SVEUČILIŠTE U SPLITU

FAKULTET ELEKTROTEHNIKE, STROJARSTVA I BRODOGRADNJE

SEMINARSKI RAD IZ KOLEGIJA STROJNO UČENJE

PREPOZNAVANJE ZNKOVA S REGISTARSKIH OZNAKA POMOĆU KONVOLUCIJSKE NEURALNE MREŽE

Ivan Sičaja

Split, kolovoz 2020.

SADRŽAJ

[1 UVOD 3](#_Toc49726473)

[2 REALIZACIJA ALGORITMA 4](#_Toc49726474)

[2.1 ODABIR PROGRAMSKOG OKRUŽENJA I PROGRAMSKOG JEZIKA 4](#_Toc49726475)

[2.2 PRIPREMA BAZE PODATAKA 4](#_Toc49726476)

[2.3 KREIRANJE I TRENIRAJNE MODELA KONVOLUCIJSKE NEURALNE MREŽE 5](#_Toc49726477)

[2.4 SEGMENTACIJA NIZA ZNAKOVA REGISTARSKE OZNAKE 10](#_Toc49726478)

[3 TEST 13](#_Toc49726479)

[4 ZAKLJUČAK 14](#_Toc49726480)

[5 LITERATURA 15](#_Toc49726481)

# UVOD

U ovom seminarskom radu ukratko ćemo objasniti cjelokupan postupak realizacije algoritma za prepoznavanje znakova s registarskih tablice pomoću konvolucijske neuralne mreže. Konvolucijska neuralna mreža je neuralna mreža koja posjeduje sposobnost prepoznavan karakterističnih oblika sa slike. U nastavku ćemo nabrojati faze realizacije projekta, a to su:

-odabir programskog okruženja i jezika u kojima će se program realizirati

-priprema baze podataka

-kreiranje i treniranje modela konvolucijske neuralne mreže

-segmentacija niza znakova registarske oznake

-test

Na samom kraju ćemo navesti određena opažanja kao i najbitnije zaključke vezane za realizaciju samog projekta.

# REALIZACIJA ALGORITMA

## ODABIR PROGRAMSKOG OKRUŽENJA I PROGRAMSKOG JEZIKA

Svakako jednu od najbitnijih odluka predstavlja odabir kvalitetnog programskog okruženja kao i programskog jezika koji omogućuje kreiranje i treniranje konvolucijske neuralne mreže.

Programsko okruženje koje ćemo koristiti je Visual Studio Code koji je uz sve svoje zaista impresivne mogućnosti potpuno besplatan za korištenje.

Programski jezik koji ćemo koristiti je Python, najpopularniji jezik današnjice koji uz popularnost povlači mnoštvo kvalitetne literature što je vrlo bitan faktor pri odabiru programskog jezika. Uz mnoštvo literature tu je i mnoštvo mogućnosti kao što su instalacije različitih modula od kojih ćemo posebno istaknuti module TensorFlow i Keras koji nam omogućuju kreiranje i treniranje konvolucijskih neuralnih mreže u ovom programskom jeziku.

## PRIPREMA BAZE PODATAKA

Priprema reprezentativne i mnogobrojne baze podataka kao i shvaćanje svrhe različitih operacija koji se provode nad bazom podataka u svrhu proizvodnje kvalitetnog algoritma koji će uspješno klasificirati željene objekte vrlo često predstavlja najzahtjevniji dio postupka realizacije algoritma. Ako baza podataka nije reprezentativna odnosno sastoji se od primjera koji vrlo odstupaju od stvarnih primjera koje želimo klasificirati algoritam će biti loš iz razloga što će naučiti određene stvari koje nama nisu bitne. Ako baza podataka nije mnogobrojna doći će do pretreniranja algoritma (eng. Overfitting) što je također vrlo nepoželjna osobina, a očituje se u tome da će algoritam vrlo teško moći predviđati određene uzorke koji se nalaze izvan skupa za treniranje. Cilj našeg algoritma je da sa vrlo visokom točnošću predviđa određene objekte ali istodobno da bude vrlo fleksibilan, odnosno u mogućnosti da pravilno klasificira primjere koji se ne nalaze u testnom skupu.

Veličina baze podataka ovisi od samoj svrsi, a može se kretati od nekoliko stotina primjera do nekoliko stotina tisuća primjera.

Nakon što smo zadovoljili kriterij reprezentativnosti počinjemo skup za treniranje određenim primjerima. Velike baze podataka za neke tipične primjene dostupne su na internetu, a ako se radi o specifičnim primjenama potrebno je samostalno izraditi bazu podataka. Za naš projekt koristit ćemo dvije baze podataka odnosno bazu podataka za prepoznavanje slova i bazu podataka za prepoznavanje brojeva te ćemo ih radi jednostavnosti spojiti u jednu bazu podataka koja se sastoji od 36 klasa odnosno 26 slova i 10 brojeva. Sljedeća stvar na koju trebamo pripaziti je način pohrane i funkcionalnost baze podatka. Standardni načini pohrane baze podataka slika u .png i .jpg formatima zauzimaju jako puno prostora za skladištenje, vrlo su nepregledni i vrlo ne funkcionalni iz razloga što nam dodatno kompliciraju saznavanje klase pripadnosti za određeni primjer, stoga ćemo cijelu bazu podataka za treniranje pohraniti u .csv format, a prednosti su mnogostruke:

-ušteda na prostoru za pohranu za oko 2200 puta za naš slučaj

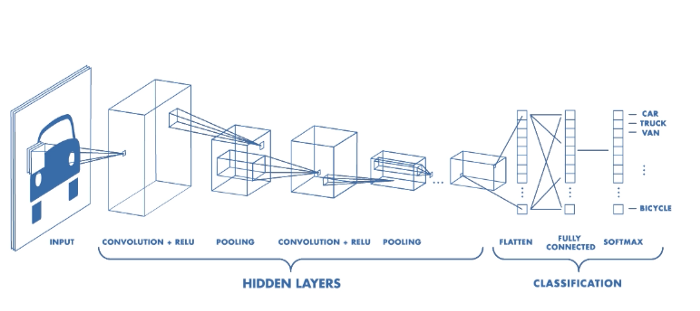
-uvelike poboljšana preglednost

-drastično povećava funkcionalnost

Svaki naš primjer predstavlja slika dimenzija 28x28 piksela što predstavlja uobičajene dimenzije odnosno MNIST (eng. Modified [National Institute of Standards and Technology](https://en.wikipedia.org/wiki/National_Institute_of_Standards_and_Technology)) standard. Naša baza podataka se sastoji od 432452 slike, odnosno 432452 retka tablice, a svaki redak se sastoji od 785 stupaca kod kojih prvi stupac predstavlja klasu kojoj pripada određena slika dok ostalih 784 stupca predstavljaju brojčane vrijednosti boje svakog piksela slike dimenzija 28x28 piksela. Ovako pohranjeni podatci omogućuju računalu vrlo brzu obradu informacija pohranjenih u slici.

## KREIRANJE I TRENIRAJNE MODELA KONVOLUCIJSKE NEURALNE MREŽE

Uobičajena struktura konvolucijske neuralne mreže prikazana je na sljedećoj slici.



Slika 2.1 Uobičajena struktura konvolucijske neuralne mreže[1]

Model konvolucijske mreže koji ćemo koristiti u projektu je sekvencijalni model (eng. Sequential) odnosno model koji omogućuje slanje ulaznih podataka na točno jedan ulaz te daje točno jedan izlaz.

Kada smo izabrali model unutar njega kreiramo određene slojeve (eng. layers) koji filtriraju određene parametre iz ulaznih primjera zahvaljujući kojima konvolucijska neuralna mreža uči kako ispravno klasificirati određeni primjer.

Slojevi koje ćemo kreirati unutar sekvencijalnog modela su:

- konvolucijski sloj

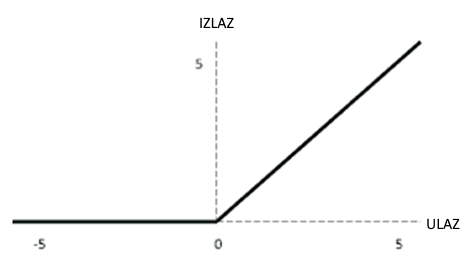
- „Maxpooling“ sloj

- „Flatten“ sloj

- „Dense“ sloj

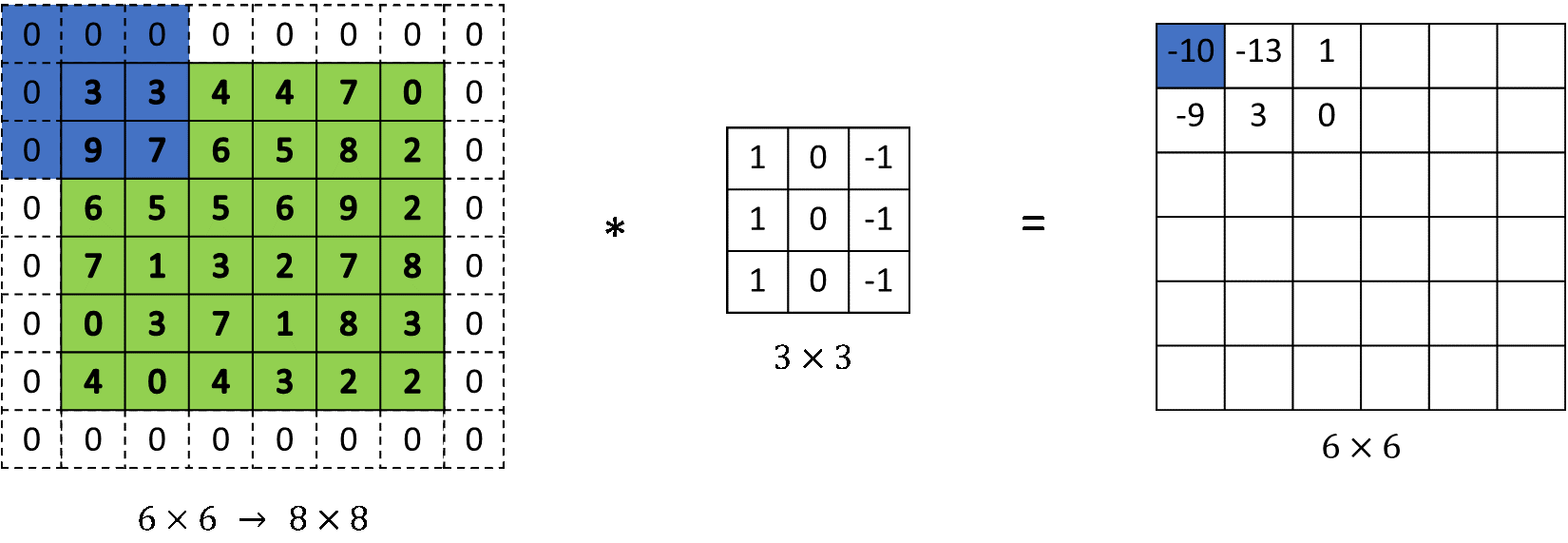
- „Dropout“ sloj

**Konvolucijski sloj** je sloj koji ima mogućnost prepoznavanja određenih oblika koji su definirani matricom određenih dimenzija i određenih vrijednosti odnosno konstanti koje zapravo predstavljaju RELU (eng. Rectified Linear Unit) aktivacijsku funkciju. RELU aktivacijska funkcija je funkcija koja eksponencijalno pojačava određene vrijednosti dok određene vrijednosti koje su joj nebitne postavlja na vrijednost nula što nam omogućuje dodatno isticanje određenih karakterističnih oblika. Oblik RELU aktivacijske funkcije prikazan je na sljedećoj slici.



Slika 2.2 Oblik RELU aktivacijske funkcije[2]

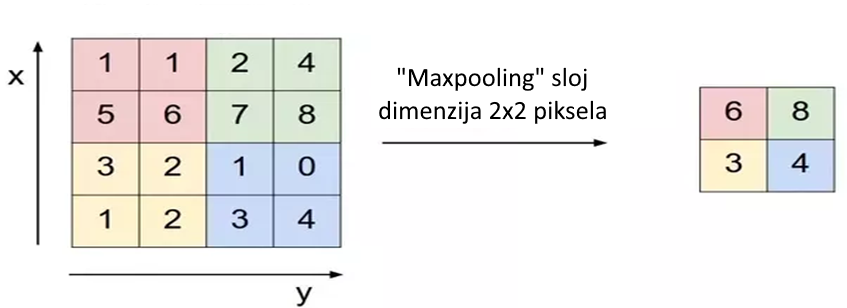
**Konvolucijski sloj** određenih dimenzija npr. dimenzija 3x3 piksela koji se sastoji od određenih konstanti se pomjera kroz cijelu sliku te sa svojim konstantama množi one vrijednosti boje piksela s kojima se trenutno preklapa te tako izmnožene vrijednosti svih piksela sumira te upisuje u novu matricu koja sada predstavlja matricu karakterističnih oblika koja je prema svojim dimenzijama manja od matrice ulazne slike. Način funkcioniranja konvolucijskog sloja prikazan je na sljedećoj slici.



Slika 2.3 Prikaz rada konvolucijskog sloja s RELU aktivacijskom funkcijom [3]

O tome koliko ćemo karakterističnih oblika paralelno prepoznavati ovisi o broju filtera koji definiramo u samom programu.

Kada smo izvukli određeni broj karakterističnih oblika potrebno je izvršiti sortiranje odnosno kreirati matricu koja će sadržavati samo bitne karakteristične oblike na osnovu kojih algoritam može naprednije učiti, a postupak procesiranja informacija smo dodatno pojednostaviti. Navedena poboljšanja omogućuje nam „**Maxpooling“ sloj**, određenih dimenzija npr. dimenzija 2x2 piksela koji uzima maksimalnu vrijednost od trenutna četiri piksela s kojima se preklapa odnosno uzima najkarakterističniji oblik te ga pohranjuje u novu reduciranu matricu oblika koja se trenutno sastoji od najkarakterističnijih oblika. Način rada „Maxpooling“ sloja prikazan je na sljedećoj slici.

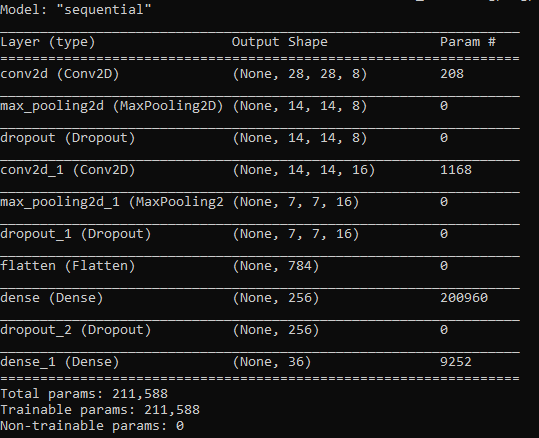


Slika 2.4 Prikaz rada „Maxpooling“ sloja [4]

Kad smo izvukli najkarakterističnije oblike potrebno je izvršiti predimenzioniranje podataka odnosno poredati sve karakteristične oblike u jedan niz odnosno skriveni sloj konvolucijske neuralne mreže koji se naziva **„Flatten“ sloj** s ciljem daljnjeg slanja podataka prema izlazu. Kako bismo povezali „Flatten“ s narednim slojem potrebno je koristit „Dense“ odnosno „Fully connected“ sloj s određenim brojem neurona. U našem slučaju **prvi „Dense“ sloj** se za naš slučaj sastoji od 256 neurona kojima je kao i u prethodnim slojevima korištena aktivacijska RELU funkcija.

Kako bismo povećali fleksibilnost algoritma odnosno spriječili pretreniranost (eng. overfitting) koristimo **„Dropout“ sloj** koji nasumice iz određenog sloja izbacuje određeni postotak neurona s ciljem da isti sloj s manje neurona donosi zaključke kao i sloj s više neurona odnosno čini neurone „sposobnijima“ za donošenje zaključaka o kojoj znamenci se radi iz razloga što manji broj neurona uči dodatne informacije odnosno koriti duge putove kako doći do određenog zaključka.

Konačno kada imamo dovoljno strojni naučene neurone potrebno je kreirati **izlazni „Dense“ sloj** s određenim brojem izlaznih klasa odnosno neurona. Za naš slučaj posljednji sloj će imati 36 klasa odnosno neurona koji predstavljaju 26 slova i 10 brojeva koje naša konvolucijska neuralna mreža treba razlikovati. Kao aktivacijska funkcija u posljednjem “Dense“ izlaznom sloju koristimo **„Softmax“ aktivacijsku funkciju** koja za svoj izlaz daje određenu vjerojatnost date kombinacije prepoznatih karakterističnih oblika koji pripadaju određenoj klasi s određenom vjerojatnošću. Budući da koristimo sekvencijalni model odnosno model koji omogućava točno jedan izlaz, izlaz iz naše konvolucijske neuralne mreža bit će ona klasa odnosno neuron iz posljednjeg sloja koji prema „Softmax“ posjeduje najvišu vjerojatnost. Struktura kreiranog modela prikazana je na sljedećoj slici.



Slika 2.5 Prikaz kreiranog modela

Što se tiče treniranja potrebno je poznavati dva parametra, a to su:

-broj primjera koji se paralelno šalju na ulaz u mrežu (eng. batch size)

-broj prolazaka kroz skup za treniranje (eng. epoch)

Ovisno o brojnosti primjera iz skupa za treniranje parametar „Batch size“ može poprimati različite vrijednosti. Uobičajene vrijednosti koje se koriste su **8, 16, 32, 64, 128, 256, 512** te **1024** primjera, koji ćemo broj izabrati ovisi uglavnom o brojnosti baze podataka za treniranje. Ako je „Batch size“ vrlo mali broj algoritam će imati veću točnost, ali će proces treniranja biti vrlo dug ako se redi o našoj bazi podataka koja se sastoji od **432452** primjera dok će za iznos parametra „Batch size“ od 1024 konačna točnost učenja će biti manja u odnosu na prvi slučaj dok će vrijeme treniranja biti višestruko kraće.

„Epochs“ nam definira broj koliko ćemo puta u svrhu treniranja prolaziti kroz skup za treniranje. Uobičajeno je da se povećanjem prolazaka kroz skup do određene razine, povećava i konačna točnost algoritma, ali se povećava i sam postupak treniranja.

Za naš projekt „Batch size“ je imao vrijednost **512** dok smo za vrijednost „Epochs“ uzeli vrijednost **60**.

Postoje dodatni parametri koje je moguće definirati u svrhu poboljšanja klasifikacije algoritma, koji nisu korišteni u našem projektu iz razloga što algoritmu dodatno otežavaju sam postupak učenja, stoga ćemo ih samo pobrojati u nastavku.

Dodatni parametri treniranja konvolucijske neuralne mreže su:

-rotiranje nasumične slike iz skupa za treniranje za određeni broj stupnjeva

-zumiranje nasumične slike iz skupa za treniranje za određeni postotak

-horizontalno pomicanje nasumične slike iz skupa za treniranje za određeni postotak

-vertikalno pomicanje nasumične slike iz skupa za treniranje za određeni postotak

-horizontalno zrcaljenje nasumične slike iz skupa za treniranje za određeni postotak

-vertikalno zrcaljenje nasumične slike iz skupa za treniranje za određeni postotak

U određenim svrhama upravo ovi dodatni parametri mogu biti presudni u ispravnoj klasifikaciji primjera, što nije slučaj za našu svrhu.

Konačna točnost algoritma iznosi **77.54%** što predstavlja gotovo idealan balans između sigurnosti zaključivanja pripadnosti primjera određenoj klasi i fleksibilnosti neuralne mreže odnosno ispravnog klasificiranja ulaznih primjera koji se ne nalaze unutar skupa za treniranje. Vrijeme treniranja je iznosilo oko **120 minuta**.

## SEGMENTACIJA NIZA ZNAKOVA REGISTARSKE OZNAKE

Kako bismo prepoznali određenu znakovnu riječ potrebno ju je raščlaniti odnosno segmentirati na slova te pomoću kreirane konvoluciske neuralne mreže određenim redoslijedom klasificiati ispravno svaki znak. Segmentaciju riječi u znakove vršit ćemo pomoću „Opencv“ modula za prepoznavanje bridova. Budući da ja svako slovo crne boje, a pozadina bijele boje vrlo je jednostavno odrediti bridove prisutne na slici. Kako bismo smanjili utjecaj šuma potrebno je pred procesirati sliku. Metode koje koristimo za pred procesiranje prikazane su u narednim slikama.

U nastavku je prikazana originalna slika.



Slika 2.6 Prikaz originalne slike

Oiginalnu sliku koja se satoji od tri kanala boje (RGB,HSV) potrebno je pretvoriti u jednokanalnu zbog pojednostvavljivanja postupka izvlačenja bitnih karakteristika iz slike.



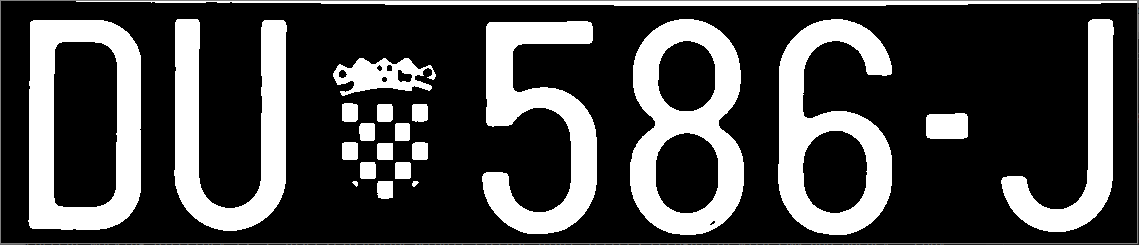
Slika 2.7 Prikaz jedno-kanalne slike „Grayscale“

Kako bismo smanjili utjecaj šuma jedno-kanalne slike koristimo učinak „Blur“ koji usrednjuje šumove, a kao rezultat se javlja to da sitni šumovi poprimaju u većoj ili manjoj mjeri boju okoline.



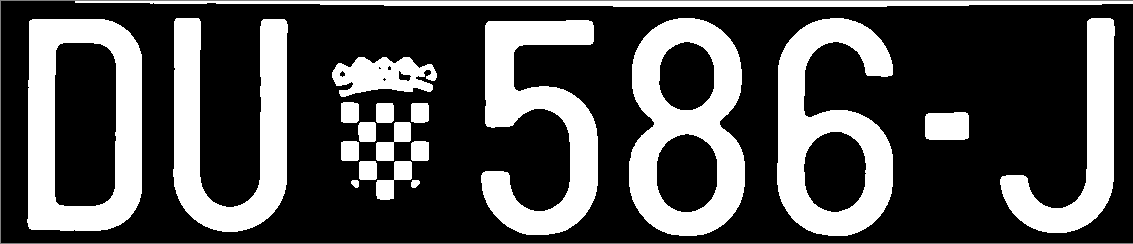
Slika 2.8 Prikaz slike nakon primjenjenog efekta“Gaussian Blur“

Budući da naš algoritam sliku segmentira na osnovu rubova potrebno rubove maksimalno naglasiti odnosno koristimo učinak „Thereshod“ te sve svodimo da samo dvije bolje pomoću učinka „Binary“.



Slika 2.9 Prikaz slike binarnih vrijednosti

Tako procesiranoj slici još jednom uklanjamo šumove pomoću učinka „Dilatation“ te je slika spremna za segmentaciju.



Slika 2.10 Prikaz slike nakon primijenjenog efekta“Dilatation“

Budući da se unutar registarske oznake nalazi i grb republike Hrvatske kao i crtica koji nam nisu potrebni za prepoznavanje, u samom algoritmu definiramo dimenzije rubova koje želimo segmentirati na način da definiramo dimenzije rubova koje želimo izvući odnosno samo one rubove čije su dimenzije veće od 70% visine ulazne slike. Konačni rezultat segmentiranja prikazan je na sljedećoj slici.



Slika 2.11 Prikaz segmenata originalne slike

# TEST

Kada je algoritam bio na 60% točnosti vrlo je dobro vršio klasifikaciju. Problemi su se pojavljivali prilikom klasifikacije sličnih znakova odnosno umjesto broja nula predviđano se slovo „O“, umjesto broja 5 predviđano je slovo „S“, umjesto broja 3 predviđano je slovo B. Dodatnim utreniravanjem algoritma na točnost klasifikacije od 77.54% navedeni problemi klasifikacije su uvelike otklonjeni, još uvijek se povremeno pojavljuju određeni problemi kod prepoznavanja određenih brojeva koji su posljedica nedovoljno reprezentativne baze podataka.

Kako najveći problem algoritma pojavljuje se segmentacija znakova registarske oznake koja sadrži puno šuma te dolazi do detekcije pogrešnog broja znakova. Ovaj problem se može uspješno riješiti dodatnim podešavanjima parametara predobrade slike što iziskuje određeno vrijeme istraživanja. Slike registarskih oznaka koje sadržavaju manju količinu šuma algoritam idealno segmentira.

# ZAKLJUČAK

Priprema reprezentativne i mnogobrojne baze podataka kao i shvaćanje svrhe različitih operacija koji se provode nad bazom podataka u svrhu proizvodnje kvalitetnog algoritma koji će uspješno klasificirati željene objekte vrlo često predstavlja najzahtjevniji dio postupka realizacije algoritma.

Povećanjem broja paralelnih slika koje šaljemo na ulaz smanjujemo konačnu točnost algoritma ali smanjujemo i vrijeme treniranja algoritma. Povećanjem epoha treniranja povećavamo točnost algoritma do određene mjere s tim da nakon određenog broja epoha može dolaziti do smanjenja konačne točnosti algoritma. Dodatni parametri za treniranje mreže usporavaju sami proces treniranja te znatno smanjuju konačnu točnost algoritma, stoga ih nismo koristili u našem projektu. Cilj je postići točnost u rasponu od 70-80% kako bismo s dovoljnom sigurnošću klasificirali određene primjere ali istovremeno očuvali fleksibilnost mreže odnosno prepoznavanje određenih primjera koji odstupaju od primjera iz skupa za treniranje. Najveći problem predstavljaju šumovi s ulazne slike koji otežavaju segmentaciju, a samim tim i prepoznavanje niza znakova koji se nalaze na registarskoj oznaci.

Bez kvalitetne baze podataka ne možemo kreirati kvalitetan algoritam.

Jadan od ciljeva seminarskog rada je razumijevanje cjelokupnog koncepta prepoznavanja određenih oblika pomoću konvolucijske neuralne mreže. Također jedan od ciljeva je izraditi vrlo točan algoritam, odnosno da veliku većinu primjera pravilno klasificira. Kako bi algoritam ispravno klasificirao gotovo sve primjere, potrebno je izmjenjivati određene dijelove algoritma odnosno „poigravati“ se određenim dijelovima faza izrade algoritma što u konačnici rezultira vrlo malim napretkom u shvaćanju funkcioniranja konvolucijskih neuralnih mreža i vrlo velikom potrošnjom vremena, što svakako nije cilj ovog projekta.

# LITERATURA

[1] Sasha Sumit: „A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way“, s interneta,<https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53> , 30. kolovoza 2020.

[2]  [Hossam H. Sultan](https://www.researchgate.net/profile/Hossam_H_Sultan):“ReLU activation function“, s interneta, <https://www.researchgate.net/figure/ReLU-activation-function_fig7_333411007> , 30. kolovoza 2020.

[3] datahacker.rs:“ #004 CNN Padding“, s interneta, <http://datahacker.rs/what-is-padding-cnn/> , 30. kolovoza 2020.

[4] Github:“ Convolutional Neural Networks (CNNs / ConvNets)“,s interneta, <https://cs231n.github.io/convolutional-networks/> , 30. kolovoza 2020.