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

Sadržaj

1 Uvod

Klasifikacija je vrsta problema nadgledanog učenja ¡FUSNOTA¿ koja podrazumeva predvidjanje kategoricke ciljne promenljive. Kategorickim se smatraju promenljive koje uzimaju konacan broj vrednosti (medju kojima nema uredjenja). Na primer, prepoznavanje da li je dobijen e-mail spam, reklama ili obicna poruka je problem klasifikacije.

Prepoznavanje i klasifikacija slika je polje u oblasti mašinskog učenja koje se veoma brzo razvija. Na primer, klasifikatori slika (mada se već koriste) će se sve više koristiti 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.

Postoje mnogi algoritmi koji se koriste za klasifikaciju slika (npr. SVM), a jedan od najboljih je CNN - konvolutivne neuronske mreže. CNN se može zamisliti kao automatski „ekstraktor" karakteristika slike. On efikasno koristi informacije o susednim pikselima da bi efektivno konvolucijom smanjio sliku, a zatim da bi predvideo kategoriju kojoj slika pripada, koristi sloj za predviđanje. 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.

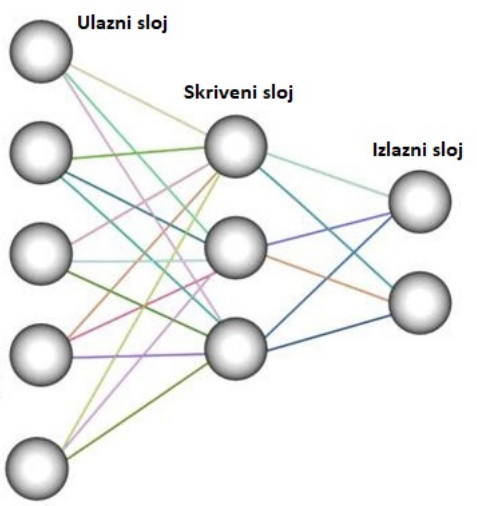
Svrha ovog rada je da prikaže osnovne osobine konvolutivnih neuronskih mreža i da pokaže na koji način se one mogu implementirati tako da korektno identifikuju saobraćajne znakove iz baze podataka sa kineskim saobraćajnim znacima. U radu je opisano nekoliko modela mreža i upoređeni su njihovi ostvareni rezultati.

2 Neuronske mreže

Neuronske mreže predstavljaju skup statističkih modela učenja inspirisane biološkim neuronima, za rešavanje klasifikacionih[1] i regresionih problema[2]. Njihove primene su mnogobrojne, a neke od njih su: kategorijzacija teksta, raspoznavanje i sinteza govora, autonomna voznja, igranje video igara, masinsko prevodenje prirodnih jezika i slicno. Kljucna prednost neuronskih mreza je da same mogu da konstruisu nove atribute nad sirovom reprezentacijom podataka[3] i odlicno baratanje sa velikim kolicinama podataka.

2.1 Arhitektura

Osnovnu jedinicu gradje neuronske mreze predstavljaju neuroni, koji su medjusobno povezani vezama sa tezinama koje se podesavaju tokom ucenja mreze. Povezani neuroni prosledjuju signale jedni drugima, a organizovani su u slojeve. Najjednostavniji oblik neuronske mreze je perceptron, koji sadrzi samo jedan ulazni i jedan izlazni sloj. Medjutim, kako on sluzi samo za ucenje linearnih modela, a u praksi se javlja potreba da i slozeniji modeli mogu da se nauce, osim perceptrona, koriste se viseslojne neuronske mreze. Viseslojna neuronska mreza osim ulaznog i izlaznog sloja, ima jedan ili vise skrivenih slojeva (slika 1). Kako se neuronske mreze uglavnom koriste za klasifikaciju uzorka u razlicite kategorije, ulazni sloj se sastoji od onoliko neurona kolika je dimenzionalnost ulaznog prostora, a broj neurona na izlazu jednak je broju klasa. Samo ucenje neuronske mreze je zapravo podesavanje tezina sve dok se ne dobije zadovoljavajuca aproksimacija izmedju ulaznih i izlaznih velicina.



Slika 1: Višeslojna neuronska mreža sa jednim skrivenim slojem

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

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

[3] Iako se nekad mogu pretpostaviti koji su atributi najinformativniji za predvidanje ciljne promenljive, izbori tih atributa su neretko losiji od onoga sto bi algoritam ucenja mogao da detektuje u sirovoj informaciji.

2.2 Razlog uvodjenja CNN

Jednostavni zadaci prepoznavanja mogu se dosta dobro resiti skupovima podataka malih velicina, na primer desetine hiljada slika. Medjutim, objekti u realisticnim postavkama pokazuju znacajnu varijabilnost, stoga, da bi bilo moguce nauciti prepoznati ih, potrebno je koristiti mnogo vece skupove za treniranje. Da bismo naucili o hiljadama objekata iz miliona slika, potreban nam je model sa velikim kapacitetom ucenja. Konvolutivne neuronske mreze (CNN), koje ce biti detaljno obradjene u ostatku rada, cine jednu takvu klasu modela. ne daju jake i uglavnom ispravne pretpostavke o prirodi slika, a njihov kapacitet se moze kontrolisati variranjem njihove dubine i sirine. Tako, u poredjenju sa standardnim (feedforward) neuronskim mrezama sa slojevima slicnih velicina, CNN imaju mnogo manje veza i parametara, tako da ih je lakse trenirati, dok je njihov teoretski najbolji ucinak verovatno samo nesto losiji.

3 Konvolutivne neuronske mreže

Konvolutivne neuronske mreze (eng. Convolutional neural network, CNN) predstavljaju podklasu neuronskih mreza koja ima najmanje jedan konvolutivni sloj (a moze ih imati i vise). Ova vrsta neuronskih mreza je inspirisana vizuelnim korteksom. Svaki put kada nešto vidimo, aktivira se niz slojeva neurona, i svaki sloj otkriva skup karakteristika kao što su linije, ivice itd. Visi nivoi slojeva otkritivaju složenije karakteristike kako bi prepoznali ono što smo videli. Konvolutivne mreze rade po istom principu i prakticno su uvek duboke neuronske mreze, upravo zbog toga sto je potrebno od sitnijih detalja, poput uspravnih, kosih i horizontalnih linija, koji obicno bivaju detektovani u nizim slojevima mreze, konstruisati slozenije oblike poput delova lica. Konvolutivne neuronske mreze se koriste u obradi signala (slike, zvuka), ali i teksta. U odnosu na ostale vrste neuronskih mreza, isticu se u prikupljanju lokalnih informacija (na primer o susednim pikselima na slici ili ,,okruzujucim"(eng. surrounding) recima u tekstu) i smanjenju slozenosti modela (brze treniranje, potrebno je manje izoraka, manja sansa da dodje do preprilagodjavanja (eng. overfitting)). Konvulativne neuronske mreze se zasnivaju na sposobnosti mreza da iz sirovog signala konstruisu atribute. Nazivaju se konvolutivnim zato sto uce **filtere** (pojam objasnjen u tabeli 1), cijom konvolutivnom primenom detektuju odredjena svojstva signala.

Postoji nekoliko arhitektura u polju konvolutivnih neuronskih mreža, koje se razlikuju po tipu, broju i rasporedu slojeva. Dve najpoznatije su:

- LeNet-5 - ovu arhitekturu je napravio Yann LeCunn krajem 20. veka, a koristila se za citanje zip kodova, cifara itd. LeNet-5 arhitektura se sastoji iz dva skupa konvolutivnih slojeva (eng. convolutional layers) i slojeva agregacije (eng. average pooling layers), koji su praćeni poravnavajućim konvolutivnim slojem (eng. flattening convolutional layer), a onda slede dva potpuno-povezana sloja (eng. two fully-connected layers) i konačno softmax klasifikator (eng. softmax classifier).
- AlexNet 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 (eng. Rectified Linear Unit, ReLu)[4] za ne-linearni deo umesto Tanh ili Sigmoid funkcije, koje su bile standardne za tradicionalne neuronske mreže.

3.1 Parametri

U ovoj sekciji je dat tabelarni prikaz parametara koji su najznacajniji za implementaciju konvolutivne mreze. U tabeli 1 je dat samo kratak opis svakog od njih radi boljeg razumevanja teksta koji sledi. Ipak, u nastavku je svaki detaljnije opisan.

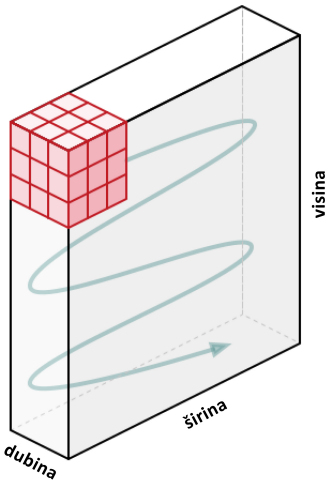
Tabela 1: Primer tabele

Filter (jezgro, kernel)	- matrica sa tezinama za konvoluciju ulaza - daje meru koliko deo ulaza liči na karakteristiku - tezine u matricama filtera su izvedene za vreme treniranja podataka
Padding	- koristi se za dodavanje kolona i redova nula da bi se odrzala konstantna velicina matrice (mape) nakon konvolucije - ovaj parametar moze da unapredi performanse tako sto zadrzava informacije u okvirima
Stride	- broj piksela koji želite preskočiti, dok prelazite ulaz vodoravno i uspravno tokom konvolucije, nakon mnozenja svakog elementa iz ulazne matrice težina s onima u filtru
Number of Channels	- It is the equal to the number of color channels for the input but in later stages is equal to the number of filters we use for the convolution operation. - The more the number of channels, more the number of filters used, more are the features learnt, and more is the chances to over-fit and vice-versa.
Pooling-layer Parameters	- ima iste parametre kao i konvolutivni sloj - uglavnom se koristi Max-Pooling opcija - The objective is to down-sample an input representation (image, hidden-layer output matrix, etc.), reducing its dimensionality by keeping the max value(activated features) in the sub-regions binned.

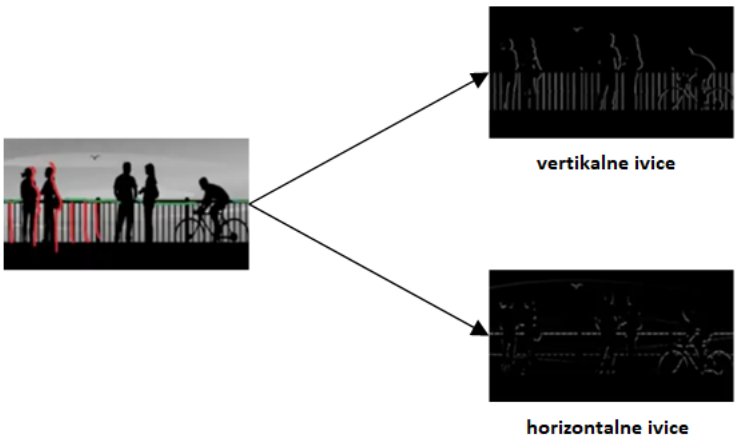
[4]ReLu funkcija se definiše kao: f(x) = max(0,x). Prednost ReLu nad sigmnoidnom funkcijom je što treniranje obavlja mnogo brže.

3.2 Konvolucija slika preko CNN

Da bi izvrsile klasifikaciju slika, konvolutivne mreze (preciznije, konvolutivni sloj, detaljnije u poglavlju ?????) obavljaju neku vrstu pretrage. Ovo se moze zamisliti kao mali pokretni (ili klizni) prozor (prikazano na slici 2) koji klizi s leva na desno preko vece slike, i nastavlja s leve strane kada dodje do kraja jednog prelaza (kao kod pisace masine). Taj pokretni (klizni) prozor - koji ustvari predstavlja filter, moze da prepozna samo jednu stvar, recimo kratku vertikalnu liniju (tri tamna piksela naslagana jedan na drugi). Slicno, neki drugi filter moze da sluzi za prepoznavanje horizontalnih linija, i on se takodje pomera preko piksela slike, trazeci podudaranja. Rezultat koji se postize filterima koji prepoznaju vertikalne i horizontalne linijije prikazan je na slici 3.

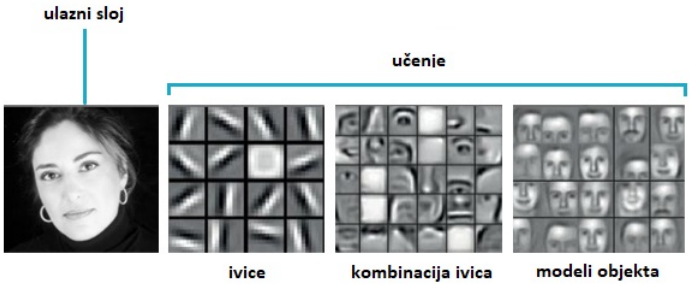


Slika 2: Kretanje filtera



Slika 3: Detektovanje ivica

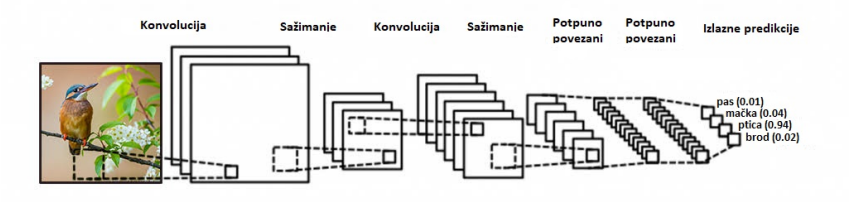
Svaki put kada dodje do poklapanja (filtera sa ulazom), ono se mapira u prostor sa karakteristikama - koji se zove **mapa karakteristika** (eng. feature maps), koji je specifican za taj vizuelni element. U tom prostoru (tj. mapi) se cuva (odnosno belezi) lokacija svakog poklapanja sa vertikalnom linijom. Konvolutivna mreza pokrece mnogo pretraga nad jednom istom slikom. Na pocetnom sloju mreze 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 slozeniji oblici. Ovo je demonstrirano na slici 4.



Slika 4: Učenje mreže

3.3 Unutrasnja struktura CNN

Unutrasnja struktura konvolutivne mreže se sastoji od nekoliko naizmeničnih konvolutivnih slojeva (eng. convolution layer) i slojeva agregacije (eng. pooling layer), pri cemu je dozvoljeno pojavljivanje iste vrste sloja vise puta (prikazano na slici 5). U dosad opisanoj strukturi neurona (neuronskih mreza) izlaz iz svakog od njih je bio skalarna veličina. Izlazi konvolutivnog sloja su dvodimenzionalni i nazivaju se mapama karakteristika (eng. feature maps). Oni se transformisu nelinearnom aktivacionom funkcijom. U konvolutivnim mrežama kao aktivacione funkcije najčešće koriste ReLU (na izlazu iz konvolutivnog sloja) i Softmax (na izlazu iz poslednjeg, potpuno povezanog, sloja mreže). O njima će biti više reči u narednim poglavljima.



Slika 5: Arhitektura konvolutivne neuronske mreže

3.4 Konvolucija

Konvolutivni sloj je glavni deo konvolutivne neuronske mreze koji radi najvise izracunavanja u mrezi. Njegova uloga je konstrukcija novih atributa. To je prvi sloj koji ekstraktuje karakteristike iz ulazne slike. Konvolucija je matematicka operacija koja uzima dva ulaza - dve matrice f i g, dimenzija m x n i p x q, a definisana je na sledeci nacin:

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

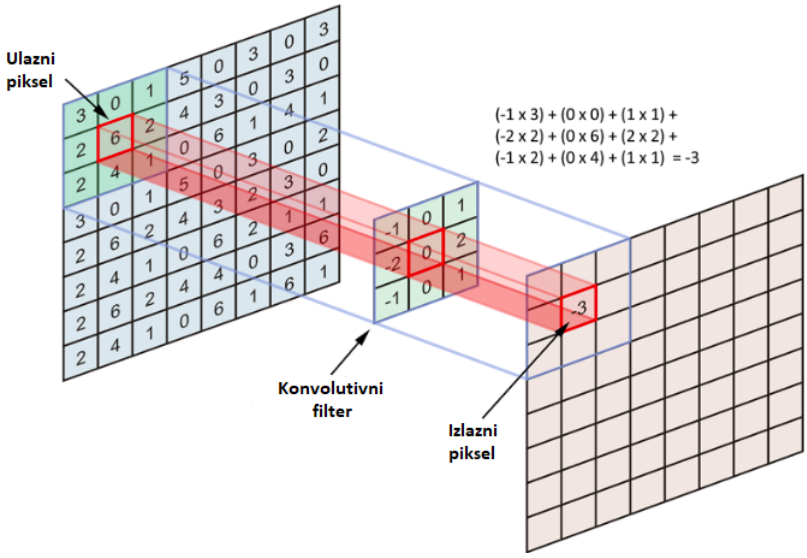
Matrica f je obicno ulaz, poput slike, dok je matrica g filter. Određena ulazna reprezentacija podatka (originalna slika) konvolvira se sa filterom sa određenim parametrima (ti parametri predstavljaju i parametre konvolutivne mreze). Procesom učenja ovi parametri (težine koje je potrebno nauciti kako bi mreža davala dobre rezultate) se podešavaju i bivaju nauceni. Filteri su najcešce manjih prostornih dimenzija od ulaza, ali uvek su jednake dubine kao i ulaz. Kao sto je vec receno, konvolucijska jezgra (filteri) filtriraju sliku, tj. mapu karakteristika kako bi izlučila 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 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.

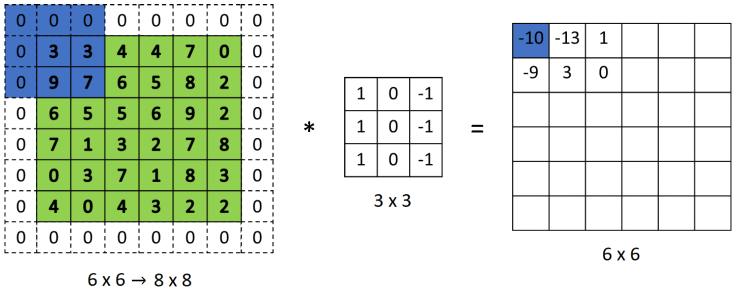
Izlaz jednog filtera će biti dvodimenzioni niz. Ako imamo više filtera, izlaz iz konvolucionog sloja će biti rezultati svakog filtera poredjanih po dubini. Operacijom konvolucije dobija se transformirana slika dimenzije I*K, gde je I dimenzija ulazne matrice slike, a K je dimenzija filtera koji je primenjen nad tom slikom. Takodje, dobijena transformacija je lokalna tj. pikseli izlaza zavise od lokalnih, susednih piksela ulaza. Ceo ovaj proces je prikazan na slici 6.

3.4.1 Prosirivanje

Formula konvolucije koja je data u poglavlju 3.4 nije definisana za sve indekse i=0, m - 1 i j=0, n - 1. Na primer, ako je i, j = 0 i k, l ¿ 0, vrednost fi - k,i - l nije definisana. Ukoliko bi se u obzir uzele samo definisane vrednosti, dimenzija konvolucije bi bila manja od dimenzije matrice f. Medjutim, to nije uvek pozeljno, i moze se izbeci tako sto se vrsi **prosirivanje** (eng. padding) matrice f, na primer nulama ili vrednostima koje su vec na obodu, tako da velicina rezultujuce matrice bude jednaka velicini matrice f pre prosirivanja. Ovo je prikazano na slici 7. Takode, prilikom racunanja konvolucije, filter se duz slike ne mora pomerati za jedan piksel, vec za neki veci **korak** (eng. stride).



Slika 6: Konvolucija primenom filtera dimenzije 3x3 na matricu dimenzije 8x8



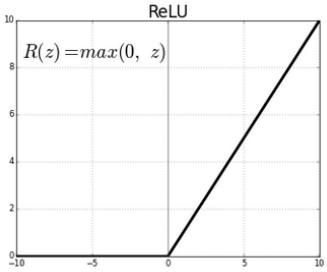
Slika 7: Primer prosirivanja

3.4.2 Aktivaciona funkcija

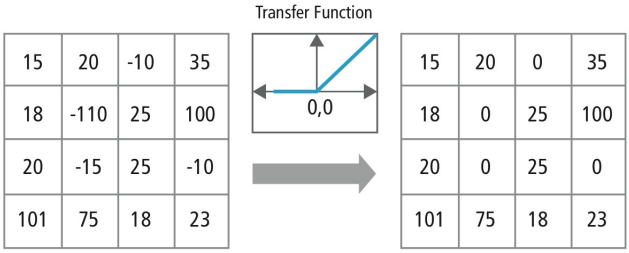
U konvolutivnim slojevima kao aktivaciona funkcija se najčešće koristi ReLu (Rectified Linear Unit) funkcija. Definisana je kao

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

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: Izgled ReLU funkcije



Slika 9: Primer rada ReLU funkcije nad mapom karakteristika

3.5 Agregacija

Uloga sloja za agregaciju je smanjenje broja parametara kada su slike prevelike, kao i smanjenje broja racunskih operacija u visim slojevima smanjuje. Agregacija smanjuje dimenzionalnost svake mape, ali zadržava važne informacije. Sve to rezultuje smanjenjem racunske zahtevnosti pri optimizaciji i pomaze u kontroli preprilagođavanja. Zato zelimo da slojevi za agregaciju prate konvolucione slojeve kako bismo postepeno smanjili prostornu veličinu (širinu i visinu) prikaza podataka.

Cesto nas konačni zadatak postavlja neko globalno pitanje o slici, npr., Da li sadrži mačku? Tako da čvorovi našeg zadnjeg sloja moraju biti osetljivi na celi ulaz. Postepenom agregacijom informacija, stvarajuci grublje i grublje mape karakteristika, postize se taj cilj da se na kraju nauci globalna reprezentacija, zadrzavajuci sve prednosti konvolucijskih slojeva na srednjim slojevima obrade.

Sloj agregacije ukrupnjuje informacije, tako sto racuna neku jednostavnu funkciju agregacije susednih jedinica prethodnog sloja, poput maksimuma (eng. Max pooling), koji vraca maksimalnu vrednost dela slike pokrivene filterom, ili proseka (eng. Average pooling), koji vraca prosecnu vrednost dela slike pokrivene filterom. Ukoliko agregira, na primer, 3 x 3 piskela, onda je broj izlaza ovog sloja 9 puta manji od broja izlaza prethodnog. Kada se racuna maksimum, dolazi do zanemarivanje informacije o tome gde je precizno neko svojstvo (poput uspravne linije) pronadeno, ali se ne gubi informacija da je pronadeno. Ovakva vrsta zanemarivanja informacije cesto ne steti cilju koji treba postici. Na primer, ako su na slici pronadeni kljun i krila, informacija o tacnoj pozicijiji najverovatnije nije bitna za odlucivanje da li se na slici nalazi ptica. Ipak, ukoliko je

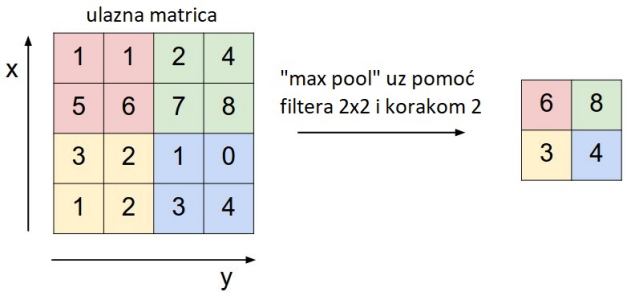
potrebno napraviti mrezu koja igra igru u kojoj su pozicije objekata na ekranu bitne, nije pozeljno koristiti agregaciju.

Average pooling was often used historically but has recently fallen out of favor compared to the max pooling operation, which has been shown to work better in practice.

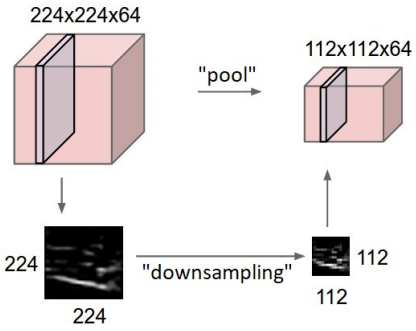
Pooling layer downsamples the volume spatially, independently in each depth slice of the input volume.

Prva slika (pool): In this example, the input volume of size [224x224x64] is pooled with filter size 2, stride 2 into output volume of size [112x112x64]. Notice that the volume depth is preserved.

Druga slika (maxpool): The most common downsampling operation is max, giving rise to max pooling, here shown with a stride of 2. That is, each max is taken over 4 numbers (little 2x2 square).



Slika 10: Maxpool



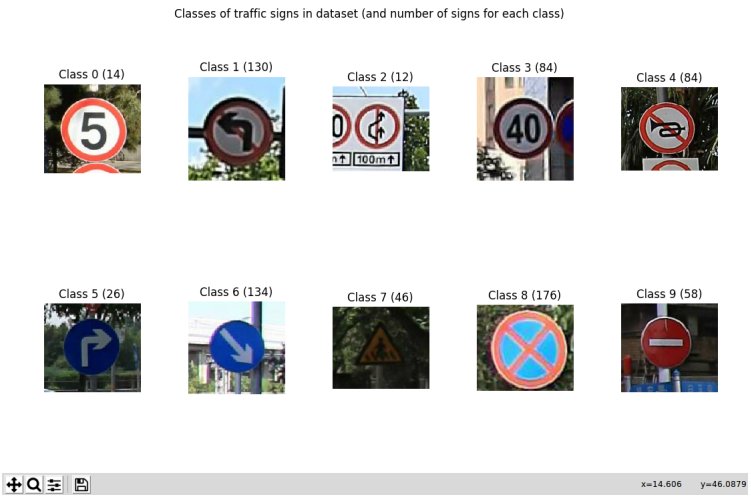
Slika 11: Pool

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 [1]. 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 mreza 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 slike. Na slici 12 je prikazan po jedan znak iz svake klase trening skupa, zajedno sa brojem elemenata te klase. Slično, ti podaci o test skupu, mogu se videti na slici 13.

Slika 12: Trening skup



Slika 13: Test skup

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, dva agregirajuća i dva potpuno povezana sloja. Detaljna arhitektura mreže se može videti na slici 14. U svim konvolutivnim slojevima veličina jezgra je 3x3, a broj filtera na izlazu iz konvolucije je 32. Takođe, u svakom se koristi ReLU aktivaciona funkcija. U sloju agregacije, sažimanje se vrši biranjem maksimalne vrednosti dela mape karakteristika koji je prekriven filterom. Kako ne bi došlo do preprilagodjavanja modela trening podacima, povremeno je, pomoću funkcije Dropoup(), 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, 64x64 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

```
_________________________________________________________________
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 14: Slojevi modela neuronske mreže

Na slici 15 možemo videti da je preciznost modela na trening skupu tokom poslednje tri epohe varirala oko 0.99 i 1.0. Na slici 16 možemo videti kako se model pokazao na test skupu. Preciznost (*accuracy*, tj. broj tačno klasifikovanih instanci / ukupan broj instanci) je jednaka 0.882. Vidimo da je i preciznost (*precision*) koja se odnosi na svaku od klasa dosta visoka (osim za klase 0 i 5)

```
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 15: Preciznost na trening podecima

```
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 16: Preciznost na test podecima

4.2 Model 2: LeNet-5

Još bolje rezultate smo dobile pomoću LeNet-5 mreže [2]. LeNet-5 mrežu je 1990ih napravio Yann LeCunn. Mreža se sastoji iz 7 slojeva, pri čemu se ne računa ulazni sloj. Sve slike smo skalirale na večičinu 32x32 piksela, jer ova mreža kao ulaz očekuje slike tih dimenzija. Prvi konvolutivni sloj na izlazu daje 6 mapa karakteristika, koristi jezgro veličine 3x3 i kao aktivacionu funkciju koristi ReLU. Nakon njega sledi sloj agregacije u kom se sažimanje vrši 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 17 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

```
Using TensorFlow backend.
Number of classes: 10
Number of images for training: 1639
classes shape:  (1639, 10)
images shape:  (1639, 32, 32, 3)
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 30, 30, 6)         168
_________________________________________________________________
average_pooling2d_1 (Average (None, 15, 15, 6)         0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 13, 13, 16)        880
_________________________________________________________________
average_pooling2d_2 (Average (None, 6, 6, 16)          0
_________________________________________________________________
flatten_1 (Flatten)          (None, 576)               0
_________________________________________________________________
dense_1 (Dense)              (None, 120)               69240
_________________________________________________________________
dense_2 (Dense)              (None, 84)                10164
_________________________________________________________________
dense_3 (Dense)              (None, 10)                850
=================================================================
Total params: 81,302
Trainable params: 81,302
Non-trainable params: 0
_________________________________________________________________
Train on 1311 samples, validate on 328 samples
```

Slika 17: Slojevi modela leNet-5 neuronske mreže

Na slici 18 možemo videti da je preciznost na trening skupu u poslednje tri iteracije bila najveća moguća, tj 1.00. Na slici 19 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.

```
Epoch 18/20
1311/1311 [==============================] - 3s 2ms/step - loss: 1.5170e-04 - acc: 1.0000 - val_loss: 0.0012 - val_acc: 1.0000
Epoch 19/20
1311/1311 [==============================] - 3s 2ms/step - loss: 1.4775e-04 - acc: 1.0000 - val_loss: 0.0012 - val_acc: 1.0000
Epoch 20/20
1311/1311 [==============================] - 3s 2ms/step - loss: 1.4572e-04 - acc: 1.0000 - val_loss: 0.0012 - val_acc: 1.0000
```

Slika 18: Preciznost na trening podecima

```
Test accuracy = 0.9109947643979057

Classification report:
              precision    recall  f1-score   support

           0       0.86      0.86      0.86        14
           1       0.82      1.00      0.90       130
           2       1.00      1.00      1.00        12
           3       0.87      0.93      0.90        84
           4       1.00      0.55      0.71        84
           5       0.71      0.92      0.80        26
           6       0.98      0.93      0.95       134
           7       0.85      1.00      0.92        46
           8       0.97      0.97      0.97       176
           9       1.00      0.93      0.96        58

   micro avg       0.91      0.91      0.91       764
   macro avg       0.91      0.91      0.90       764
weighted avg       0.92      0.91      0.91       764

Confusion matrix:
[[ 12   0   0   0   0   0   0   2   0   0]
 [  0 130   0   0   0   0   0   0   0   0]
 [  0   0  12   0   0   0   0   0   0   0]
 [  0   4   0  78   0   2   0   0   0   0]
 [  0  24   0   8  46   0   0   2   4   0]
 [  0   0   0   0   0  24   0   0   2   0]
 [  0   0   0   4   0   6 124   0   0   0]
 [  0   0   0   0   0   0   0  46   0   0]
 [  2   0   0   0   0   2   2   0 170   0]
 [  0   0   0   0   0   0   0   4   0  54]]
```

Slika 19: Preciznost na test podecima

5 Zaključak

Literatura

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

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

A Dodatak

A.1 Podnaslov 1

Dodatna objasnjenja

By visualizing the output from different convolution layers in this manner, the most crucial thing that you will notice is that the layers that are deeper in the network visualize more training data specific features, while the earlier layers tend to visualize general patterns like edges, texture, background etc.

This knowledge is very important when you use Transfer Learning whereby you train some part of a pre-trained network (pre-trained on a different dataset, like ImageNet in this case) on a completely different dataset. The general idea is to freeze the weights of earlier layers, because they will anyways learn the general features, and to only train the weights of deeper layers because these are the layers which are actually recognizing your objects.

Convolved Feature, Activation Map or Feature Map is the output volume formed by sliding the filter over the image and computing the dot product. Perceptrons come first in 1950s, and it uses a brittle activation function to do classification, so if w*x is greater than some value it predicts positive, otherwise negative.

Neurons uses a softer activation function by introducing a sigmoid function, a tanh function or other activation functions to pass on values to other neurons in the network.

So perceptrons do not use in a network setting, they do classification on their own, hence they can't classify XOR, however neurons can because they all contribute forward to the final output, using more complicated structure(i.e. multiple layers in network), they are able to classify XOR and other complicated problems.

A CNN, in specific, has one or more layers of convolution units. A convolution unit receives its input from multiple units from the previous layer which together create a proximity. Therefore, the input units (that form a small neighborhood) share their weights.

The convolution units (as well as pooling units) are especially beneficial as:

They reduce the number of units in the network (since they are many-to-one mappings). This means, there are fewer parameters to learn which reduces the chance of overfitting as the model would be less complex than a fully connected network. They consider the context/shared information in the small neighborhoods. This future is very important in many applications such as image, video, text, and speech processing/mining as the neighboring inputs (eg pixels, frames, words, etc) usually carry related information.