Prepoznavanje saobraćajnih znakova pomoću konvolutivnih neuronskih mreža

Seminarski rad u okviru kursa
Računarska inteligencija
Matematički fakultet

Jana Jovičić 215/2015, Jovana Pejkić 435/2016
jana.jovicic755@gmail.com, jov4ana@gmail.com

16. maj 2019.

Sažetak

Zahvaljujući konvolutivnim neuronskim mrežama, u oblasti mašinskog učenja je unapređen proces klasifikacije slika i prepoznavanja objekata na slikama. U ovom radu će biti opisana opšta arhitektura konvolutivnih neuronskih mreža, način na koji funkcioniše izdvajanje karakteristika objekata sa slika i njeno klasifikovanje na osnovu tih karakteristika. Biće navedeni i načini na koje je moguće optimizovati mrežu. Takođe, biće prikazane implementacije tri vrste mreža, među kojima su LeNet-5 i AlexNet, a zatim će biti analiziran njihov uspeh prilikom prepoznavanja i klasifikacije saobraćajnih znakova.

Sadržaj

1 Uvod

Prepoznavanje i klasifikacija slika je polje u oblasti mašinskog učenja koje se veoma brzo razvija. Na primer, klasifikatori slika se sve više koristite za: zamenu lozinke prepoznavanjem lica, prepoznavanje otisaka prstiju, analizu krvnih slika, identifikaciju geografskih karakteristika iz satelitskih snimaka, analizu aero i satelitskih snimaka, detekciju urbanih područja, detekciju puteva, autonomnu vožnju i otkrivanje prepreka itd.

Klasifikacija je vrsta problema nadgledanog učenja[1] koja podrazumeva predviđanje kategoričke ciljne promenljive. Kategoričkim se smatraju promenljive koje uzimaju konačan broj vrednosti (među kojima nema uređenja). Na primer, prepoznavanje da li je dobijen e-mail *smeće (eng.* spam*)*, reklama ili obična poruka je problem klasifikacije.

Postoje mnogi algoritmi koji se koriste za klasifikaciju slika (npr. SVM[2]), a jedan od najboljih predstavljaju *konvolutivne neuronske mreže (eng.* convolutional neural network, CNN*)*. CNN se može zamisliti kao automatski „ekstraktor" karakteristika slike. On efikasno koristi informacije o susednim pikselima kako bi konvolucijom i agregacijom smanjio sliku, a zatim predvideo kategoriju kojoj slika pripada [3]. Iako su konvolutivne neuronske mreže projektovane tako da prednost postižu u radu sa 2D strukturama, kao što su slike ili ulazi poput *govornog signala (eng.* speech signal*)*, najnovije studije pokazuju da postižu značajne rezultate i sa 3D strukturama.

U ovom radu je predstavljeno rešavanje problema klasifikacije slika saobraćajnih znakova pomoću neuronskih mreža (koje su opisane u poglavlju 2), i to konvolutivnih neuronskih mreža (koje su opisane u poglavlju 3). Cilj ovog rada je da, kroz rešavanje ovog problema, prikaže osnovne osobine konvolutivnih neuronskih mreža kao i njihovu implementaciju. U radu je, u poglavlju 4, najpre opisano nekoliko razlicitih modela mreža, a onda su upoređeni i analizirani ostvareni rezultati.

[1] *Nadgledano učenje (eng.* supervised learning*)* je proces učenja funkcije koja preslikava ulaz u odgovarajući izlaz, zasnovan na datim ulzno-izlaznim parovima. Dve osnovne vrste problema nadgledanog učenja su regresija i klasifikacija. Regresija je problem predviđanja neprekidne, a klasifikacija kategoričke ciljne promenljive.

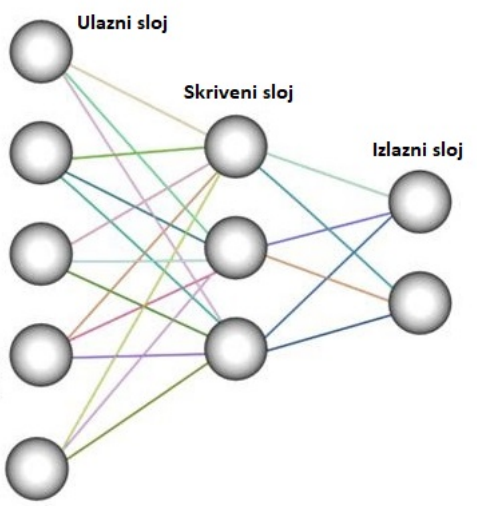
[2] *Podržavajući vektori (eng.* Support vector machines, SVM*)* predstavljaju model nadgledanog učenja koji se koristi za klasifikacionu i regresionu analizu.

2 Neuronske mreže

Neuronske mreže predstavljaju skup statističkih modela učenja inspirisan biološkim neuronima, za rešavanje klasifikacionih[3] i regresionih problema[4]. Njihove primene su mnogobrojne, a neke od njih su: kategorizacija teksta, raspoznavanje i sinteza govora, autonomna vožnja, igranje video igara, mašinsko prevođenje prirodnih jezika i slično. Ključna prednost neuronskih mreža je da same mogu da konstruišu nove atribute nad sirovom reprezentacijom podataka[5] i odlično baratanje sa velikim količinama podataka. Opis građe neuronske mreže dat je u poglavlju 2.1.

2.1 Unutrašnja struktura neuronskih mreža

Osnovnu jedinicu građe neuronske mreže predstavljaju neuroni, koji su međusobno povezani vezama sa težinama koje se podešavaju tokom učenja mreže. Povezani neuroni prosleđuju signale jedni drugima, a organizovani su u slojeve. Najjednostavniji oblik neuronske mreže je **perceptron**, koji sadrži samo jedan ulazni i jedan izlazni sloj. Međutim, kako on služi samo za učenje linearnih modela, a u praksi se javlja potreba da i složeniji modeli mogu da se nauče, osim perceptrona, koriste se višeslojne neuronske mreže. Višeslojna neuronska mreža osim ulaznog i izlaznog sloja, ima jedan ili više skrivenih slojeva (slika 1). Kako se neuronske mreže uglavnom koriste za klasifikaciju uzorka u različite kategorije, ulazni sloj se sastoji od onoliko neurona kolika je dimenzionalnost ulaznog prostora, a broj neurona na izlazu jednak je broju klasa. Samo učenje neuronske mreže je zapravo podešavanje težina sve dok se ne dobije zadovoljavajuća aproksimacija između ulaznih i izlaznih veličina.



Slika 1: Višeslojna neuronska mreža sa jednim skrivenim slojem

[3]Klasifikacioni problem - ako je izlazna promenljiva kategoričkog tipa, npr. ,,zdrav"i ,,bolestan".

[4]Regresioni problem - ako je izlazna promenjiva neprekidnog tipa, npr. ,,plata"ili ,,težina".

[5]Iako se nekad mogu pretpostaviti koji su atributi najinformativniji za predviđanje ciljne promenljive, izbori tih atributa su neretko lošiji od onoga što bi algoritam učenja mogao da detektuje u sirovoj informaciji.

2.2 Zadaci prepoznavanja i uvođenje CNN

Jednostavni zadaci prepoznavanja mogu se dosta dobro rešiti skupovima podataka malih veličina, npr. desetine hiljada slika. Međutim, objekti u realističnim postavkama pokazuju značajnu raznovrsnost, stoga, da bi bilo moguće naučiti prepoznati ih, potrebno je koristiti mnogo veće skupove za treniranje. Da bi se naučilo o hiljadama objekata iz miliona slika, potreban je model sa velikim kapacitetom učenja. Konvolutivne neuronske mreže, koje su detaljno obrađene u ostatku rada, čine jednu takvu klasu modela. One daju jake i uglavnom ispravne pretpostavke o prirodi slika, a njihov kapacitet se može kontrolisati variranjem njihove dubine i širine. Tako, u poređenju sa standardnim neuronskim mrežama (*mrežama sa propagacijom unapred (eng.* feedforward neural network*)*) sa slojevima sličnih veličina, CNN imaju mnogo manje veza i parametara, tako da ih je lakše trenirati, dok je njihov teoretski najbolji učinak samo nešto lošiji.

3 Konvolutivne neuronske mreže (CNN)

Konvolutivne neuronske mreže predstavljaju potklasu neuronskih mreža koja ima najmanje jedan konvolutivni sloj (a može ih imati i više). Ova vrsta neuronskih mreža je inspirisana vizuelnim korteksom. Svaki put kada osoba nešto vidi, aktivira se niz slojeva neurona, i svaki sloj otkriva skup karakteristika kao što su linije, ivice itd. Viši nivoi slojeva otkrivaju složenije karakteristike kako bi prepoznali ono što je osoba videla. Konvolutivne mreže rade po istom principu i praktično su uvek *duboke neuronske mreže (eng.* deep neural network*)*[6], upravo zbog toga što je potrebno od sitnijih detalja, poput uspravnih, kosih i horizontalnih linija, koji obično bivaju detektovani u nižim slojevima mreže, konstruisati složenije oblike poput delova lica. Konvolutivne neuronske mreže se koriste u obradi signala (slike, zvuka), ali i teksta. U odnosu na ostale vrste neuronskih mreža, ističu se u prikupljanju lokalnih informacija (npr. o susednim pikselima na slici ili *,,okružujućim"(eng.* surrounding*)* rečima u tekstu) i smanjenju složenosti modela (brže treniranje, potrebno je manje uzoraka, manja šansa da dodje do *preprilagodjavanja (eng.* overfitting*)*). Konvulativne neuronske mreže se zasnivaju na sposobnosti mreža da iz sirovog signala konstruišu atribute. Nazivaju se konvolutivnim zato što uče **filtere** (pojam objašnjen u tabeli 1), čijom konvolutivnom primenom detektuju određena svojstva signala.

Postoji nekoliko različitih arhitektura u polju konvolutivnih neuronskih mreža. One se razlikuju po tipu, broju i rasporedu slojeva, i uglavnom predstavljaju bolju verziju neke od prethodnih mreža (o ovome više reči u poglavlju 3.3). Takođe, ono što treba istaći jeste podešavanje parametara mreže. Iako nije moguće izračunati u kom slučaju koji parametar treba imati koju vrednost, u narednom poglavlju (3.1) su opisani neki od bitnijih parametara koje svakako treba podesiti pri učenju mreže.

3.1 Parametri

U ovoj sekciji je dat tabelarni prikaz parametara koji su najznačajniji za implementaciju konvolutivne mreže. U tabeli 1 je dat kratak opis

[6]Duboke neuronske mreže su neuronske mreže koje se sastoje od više skrivenih slojeva između ulaznih i izlaznih slojeva (koji na izlazu predviđaju ciljnu promenljivu), koje mogu modelovati i kompleksne nelinearne veze.

svakog od njih radi boljeg razumevanja teksta koji sledi. Ipak, u nastavku je svaki detaljnije opisan.

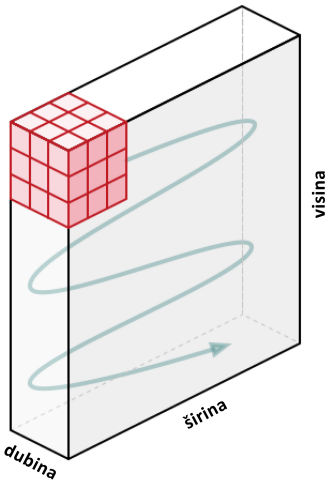
Tabela 1: Parametri konvolutivne neuronske mreže

Filter/jezgro (eng. filter/kernel*)*	- matrica sa težinama za konvoluciju ulaza (poglavlje 3.2) - daje meru koliko deo ulaza liči na karakteristiku - težine u matricama filtera su izvedene za vreme treniranja podataka
Proširenje (eng. Padding*)*	- koristi se za dodavanje kolona i redova nula (ili drugih vrednosti) da bi se održala konstantna veličina matrice (mape) nakon konvolucije (poglavlje 3.4.1) - ovaj parametar može da unapredi performanse tako što zadržava informacije u okvirima (odnosno proširenjima)
Korak (eng. Stride*)*	- broj piksela koji se preskače dok se ulaz prelazi vodoravno i uspravno tokom konvolucije, nakon množenja svakog elementa iz ulazne matrice težina s onima u filteru (poglavlje 3.4.1)
Broj kanala (eng. Number of Channels*)*	- u početku je jednak broju *kanala boja (eng.* color channels*)* ulaza, a u kasnijim stadijumima postaje jednak broju filtera koji su korišćeni za operaciju konvolucije - što je veći broj kanala, veći je i broj korišćenih filtera, a time je naučeno više karakteristika, pa postoji šansa da dođe do preprilagođavanja (i obrnuto)
Parametri sloja agregacije	- ima iste parametre kao i konvolutivni sloj - uglavnom se koristi *Max-Pooling* opcija (poglavlje 3.5) - smanjuje dimenzionalnost ulazne reprezentacije (slike, izlazne matrice skrivenog sloja, itd.) tako što čuva maksimalnu vrednost podregiona

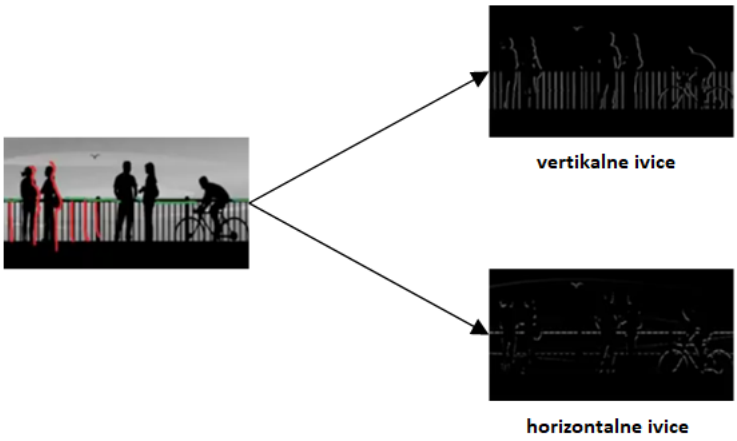
3.2 Konvolucija slika preko CNN

Da bi izvršile klasifikaciju slika, konvolutivne neuronske mreže (preciznije, konvolutivni sloj, detaljnije u poglavlju 3.4) obavljaju neku vrstu pretrage. Ovo se može zamisliti kao mali pokretni (ili klizni) prozor (prikazano na slici 2) koji klizi s leva na desno preko veće slike, i nastavlja s leve

strane kada dođe do kraja jednog prelaza (kao kod pisaće mašine). Taj pokretni prozor - koji ustvari predstavlja filter, može da prepozna samo jednu stvar, recimo kratku vertikalnu liniju (tri tamna piksela naslagana jedan na drugi). Slično, neki drugi filter može da služi za prepoznavanje horizontalnih linija, i on se takođe pomera preko piksela slike, tražeći podudaranja. Rezultat koji se postiže filterima koji prepoznaju vertikalne i horizontalne linijije prikazan je na slici 3.



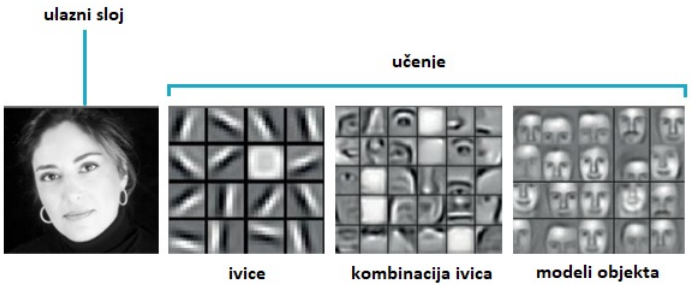
Slika 2: Kretanje filtera



Slika 3: Detektovanje vertikalnih i horizontalnih ivica

Svaki put kada dođe do poklapanja (filtera sa ulazom), ono se mapira u prostor sa karakteristikama - koji se zove ***mapa karakteristika*** *(eng.* feature maps*)*, koji je specifican za taj vizuelni element. U tom prostoru (tj. mapi) se čuva (odnosno beleži) lokacija svakog poklapanja sa vertikalnom linijom. Konvolutivna mreža pokreće mnogo pretraga nad jednom

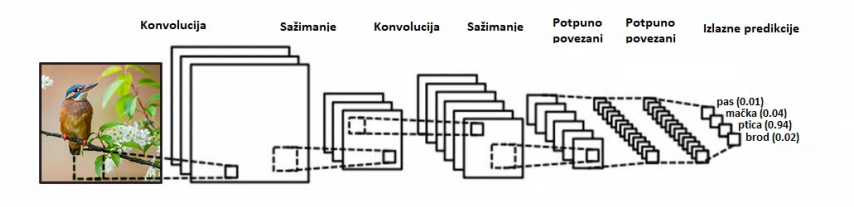
istom slikom. Na početnom sloju mreže koriste se filteri koji prepoznaju horizontalnu liniju, vertikalnu liniju i dijagonalnu liniju, da bi kreirali mapu ivica slike. U narednim koracima (odnosno slojevima) posmatraju se kombinacije ovih ivica i tako prepoznaju složeniji oblici. Ovo je demonstrirano na slici 4. U narednim poglavljima je objašnjeno iz čega se konvolutivne neuronske mreže sastoje i kako ti gradivni elementi zapravo funkcionišu.



Slika 4: Učenje mreže

3.3 Unutrašnja struktura CNN

Unutrašnja struktura konvolutivne mreže se sastoji od nekoliko naizmeničnih ***konvolutivnih slojeva*** *(eng.* convolution layer*)* i ***slojeva agregacije*** *(eng.* pooling layer, dense layer*)*, pri čemu je dozvoljeno pojavljivanje iste vrste sloja više puta (prikazano na slici 5). Nakon uzastopnog smenjivanja ova dva sloja sledi ***potpuno povezani sloj*** *(eng.* fully connected layer*)*, koji se takođe može ponavljati više puta. Izlazi konvolutivnog sloja su dvodimenzionalni i nazivaju se mapama karakteristika. Oni se transformisu nelinearnom **aktivacionom funkcijom**. U konvolutivnim mrežama kao aktivacione funkcije najčešće se koriste ***prečišćena linearna jedinica*** *(eng.* Rectified Linear Unit, ReLu*)*[7] (na izlazu iz konvolutivnog sloja) i ***softmax klasifikator*** *(eng.* softmax classifier*)* (na izlazu iz poslednjeg, potpuno povezanog, sloja mreže). U nastavku rada je svaki od tri sloja obrađen pojedinačno u poglavljima 3.4, 3.5, 3.6, dok se o aktivacionim funkcijama detaljnije govori u poglavlju 3.4.2.



Slika 5: Arhitektura konvolutivne neuronske mreže

[7]ReLu funkcija se definiše kao: f(x) = max(0,x). Prednost ReLu funkcije nad sigmoidnom funkcijom (koja se takođe može koristiti kao aktivaciona funkcija) je što treniranje obavlja mnogo brže.

Kako su moguće različite kombinacije različitih slojeva, prirodno je da postoji više arhitektura konvolutivnih neuronskih mreža. Svaka od tih arhitektura ima svoje prednosti i mane, i u zavisnosti od problema drugačija arhitektura daje bolje (odnosno gore) rezultate. U sledećoj listi su nabrojane i ukratko opisane samo neke od najpoznatijih arhitektura konvolutivnih neuronskih mreža:

- **LeNet-5** - Ovu arhitekturu je napravio Yann LeCunn krajem 20. veka, a koristila se za čitanje zip kodova, cifara itd. LeNet-5 arhitektura se sastoji iz dva skupa konvolutivnih slojeva i slojeva agregacije, koji su praćeni *poravnavajućim konvolutivnim slojem (eng.* flattening convolutional layer*)*, za kojim slede dva potpuno-povezana sloja, a zatim softmax klasifikator.
- **AlexNet** - Ovo je jedna od prvih dubokih neuronskih mreža. Sastoji se iz pet konvolutivnih slojeva praćena sa tri potpuno povezana sloja. Ovu mrežu je napravio Alex Krizhevsky, koji je koristio prečišćenu linearnu jedinicu - ReLu za ne-linearni deo umesto tanh ili sigmoid funkcije, koje su bile standardne za tradicionalne neuronske mreže.
- **VGG16** - Ova arhitektura predstavlja unapređenje arhitekture AlexNet. U njoj su veliki filteri zamenjeni sa više malih 3x3 filtera, jedan za drugim[8].
- **GoogLeNet** - Ova arhitektura ima čak 22 sloja. Brža je i manja od Alexnet, ali i mnogo preciznija. Ideja ove mreže jeste da se napravi model koji će moći da se koristi i na *pametnom telefonu (eng.* smartphone*)*.

3.4 Konvolutivni sloj

Konvolutivni sloj je glavni deo konvolutivne neuronske mreže koji radi najviše izračunavanja u mreži. Njegova uloga je konstrukcija novih atributa. To je prvi sloj koji ekstraktuje karakteristike iz ulazne slike. Konvolucija je matematička operacija koja uzima dva ulaza - dve matrice f i g, dimenzija m×n i p×q, a definisana je na sledeći način:

$$(f * g)_{ij} = \sum_{k=0}^{p-1} \sum_{l=0}^{q-1} f_{i-k,i-l} * g_{k,l}$$

Matrica f predstavlja ulaz, poput slike, a matrica g filter [4]. Određena ulazna reprezentacija podatka (originalna slika) konvolvira se sa filterom sa određenim parametrima (ti parametri predstavljaju i parametre konvolutivne neuronske mreže). Procesom učenja ovi parametri (težine koje je potrebno naučiti kako bi mreža davala dobre rezultate) se podešavaju i bivaju naučeni. Filteri su najcešce manjih prostornih dimenzija od ulaza, ali uvek su jednake dubine kao i ulaz. Kao što je vec rečeno, konvolucijska jezgra (odnosno filteri) filtriraju sliku, tj. mapu karakteristika kako bi dobili neku korisnu informaciju kao što je recimo određeni oblik, boja ili ivica.

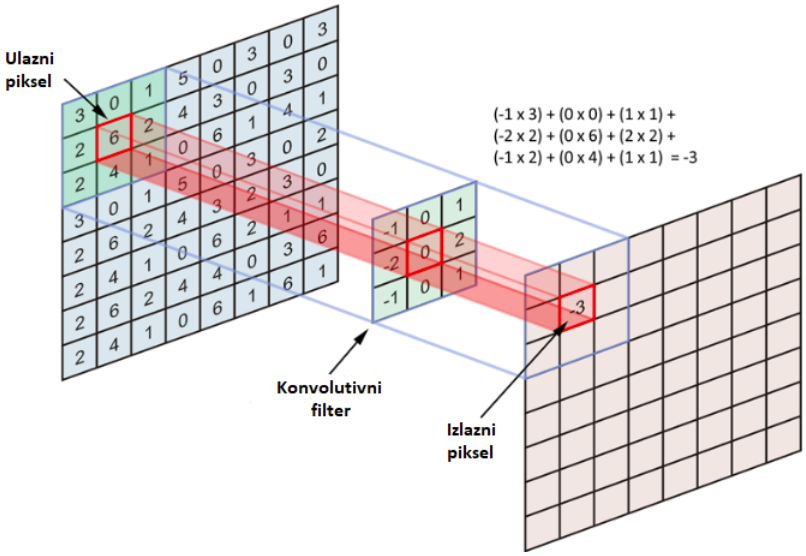
Tokom prvog prolaza svaki filter se pomera po širini i visini ulaza i računa se skalarni proizvod ulaza i vrednosti filtera (prikazano u formuli iznad). Ako obe matrice imaju visoke vrednosti na istim pozicijama, onda će i vrednost skalarnog proizvoda biti velika. A, ako nemaju, onda će i

[8]Više manjih filtera se pokazuje boljim nego jedan veliki filter, zato što više ne-linearnih slojeva povećavaju dubinu mreže što im omogućava da uče kompleksnije karakteristike.

vrednost skalarnog proizvoda biti mala. Na taj način, samo na osnovu te vrednosti, može se zaključiti da li sadržaj slike odgovara sadržaju koji filter traži.

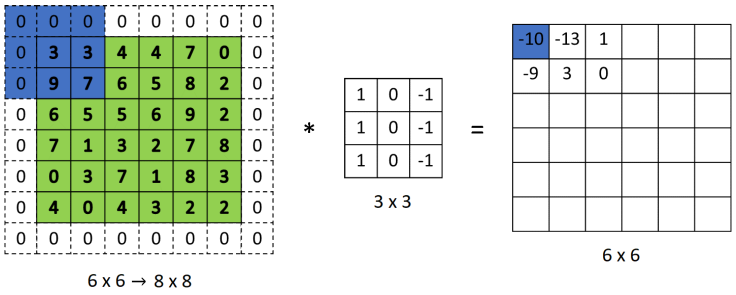
Izlaz jednog filtera je dvodimenzioni niz. Ako postoji više filtera, izlaz iz konvolutivnog sloja su rezultati svih filtera (mape karakteristika) poredjani po dubini. Broj mapa karakteristika na izlazu, odnosno broj primenjenih filtera, predstavlja **dubinu mreže**. Operacijom konvolucije dobija se transformirana slika dimenzije I*K, gde je I dimenzija ulazne matrice slike, a K dimenzija filtera koji je primenjen nad tom slikom. Takođe, dobijena transformacija je lokalna tj. pikseli izlaza zavise od lokalnih, susednih piksela ulaza. Ceo proces konvolucije, za koji je upotrebljen filter dimenzije 3x3 i ulazna matrica dimenzije 8x8, prikazan je na slici 6. U podpoglavljima 3.4.1 i 3.4.2 su opisani parametri koji bitno utiču na ishod rada mreže.

Slika 6: Konvolucija matrice dimenzija 3×3 filterom dimenzija 8×8



3.4.1 Proširivanje

Formula konvolucije koja je data u poglavlju 3.4 nije definisana za sve indekse i = $\overline{0, m-1}$ i j = $\overline{0, n-1}$. Na primer, ako je i, j = 0 i k, l > 0, vrednost $f_{i-k,i-l}$ nije definisana. Ukoliko bi se u obzir uzele samo definisane vrednosti, dimenzija konvolucije bi bila manja od dimenzije matrice f. Međutim, to nije uvek poželjno, i može se izbeći tako što se vrši **proširivanje** matrice f, na primer nulama ili vrednostima koje su već na obodu, tako da veličina rezultujuće matrice bude jednaka veličini matrice f pre proširivanja. Ovo je prikazano na slici 7. Takođe, prilikom računanja konvolucije, filter se duž slike ne mora pomerati za jedan piksel, već za neki veći **korak** (pojam opisan u tabeli 1) [4].



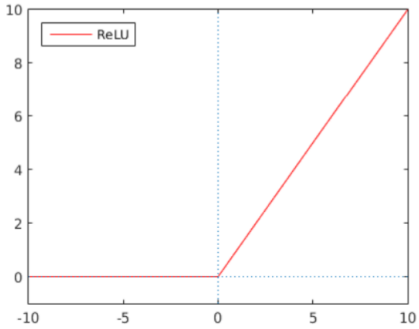
Slika 7: Primer proširivanja ulazne matrice nulama

3.4.2 Aktivaciona funkcija

U konvolutivnim slojevima kao aktivaciona funkcija se najčešće koristi ReLu (Rectified Linear Unit) funkcija. Definisana je kao

$$f(z) = max(0, z).$$

To znači da će negativne piksele mape karakteristika nastale konvolucijom da postavi na nulu i neće aktivirati odgovarajuće neurone. A pozitivne piksele neće menjati. Na taj način, ReLu ne aktivira sve neurone u istom trenutku, čime ubrzava rad mreže i čini je efikasnijom. Na slici 8 se može videti kako ReLU funkcija izgleda, a na slici 9 kako se pomoću nje transformiše mapa karakteristika.

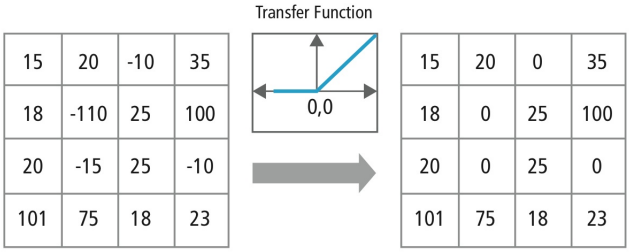


Slika 8: Grafik ReLU funkcije

Osim ReLu funkcijue, koriste se (u manjoj meri) i sigmoidna funkcija i tangens hiperbolički. Grafici ovih funkcija su prikazani na slici 10. Sigmoidna funkcija je praktično konstantna osim u okolini nule, te dovodi do anuliranja gradijenta, što otežava učenje, pa se u praksi baš i ne koristi. Sigmoidna funkcija se definiše ovako:

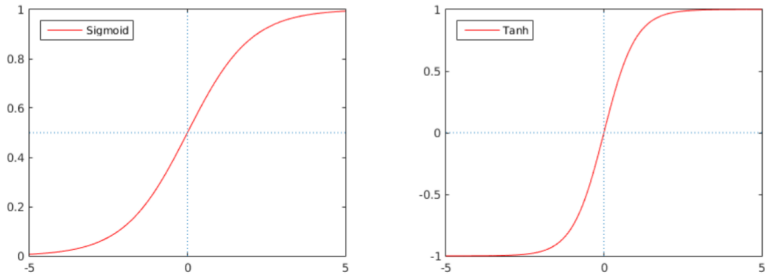
$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

Tangens hiperbolički je funkcija vrlo srodna sigmoidnoj (a nešto je duže od nje u upotrebi). Tangens hiperbolički je u okolini nule bliska identitetu, što čini model sličnijim linearnom i u nekoj meri olakšava optimizaciju. Tangens hiperbolički definisan je ovako:



Slika 9: Primer rada ReLU funkcije nad mapom karakteristika

$$\tanh(x) = \frac{e^{2x} - 1}{e^{2x} + 1}$$



Slika 10: Grafici funkcija sigmoid i tanh

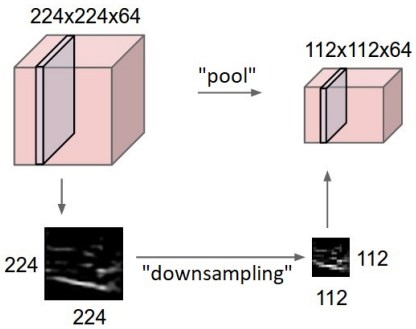
3.5 Sloj agregacije

Uloga sloja agregacije je smanjenje broja parametara kada su slike prevelike, kao i smanjenje broja operacija u višim slojevima mreže. Agregacija smanjuje dimenzionalnost svake mape, ali zadržava važne informacije. Sve to rezultuje smanjenjem računske zahtevnosti pri optimizaciji i pomaže u kontroli *overfitinga*. Zato je poželjno da slojevi za agregaciju prate konvolutivne slojeve kako bi postepeno smanjili prostornu veličinu (širinu i visinu) prikaza podataka.

Često konačni zadatak postavlja neko globalno pitanje o slici, na primer, da li sadrži mačku? Stoga, čvorovi zadnjeg sloja moraju biti osetljivi na celi ulaz. Postepenom agregacijom informacija, stvarajući sve grublje i grublje mape karakteristika, postiže se taj cilj da se na kraju nauči globalna reprezentacija, zadržavajući sve prednosti konvolutivnih slojeva na srednjim slojevima obrade.

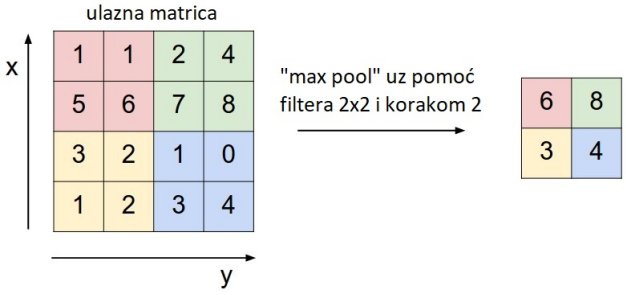
Sloj agregacije ukrupnjuje informacije, tako sto računa neku jednostavnu funkciju agregacije susednih jedinica prethodnog sloja, poput *maksimuma (eng.* Max pooling*)*, koji vraća maksimalnu vrednost dela slike pokrivene filterom, ili *proseka (eng.* Average pooling*)*, koji vraća prosečnu vrednost dela slike pokrivene filterom. Funkcija koja vraća prosečnu vrednost se u praksi toliko ne koristi, već se javlja iz istorijskih razloga, dok se funkcija koja računa maksimalnu vrednost mnogo bolje pokazala u praksi, te se ona skoro uvek koristi. Ukoliko agregira, na primer, 3×3 piskela, onda je broj izlaza ovog sloja 9 puta manji od broja izlaza prethodnog. Kada se računa maksimum, dolazi do zanemarivanja informacija o tome gde je precizno neko svojstvo (poput uspravne linije) pronađeno, ali se ne gubi informacija da je ono pronađeno. Ovakva vrsta zanemarivanja informacije često ne šteti cilju koji treba postići. Na primer, ako su na slici pronađeni kljun i krila, informacija o tačnoj poziciji najverovatnije nije bitna za odlučivanje da li se na slici nalazi ptica. Ipak, ukoliko je potrebno napraviti mrežu koja igra igru u kojoj su pozicije objekata na ekranu bitne, nije poželjno koristiti agregaciju. Slede dva ilustrovana primera, a onda poglavlje o sloju koji se nadovezuje na sloj agregacije (poglavlje 3.6).

Na slici 11 je prikazana operacija agregacije primenjena na ulaz dimenzija 224×224×64, uz pomoć filtera veličine 2 i koraka veličine 2. Izlaz koji se dobija je dimenzija 112×112×64. Može se primetiti da je dubina matrice ostala očuvana. Naime, sloj agregacije redukuje veličinu posebno (tj. nezavisno) za svaki *sloj dubine (eng.* depth slice*)* ulazne veličine.



Slika 11: Proces *agregacije (eng.* pooling*)*

Na slici 12 je prikazan izlaz koji se dobija kada se na ulaznu matricu dimenzije 4×4 primeni *Maxpool* metod uz pomoć filtera dimenzije 2×2 i koraka 2. Naime, u ovom primeru se računa maksimalna vrednost nad svakim blokom od četiri broja.



Slika 12: Agregacija metodom *Maxpool*

3.6 Potpuno povezani sloj

Nakon što su u konvolutivnom sloju i sloju agregacije sakupljene informacije o slici, sledeće što je potrebno uraditi jeste zaključiti kojoj klasi pripada ta slika. Za to se koristi jedan ili više potpuno povezanih slojeva.

U potpuno povezanom sloju, svaki neuron je povezan sa svakim neuronom iz prethodnog sloja, kao što je to slučaj kod običnih neuronskih mreža. Taj sloj kao ulaz očekuje vektor, što znači da je prvo potrebno *ispraviti (eng.* flatten*)* mapu karakteristika, dobijenu nakon konvolucije i agregacije, u vektor. U svim slojevima, osim u poslednjem, može da se koristi ReLU aktivaciona funkcija.

Poslednji potpuno povezani sloj treba da ima onoliko neurona koliko ima i klasa, kako bi mogao da odredi koju klasu da pridruži slici. U ovom sloju je najbolje koristiti softmax aktivacionu funkciju. Softmax funkcija određuje verovatnoću pripadnosti slike svakoj od klasa. Nakon njene primene, dobija se vektor vrednosti koje odgovaraju verovatnoćama i koje u zbiru daju vrednost 1. Zahvaljujući njoj, slika se klasifikuje u onu klasu kojoj odgovara najveća verovatnoća pripadnosti. Softmax funkcija je definisana kao:

$$S(z_i) = \frac{e_i^z}{\sum_{j=1}^{K} e_j^z}, i = 1, ..., K, z = (z_1, ..., z_K)$$

gde je z ulazni vektor od K realnih brojeva.

U sloju koji koristi softmax aktivacionu funkciju, obično se za funkciju određivanja greške klasifikacije (tj. funkciju gubitka, eng. loss function) uzima *categorical crossentropy*. Ona se koristi ukoliko su podaci takvi da jedna instanca tih podataka može da pripada samo jednoj klasi. *Categorical crossentropy* upoređuje raspodelu verovatnoća predviđenih klasa sa raspodelom koja odgovara pravim klasama. Ova funkcija je definisana kao:

$$H(y, \hat{y}) = -\sum_i y_i \cdot log(\hat{y}))$$

gde y predstavlja vektor pravih raspodela, a $\hat{y}$ vektor predviđenih raspodela. Dakle, funkcija gubitka služi da kvantifikuje kvalitet klasifikacije, tj. da oceni koliko je dobar skup težina koji služi za učenje mreže.

3.7 Optimizacija mreže

Cilj optimizacije je da pronađe skup težina tako da se funkcija gubitka minimizuje. Konvolutivne neuronske mreže se često optimizuju pomoću stohastičkog gradijentnog spusta ili Adam optimizatora.

Kod gradijentnog spusta, težine se traže u pravcu koji je suprotan pravcu gradijenta funkcije gubitka, jer se na taj način može stići do lokalnog minimuma. A to nam odgovara jer nam je cilj da minimizujemo tu funkciju gubitka. Međutim, samo kretanje u tom pravcu ne garantuje da će taj minimum biti pronađen, jer je u zavisnosti od dužine koraka moguće preći preko minimuma. Zato je potrebno odrediti pogodnu dužinu koraka, tj. pronaći najbolji parametar učenja (eng. learning rate).

Kod optimizacije gradijentnim spustom, za ažuriranje makar samo jedne težine, potrebno je proći kroz sve podatke iz trening skupa. Zbog toga se češće koristi optimizacija stohastičkim gradijentnim spustom (SGD, eng. Stochastic Gradient Descent), kod koje je moguće koristiti samo podskup trening podataka. Zove se stohastički jer nasumično bira trening uzorke na osnovu kojih će ažurirati parametre. Tim nasumičnim izborom je omogućeno pronalaženje drugog, potencijalno boljeg, lokalnog minimuma.

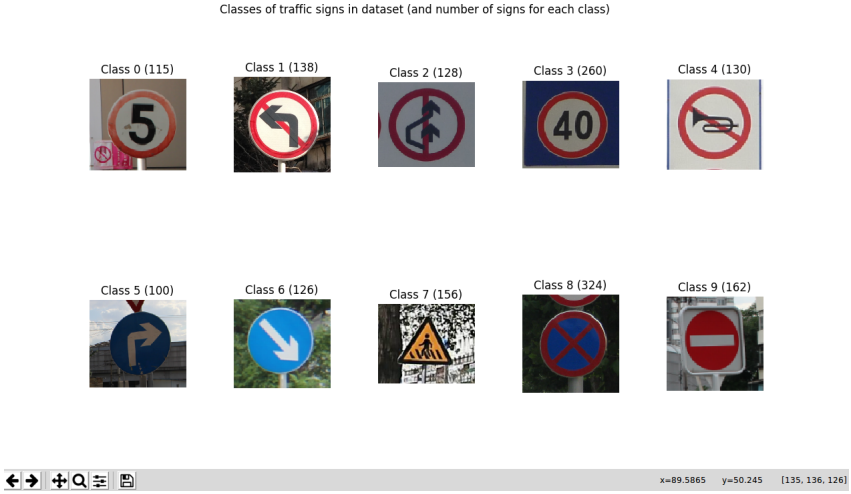
Optimizator Adam predstavlja unapređenu verziju SGD-a. SGD održava jednu vrednost parametra učenja za sva ažuriranja težina i ta vrednost se ne menja tokom treninga. Za razliku od njega, Adam može da održava različite parametre učenja za različite težine i da ih menja tokom procesa učenja [1].

4 Implementacija i eksperimentalni rezultati

Neuronsku mrežu smo implementirale u programskom jeziku Python korišćenjem Keras biblioteke. Kao podatke za trening i testiranje, koristile smo slike iz baze podataka kineskih saobraćajnih znakova [2]. Baza sadrži 6164 slika saobraćajnih znakova podeljenih u 58 klasa, pri čemu trening skup sadrži 4170, a test skup 1994 slike. Međutim, nisu sve klase imale podjednak broj slika. Za neke je postojalo 5 slika na kojima bi mreža mogla da trenira, a za neke oko 400. Takođe, zbog prevelike količine podataka, izvršavanje programa je bilo mnogo sporo. Zbog svega toga, odlučile smo da koristimo samo jedan deo te baze i izdvojile 10 klasa koje su imale približno jednak broj slika. Nakon toga, dobile smo trening skup od 1693 slika i test skup od 764 slika. Na slici 13 je prikazan po jedan znak iz svake klase trening skupa, zajedno sa brojem elemenata te klase. Slično, isti podaci o test skupu, mogu se videti na slici 14. U narednim poglavljima će biti predstavljeni neki od modela konvolutivnih neuronskih mreža koje smo pravile, a koje su davale najbolje rezultate.

4.1 Model 1

Jedan od prvih modela koji je imao veći uspeh na test podacima prikazan je u kodu 1. Sastoji se iz četiri konvolutivna sloja, dva sloja agregacije i dva potpuno povezana sloja. Detaljna arhitektura mreže se može videti na slici 15. U svim konvolutivnim slojevima veličina jezgra je 3×3, a broj filtera na izlazu iz konvolutivnog sloja je 32. Takođe, u svakom se koristi ReLU aktivaciona funkcija. U sloju agregacije, sažimanje se vrši biranjem



Slika 13: Trening skup

Slika 14: Test skup

maksimalne vrednosti dela mape karakteristika koji je prekriven filterom. Kako ne bi došlo do preprilagodjavanja modela trening podacima, povremeno je, pomoću funkcije *Dropout()*, isključivan određen broj nasumično odabranih neurona. Nakon agregacija je isključeno 20% neurona, a pre poslednjeg, potpuno povezanog, sloja 50%. Poslednji potpuno povezani sloj ima onoliko neurona koliko ima klasa i koristi softmax aktivacionu funkciju.

Učenje modela je sprovedeno u 30 epoha. *Batch size* je postavljen na 32, što znači da će u svakoj iteraciji biti uzeta 32 primerka iz trening skupa koja će biti propagirana kroz mrežu. Optimizacija modela je izvršena pomoću gradijentnog spusta. Takođe, pre početka treninga, sve slike su skalirane na istu veličinu, 64×64 piksela.

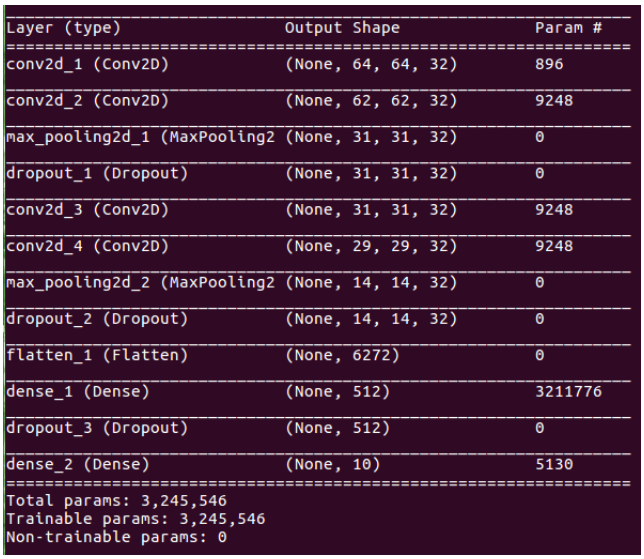
```
def cnn_model():

    model = Sequential()

    model.add(Conv2D(filters = 32, kernel_size = (3, 3),
        padding='same', input_shape=(IMG_SIZE, IMG_SIZE, 3),
        data_format="channels_last", activation='relu'))
    model.add(Conv2D(filters = 32, kernel_size = (3, 3),
        activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(0.2))

    model.add(Conv2D(filters = 32, kernel_size = (3, 3),
        padding='same', activation='relu'))
    model.add(Conv2D(filters = 32, kernel_size = (3, 3),
        activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(0.2))

    model.add(Flatten())
    model.add(Dense(512, activation='relu'))
    model.add(Dropout(0.5))
    model.add(Dense(NUM_OF_CLASSES, activation='softmax'))

    model.summary()
    return model


batch_size = 32
epochs = 30
lr = 0.01   #learning rate

model = cnn_model()

# optimizacija pomocu gradijentnog spusta
sgd = SGD(lr=lr, decay=1e-6, momentum=0.9, nesterov=True)

model.compile(loss='categorical_crossentropy', optimizer=sgd,
        metrics=['accuracy'])

def lr_schedule(epoch):
    return lr * (0.1 ** int(epoch / 10))

model.fit(images, classes,
          batch_size=batch_size,
          epochs=epochs,
          validation_split=0.2,
          callbacks=[LearningRateScheduler(lr_schedule),
                 ModelCheckpoint('model.h5', save_best_only=True)])
```

Kod 1: Model 1

Na slici 16 se može videti da je preciznost modela na trening skupu tokom poslednje tri epohe varirala oko 0.99 i 1.00. Na slici 17 se može videti kako se model pokazao na test skupu. *Preciznost (eng.* accuracy*)*, tj. broj tačno klasifikovanih instanci / ukupan broj instanci) je jednaka 0.882. Može se videti da je i *preciznost (eng.* precision*)* koja se odnosi na svaku od klasa dosta visoka (osim za klase 0 i 5).

4.2 Model 2: LeNet-5

Još bolje rezultate smo dobile pomoću LeNet-5 mreže (opisana u poglavlju 3.3) [5]. Mreža se sastoji iz 7 slojeva, pri čemu se ne računa ulazni sloj. Sve slike smo skalirale na večičinu 32×32 piksela, jer ova mreža kao ulaz očekuje slike tih dimenzija. Prvi konvolutivni sloj na izlazu daje 6 mapa karakteristika, koristi jezgro veličine 3×3 i kao aktivacionu funkciju koristi ReLU. Nakon njega sledi sloj agregacije u kom se sažimanje vrši



Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 64, 64, 32)	896
conv2d_2 (Conv2D)	(None, 62, 62, 32)	9248
max_pooling2d_1 (MaxPooling2	(None, 31, 31, 32)	0
dropout_1 (Dropout)	(None, 31, 31, 32)	0
conv2d_3 (Conv2D)	(None, 31, 31, 32)	9248
conv2d_4 (Conv2D)	(None, 29, 29, 32)	9248
max_pooling2d_2 (MaxPooling2	(None, 14, 14, 32)	0
dropout_2 (Dropout)	(None, 14, 14, 32)	0
flatten_1 (Flatten)	(None, 6272)	0
dense_1 (Dense)	(None, 512)	3211776
dropout_3 (Dropout)	(None, 512)	0
dense_2 (Dense)	(None, 10)	5130

Total params: 3,245,546
Trainable params: 3,245,546
Non-trainable params: 0

Slika 15: Slojevi modela 1 neuronske mreže

```
Epoch 28/30
1311/1311 [==============================] - 110s 84ms/step - loss: 0.0046 - acc: 0.9985 - val_loss: 4.6311e-04 - val_acc: 1.0000
Epoch 29/30
1311/1311 [==============================] - 103s 78ms/step - loss: 0.0014 - acc: 1.0000 - val_loss: 4.4722e-04 - val_acc: 1.0000
Epoch 30/30
1311/1311 [==============================] - 110s 84ms/step - loss: 0.0035 - acc: 0.9992 - val_loss: 4.3213e-04 - val_acc: 1.0000
```

Slika 16: Preciznost modela 1 na trening podacima

```
Test accuracy = 0.8821989528795812

Classification report:
              precision    recall  f1-score   support

           0       0.50      0.86      0.63        14
           1       0.96      0.78      0.86       130
           2       1.00      0.83      0.91        12
           3       0.91      0.93      0.92        84
           4       0.91      0.95      0.93        84
           5       0.67      0.46      0.55        26
           6       1.00      0.81      0.89       134
           7       0.76      0.96      0.85        46
           8       0.84      0.99      0.91       176
           9       0.93      0.93      0.93        58

   micro avg       0.88      0.88      0.88       764
   macro avg       0.85      0.85      0.84       764
weighted avg       0.90      0.88      0.88       764

Confusion matrix:
[[ 12   0   0   0   0   0   2   0   0]
 [  6 102   0   0   8   4   0   6   0   4]
 [  0   0  10   2   0   0   0   0   0   0]
 [  2   2   0  78   0   0   0   2   0   0]
 [  2   0   0   0  80   0   0   2   0   0]
 [  0   0   0   0   0  12   0   0  14   0]
 [  2   0   0   4   0   0 108   0  20   0]
 [  0   0   0   2   0   0   0  44   0   0]
 [  0   0   0   0   0   2   0   0 174   0]
 [  0   2   0   0   0   0   0   2   0  54]]
```

Slika 17: Preciznost modela 1 na test podacima

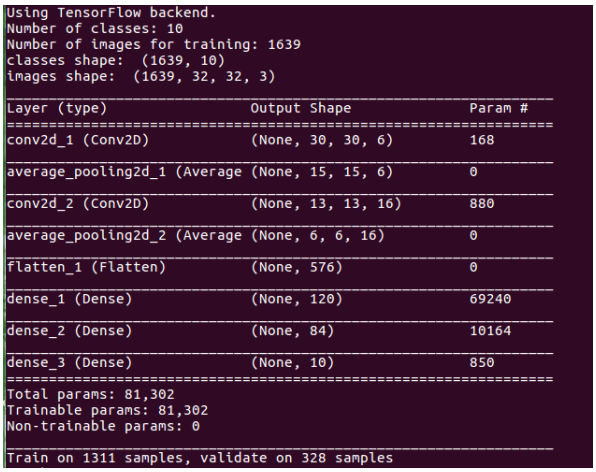
biranjem prosečne vrednosti dela mape karakteristika koji je prekriven filterom. Još jednom se ponavljaju ova dva sloja, s tim što sada konvolutivni sloj vraća 16 mapa karakteristika. Nakon ispravljanja mape karakteristika u vektor, taj vektor se prosleđuje potpuno povezanom sloju koji se sastoji iz 120 neurona i koristi ReLU aktivacionu funkciju. Nakon toga sledi potpuno povezani sloj od 84 neurona koji, takođe, koristi ReLU aktivacionu funkciju. I, na kraju, ostaje još jedan potpuno povezani sloj koji ima onoliko neurona koliko ima klasa i koji koristi softmax aktivacionu funkciju. U kodu 2 je data implementacija LeNet-5 mreže, a na slici 18 je prikazana njena detaljna arhitektura.

```
def cnn_model():

    model = Sequential()

    model.add(Conv2D(filters=6, kernel_size=(3, 3),
        activation='relu', input_shape=(IMG_SIZE, IMG_SIZE, 3),
        data_format="channels_last"))
    model.add(AveragePooling2D())

    model.add(Conv2D(filters=16, kernel_size=(3, 3),
        activation='relu'))
    model.add(AveragePooling2D())

    model.add(Flatten())
    model.add(Dense(units=120, activation='relu'))
    model.add(Dense(units=84, activation='relu'))
    model.add(Dense(units=NUM_OF_CLASSES, activation = 'softmax'))

    model.summary()
    return model


batch_size = 32
epochs = 20
lr = 0.01

model = cnn_model()

model.compile(loss=keras.losses.categorical_crossentropy,
        optimizer=keras.optimizers.Adam(),
        metrics=['accuracy'])

def lr_schedule(epoch):
    return lr * (0.1 ** int(epoch / 10))

model.fit(images, classes,
          batch_size=batch_size,
          epochs=epochs,
          validation_split=0.2,
          callbacks=[LearningRateScheduler(lr_schedule),
                 ModelCheckpoint('model.h5', save_best_only=True)])
```

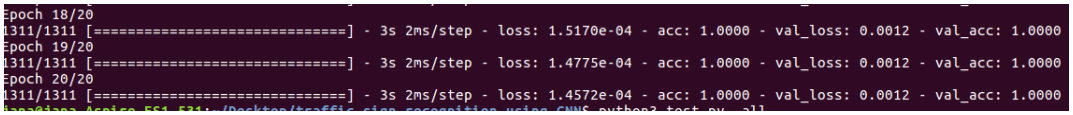
Kod 2: LeNet-5

Na slici 19 možemo videti da je preciznost na trening skupu u poslednje tri iteracije bila najveća moguća, tj 1.00. Na slici 20 je prikazan rezultat rada modela na test podacima. Preciznost (*accuracy*) iznosi 0.91. Takođe vidimo da je preciznost (*precision*) za pojedinačne klase visoka. Na osnovu matrice konfuzije može se videti da veoma malo podataka pogrešno klasifikuje.

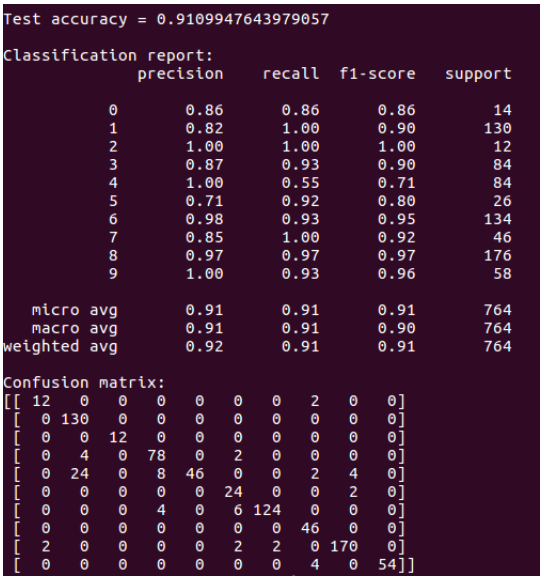
Na slici 28 prikazan je rast preciznosti na trening i test podacima kroz epohe. Na slici 22 se može videti kako je funkcija gubitka opadala sa povećanjem broja epoha treniranja. Na slici 23 je detaljnije prikazana matrica konfuzije.



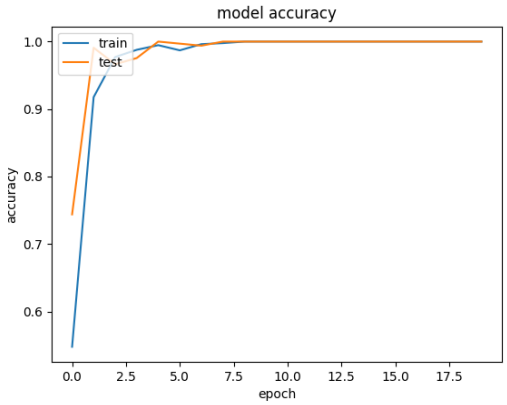
Slika 18: Slojevi modela leNet-5 neuronske mreže



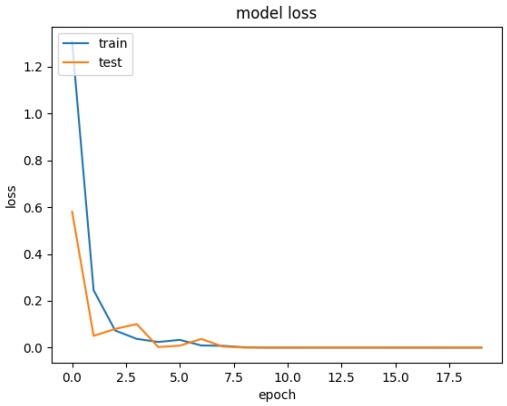
Slika 19: Preciznost leNet-5 modela na trening podacima



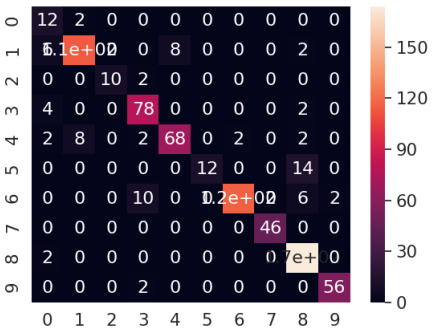
Slika 20: Preciznost leNet-5 modela na test podacima



Slika 21: Preciznost leNet-5 modela



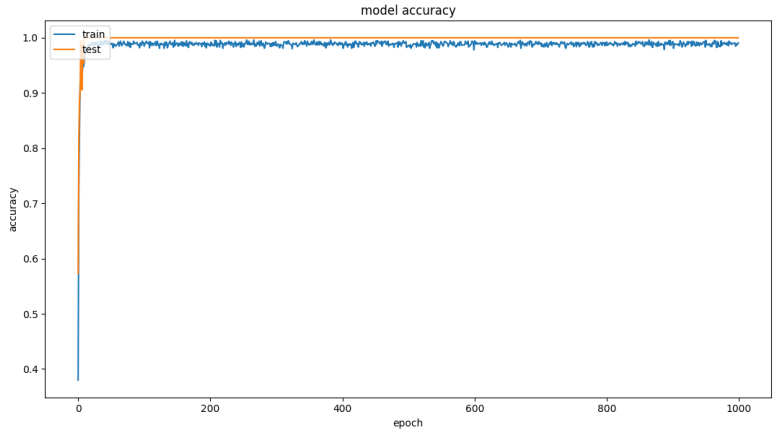
Slika 22: Gubitak leNet-5 modela



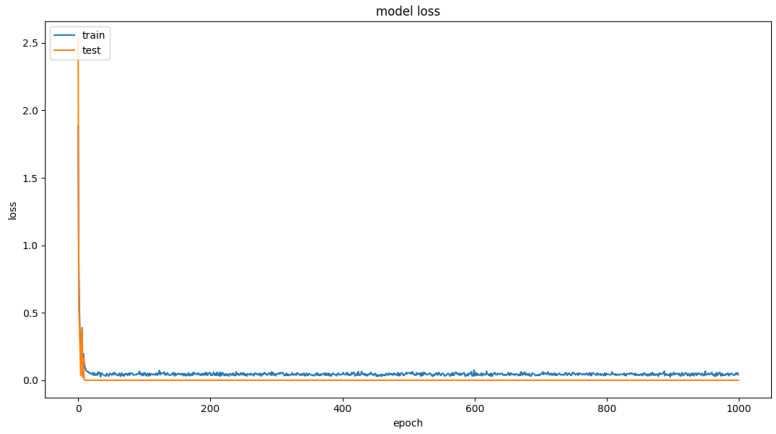
Slika 23: Matrica konfuzije leNet-5 modela

4.3 Model 3: AlexNet

Model AlexNet, čija je implementacija prikazana u ovoj sekciji, daje najbolje rezultate (bolje od ostala dva modela). Prilikom testiranja dostiže preciznost čak 0.95. Grafici postignute preciznosti i gubitka tokom treniranja podataka prikazani su redom na slikama 24 i 25. Statistike (između ostalih i matrica konfuzije) testiranja mogu se videti na slici 26. AlexNet mreža se sastoji iz 5 konvolutivna sloja (sa jezgrom veličine 3×3 i aktivacionom funkcijom ReLu), gde nakon svakog izlaz normalizuje pre nego što se prosledi narednom sloju. Takođe, pre nego što usledi normalizacija nakon prvog, drugog i petog konvolutivnog sloja, odvija se sažimanje (odnosno agregacija) sa parametrom *padding = 'valid'*, što znači da nema proširenja. Nakon ponavaljanja ova 2 (odnosno 3) sloja, sledi poravnavajući sloj (koji „ispravlja"mapu karakteristika u vektor), a zatim 3 *Dense* sloja. Nakon svakog *Dense* sloja je izvršena funkcija *Dropout()* koja sprečava preprilagođavanje modela, a onda su podaci normalizovani. Na samom kraju mreže nalazi se izlazni sloj koji preslikava ulaz u dati broj klasa, a kao funkciju aktivacije koristi *Softmax*. U kodu 3 je data implementacija AlexNet mreže, a na slici 27 je prikazana njena detaljna arhitektura.



Slika 24: Preciznost AlexNet modela pri treniranju za 1000 epoha



Slika 25: Gubitak AlexNet modela pri treniranju za 1000 epoha

```
Using TensorFlow backend.
2019-05-24 00:54:01.872038: I tensorflow/core/platform/cpu_feature_guard.cc:137]
Your CPU supports instructions that this TensorFlow binary was not compiled to use: SSE4.1 SSE4.2
Test accuracy = 0.9502617801047121

Classification report:
              precision    recall  f1-score   support

           0       0.75      0.86      0.80        14
           1       0.98      0.97      0.98       130
           2       1.00      0.83      0.91        12
           3       0.82      1.00      0.90        84
           4       0.94      0.81      0.87        84
           5       0.76      1.00      0.87        26
           6       1.00      0.94      0.97       134
           7       1.00      1.00      1.00        46
           8       0.99      0.97      0.98       176
           9       1.00      1.00      1.00        58

    accuracy                           0.95       764
   macro avg       0.93      0.94      0.93       764
weighted avg       0.96      0.95      0.95       764

Confusion matrix:
[[ 12   2   0   0   0   0   0   0   0   0]
 [  0 126   0   2   0   0   0   0   2   0]
 [  2   0  10   0   0   0   0   0   0   0]
 [  0   0   0  84   0   0   0   0   0   0]
 [  0   0   0  10  68   6   0   0   0   0]
 [  0   0   0   0   0  26   0   0   0   0]
 [  0   0   0   4   4   0 126   0   0   0]
 [  0   0   0   0   0   0   0  46   0   0]
 [  2   0   0   2   0   2   0   0 170   0]
 [  0   0   0   0   0   0   0   0   0  58]]
```

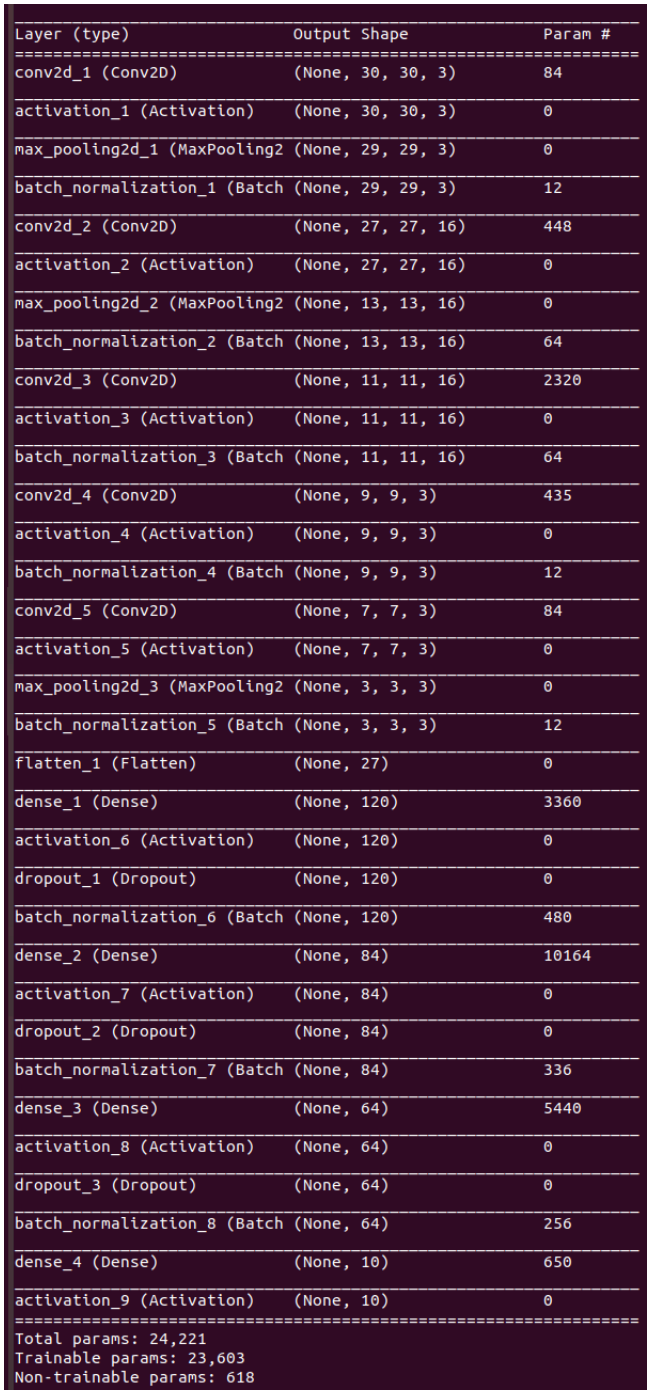
Slika 26: Matrica konfuzije i ostale statistike AlexNet modela za 1000 epoha

```
def alexNet():

    model = Sequential()

    # 1. kovolucioni sloj
    model.add(Conv2D(filters=3,
                     input_shape=(IMG_SIZE, IMG_SIZE, 3),
                     kernel_size=(3,3),
                     padding='valid',
                     data_format="channels_last"))
    model.add(Activation('relu'))
    # Pooling - agregacija
    model.add(MaxPooling2D(pool_size=(2,2),
```

```
            strides=(1,1),
            padding='valid'))
    # Batch Normalisation - normalizacija
    # (pre nego sto se prosledi narednom sloju)
    model.add(BatchNormalization())

    # 2. kovolucioni sloj
    model.add(Conv2D(filters=16,
                     kernel_size=(3,3),
                     strides=(1,1),
                     padding='valid'))
    model.add(Activation('relu'))
    # Pooling - agregacija
    model.add(MaxPooling2D(pool_size=(2,2),
                           strides=(2,2),
                           padding='valid'))
    # Batch Normalisation - normalizacija
    model.add(BatchNormalization())

    # 3. kovolucioni sloj
    model.add(Conv2D(filters=16,
                     kernel_size=(3,3),
                     strides=(1,1),
                     padding='valid'))
    model.add(Activation('relu'))
    # Batch Normalisation - normalizacija
    model.add(BatchNormalization())

    # 4. kovolucioni sloj
    model.add(Conv2D(filters=3,
                     kernel_size=(3,3),
                     strides=(1,1),
                     padding='valid'))
    model.add(Activation('relu'))
    # Batch Normalisation - normalizacija
    model.add(BatchNormalization())

    # 5. kovolucioni sloj
    model.add(Conv2D(filters=3,
                     kernel_size=(3,3),
                     strides=(1,1),
                     padding='valid'))
    model.add(Activation('relu'))
    # Pooling - agregacija
    model.add(MaxPooling2D(pool_size=(2,2),
                           strides=(2,2),
                           padding='valid'))
    # Batch Normalisation - normalizacija
    model.add(BatchNormalization())


    # Sloj za poravnanje
    model.add(Flatten())
    # 1. Dense sloj
    model.add(Dense(units = 120))
    model.add(Activation('relu'))
    # Dodajemo Dropout da sprecimo overfitting
    model.add(Dropout(0.4))
    # Batch Normalisation - normalizacija
    model.add(BatchNormalization())

    # 2. Dense sloj
    model.add(Dense(84))
    model.add(Activation('relu'))
    # Dodajemo Dropout
    model.add(Dropout(0.4))
    # Batch Normalisation - normalizacija
    model.add(BatchNormalization())

    # 3. Dense sloj
    model.add(Dense(64))
    model.add(Activation('relu'))
    # Dodajemo Dropout
    model.add(Dropout(0.4))
```

```
    # Batch Normalisation - normalizacija
    model.add(BatchNormalization())

    # Izlazni sloj
    model.add(Dense(NUM_OF_CLASSES))
    model.add(Activation('softmax'))

    model.summary()
    return model
```

Kod 3: AlexNet



Slika 27: Slojevi AlexNet modela

4.4 Analiza modela

U ovom poglvlju će biti upoređeni rezultati rada tri modela koje smo implementirale. U narednim tabelama je prikazana preciznost modela na test skupu, kao i vreme njegovog treniranja, u zavisnosti od toga kako su podešeni različiti parametri pri treniranju. U tabeli 2 su prikazani rezultati rada modela_1 u 10 epoha, u tabeli 3 modela LeNet5 u 10 epoha, u tabeli 4 modela LeNet5 u 100 epoha i u tabeli 5 modela AlexNet u 10 i 100 epoha.

Tabela 2: Model 1

Model 1, 10 epoha				
Funkcija aktivacije	optimizator	Batch size	preciznost na test skupu	vreme izvrsavanja
relu	adam	32	0.848	233.333
relu	adam	64	0.767	218.867
relu	sgd	32	0.879	213.162
relu	sgd	64	0.845	257.118
sigmoid	adam	32	0.230	255.351
sigmoid	adam	64	0.230	243.452
sigmoid	sgd	32	0.230	252.360
sigmoid	sgd	64	0.230	265.278
tanh	adam	32	0.230	231.808
tanh	adam	64	0.110	228.060
tanh	sgd	32	0.908	223.549
tanh	sgd	64	0.879	228.474

Tabela 3: Model 2

LeNet5, 10 epoha				
Funkcija aktivacije	optimizator	Batch size	preciznost na test skupu	vreme izvrsavanja
relu	adam	32	0.825	29.112
relu	adam	64	0.827	31.737
relu	adam	128	0.835	23.660
relu	sgd	32	0.830	27.591
relu	sgd	64	0.796	25.008
relu	sgd	128	0.487	22.703
sigmoid	adam	32	0.230	29.442
sigmoid	adam	64	0.230	25.430
sigmoid	adam	128	0.230	24.470
sigmoid	sgd	32	0.230	29.697
sigmoid	sgd	64	0.230	28.206
sigmoid	sgd	128	0.230	23.382
tanh	adam	32	0.866	29.580
tanh	adam	64	0.845	25.425
tanh	adam	128	0.872	24.269
tanh	sgd	32	0.843	29.354
tanh	sgd	64	0.817	26.013
tanh	sgd	128	0.772	23.323

Zaključi:

- Sigmoidnu funkciju ne treba koristiti ni u modelu_1 ni u LeNet modelu, jer sve slike klasifikuje u istu klasu (8. klasu)
- LeNet model je mnogo brži od modela_1: za približno isto vreme u modelu_1 se izvrsi 10 epoha, a u LeNet modelu 100 epoha
- Ali, kada se posmatra izvrsavanje u 10 epoha, model_1 daje malo bolje rezultate od LeNet (ako se izuzme kombinacija tangentne aktivacione funkcije i optimizacije pomocu Adama)
- Model_1 daje najveću preciznost (0.908) na test skupu kada se kao aktivaciona funkcija koristi tangenta funkcija, kada se optimizuje pomocu stohastickog gradijentnog spusta i kada je batch size = 32
- Model_1 ima nesto veću preciznost kada je batch size = 32, u odnosu na to kada je batch size = 64
- Kada poredimo izvršavanje LeNet modela u 10 i 100 epoha, vidimo da je preciznost tek malo bolja u 100 epoha, nego u 10 - za 0.01 ili 0.02. A izvrsavanje traje oko 10 puta duže. Zato se više isplati uzeti manji broj epoha.
- Kod AlexNet modela, vreme izvšavanja je mnogo veće (skoro 10 puta veće) za 100 nego za 10 epoha, što je i očekivano.
- Najmanja preciznost za AlexNet model se postiže za funkciju sigmoid, 10 epoha i *Batch size* 256, a iznosi 0.696. Takođe, to je jedini slučaj da je preciznost ispod 0.71, a uglavnom su preciznosti u intervalu [0.801, 0.934], iz čega zaključujemo da AlexNet model ima

Tabela 4: Model 2

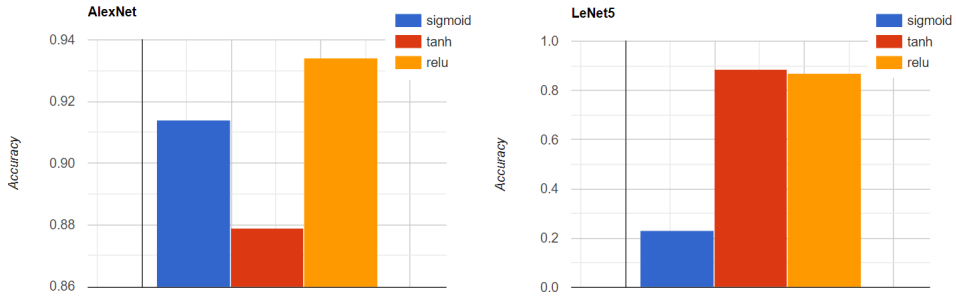
LeNet5, 100 epoha				
Funkcija aktivacije	optimizator	Batch size	preciznost na test skupu	vreme izvrsavanja
relu	adam	32	0.869	285.869
relu	adam	64	0.822	229.086
relu	adam	128	0.843	215.497
relu	sgd	32	0.796	255.521
relu	sgd	64	0.811	252.399
relu	sgd	128	0.730	221.506
sigmoid	adam	32	0.230	286.774
sigmoid	adam	64	0.230	249.128
sigmoid	adam	128	0.230	242.137
sigmoid	sgd	32	0.230	240.445
sigmoid	sgd	64	0.230	238.546
sigmoid	sgd	128	0.230	235.574
tanh	adam	32	0.887	274.488
tanh	adam	64	0.853	240.747
tanh	adam	128	0.866	212.645
tanh	sgd	32	0.816	274.816
tanh	sgd	64	0.780	235.054
tanh	sgd	128	0.780	219.117

dobru preciznost, kao i to da ona ne osciluje mnogo izvan pomenutog intervala.

- Prosečna preciznost za 10 epoha iznosi 0.824 (približno 82%), a prosečna preciznost za 100 epoha iznosi 0.907 (približno 90%), iz čega se može zaključiti da je za AlexNet model i dati skup podataka bolje trenirati model u što više epoha. Ovo potvrđuje i činjenica (poglavlje 4.3 da je u 1000 epoha ovaj model dostigao najveću preciznost (veću od sva tri modela) koja iznosi 0.95 (odnosno 95%).
- Može se uočiti da AlexNet model sporije trenira podatke u odnosu na LeNet5 mrežu, ali mnogo brže u odnosu na Model 1. Naime, najmanje vreme izvršavanja AlexNet mreže je postignuto za 10 epoha (funkciju ReLu i *Batch size* 256), a iznosi 33.786 sekunde, dok je maksimalno vreme koje LeNet5 model dostiže za 10 epoha (funkciju ReLu i *Batch size* 64) 31.737 sekundi.

Tabela 5: Rezultati za AlexNet model za različite vrednosti parametara: broj epoha (10 i 100), funkciju aktivacije (relu, tanh i sigmoid) i *Batch size* (32, 64 i 256).

AlexNet					
Br. epoha	Funkcija aktivacije	Optimizator	*Batch size*	Preciznost na test skupu	Vreme izvrsavanja (sekunde)
10	relu	adam	32	0.838	44.356
10	relu	adam	64	0.848	37.813
10	relu	adam	256	0.715	33.786
10	sigmoid	adam	32	0.919	43.755
10	sigmoid	adam	64	0.845	38.291
10	sigmoid	adam	256	0.696	36.811
10	tanh	adam	32	0.801	43.664
10	tanh	adam	64	0.906	39.264
10	tanh	adam	256	0.851	34.403
100	relu	adam	32	0.934	372.984
100	relu	adam	64	0.934	334.667
100	relu	adam	256	0.906	295.523
100	sigmoid	adam	32	0.914	372.971
100	sigmoid	adam	64	0.914	334.225
100	sigmoid	adam	256	0.856	292.743
100	tanh	adam	32	0.879	373.153
100	tanh	adam	64	0.921	339.918
100	tanh	adam	256	0.906	296.921



Slika 28: Grafici na kojima je prikazana preciznost za funkcije sigmoid, tanh i relu, kod AlexNet i LeNet-5 modela redom

5 Zaključak

U ovom radu obrađene su konvolutivne neuronske mreže kroz implementaciju tri modela za klasifikaciju slika saobraćajnih znakova. Kako je korišćeni skup podataka relativno mali, ne može se postići pun potencijal konvolutivnih mreža, ali smo kroz isprobavanje različitih vrednosti parametara uspele da postignemo dobru preciznost kojom modeli vrše klasifikaciju. Dva prikazana modela predstavljaju poznate arhitekture kon-

volutivnih mreža - to su LeNet-5 i AlexNet, dok smo arhitekturu jednog modela same osmislile.

Kako CNNs rade sa sve većim skupovima podataka, to mreža mora biti kompleksnija, a samim tim ona zahteva sve veće količine resursa - posebno memorijske resurse (zbog svih parametara koje je potrebno podesiti prilikom učenja modela). Ipak, CNN predstavljaju odlično rešenje za problem klasifikacije slika, i sve više se koriste.

Literatura

[1] a. Adam optimizer. on-line at: https://towardsdatascience.com/adam-latest-trends-in-deep-learning-optimization-6be9a291375c.

[2] a. Chinese Traffic Sign Database. on-line at: http://www.nlpr.ia.ac.cn/pal/trafficdata/recognition.html.

[3] Saad Albawi. Understanding of a Convolutional Neural Network. *The International Conference on Engineering and Technology*, 2017. on-line at: https://www.researchgate.net/publication/319253577_Understanding_of_a_Convolutional_Neural_Network.

[4] Anđelka Zečević Mladen Nikolić. *Mašinsko učenje*. 2019.

[5] Leon Bottou Yann LeCun, Patric Haffner. Object recognition with Gradient-Based learning. 1998.