Primjena evaluacijskih tehnika na primjeru konvolucijske neuronske mreže

Lovreković, Lora

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

ZAVRŠNI RAD

Lora Lovreković

Zagreb, 2021.

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

Primjena evaluacijskih tehnika na primjeru konvolucijske neuronske mreže

Mentor:
Doc. dr. sc. Tomislav Stipančić, dipl. ing.

Student:
Lora Lovreković

Zagreb, 2021.

Izjavljujem da sam ovaj rad izradila samostalno koristeći znanja stečena tijekom studija i navedenu literaturu.

Zahvaljujem mentoru doc. dr. sc. Tomislavu Stipančiću na pruženoj pomoći i podršci tijekom pisanja ovoga rada.

Željela bih zahvaliti svojoj obitelji na bezuvjetnoj potpori i prijateljima koji su mi uljepšali i olakšali studentske dane.

Lora Lovreković

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE
Središnje povjerenstvo za završne i diplomske ispite
Povjerenstvo za završne ispite studija strojarstva za smjerove:
proizvodno inženjerstvo, računalno inženjerstvo, industrijsko inženjerstvo i menadžment, inženjerstvo materijala i mehatronika i robotika

Sveučilište u Zagrebu Fakultet strojarstva i brodogradnje	
Datum	Prilog
Klasa: **602 - 04 / 21 - 6 / 1**	
Ur.broj: **15 - 1703 - 21 -**	

ZAVRŠNI ZADATAK

Student: **Lora Lovreković** Mat. br.: 0035214434

Naslov rada na hrvatskom jeziku: **Primjena evaluacijskih tehnika na primjeru konvolucijske neuronske mreže**

Naslov rada na engleskom jeziku: **Application of evaluation techniques on the example of a convolutional neural network**

Opis zadatka:

Alati za evaluaciju rada modela strojnog učenja omogućuju uvid u skrivene ovisnosti parametara neronskih mreža te omogućuju fina podešavanja kako bi njihovi modeli ostvarivali bolje rezultate klasifikacije ili nekog drugog zadatka.

U radu je potrebno trenirati konvolucijsku neuronsku mrežu na *CIFAR-10* bazi slika te evaluirati rad mreže. Posebno je potrebno proučiti evaluacijske tehnike analize modela strojnog učenja. Razvijeni model konvolucijske neuronske mreže ostvaren u *Python* programskom jeziku potrebno je posebno evaluirati te dati analizu s obzirom na *Precision, Accuracy* i *Recall* evaluacijske funkcije.

U radu je potrebno dati osvrt na moguća poboljšanja rada mreže.

Također, u radu je potrebno navesti korištenu literaturu te eventualno dobivenu pomoć.

Zadatak zadan:
30. studenoga 2020.

Zadatak zadao:

Doc. dr. sc. Tomislav Stipančić

Datum predaje rada:
1. rok: 18 veljače 2021.
2. rok (izvanredni): 5. srpnja 2021.
3. rok: 23. rujna 2021.

Predviđeni datumi obrane:
1. rok: 22.2. – 26.2.2021.
2. rok (izvanredni): 9.7.2021.
3. rok: 27.9. – 1.10.2021.

Predsjednik Povjerenstva:

Prof. dr. sc. Branko Bauer

SADRŽAJ

POPIS SLIKA

POPIS TABLICA

POPIS OZNAKA

Oznaka	Jedinica	Opis
D		Stara dimenzija slike
D_{nova}		Nova dimenzija slike
F		Dimenzija filtera
P		Broj nadopunjenih nula
S		Korak
y		Ulazni vektor
$S(y_i)$		Vjerojatnost
TP		Broj točnih pozitiva
TN		Broj točnih negativa
NP		Broj netočnih pozitiva
NN		Broj netočnih negativa

SAŽETAK

U ovome je radu trenirana konvolucijska neuronska mreža na CIFAR-10 bazi slika i evaluiran je rad mreže primjenom evaluacijskih funkcija. U teorijskom dijelu objašnjen je princip rada umjetnih neuronskih mreža te arhitektura i funkcija konvolucijskih neuronskih mreža. Uveden je pojam matrica konfuzije kako bi se mogle objasniti evaluacijske funkcije točnost, preciznost i opoziv, a opisana je i struktura CIFAR-10 skupa podataka.

Kod je pisan u programskom jeziku Python (verzija 3.7), a za izradu koda korištene su knjižnice tensorflow, matplotlib, numpy i scikit-learn. Uspoređen je rad mreže na serijama za trening i na testnoj seriji te je model mreže evaluiran i koristeći predviđanja. Predložena su i moguća poboljšanja mreže.

Ključne riječi: konvolucijska neuronska mreža, CIFAR-10, matrica konfuzije, Accuracy, Precision, Recall, Python

SUMMARY

In this thesis convolutional naural network is trained on CIFAR-10 database and it's work is evaluated using evaluation functions. Theoretical part of the thesis explains artificial neural networks work principles along with convolutional neural networks architecture and funcion. The concept of confusion matrix is introduced to help with explanation of evaluation functions accuracy, precision and recall, and the structure of CIFAR-10 database is described as well.

The code is written in Python programming language (version 3.7), and libraries tensorflow, matplotlib, numpy and scikit-learn were used to write the code. Network performance on training batches was compared with the performance on a test bach, but the network model was evaluated using predictions too. Possible network improvemets are also sugested in this thesis.

Key words: convolutional neural network, CIFAR-10, confusion matrix, Accuracy, Precision, Recall, Python

1. UVOD

Umjetna inteligencija područje je računalne znanosti koje u današnje vrijeme prati izuzetan napredak i razvoj. Ona pomaže pri rješavanju problema u raznim područjima poput medicine, robotike, upravljanja te olakšava svakodnevne zadatke u životima običnih ljudi. Jedan od glavnih ciljeva umjetne inteligencije ostvarenje je imitacije ljudskog mozga kao najsloženijeg oblika inteligencije trenutno poznatog čovjeku. Razvoj medicine, istraživanja i sve bolje razumijevanje ljudskog neurološkog sustava, omogućuju znanstvenicima i inženjerima da se sve bliže primiču ostvarenju svog cilja. Po uzoru na ljudski živčani sustav, nastale su umjetne neuronske mreže, jedan od najkorištenijih alata u procesima strojnog učenja. Neuronske mreže sastoje se od mnogo umjetnih neurona, od kojih svaki može pohranjivati informacije ne koristeći se pritom memorijom računala i obrađivati ih istodobno u puno većoj količini nego što to radi središnja procesorska jedinica računala. U funkcionalnosti neuronskih mreža nedvojbeno leži ogroman potencijal za budućnost, a trenutno najbolje rezultate ostvaruju u predviđanjima i klasifikaciji. Iznimno dobri rezultati u obradi i analizi slika postižu se korištenjem posebne vrste neuronskih mreža zvane konvolucijske neuronske mreže.

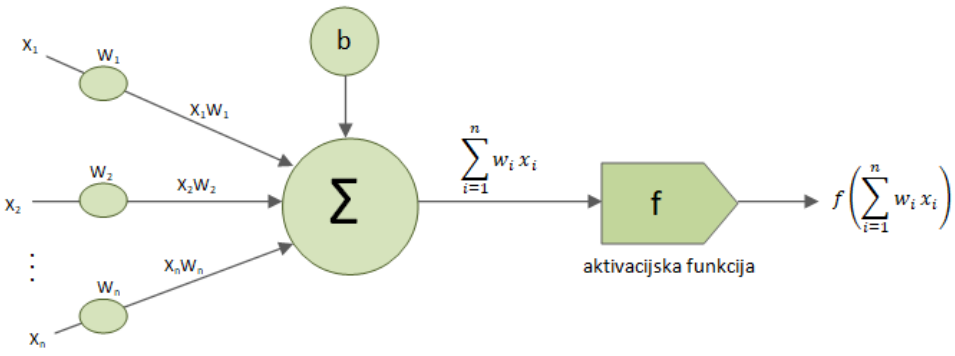
U ovome radu konvolucijske neuronske mreže korištene su za raspoznavanje i klasifikaciju slika iz CIFAR-10 skupa podataka. Raznim istraživanjima utvrđena je ljudska točnost klasifikacije slika iz ovog seta podataka od samo 94%. Slike su vrlo malih dimenzija tako da čak ni ljudski mozak ne može postići stopostotni učinak, predstavljajući tako izazov mnogim inženjerima, programerima i znanstvenicima. Kako bi se mogla procijeniti kvaliteta pojedinih neuronskih mreža, razvijene su evaluacijske tehnike koje pomažu i u njihovom razumijevanju. Upravo je analiza rada konvolucijskih neuronskih mreža korištenjem takvih evaluacijskih tehnika fokus ovog završnog rada, stoga će u nastavku biti objašnjeni bitni pojmovi potrebni za shvaćanje procesa evaluacije.

2. UMJETNE NEURONSKE MREŽE

Umjetne neuronske mreže predstavljaju jednu od najkorištenijih tehnologija za klasifikaciju slika, odnosno u području računalnog vida uopće. [1] Motivacija iza istraživanja i razvijanja umjetnih neuronskih mreža krije se u njihovoj sličnosti s biološkim sustavom sastavljenog od velikog broja nervnih ćelija, čija je jedna od najvažnijih karakteristika paralelno procesuiranje podataka i sposobnost učenja. [2] U ovom poglavlju objašnjene su ključne stavke neuronskih mreža koje predstavljaju podlogu za razumijevanje konvolucijskih neuronskih mreža.

2.1. Model neurona

Umjetni neuron po uzoru na biološki prima ulaze (potencijale) od susjednih neurona, obrađuje ih i pretvara u izlazne signale koje tada šalje sljedećim neuronima. Spojevi dvaju neurona nazivaju se sinapse i imaju svoju težinsku vrijednost, odnosno težinu. Težine su prilagodljivi parametri, koji se množe s ulaznim varijablama, a zbroj svih umnožaka ulaza i težina za jedan neuron predstavlja njegovu težinsku sumu. Na tu se sumu zatim primjenjuje aktivacijska funkcija te se u konačnici dobije izlazni signal. [3] Označimo li ulazne signale s $x_1, x_2, \ldots, x_n$, težine sinapsa s $w_1, w_2, \ldots, w_n$ te pomak po osi apscisa (eng. bias) s b, možemo prikazati građu umjetnog neurona i proces prijenosa signala na sljedeći način:

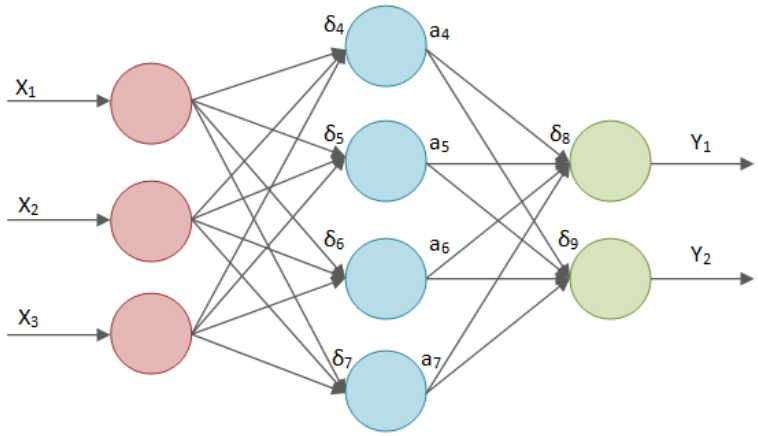


Slika 1. Građa umjetnog neurona

2.2. Model umjetne neuronske mreže

Na slici [Slika 2] je prikazan osnovni model neuronske mreže, višeslojna mreža bez povratnih veza. Krugovi predstavljaju neurone, a svaki stupac neurona predstavlja jedan sloj neuronske mreže. Ona se mora sastojati od ulaznog sloja (crveno) i izlaznog sloja (zeleno) te može imati

jedan ili više skrivenih slojeva (plavo). Uobičajeno je da svaki neuron (osim onih u ulaznom sloju, koji ulaze dobivaju izravno iz podataka) ima vezu sa svakim neuronom iz prethodnog sloja te se takvi slojevi tada nazivaju potpuno povezani slojevi. Sinapse služe za izračunavanje pobude delta δ, a malim slovom a označava se izlaz iz skrivenih slojeva nakon primjene aktivacijske funkcije na pobudu. [1]



Slika 2. Građa umjetne neuronske mreže

2.3. Proces učenja neuronskih mreža

Učenje (eng. learning, training) neuronskih mreža podrazumijeva iterativni postupak optimiranja težinskih faktora na osnovi proračunate greške između proračunate vrijednosti i stvarne vrijednosti mjerene veličine, kako bi mreža mogla generirati točne izlaze. Takvo podešavanje težina odvija se po jednom od pravila učenja poput npr. pravila širenja unatrag ili algoritma unazadne propagacije izlazne pogreške (eng. back propagation). Prilikom jednog prolaza informacije kroz neuronsku mrežu generira se vrijednost koja se uspoređuje sa stvarnom vrijednošću te se na temelju razlike tih dviju vrijednosti ispravljaju težinski faktori. Njihovom korekcijom neuronska mreža uči predviđati stvarne vrijednosti i smanjuje se razlika stvarnih i predviđenih vrijednosti izlaza.

Provjerom mreže na novom skupu podataka sprječava se tako zvano „pretreniranje“ mreže, koje se javlja kada mreža opisuje skup podataka na kojem je razvijana, a izvan tog skupa daje lošije rezultate. [4]

3. KONVOLUCIJSKE NEURONSKE MREŽE

Konvolucijska neuronska mreža (eng. convolutional neural network – CNN) specijalizirana je vrsta neuronske mreže, koja primijenjena u radu s dvodimenzionalnim slikovnim podatcima daje izvrsne rezultate, iako može biti korištena i za analizu jednodimenzionalnih i trodimenzionalnih podataka. Mreža je dobila ime po svom konvolucijskom sloju, koji izvodi operaciju „konvolucije“. Konvolucija je linearna operacija koja uključuje množenje težinskih faktora s ulazima, slično kao i kod uobičajenih neuronskih mreža. [5]

Glavno svojstvo ovakvih mreža sposobnost je prepoznavanja obilježja slika poput svijetlih ili tamnih mrlja, različito usmjerenih rubova, uzoraka i slično. Upravo na tim obilježjima temelji se prepoznavanje apstraktnih obilježja kao što su na primjer mačje uši, pseća njuška, ljudsko oko ili osmerokutni znak za obavezno zaustavljanje. Klasičnu neuronsku mrežu bilo bi teško naučiti prepoznavati takva obilježja na temelju piksela ulaznih slika jer se vrijednosti piksela mogu drastično mijenjati ovisno o položaju, orijentaciji i veličini objekta na slici. Bile bi potrebne ogromne količine podataka za učenje jer bi takva mreža mogla prepoznati objekt samo u uvjetima u kakvima se pojavljuje u setu za učenje. Konvolucijske neuronske mreže pomažu u smanjivanju količine podataka potrebnih za učenje. [3]

3.1. Arhitektura konvolucijske neuronske mreže

Uzme li se u obzir da su ulazni podatci slike, konvolucijske neuronske mreže prikazuju ih u tri dimenzije (volumni raspored). Te dimenzije predstavljaju visinu i širinu, koje ovise o visini i širini slike, te dubinu. Dubina definira boje na slikama, te sukladno tome najčešće ima tri (za RGB sustav boja) ili jednu vrijednost (za crno-bijele ili jednobojne slike).

Konvolucijska neuronska mreža sastavljena je od niza slojeva, a svaki sloj vrši određenu transformaciju prethodnog sloja. Na kraju mreža proizvede izlaz, odnosno mapu značajki (eng. feature map). Osnovne komponente konvolucijskih neuronskih mreža su:

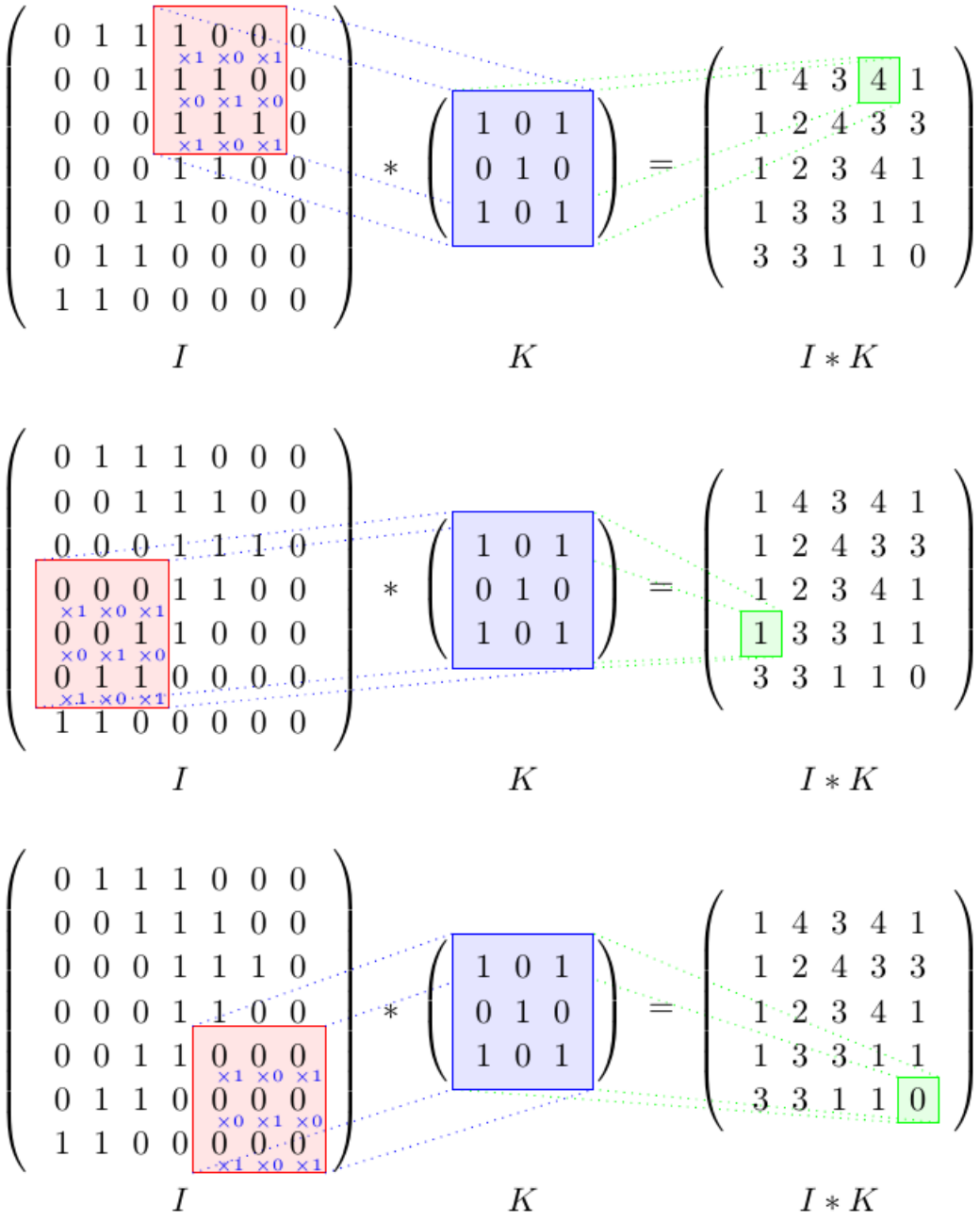
- **Ulazni sloj –** odnosi se na ulaznu sliku i čuva vrijednosti piksela. Na primjer za sliku visine 32, širine 32 i dubine 3, dimenzije će biti 32x32x3.
- **Konvolucijski sloj** – konvulira sliku određenim brojem filtera u svrhu dobivanja izlazne slike.

- **Aktivacijska funkcija –** dodaje nelinearnosti u skrivene slojeve mreže, pri čemu se ne mijenjaju dimenzije ulaznih podataka.
- **Sloj sažimanja –** reducira visinu i širinu 2D aktivacijskih mapa uz zadržavanje dubine.
- **Potpuno povezani slojevi** – kao i kod klasičnih neuronskih mreža sadrže neurone, od kojih je svaki povezan sa svim neuronima iz prethodnog sloja, a izlaz je vektor rezultata. [6]

3.1.1. Konvolucijski sloj

U konvolucijskom se sloju provodi operacija konvolucije, po čemu je mreža i dobila ime. Što je taj sloj dublji, to mreža može prepoznati više obilježja na slici. Tako na primjer prvi sloj može prepoznati rubove na slici, drugi sloj spaja te rubove u složenije oblike poput krugova ili pravokutnika, a treći sloj detektira još složenije značajke poput dijelova tijela.

Konvolucijski sloj (CONV sloj) sadrži filtere, odnosno kernele, koji se sastoje od težina, koje je potrebno naučiti tijekom procesa treniranja mreže. Filteri su manjih prostornih dimenzija od dimenzija slike, ali im dimenzija dubine ostaje ista. Primjerice za ulaznu sliku dimenzija 32x32x3, filter može biti dimenzija 5x5, ali moraju postojati tri takve matrice, po jedna za svaku komponentu boje. Proces konvolucije sastoji se od toga da se filter pomiče unaprijed po širini i visini ulaznog volumena slike počevši od gornjeg lijevog kuta. Pri svakom pomaku filtera računa se dvodimenzionalna konvolucija između vrijednosti filtera i ulaznih vrijednosti te se dobiva skalarni produkt. Krajnji rezultat konvolucije bit će dvodimenzionalna aktivacijska mapa, koja daje odgovor tog filtera na svakoj poziciji. Učenje filtera provodi se tako što se oni aktiviraju kada prepoznaju određeno obilježje. Aktivacijske mape spremaju se duž dimenzije dubine i proizvode izlazni volumen. [2]



Slika 3. Primjer konvolucije s filterom veličine 3x3

Dimenzije izlazne mape ovise o sljedećim hiperparametrima:

- **Veličina filtera** – određena visinom i širinom matrice filtera, na primjer filter veličine 3x3 imat će 9 težina.
- **Broj filtera**, odnosno **dubina** – svaki od filtera traži određene značajke u ulaznom volumenu, stoga je prednost veće dubine zapažanje više obilježja slike.

- **Dopunjavanje** – utječe na širinu i visinu izlazne mape nadopunjavanjem mape značajki nulama oko rubova. Također, time se omogućava veći broj konvolucija filterom.
- **Korak** – broj piksela za koji se filter pomiče u horizontalnom i vertikalnom smjeru prilikom konvolucije. Ne preskačemo li piksele, korak će biti 1, a s povećanjem koraka, preskače se veći broj piksela i smanjuje se izlazni volumen. [6]

Dimenzije nove slike računaju se prema sljedećoj formuli:

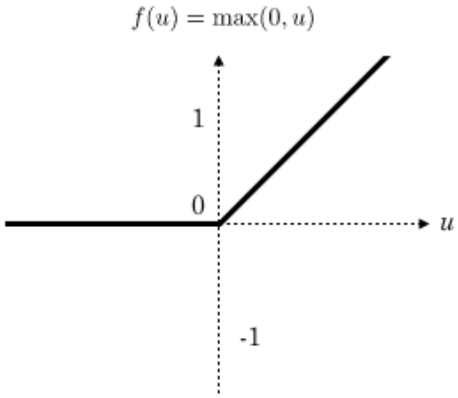
$$D_{nova} = \frac{D - F + 2P}{S} + 1 \qquad \textbf{(1)}$$

pri čemu je *D* stara dimenzija slike, *F* dimenzija filtera, *P* broj nadopunjenih nula, a *S* korak. Dubina izlazne mape jednaka je broju filtera koji se koriste. [1]

3.1.2. Aktivacijska funkcija

Nakon konvolucije uobičajeno slijedi primjena nelinearne aktivacijske funkcije, faza često zvana kao faza detekcije (eng. detection stage). Neke od poznatih funkcija su: funkcija skoka, linearna funkcija, sigmoidna, logistička, tahn funkcija, ReLu i Softmax. Budući da će se za izradu konvolucijske neuronske mreže u ovom radu koristiti funkcija ReLu, u nastavku će samo ona biti pobliže objašnjena.

ReLu (Rectified Linear Unit) aktivacijska funkcija nelinearna je funkcija, čija derivacija nije konstantna, što znači da je moguća primjena algoritma propagacije unatrag. Odredi li se da je x ulaz funkcije, ako je x pozitivan, i izlaz će biti x, ali ukoliko je x negativan, izlaz postaje 0. Na taj je način aktiviran manji broj neurona i mreža je manje opterećena. Ipak, ReLu funkcija ima i značajni nedostatak zvan „Dying ReLu problem“, koji se pojavljuje kada je ulaz u funkciju jednak 0 ili je negativan. U tom slučaju gradijent funkcije postaje 0 i nije moguća primjena algoritma propagacije unatrag.



Slika 4. Graf ReLu aktivacijske funkcije

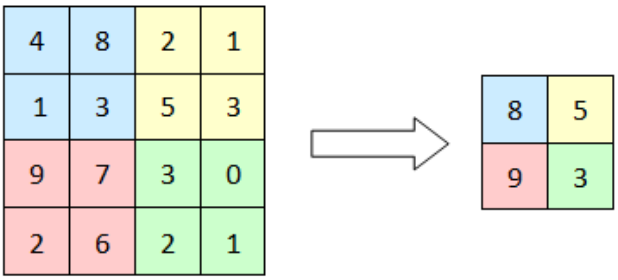
Takav problem u praksi se često rješava primjenom Leaky ReLu aktivacijske funkcije, koja u negativnom dijelu osi y ima blagi pozitivni nagib, što joj omogućuje primjenu algoritma propagacije unatrag. [2]

3.1.3. Sloj sažimanja

U sloju sažimanja (eng. pooling layer) također se pojavljuju filteri, no oni se bitno razlikuju od filtera u konvolucijskom sloju. Ovi filteri ne sadrže težine, a služe za sažimanje dimenzija slike (aktivacijske mape iz prethodnog sloja), odnosno oni izabiru vrijednosti u krugu obuhvaćenom filterom tih dimenzija. Za operaciju sažimanja najčešće se koriste sažimanje maksimalnih vrijednosti (eng. max pooling) i prosječno sažimanje (eng. average pooling). Sažimanje maksimalnih vrijednosti svodi se na pronalazak maksimalne vrijednosti unutar odabrane okoline (filtera), čime se čuvaju izraženiji pikseli slike. Koristi li se korak, dimenzije izlazne mape ovoga sloja računaju se po sljedećoj formuli:

$$D_{nova} = \frac{D - F}{S} + 1 \qquad \textbf{(2)}$$

gdje je D stara dimenzija, S korak, a F dimenzija filtera (širina ili visina). [1]



Slika 5. Sažimanje maksimalnih vrijednosti filterom veličine 2x2 i korakom 2

Prosječno sažimanje razlikuje se po tome što u zadanoj okolini pronalazi prosječnu vrijednost. Proces sažimanja pomaže kod invarijantnosti malih translacija ulaznih podataka. To znači da ako se ulazna vrijednost translatira za mali iznos, izlazna vrijednost neće se naročito promijeniti nakon primjene bilo koje od operacija sažimanja. Invarijantnost je vrlo korisna u situacijama kada je bitno znati nalazi li se neko obilježje na slici, a nije bitan podatak o lokaciji na slici. [2]

3.1.4. Potpuno povezani slojevi

Potpuno povezani sloj (eng. fully connected layer – FC) tipično se koristi na samom kraju konvolucijske neuronske mreže i identičan je potpuno povezanim slojevima u klasičnoj unaprijednoj neuronskoj mreži. Svaki neuron ovog sloja povezan je sa svakim elementom aktivacijske mape iz prethodnog sloja. Nakon nekoliko ovakvih slojeva slijedi izlaz, uobičajeno u vektorskom obliku. [6] Za klasifikaciju slika izlaz će sadržavati vrijednosti za svaku klasu, a za aktivaciju će se uobičajeno koristiti Softmax funkcija, koja transformira rezultate na izlazu u vjerojatnosti, čineći ih pogodnih za klasifikaciju:

$$S(y_i) = \frac{e^{y_i}}{\sum_j e^{y_j}} \quad \textbf{(3)}$$

4. EVALUACIJSKE FUNKCIJE

Nakon što je izrađen i treniran model neuronske mreže, mora se provjeriti njegova učinkovitost na testnom skupu podataka. Ta evaluacija pruža uvid u to koliko mreža dobro radi i kako se kvaliteta njenog rada može unaprijediti. U ovom će poglavlju biti objašnjene tehnike za evaluaciju rada konvolucijskih neuronskih mreža.

4.1. Matrica konfuzije

Slijedi objašnjenje matrica konfuzije, koje će olakšati shvaćanje principa rada evaluacijskih funkcija.

4.1.1. Binarna klasifikacija

Kod binarne klasifikacije svaki ulazni podatak pripada jednoj od dvije klase, obično označene s 1 i 0 ili pozitivno i negativno (eng. positive i negative). Proučimo problem binarne klasifikacije s klasama pozitivno i negativno na primjeru pozitivnih i negativnih ocjena na ispitu. Slijedi ispis oznaka za sedam uzoraka koji se koriste za učenje modela:

```
pozitivno, negativno, negativno, pozitivno, pozitivno, pozitivno, negativno
```

Te oznake predstavljaju temeljnu istinu (eng. ground-truth labels) uzoraka. „Temeljna istina“ točna je oznaka svakog ulaznog podatka u našem modelu, tj. idealni očekivani rezultat i nužna je za nadgledano strojno učenje. Ipak, kada se unesu podatci, povratni rezultat ne mora nužno biti oznaka klase, već može biti rezultat na ispitu, kao na primjer:

```
0.6, 0.2, 0.55, 0.9, 0.4, 0.8, 0.5
```

Uz uvjet da je prag za pozitivnu ocjenu 0.5, izlazni rezultat izgleda ovako:

```
pozitivno (0.6), negativno (0.2), pozitivno (0.55), pozitivno (0.9),
negativno (0.4), pozitivno (0.8), pozitivno (0.5)
```

Ove oznake predstavljaju predviđanja uzoraka (eng. predicted labels). Slijedi usporedba temeljne istine i predviđanja:

TEMELJNA ISTINA	pozitivno	negativno	negativno	pozitivno	pozitivno	pozitivno	negativno
PREDVIĐANJA	pozitivno	negativno	pozitivno	pozitivno	negativno	pozitivno	pozitivno

Tablica 1. Usporedba temeljne istine i predviđanja za primjer ocjena na ispitu

Na prvi pogled odmah se mogu uočiti 4 točne i 3 netočne pretpostavke. Kako bi se dobio bolji uvid u učinkovitost modela, koristi se matrica konfuzije (eng. confusion matrix). To je matrica veličine 2x2, u kojoj se oznake „pozitivno“ i „negativno“ za retke odnose na temeljnu istinu, a za stupce na predviđanja.

		Predviđanja	
		Pozitivno	Negativno
Temeljna istina	Pozitivno	Točno Pozitivno	Netočno Negativno
	Negativno	Netočno Pozitivno	Točno Negativno

Tablica 2. Matrica konfuzije

Četiri ćelije matrice predstavljaju četiri mogućnosti za točna i netočna predviđanja. Točno znači da se podudaraju temeljna istina i predviđanje, a Netočno označava neusklađenost predviđanja i temeljne istine.

Točno Pozitivno = TP = točno predviđena klasa pozitivno

Točno Negativno = TN = točno predviđena klasa negativno

Netočno Pozitivno = NP = netočno predviđena klasa pozitivno

Netočno Negativno = NN = netočno predviđena klasa negativno

Ako se u matricu konfuzije unesu podatci iz primjera pozitivnih i negativnih ocjena na ispitu dobije se sljedeći prikaz:

		Predviđanja	
		Pozitivno	Negativno
Temeljna istina	Pozitivno	3	1
	Negativno	2	1

Tablica 3. Matrica konfuzije za ocjene na ispitu

4.1.2. Višeklasna klasifikacija

Promotrimo primjer s 9 uzoraka, od kojih svaki pripada jednoj od sljedećih klasa: bijelo, crno ili crveno. Slijedi prikaz temeljne istine ulaznih podataka i oznaka predviđanja, koje se dobiju nakon unosa podataka u model:

Temeljna istina: `crveno, crno, crveno, bijelo, bijelo, crveno, crno, crveno, bijelo`

Predviđanja: `crveno, bijelo, crno, bijelo, crveno, crveno, crno, bijelo, crveno`

Prije stvaranja konfuzijske matrice mora se odrediti ciljana klasa. Ako se odredi da je ciljana klasa crvena, ona će sada biti označena kao „pozitivno“, a sve ostale klase kao „negativno“:

Temeljna istina: `pozitivno, negativno, pozitivno, negativno, negativno, pozitivno, negativno, pozitivno, negativno`

Predviđanja: `pozitivno, negativno, negativno, negativno, pozitivno, pozitivno, negativno, negativno, pozitivno`

Sada kada su definirane samo dvije klase, matrica konfuzije računa se po principu iz prethodnog poglavlja uz napomenu da se računa matrica samo za crvenu klasu.

klasa CRVENO			
		Predviđanja	
		Pozitivno	Negativno
Temeljna istina	Pozitivno	2	2
	Negativno	2	3

Tablica 4. Matrica konfuzije za klasu crveno

Po istome principu slijedile bi i matrice za bijelu i crnu klasu.

klasa BIJELO			
		Predviđanja	
		Pozitivno	Negativno
Temeljna istina	Pozitivno	1	2
	Negativno	2	4

Tablica 5. Matrica konfuzije za klasu bijelo

klasa CRNO			
		Predviđanja	
		Pozitivno	Negativno
Temeljna istina	Pozitivno	1	1
	Negativno	1	6

Tablica 6. Matrica konfuzije za klasu crno

Popularna Pythonova knjižnica (eng. library) Scikit-learn sadrži modul kojim se računa metrika u matricama konfuzije. Za problem binarne klasifikacije koristi se funkcija `confusion_matrix()`, kojoj pripadaju parametri:

- `y_true`: oznake temeljne istine
- `y_pred`: oznake predviđanja

Isti parametri koriste se i za probleme s više klasa, ali u tom se slučaju poziva funkcija `multilabel_confusion_matrix()`. U ovom se slučaju pojavljuje i treći parametar imena `labels` (oznake) s listom oznaka klasa. [7]

4.2. Točnost

Točnost (eng. Accuracy) je evaluacijska funkcija koja opisuje kako model radi kroz sve klase, a posebno je korisna kada su sve klase jednako važne. Računa se kao omjer broja točnih pretpostavki i broja ukupnih pretpostavki:

$$Točnost = \frac{Broj\ točnih\ pretpostavki}{Ukupan\ broj\ pretpostavki} \quad \textbf{(4)}$$

U binarnoj klasifikaciji može se računati s obzirom na pozitive i negative:

$$Točnost = \frac{TP + TN}{TP + TN + NP + NN} \quad \textbf{(5)}$$

Izračunajmo točnost za primjer modela koji klasificira 100 tumora kao maligni (klasa pozitivno) ili benigni (klasa negativno):

		Predviđanja	
		Pozitivno (maligni)	Negativno (benigni)
Temeljna istina	Pozitivno (maligni)	TP = 1	NP = 1
	Negativno (benigni)	NN = 8	TN = 90

Tablica 7. Matrica konfuzije za primjer tumora

$$Točnost = \frac{1 + 90}{1 + 90 + 1 + 8} = 0{,}91 = 91\% \quad \textbf{(6)}$$

Dobije se točnost od 91%, što znači da je 91 tumor od ukupnih 100 ispravno klasificiran. Na prvi pogled moglo bi se reći da ovaj model generira prilično dobre rezultate, ali pažljivijom analizom rezultata, lako je uočiti da je od ukupnih 9 malignih tumora ($TP + NN = 1 + 8 = 9$) točno prepoznat samo 1. To je jako loš ishod jer upućuje na činjenicu da 8 malignih tumora neće uopće biti dijagnosticirano i, sukladno tome, neće biti ni liječeno.

Pri klasifikaciji u kojoj klase nemaju istu važnost, kao što je slučaj u ovom primjeru, nije dovoljno izračunati samo točnost da bi se mogla uspješno procijeniti kvaliteta rada modela, već je potrebno izračunati preciznost i opoziv. [8]

4.3. Preciznost

Evaluacijska funkcija preciznost (eng. Precision) daje odgovor na pitanje koliko je uzoraka svrstanih u klasu pozitivno ispravno klasificirano, stoga se računa kao omjer točno predviđenih pozitiva i ukupnih pozitiva.

$$Preciznost = \frac{TP}{TP + NP} \quad \textbf{(7)}$$

Što je preciznost veća, to znači da je veći broj pozitiva ispravno klasificiran, a manji broj predviđenih pozitiva klasificiran je netočno. Zaključno, što je preciznost veća, to više možemo vjerovati modelu da donese točnu pretpostavku.

Promotri li se primjer klasifikacije malignih i benignih tumora, uočava se sljedeći ishod:

$$Preciznost = \frac{1}{1 + 1} = 0{,}5 = 50\% \quad \textbf{(8)}$$

Predvidi li ovaj model da je tumor maligni, bit će točan u 50% slučajeva.

4.4. Opoziv

Evaluacijska funkcija opoziv (eng. Recall) pokušava dati odgovor na pitanje koji je udio pozitiva temeljne istine ispravno klasificiran, stoga se računa kao omjer točno pretpostavljenih pozitiva i ukupnog broja pozitiva.

$$Opoziv = \frac{TP}{TP + NN} \quad \textbf{(9)}$$

Funkcija opoziv fokusirana je samo na klasifikaciju uzoraka iz klase pozitivno, neovisno o klasifikaciji uzoraka iz klase negativno. Uvrštavajući rezultate iz primjera s tumorima dobiva se sljedeći ishod:

$$Opoziv = \frac{1}{1 + 8} = 0{,}1111 = 11{,}11\% \quad \textbf{(10)}$$

Iako preciznost upućuje na to da je od ukupnog broja identificiranih malignih tumora njih 50% ispravno dijagnosticirano, tek nakon izračunatog opoziva zaključuje se da velik dio malignih tumora uopće nije točno prepoznat. [9]

5. CIFAR-10 set podataka

U ovom radu konvolucijska neuronska mreža trenirat će se na CIFAR-10 skupu podataka.

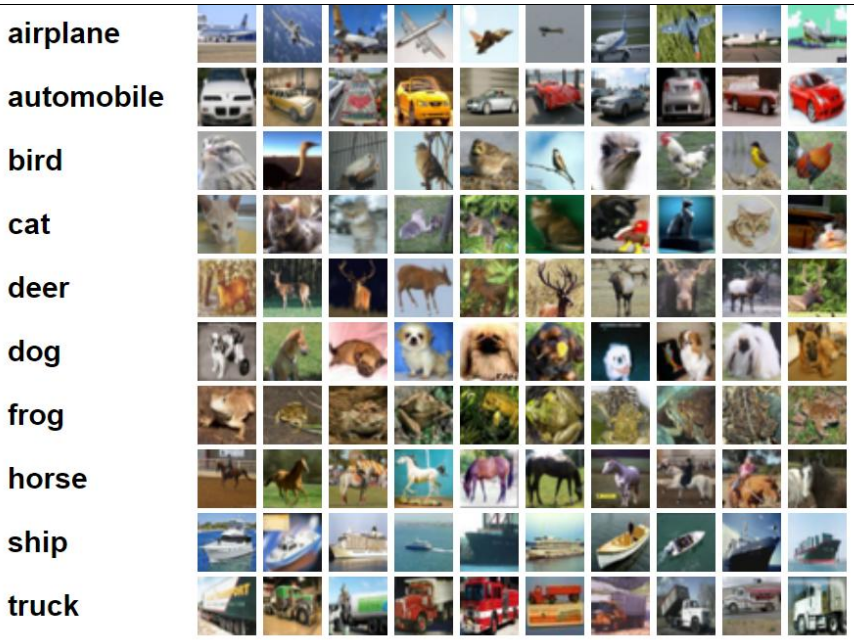
Alex Krizhevsky, Vinod Nair i Geoffrey Hinton zajedno su generirali skup podataka, koji se sastoji od 80 milijuna sitnih slika, a CIFAR-10 i CIFAR-100 podskupovi su tog skupa podataka.

5.1. CIFAR-10

CIFAR-10 skup je podataka uspostavljen za raspoznavanje objekata, koji sadrži 60 000 slika u boji veličine 32x32 piksela. Slike su podijeljene u 10 klasa (eng. class) od kojih svaka sadrži 6 000 slika. Klase (avioni, automobili, ptice, mačke, jeleni, psi, žabe, konji, brodovi, kamioni) su međusobno posve isključive, što znači da nema preklapanja između klasa. Na primjer, klasa „automobili“ sadrži slike limuzina, džipova i slično, a klasa „kamioni“ samo slike velikih kamiona. Nijedna od te dvije klase ne sadrži slike kamioneta, koji bi se teoretski mogli svrstati u obje skupine.

U CIFAR-10 skupu nalazi se 50 000 slika za učenje (eng. training image) i 10 000 testnih slika (eng. test image). Sam skup podataka podijeljen je u pet serija za učenje (eng. training batch) i jednu testnu seriju (eng. test batch) od kojih svaka sadrži 10 000 slika. Testna serija sadrži točno 1 000 nasumično odabranih slika iz svake klase, ali serije za učenje ne moraju sadržavati jednak broj slika iz svih klasa, kada se zbroje slike iz svih serija za učenje, dobije se ukupno 5 000 slika iz svake klase.

Na slici [Slika 6] je prikazan set podataka s 10 nasumično odabranih slika iz svake klase:



Slika 6. CIFAR-10

5.2. Python

Slijedi objašnjenje izgleda skupa podataka u programskom jeziku Python.

Arhiva sadrži datoteke `data_batch_1`, `data_batch_2`, …, `data_batch_5` (serije za učenje), kao i `test_batch` (testna serija). Svaka od ovih datoteka predstavlja Pythonov „pickled“ objekt te se one otvaraju na sljedeći način, nakon čega se dobiva povratni rječnik (eng. dictionary):

```
def unpickle(file):
    import pickle
    with open(file, 'rb') as fo:
        dict = pickle.load(fo, encoding='bytes')
    return dict
```

Na ovaj način, svaka datoteka serije sadrži rječnik sa sljedećim elementima:

- Podatci (eng. data) – numpy niz (eng. array) veličine 10 000x3072. Svaki od 10 000 redaka pohranjuje po jednu sliku u boji veličine 32x32 piksela. Prva 1024 (32x32=1024) člana retka predstavljaju vrijednosti crvene boje, sljedeća 1024 vrijednosti zelene boje, a zadnja 1024 plave boje. Slika je pohranjena po retcima s lijeva na desno (kao što se, na primjer, inače čita tekst), što znači da prva 32 člana niza predstavljaju vrijednosti crvene boje za prvi redak slike.

- Oznake (eng. labels) – lista 10 000 brojeva od 0 do 9. Broj indeksa nekog člana liste *i* upućuje na *i*-tu sliku, odnosno *i*-ti redak iz niza podataka.

Skup podataka sadrži i datoteku imena `batches.meta`, koja također sadrži Python rječnik, a ima sljedeće unose:

- Oznake imena (eng. label names) – lista 10 elemenata, koja daje konkretna imena brojevima u listi oznaka. Na primjer: `label_names[0] == "airplane"`, `label_names[1] == "automobile"`

[10]

6. Klasifikacija CIFAR-10 skupa podataka pomoću konvolucijske neuronske mreže

Za izradu konvolucijske neuronske mreže za klasifikaciju slika iz CIFAR-10 skupa podataka u ovom je radu korištena Pythonova knjižnica TensorFlow i njeno Keras sekvencijalno sučelje za programiranje aplikacija (eng. Keras Sequential API), stoga je za izradu i treniranje mreže potrebno tek nekoliko linija koda. [11]

6.1. Model mreže

Pomoću Keras API preuzet je CIFAR-10 skup podataka, odnosno slike i oznake za trening (`train_images`, `train_labels`) te testne slike i oznake (`test_images`, `test_labels`).

```
(train_images, train_labels), (test_images, test_labels) =
datasets.cifar10.load_data()
```

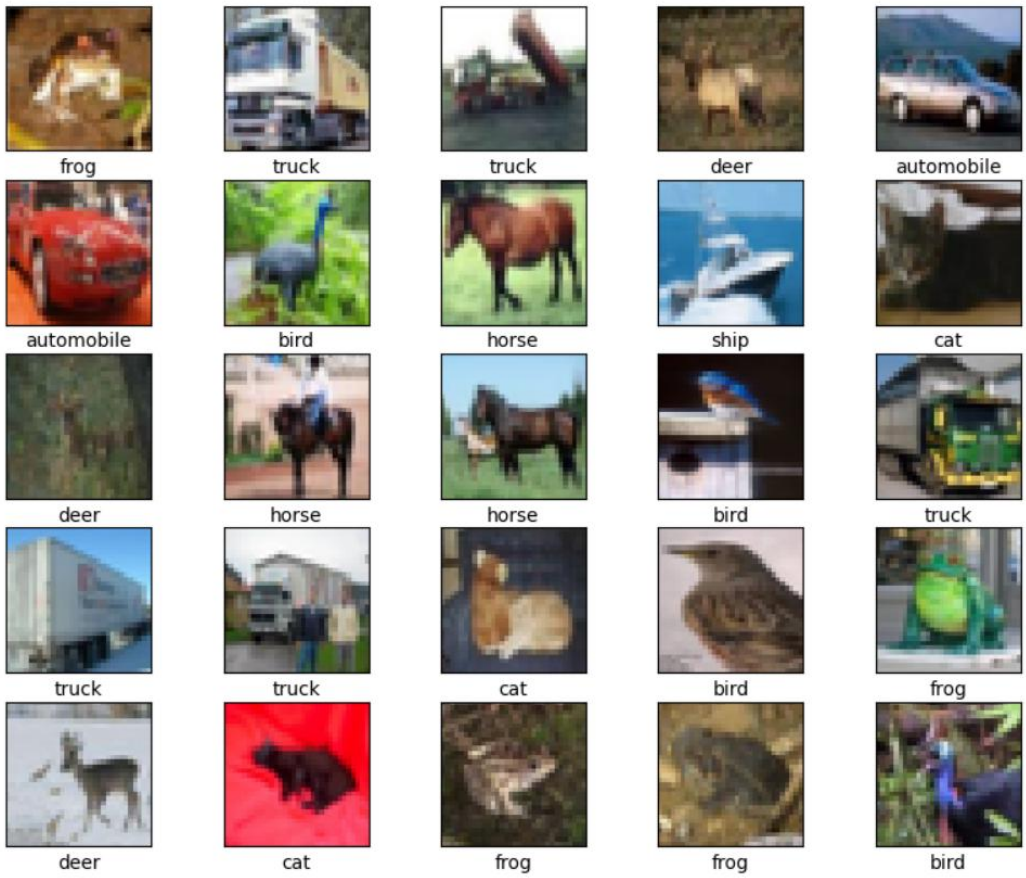
Budući da su boje prikazane RGB sustavom u kojem vrijednosti mogu varirati [0, 255], prvo se moraju normalizirati, odnosno podijeliti s 255 kako bi se dobile vrijednosti [0,1].

```
train_images, test_images = train_images / 255.0, test_images
/ 255.0
```

Kreira se lista imena klasa te se, kako bi se provjerila ispravnost podataka, generira mreža prvih 25 slika iz serije za učenje uz prikaz imena pripadajuće klase ispod svake slike:

```
class_names = ['airplane', 'automobile', 'bird', 'cat',
'deer', 'dog', 'frog', 'horse', 'ship', 'truck']
plt.figure(figsize=(10, 10))
for i in range(25):
    plt.subplot(5, 5, i+1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(train_images[i], cmap=plt.cm.binary)
    plt.xlabel(class_names[train_labels[i][0]])
plt.show()
```

Slijedi prikaz generirane mreže:



Slika 7. Mreža prvih 25 CIFAR slika

Lako je uočljivo da su slike vrlo niske rezolucije, a upravo je to najvjerojatniji uzrok ograničenog djelovanja koje vrhunski algoritmi mogu postići na ovom skupu podataka.

Koristeći Keras Sequential API sljedećih šest linija koda definiraju konvolucijsku bazu služeći se čestim obrascem – niz Conv2D konvolucijskih slojeva i MaxPooling2D slojeva sažimanja.

```
model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu',
input_shape=(32, 32, 3)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
```

U konvolucijskim slojevima definiran je broj filtera te njihove dimenzije, aktivacijska funkcija (ReLu) te dimenzije ulaza. Mreža za ulaz prima tenzore oblika (visina slike, širina slike, kanali boje) ignorirajući pritom veličinu serije. U ovom slučaju tenzor oblika je (32, 32, 3) jer su to formati CIFAR slika. U sloju sažimanja definirane su dimenzije filtera za sažimanje.

Trenutno model izgleda ovako:

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 30, 30, 32)        896
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 15, 15, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 13, 13, 64)        18496
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 6, 6, 64)          0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 4, 4, 64)          36928
=================================================================
Total params: 56,320
Trainable params: 56,320
Non-trainable params: 0
_________________________________________________________________
```

Izlaz svakog Conv2D i MaxPooling2D sloja trodimenzionalni je tenzor oblika, koji kao i na samom ulazu predstavlja visinu, širinu i dubinu. Dimenzije visine i širine smanjuju se što se više ide u dubinu mreže.

Zadnji izlazni tenzor konvolucijske baze provlači se kroz još jedan ili više gustih slojeva (eng. Dense layers) kako bi se izvršila klasifikacija. Potrebno je prije toga „izravnati“ (eng. flatten) trodimenzionalni izlaz u jednodimenzionalni jer gusti slojevi za ulaz primaju samo jednodimenzionalne vektore. Dakle, nakon što se tenzor izravna i prođe kroz jedan gusti sloj, finalno prolazi kroz gusti sloj, ujedno i zadnji sloj mreže, za koji je određeno da ima 10 izlaza

za 10 mogućih klasa. To znači da će izlaz nakon prolaska kroz mrežu biti vektor duljine 10 s 10 različitih vrijednosti.

```
model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(10))

model.summary()
```

Struktura modela u konačnici izgleda ovako:

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 30, 30, 32)        896
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 15, 15, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 13, 13, 64)        18496
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 6, 6, 64)          0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 4, 4, 64)          36928
_________________________________________________________________
flatten (Flatten)            (None, 1024)              0
_________________________________________________________________
dense (Dense)                (None, 64)                65600
_________________________________________________________________
dense_1 (Dense)              (None, 10)                650
=================================================================
Total params: 122,570
Trainable params: 122,570
Non-trainable params: 0
_________________________________________________________________
```

Sada, kada su definirani slojevi konvolucijske neuronske mreže, funkcijom `model.compile` sastavlja se model za učenje. Određeno je korištenje optimizacije koja implementira algoritam „Adam“ te računanje gubitaka (eng. loss) i točnosti za svaku epohu. Za funkciju gubitaka koristi se SparseCategoricalCrossentropy uz uvjet from_logits = True. To je ključno jer izlaz zadnjeg sloja mreže nije u formatu vjerojatnosti, stoga se na ovaj način prvo konvertira Softmax funkcijom u vjerojatnosti, što omogućava lakši odabir najvjerojatnije klase za pojedinu sliku. [12]

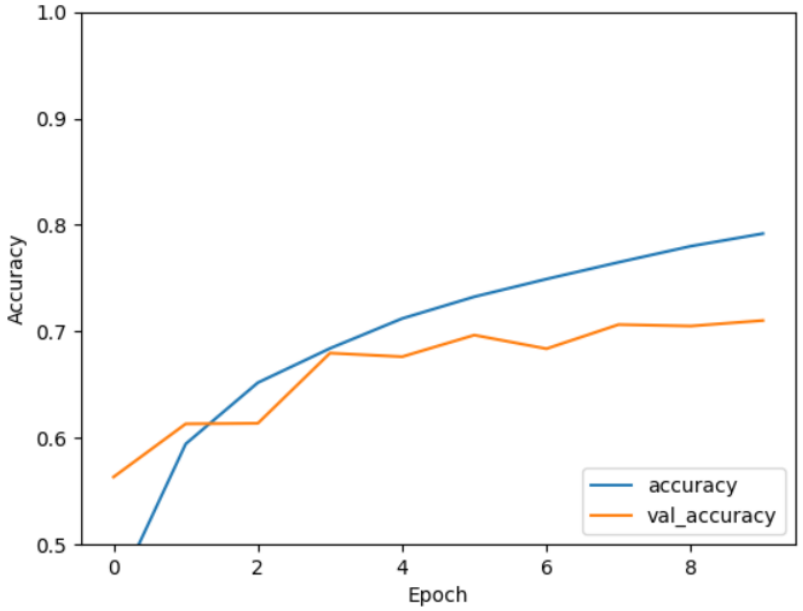
Sljedeći je korak treniranje modela koristeći funkciju `model.fit`. Slike za učenje (`train_images`) i oznake za učenje (`train_labels`), odnosno njihova temeljna istina, koriste se za učenje modela kroz 10 epoha. Broj epoha određuje broj iteracija modela kroz sve slike i oznake za učenje. Gubitak i točnost modela strojnog učenja finalno se računa na zasebnom setu podataka, u ovom slučaju su to testne slike (`test_images`) i testne oznake (`test_labels`).

```
model.compile(optimizer='adam',
loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits
=True),metrics=['accuracy'])

history = model.fit(train_images, train_labels, epochs=10,
validation_data=(test_images, test_labels)).
```

6.2. Rezultati i analiza

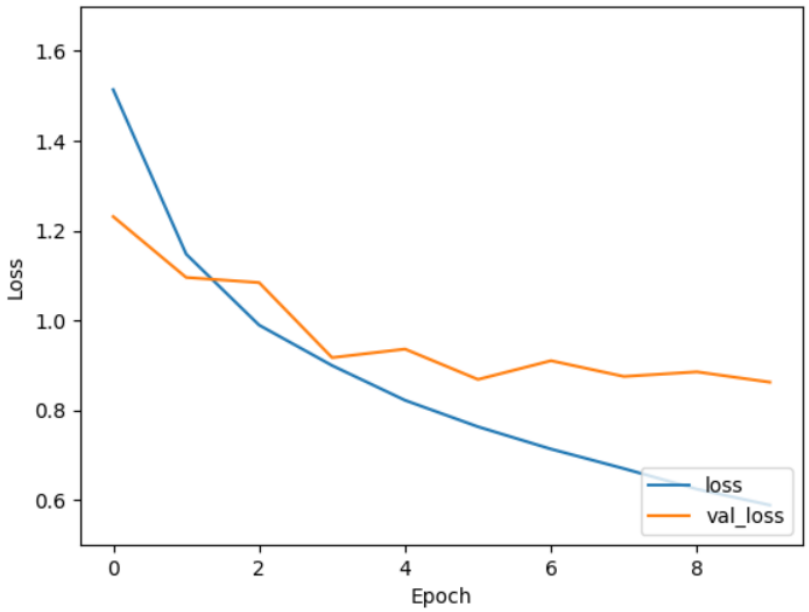
Budući da je točnost praćena tijekom svih epoha na skupu podataka i za trening i za testiranje, nakon provedenog učenja dobiva se sljedeći prikaz:



Slika 8. Grafički prikaz točnosti tijekom svih epoha

Plava linija na grafu predstavlja točnost mreže na skupu podataka koji čine slike za trening, a narančasta linija predstavlja točnost mreže za testne slike. U oba se slučaja uočava porast kroz epohe, s tim da točnost slika za trening doseže veću vrijednost. Na kraju zadnje epohe dobije se da je točnost slika za trening 0,7975, odnosno 79,75%, a točnost testnih slika 0,7103 iliti 71,03%, što je nešto lošiji rezultat.

Uz točnost, tijekom svih epoha računat je gubitak:



Slika 9. Grafički prikaz vrijednosti gubitaka tijekom svih epoha

Kao i na prethodnom grafu (Slika 8.) plava linija predstavlja gubitke, odnosno grešku mreže na slikama za trening, a narančasta linija greške na testnoj seriji. Na kraju zadnje epohe, vrijednost greške na skupu slika za trening iznosi 0,5736, a greška na testnoj seriji jednaka je 0,8627. Činjenica da se greška na skupu za učenje kontinuirano smanjuje, a na testnim podatcima ostaje približno ista (iznad 0,8) ukazuje na to da je mreža pretrenirana, odnosno da ima prostora za poboljšanje mreže.

Za evaluaciju modela koristeći predviđanja koristi se Pythonova knjižnica scikit-learn jer knjižnica Keras ne podržava računanje funkcija preciznost i opoziv. Odrede se temeljna istina (`y_true`) i predviđanja (`y_pred`) te se metrikom scikit-learn knjižnice po principu matrica konfuzija objašnjenih u jednom od prethodnih poglavlja izračunaju vrijednosti točnosti, preciznosti i opoziva. Preciznost i opoziv računaju se pomoću makro prosjeka zato što su sve klase jednake važnosti i zato što testna serija sadrži jednak broj slika iz svake klase.

```
predictions = model.predict(x=test_images,
steps=len(test_images), verbose=0)
y_true = test_labels
y_pred = np.argmax(predictions, axis=-1)

accuracy = accuracy_score(y_true, y_pred)
print('Accuracy: %f' % accuracy)

precision = precision_score(y_true, y_pred, average='macro')
print('Precision: %f' % precision)

recall = recall_score(y_true, y_pred, average='macro')
print('Recall: %f' % recall)
```

Dobije se sljedeći rezultat:

```
Accuracy: 0.710300
Precision: 0.713408
Recall: 0.710300
```

Uzimajući u obzir da je trenutni rekord točnosti klasifikacije CIFAR-10 seta podataka 99,5%, rezultat od otprilike 70% ne izgleda naročito dobro. Ipak, za potrebe ovoga rada je zadovoljavajuć jer konvolucijska neuronska mreža nije suviše kompleksna, kako bi se omogućilo lakše razumijevanje njene funkcije. Također, rekordni rezultat nije postignut

korištenjem konvolucijskih neuronskih mreža, već nešto uspješnijih vidnih transformatora (eng. Vision Transformer – ViT). [13]

Rezultat upućuje na to da se po završetku učenja mreže na testnoj seriji postiže točnost od 71,03%. Budući da se testna serija sastoji od 10 000 slika, to znači da su ispravno klasificirane 7 103 slike.

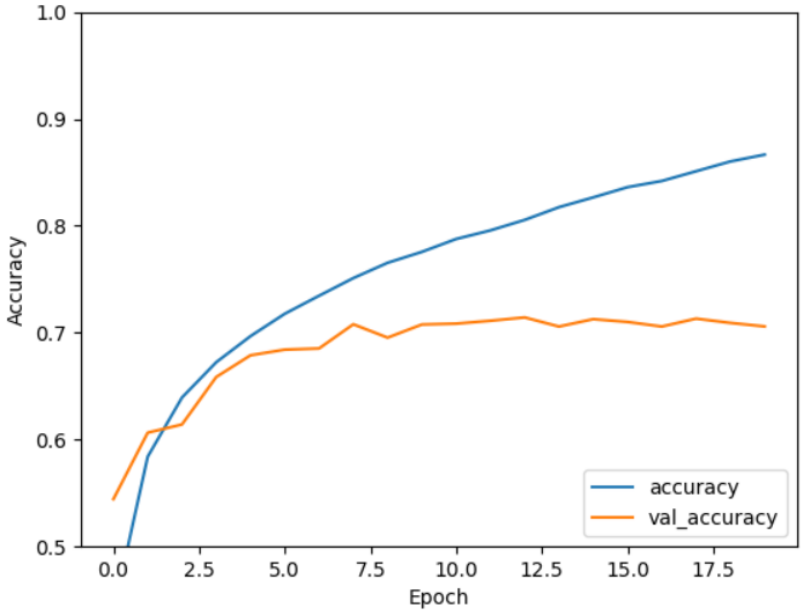
Makro prosjek preciznosti daje informaciju o prosječnoj preciznosti za sve klase, koja u ovome radu iznosi 71,34%. Dakle, za svaku je klasu izračunata preciznost, odnosno omjer točnih predviđanja za tu klasu i ukupnih predviđanja za tu klasu. Na primjer, ako je preciznost za klasu avioni jednak 80%, za klasu automobili 72% i za klasu ptice 63%, makro preciznost tih triju klasa bio bi 71,67%.

Po istome principu kao i za makro preciznost računa se i makro opoziv, koji za primjer konvolucijske mreže u ovome radu iznosi 71,03%, jednako kao i točnost.

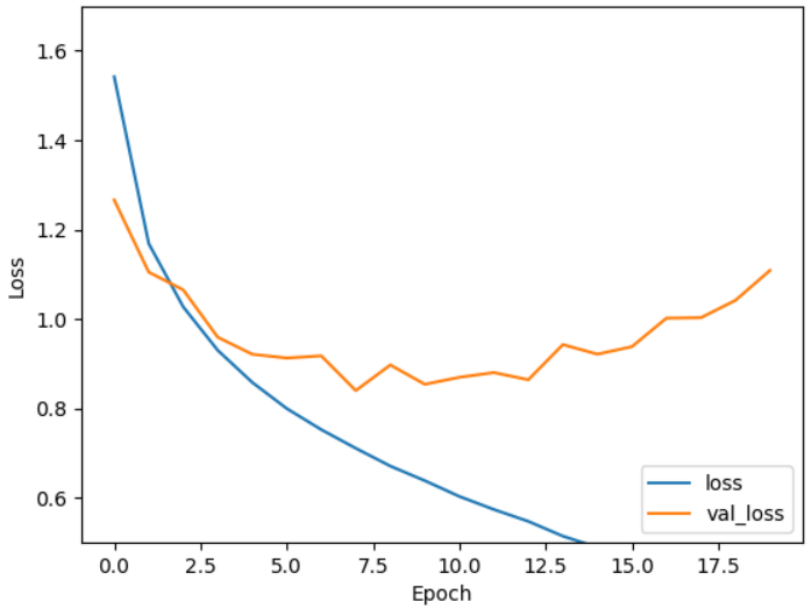
6.3. Moguća poboljšanja

Jedan od načina kako poboljšati uspješnost rada mreže svakako je produbljivanje mreže, odnosno dodavanje više konvolucijskih slojeva. U prethodnim poglavljima objašnjeno je da svaki sljedeći konvolucijski sloj može raspoznavati kompleksnija obilježja od prethodnog, stoga bi sukladno tome veća količina konvolucijskih slojeva u mreži trebala značiti i bolje rezultate. Produbljenje mreže zahtjeva i više podataka za učenje, stoga bi trebalo povećati broj slika u serijama za trening. Također, s produbljenjem mreže povećava se broj težina koje se trebaju naučiti tako da sve važnija postaje i regularizacija kako bi se smanjio rizik pretreniranosti mreže.

Logično bi bilo i povećati broj epoha tijekom učenja. No promotrimo što se događa s rezultatima nakon što se broj epoha za učenje modela mreže u ovome radu poveća s 10 na 20:



Slika 10. Grafički prikaz točnosti tijekom 20 epoha



Slika 11. Grafički prikaz vrijednosti gubitaka tijekom 20 epoha

Na grafovima točnosti (Slika 10.) i gubitaka (Slika 11.) vidljivo je da na ovom skupu podataka naša mreža postaje pretrenirana nakon otprilike 7 epoha, stoga, poveća li se broj podataka i dubina mreže, svakako treba paziti na to da broj epoha ne bude prevelik. Nakon što je mreža trenirana, rezultati točnosti i gubitaka za podatke za treniranje iznose 87,44%, odnosno 0,3528, što su znatno bolji rezultati od onih postignutih nakon samo 10 epoha. Ipak, konačni rezultat točnosti za testne podatke ponovno iznosi oko 70% (točnije 70,59%), isto

kao i preciznost (70,37%) i opoziv (70,59%), a greška iznosi 1,1088. Budući da je za primjenu mreže bitno kakve rezultate daje na novim podatcima, može se zaključiti da je u ovome slučaju s povećanjem epoha pogoršana učinkovitost mreže. S većim brojem epoha produljuje se i vrijeme potrebno za učenje mreže, na što također valja pripaziti.

7. ZAKLJUČAK

U ovome radu sažeta je teorijska podloga potrebna za razumijevanje strukture i principa rada konvolucijskih neuronskih mreža te tehnika korištenih za evaluaciju njihovog rada. Konvolucijska mreža u ovome je radu korištena za klasifikaciju CIFAR-10 baze slika. Model mreže izrađen programskim jezikom Python treniran je na pet serija za trening i testiran na jednoj testnoj seriji. Tijekom svake epohe računata je točnost i greška mreže. Rezultati pokazuju veću uspješnost mreže na podatcima za trening, što ukazuje na pretreniranost mreže. Kvaliteta predviđanja mreže na testnim podatcima analizirana je pomoću evaluacijskih tehnika točnost, preciznost i opoziv jer je potrebno poznavati sva tri parametra kako bi se mogao donijeti adekvatni zaključak o učinkovitosti rada mreže. Sva tri parametra iznose oko 70%, što nije naročito dobar rezultat. Bolji rezultati mogu se postići određenim unaprjeđenjima mreže, na primjer povećanjem kvalitete i kvantitete ulaznih podataka, produbljivanjem mreže, odnosno implementacijom više konvolucijskih slojeva u mrežu te primjenom odgovarajućeg broja epoha za učenje.

LITERATURA

[1] J. Mrđen, *Prepoznavanje rukopisa strojnim učenjem i primjena u identifikaciji popunjenih polja na uplatnici,* Zagreb: Sveučilište u Zagrebu, Fakultet elektrotehnike i računarstva, 2018..

[2] L. Mehinović, *Konvolucijske neuronske mreže za klasifikaciju objekata,* Pula: Sveučilište Jurja Dobrile u Puli, Fakultet informatike u Puli, 2020..

[3] »Elements of AI – besplatni online tečaj o osnovama umjetne inteligencije,« CroAI, Agencija 404, Sveučilište u Zagrebu, Reaktor i Sveučilište u Helsinkiju, [Mrežno]. Available: https://course.elementsofai.com/hr/5/2. [Pokušaj pristupa 24. lipanj 2021.].

[4] Ž. Ujević Andrijić, »Osvježimo znanje: Umjetne neuronske mreže,« *Kemija u industriji : Časopis kemičara i kemijskih inženjera Hrvatske,* pp. 219-220, lipanj 2019..

[5] J. Brownlee, »Machine Learning Mastery,« 17 travanj 2019.. [Mrežno]. Available: https://machinelearningmastery.com/convolutional-layers-for-deep-learning-neural-networks/. [Pokušaj pristupa 28 lipanj 2021.].

[6] K. Džomba, *Konvolucijske neuronske mreže,* Zagreb: Sveučilište u Zagrebu, Prirodoslovno-matematički fakultet, 2018..

[7] A. F. Gad, »Evaluating Deep Learning Models: The Confusion Matrix, Accuracy, Precision, and Recall,« Paperspace Blog, listopad 2020.. [Mrežno]. Available: https://blog.paperspace.com/deep-learning-metrics-precision-recall-accuracy/. [Pokušaj pristupa 30 lipanj 2021.].

[8] »Google Developers,« 10 veljača 2020.. [Mrežno]. Available: https://developers.google.com/machine-learning/crash-course/classification/accuracy. [Pokušaj pristupa 28 lipanj 2021.].

[9] »Google Developers,« 10 veljača 2020.. [Mrežno]. Available: https://developers.google.com/machine-learning/crash-course/classification/precision-and-recall. [Pokušaj pristupa 28 lipanj 2021.].

[10] A. Krizhevsky. [Mrežno]. Available: https://www.cs.toronto.edu/~kriz/cifar.htm. [Pokušaj pristupa 24. lipanj 2021.].

[11] »TensorFlow,« 17 lipanj 2021.. [Mrežno]. Available: https://www.tensorflow.org/tutorials/images/cnn. [Pokušaj pristupa 29. lipanj 2021.].

[12] S. Kalla, »Medium,« Medium, 6 lipanj 2021.. [Mrežno]. Available: https://medium.com/nerd-for-tech/classification-on-cifar-10-32abe456302. [Pokušaj pristupa 24. kolovoz 2021.].

[13] »Papers With Code,« [Mrežno]. Available: https://paperswithcode.com/sota/image-classification-on-cifar-10. [Pokušaj pristupa 24 kolovoz 2021.].

PRILOZI

I. Python kod

I. Python kod

```
import tensorflow as tf
from tensorflow.keras import datasets, layers, models
import matplotlib.pyplot as plt
import numpy as np
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score

(train_images, train_labels), (test_images, test_labels) =
datasets.cifar10.load_data()

train_images, test_images = train_images / 255.0, test_images / 255.0

class_names = ['airplane', 'automobile', 'bird', 'cat', 'deer',
               'dog', 'frog', 'horse', 'ship', 'truck']

plt.figure(figsize=(10, 10))
for i in range(25):
    plt.subplot(5, 5, i+1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(train_images[i], cmap=plt.cm.binary)
    plt.xlabel(class_names[train_labels[i][0]])
plt.show()

model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(32, 32,
3)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.summary()

model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(10))
```

```
model.summary()

model.compile(optimizer='adam',
loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])

history = model.fit(train_images, train_labels, epochs=10,
                    validation_data=(test_images, test_labels))

plt.plot(history.history['accuracy'], label='accuracy')
plt.plot(history.history['val_accuracy'], label='val_accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.ylim([0.5, 1])
plt.legend(loc='lower right')
plt.show()

plt.plot(history.history['loss'], label='loss')
plt.plot(history.history['val_loss'], label='val_loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.ylim([0.5, 1.7])
plt.legend(loc='lower right')
plt.show()

test_loss, test_acc = model.evaluate(test_images,  test_labels, verbose=2)
print(test_acc)

predictions = model.predict(x=test_images, steps=len(test_images),
verbose=0)
y_true = test_labels
y_pred = np.argmax(predictions, axis=-1)

accuracy = accuracy_score(y_true, y_pred)
print('Accuracy: %f' % accuracy)

precision = precision_score(y_true, y_pred, average='macro')
print('Precision: %f' % precision)

recall = recall_score(y_true, y_pred, average='macro')
print('Recall: %f' % recall)
```