos, jer neke značajke imaju veću numeričku vrijednost, a neke jako malu. Tako sprječava da mreža postane nepristrana (eng. unbiased). Također ubrzava optimizaciju jer ne dopušta da težine eksplodiraju, tj. ograničava ih u određenom rasponu vrijednosti. Čini se da je ispravna normalizacija ključna za učinkovito treniranje modela.

Često se koristi serijska (eng. batch) normalizacija kao tehnika normaliziranja konvolucijskih neuronskih mreža. Serijska normalizacija normalizira aktivacijske vektore koristeći srednje vrijednosti i varijacije trenutne serije.

Normalizacijski sloj se primjenjuje neposredno prije ili odmah nakon aktivacijskog sloja.

2.5 Potpuno povezani sloj

Poputno povezani sloj u konvolucijskoj neuronskoj mreži se pojavljuje na kraju arhitekture. Neuroni u potpuno povezanom sloju imaju punu vezu sa svim aktivacijama u prethodnom sloju, kao što je slučaj u tradicionalnim neuronskim mrežama. Potpuno povezani sloj će izračunati vektor stupac dimenzija 1xk, odnosno rezultat za pojedinu klasu.



Slika 11: Potpuno povezani sloj

Izlaz konvolucijskih slojeva potrebno je poravnati (eng. flattening), odnosno dobiti jednodimenzionalni vektor koji postaje ulaz u potpuno povezanom sloju.

3 Optimizacija CNN-a

Ideja optimiziranja neuronskih mreža je ubrzati pronalazak optimalnih težina. Funkcija gubitka (eng. loss function) nam omogućuje kvantificiranje kvalitete bilo kojeg skupa težina, odnosno govori koliko je dobar ili loš taj skup težina na nekom problemu. Cilj optimizacije je pronaći težine koje minimiziraju funkciju gubitka.

3.1 Stohastički gradijentni silazak (eng. Stochastic Gradient Descent or SGD)

Stohastički gradijentni silazak trenutno je daleko najčešći način optimizacije funkcija gubitka neuronske mreže. Također je jako jednostavan optimizacijski algoritam koji prvo računa vrijednost funkcije gubitka i gradijenta na nekom randomiziranom podatku, a zatim ažurira težine u negativnom smjeru izračunatog gradijenta.

Algoritam dakle izgleda ovako:

```
for i in range(nb_epochs):
  np.random.shuffle(data)
  for example in data:
    params_grad = evaluate_gradient(loss_function, example, params)
    params = params - learning_rate * params_grad
```

Slika 12: SGD algoritam - Python

3.2 Adagrad

Ideja Adagrada je prilagoditi brzinu učenja (eng. learning rate) parametrima izvodeći manja ažuriranja za težine povezane sa značajkama koje se češće pojavljuju i veća ažuriranja za značajke koje se rjeđe pojavljuju. Adagrad algoritam je time puno robusniji od SGD i jako je koristan pri rješavanju problema koje sadrže rijetke podatke (eng. sparse data).

Pravilo ažuriranja težina ovog algoritma glasi:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\sum g_{t-1}^2 + \epsilon}} g_t \tag{1}$$

gdje je g_t je gradijent trenutne iteracije, a g_{t-1} gradijent prošle iteracije s obzirom na sve težine θ. ϵ je jako mala konstanta reda veličine 10^{-8}. Ona je tu kako bi se izbjeglo dijeljenje s nulom, u slučaju kada su prošli gradijenti jednaki upravo nuli.

Glavna slabost Adagrada je njegova akumulacija kvadrata gradijenta u nazivniku. Budući da je svaki pozitivan, akumulirani zbroj nastavlja rasti tijekom treninga. To dovodi do smanjenja brzine učenja koja na kraju postaje beskrajno mala, tj. u tom trenutku mreža više nije u mogućnosti učiti.

3.3 ADAM

Optimizacijski algoritam ADAM se najčešće koristi u računalnoj viziji (eng. Computer Vision) i obradi prirodnog jezika (eng. natural language processing).

ADAM je kombinacija algoritma SGD i algoritma RMSP (eng. Root Mean Square Propagation). RMSP adaptivni je algoritam učenja koji nastoji poboljšati Adagrad. Umjesto da se uzme kumulativni zbroj kvadrata gradijenata kao u Adagradu, on uzima eksponencijalni pokretni prosjek (eng. exponential moving average).

Pravilo ažuriranja težina ovog algoritma glasi:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{v_t} + \epsilon} m_t, \quad \begin{matrix} m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \\ v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \end{matrix} \tag{2}$$

gdje je g_t je gradijent, m_t je procjena srednje vrijednosti, a v_t je suma kvadrata prošlih gradijenata.
Parametri u ADAM algoritma su:

1. ϵ - mala pozitivna konstanta koja se koristi kako bi se izbjeglo djeljenje s 0
2. $\beta 1$ i $\beta 2$ - stope opadanja prosjeka gradijenata kao u algoritmima Adagrad i RMSP
3. α - brzina učenja

Metoda je iznimno učinkovita pri rješavanju problema koji uključuju izuzetno velike skupove podataka ili parametara.

4 Regularizacija

Regularizacija je tehnika koja unosi male izmjene u algoritam učenja tako da se model bolje generalizira, čini ih manje podložnim pretreniranju (eng. overfitting). To također poboljšava performanse modela na još neviđenim podacima, odnosno podacima za testiranje.

4.1 L1, L2 regularizacija

L1 i L2 su najčešći tipovi regularizacije. Oni ažuriraju opću funkciju troška dodavanjem nekog drugog izraza.

Dodavanjem regularizacije, apsolutne vrijednosti težina se smanjuju jer se pretpostavlja da neuronska mreža s manjim težinama vodi do jednostavnijih modela. Jednostavniji model je time manje pretreniran.

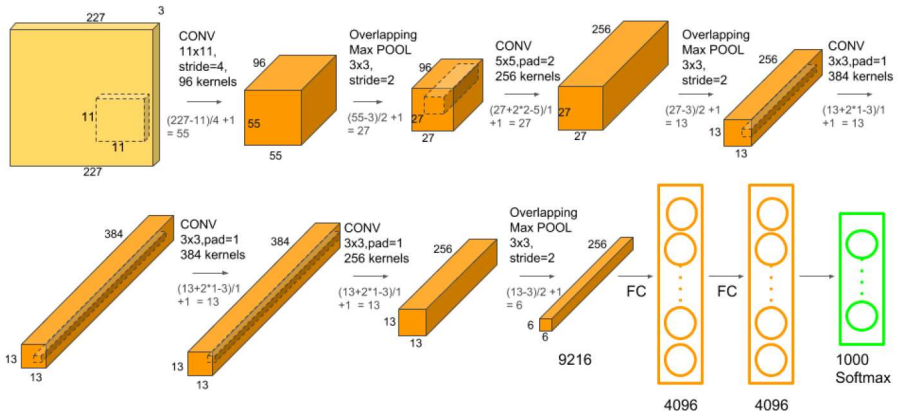
4.2 Dropout

Dropout je iznimno učinkovita, jednostavna i nedavno uvedena tehnika regularizacije koja se često koristi prilikom treniranja konvolucijske neuronske mreže. Dropout se može tumačiti kao randomizirano odabiranje dijela neuronske mreže unutar cijele neuronske mreže na kojem će se ažurirati parametri, odnosno to je tehnika u kojoj se nasumično odabrani neuroni zanemaruju tijekom treninga.

5 Pregled postojećih mreža

5.1 AlexNet

Prva konvolucijska mreža koja je popularizirala konvolucijske neuronske mreže u računalnoj viziji bila je AlexNet, razvili su je Alex Krizhevsky, Ilya Sutskever i Geoff Hinton. AlexNet je 2012. godine na ImageNet izazovu značajno nadmašila sva ostala rješenja.

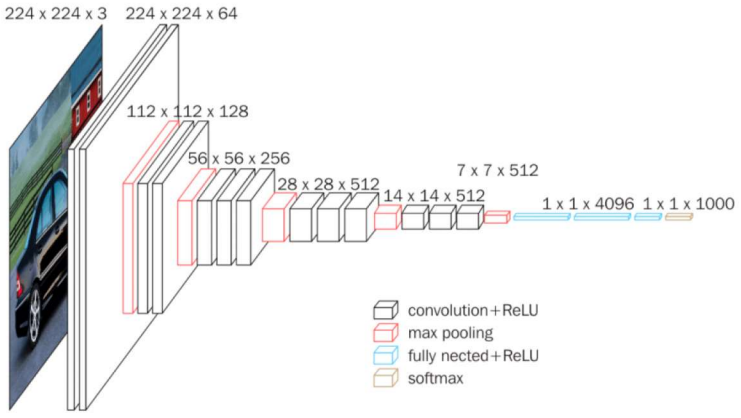


Slika 13: AlexNet - arhitektura

Arhitektura mreže se sastoji od 5 konvolucijskih, 3 pooling i 3 potpuno povezanih slojeva s preko 60 milijuna parametara. AlexNet koristi ReLU aktivacijsku funkciju, dropout i stohastički gradijentni silazak kao optimizacijsku tehniku.

5.2 VGGNet

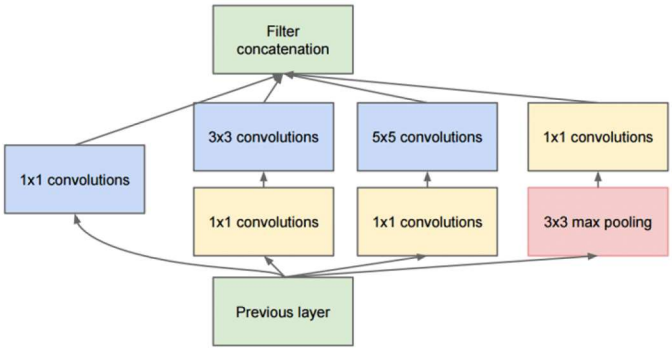
Generalna ideja VGGNet-a je proširiti AlexNet te koristiti manje filtere i dublje neuronske mreže, stoga se ova mreža sastoji od 16 konvolucijskih slojeva s filterima veličine 3x3. Time se značajno povećava broj parametara koji je u ovakvoj mreži veći od 130 milijuna.



Slika 14: VGG - arhitektura

5.3 GoogLeNet

Doprinos ove arhitekture je razvoj "početnog" modela (eng. inception model) koji je dramatično smanjio broj parametara u mreži na samo 5 milijuna. Osim toga, koristi "average pooling" umjesto potpuno povezanih slojeva na vrhu mreže, eliminirajući veliku količinu parametara za koje se ispostavilo da nisu važni.

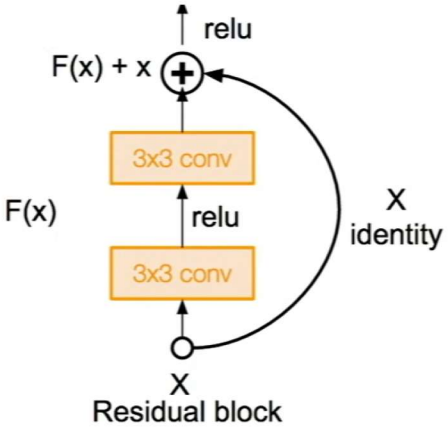


Slika 15: GoogLeNet - Inception model

Ideja u pozadini inception modula je dizajnirati manje, lokalne konvolucijske mreže (mreže u mreži) pa ih složiti u cjelinu. Unutar inception modela se paralelno odvijaju konvolucije na izlazu od prijašnjeg modela. Na kraju je potrebno spojiti sve izlaze filtera u jedan volumen koji će biti ulaz za sljedeći model.

5.4 ResNet

Vrlo duboka mreža od čak 152 sloja koja koristi rezidualne veze (eng. residual connection) kako bi smanjila mogućnost pretreniranja. Rezidualni blok je skup slojeva postavljenih tako da se izlaz nekog konvolucijskog sloja uzima i dodaje drugom sloju dublje u bloku.

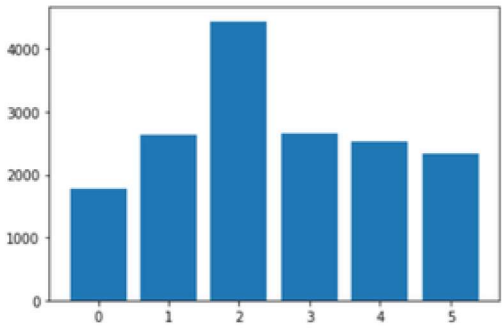


Slika 16: ResNet - residual block

Ideja je poslagati jako veliki broj rezidualnih blokova i provoditi serijsku normalizaciju nakon svakog konvolucijskog sloja. Skup ovih rezidualnih blokova postiže 3,57% pogreške na ImageNet skupu podataka za testiranje.

6 Implementacija konvolucijske neuronske mreže

Popratni projekt implementirali smo kolega Josip Pavičić i ja. Implementiran je koristeći programski jezik Python i PyTorch biblioteku. Koristili smo skup podataka za treniranje koji se sastoji od 16380 slika Homera, Abrahama, Lise, Barta, Marge i Skinnera iz popularne serije Simpsoni raspoređenih u 6 klasa kao na slici 17.



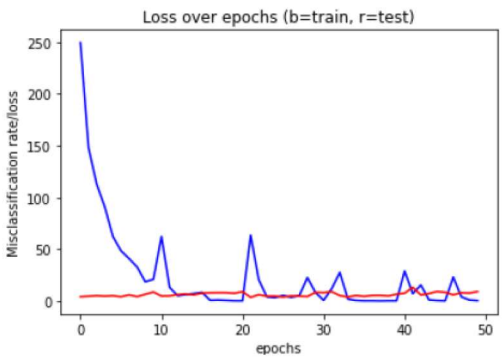
Slika 17: Simpson dataset

6.1 Treniranje modela

Računanje funkcije gubitka na trening i test podacima u svakom ciklusu treniranja važna je informacija o treniranosti modela. Model može biti pretreniran, podtreniran i model koji je dovoljno dobar u praksi.

6.1.1 Pretreniran model i funkcija gubitka

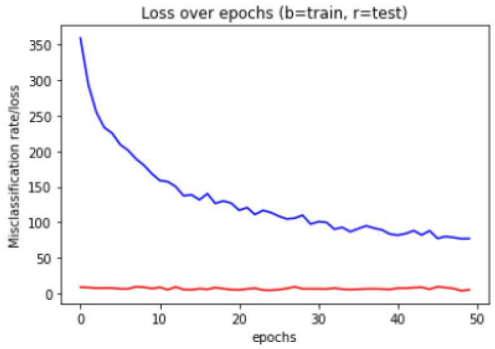
Model je pretreniran u terminima funkcije gubitka kad je funkcija gubitka na skupu podataka za treniranje manja od funkcije gubitka na skupu podataka za testiranje. Također funkcija gubitka na skupu podataka za testiranje raste (na slici 18. je vidljiv blagi rast crvene linije) Model je pretreniran kad je toliko dobro istreniran na podacima za treniranje da nauči i njihov šum. Dakle nije generaliziran jer nauči svaki podatak perfektno tako da će, do sad neviđeni, podatak krivo klasificirati.



Slika 18: Graf funkcije gubitka na pretreniranom modelu

6.1.2 Podtreniran model i funkcija gubitka

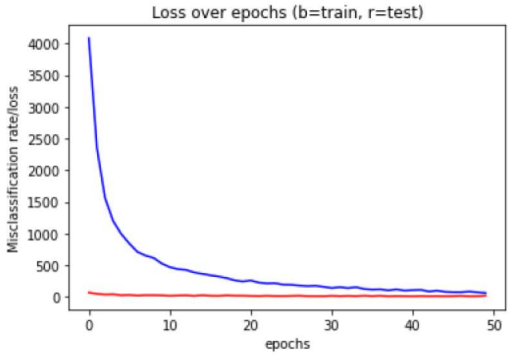
Podtrenirani model ima lošu matricu konfuzije i njene metrike. Sadrži premalo parametara, odnosno pre jednostavan je za klasifikacijski problem.



Slika 19: Graf funkcija gubitka na podtreniranom modelu

6.1.3 Model koji je dovoljno dobar u praksi

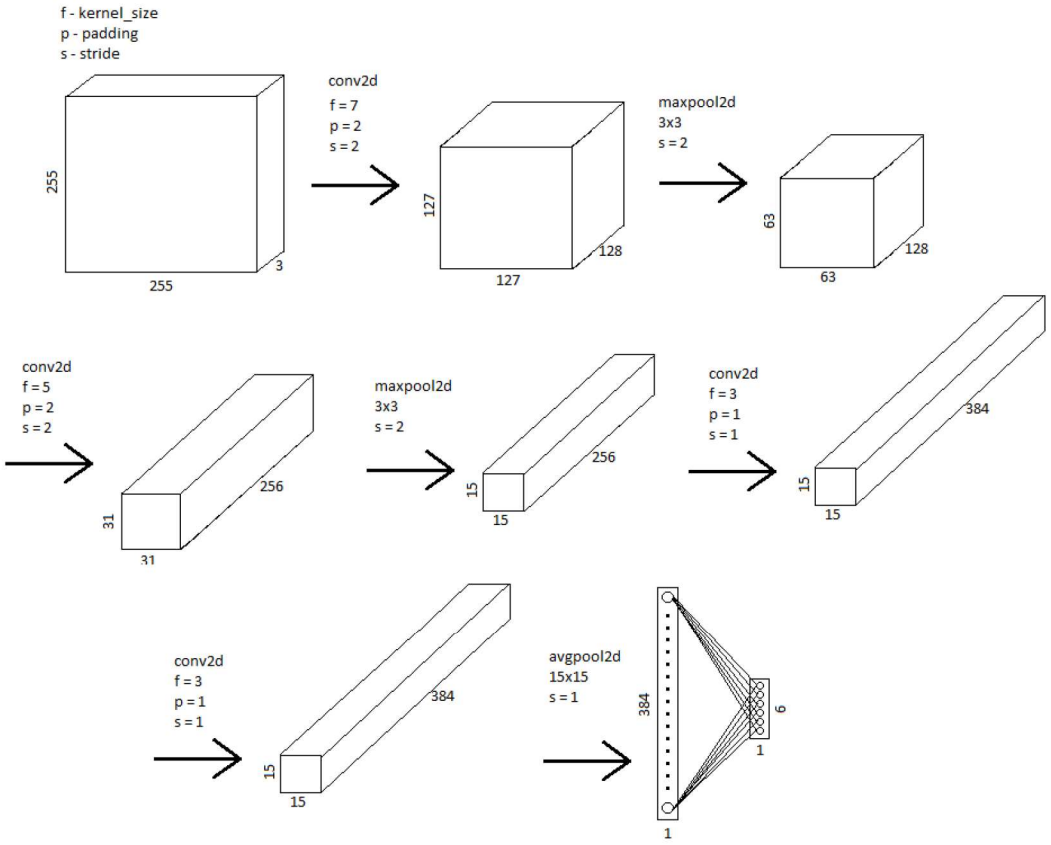
Nakon korekcija na neuronskoj mreži, popravaka pretreniranosti, podtreniranosti i arhitekture mreže, dobit ćemo model koji dobro klasificira neviđene podatke.



Slika 20: Graf funkcija gubitka na našem modelu

6.2 Arhitektura implementirane mreže

Implementirana konvolucijska neuronska mreža sastoji se od 4 konvolucijska, 4 normalizacijska sloja, 4 aktivacijska, 3 pooling i jednim potpuno povezanim slojem kao na slici 21.



Slika 21: Arhitektura implementirane CNN

Nakon svakog konvolucijskog sloja, koji generira mapu značajki slijedi normalizacijski sloj kojim sprječavamo nepristranost (eng. unbias) te optimiziramo treniranje. Nakon svakog normalizacijskog sloja slijedi aktivacijski. Kao aktivacijska funkcija, najbolje nam se pokazala Parametric ReLU (PReLU) koja ubrzava konvergenciju gradijenata te rješava problem "umirućeg ReLU-a". MaxPooling slojeve koristimo kako bi smanjili ukupni broj parametara, a globalni pooling srednje vrijednosti koristimo kako bi smanjili broj potpuno povezanih slojeva, a time i broj parametara. Globalni pooling srednje vrijednosti se, također, koristi pri daljnjoj lokalizaciji objekata na slikama uz pomoć CAM filtera (više o ovoj temi možete pronaći u završnom radu kolege Pavičića). Prije slanja informacija u potpuno povezani sloj, poravnali smo izlaz konvolucijskog dijela neuronske mreže u jednodimenzionalno polje koristeći funkciju flatten iz torch biblioteke. Kao regularizacijsku tehniku koristili smo Dropout od 60% ispred linearnog sloja. Tako mreža postaje manje osjetljiva na specifičnu težinu neurona te postiže bolju generalizaciju i sprječava pretreniranje. Kao najbolji optimizacijski algoritam ispostavio se ADAM s početnom vrijednosti brzine učenja od 0.0001.

6.3 Evaluacija

Pri evaluaciju računamo matricu konfuzije, tijekom treniranja spremamo iznose funkcije gubitka za skupove trening i test podataka te računamo točnost modela

Matrica zabune način je ocjenjivanja klasifikacijskog modela. To je usporedba između stvarne klase podatka i klase koju model predvidi za određeni podatak. Dobivamo matricu zabune ispunjenu s točno pozitivnim (TP), točno negativnim (TN), netočno pozitivnim (FP) i netočno negativnim (FN) klasificiranim slikama. Pomoću te matrice možemo izračunati:

- točnost (eng. accuracy) - generalna točnost modela

$$\frac{TP + TN}{TP + TN + FP + FN} \tag{3}$$

- netočnost (eng. misclassification) - broj podataka koji je netočno klasificiran

$$\frac{FP + FN}{TP + TN + FP + FN} \tag{4}$$

- preciznost (eng. precision) - udio točnih predikcija

$$\frac{TP}{TP + FP} \tag{5}$$

- osjetljivost (eng. recall) - udio stvarnih podataka koji su ispravno klasificirani

$$\frac{TP}{TP + FN} \tag{6}$$

```
Class:  abraham_grampa_simpson
[tn, fp, fn, tp]:  [249, 1, 6, 44]
Accuracy:  97.66666666666667
Misclassification:  2.3333333333333335
Precision:  97.77777777777777
Recall:  88.0
****************************************
Class:  bart_simpson
[tn, fp, fn, tp]:  [247, 3, 1, 49]
Accuracy:  98.66666666666667
Misclassification:  1.3333333333333335
Precision:  94.23076923076923
Recall:  98.0
****************************************
Class:  homer_simpson
[tn, fp, fn, tp]:  [248, 2, 0, 50]
Accuracy:  99.33333333333333
Misclassification:  0.6666666666666667
Precision:  96.15384615384616
Recall:  100.0
```

```
Class:  lisa_simpson
[tn, fp, fn, tp]:  [247, 3, 5, 45]
Accuracy:  97.33333333333334
Misclassification:  2.666666666666667
Precision:  93.75
Recall:  90.0
****************************************
Class:  marge_simpson
[tn, fp, fn, tp]:  [245, 5, 0, 50]
Accuracy:  98.33333333333333
Misclassification:  1.6666666666666667
Precision:  90.9090909090909
Recall:  100.0
****************************************
Class:  principal_skinner
[tn, fp, fn, tp]:  [248, 2, 4, 46]
Accuracy:  98.0
Misclassification:  2.0
Precision:  95.83333333333334
Recall:  92.0
```

Slika 22

Iz slike 20 vidimo da na 300 trening podataka imamo 284 točno pozitivnih ($44 + 49 + 50 + 45 + 50 + 46 = 284$). Točnost ovog modela je dakle $\frac{284}{300} = 94.67\%$

7 Zaključak

U ovom radu pokazali smo da se, koristeći ispravnu arhitekturu neuronske mreže, može doći do modela koji, u praksi, dovoljno dobro prepoznaje oblike na fotografijama. Rješenje je dano konstruirajući konvolucijsku neuronsku mrežu u Python-u koristeći PyTorch biblioteku. Konvolucijske neuronske mreže (CNN) postigle su zapanjujuća postignuća u raznim područjima, uključujući medicinska istraživanja, radiologiji, obradi prirodnog jezika te klasifikaciji i lokalizaciji slika. Upoznavanje s ključnim pojmovima i prednostima CNN-a, kao i ograničenjima dubinskog učenja bitno je kako bi se to iskoristilo u navedenim znanstvenim područjima.

Literatura

[1] *CS231n: Convolutional Neural Networks for Visual Recognition*, dostupno na `http://cs231n.stanford.edu/2017/`

[2] *Lecture Collection: Convolutional Neural Networks for Visual Recognition*, dostupno na `https://www.youtube.com/playlist?list=PL3FW7Lu3i5JvHM8ljYj-zLfQRF3EO8sYv`

[3] *Lecture Notes for CS231n*, dostupno na `https://cs231n.github.io/`

[4] *Završni projekt na ovu temu*, dostupno na `https://github.com/FilipVuk123/ImageClassificationLocalization/` ili `https://github.com/pavicicjosip/CNN`

[5] `https://towardsdatascience.com/`

[6] `https://www.kaggle.com/`

[7] `https://adeshpande3.github.io/`