Klasifikacija fonta datoteke

Zubčić, Ivan

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I

INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Stručni studij Računarstvo

KLASIFIKACIJA FONTA DATOTEKE

Završni rad

Ivan Zubčić

Osijek, 2023.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

Obrazac Z1S: Obrazac za imenovanje Povjerenstva za završni ispit na preddiplomskom stručnom studiju

Osijek, 20.09.2023.

Odboru za završne i diplomske ispite

Imenovanje Povjerenstva za završni ispit na preddiplomskom stručnom studiju

Ime i prezime Pristupnika:	Ivan Zubčić
Studij, smjer:	Stručni prijediplomski studij Računarstvo
Mat. br. Pristupnika, godina upisa:	AI 4643, 27.07.2017.
OIB Pristupnika:	32404983333
Mentor:	Marina Peko, dipl. ing.
Sumentor:	,
Sumentor iz tvrtke:	
Predsjednik Povjerenstva:	mr. sc. Željko Štanfel
Član Povjerenstva 1:	Marina Peko, dipl. ing.
Član Povjerenstva 2:	dr. sc. Ivana Hartmann-Tolić
Naslov završnog rada:	Klasifikacija fonta datoteke
Znanstvena grana završnog rada:	**Programsko inženjerstvo (zn. polje računarstvo)**
Zadatak završnog rada	Cilj je istrenirati model koji prepoznaje slova. Prvi dio zadatka je iščitati font filoeve(ttf, koristimo fontove kao različite načine na koji je moguće ispisati slovo/znak). Nakon toga treba renderirati svako slovo/znak(ASCII) tako da imamo veći broj uzoraka. I na kraju istrenirati model za prepoznavanje znakova.
Prijedlog ocjene pismenog dijela ispita (završnog rada):	Izvrstan (5)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 3 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 3 bod/boda Jasnoća pismenog izražavanja: 3 bod/boda Razina samostalnosti: 3 razina
Datum prijedloga ocjene od strane mentora:	20.09.2023.

Potvrda mentora o predaji konačne verzije rada:	*Mentor elektronički potpisao predaju konačne verzije.*
	Datum:

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA
I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

IZJAVA O ORIGINALNOSTI RADA

Osijek, 15.10.2023.

Ime i prezime studenta:	Ivan Zubčić
Studij:	Stručni prijediplomski studij Računarstvo
Mat. br. studenta, godina upisa:	AI 4643, 27.07.2017.
Turnitin podudaranje [%]:	3

Ovom izjavom izjavljujem da je rad pod nazivom**: Klasifikacija fonta datoteke**

izrađen pod vodstvom mentora Marina Peko, dipl. ing.

i sumentora ,

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

IZJAVA

o odobrenju za pohranu i objavu ocjenskog rada

kojom ja __Ivan Zubčić_____, OIB: __32404983333_____, student/ica Fakulteta elektrotehnike, računarstva i informacijskih tehnologija Osijek na studiju _Stručni prijediplomski studij Računarstvo_, kao autor/ica ocjenskog rada pod naslovom: _Klasifikacija fonta datoteke**,**

dajem odobrenje da se, bez naknade, trajno pohrani moj ocjenski rad u javno dostupnom digitalnom repozitoriju ustanove Fakulteta elektrotehnike, računarstva i informacijskih tehnologija Osijek i Sveučilišta te u javnoj internetskoj bazi radova Nacionalne i sveučilišne knjižnice u Zagrebu, sukladno obvezi iz odredbe članka 83. stavka 11. *Zakona o znanstvenoj djelatnosti i visokom obrazovanju* (NN 123/03, 198/03, 105/04, 174/04, 02/07, 46/07, 45/09, 63/11, 94/13, 139/13, 101/14, 60/15).
Potvrđujem da je za pohranu dostavljena završna verzija obranjenog i dovršenog ocjenskog rada. Ovom izjavom, kao autor/ica ocjenskog rada dajem odobrenje i da se moj ocjenski rad, bez naknade, trajno javno objavi i besplatno učini dostupnim:

a) široj javnosti
b) studentima/icama i djelatnicima/ama ustanove
c) široj javnosti, ali nakon proteka 6 / 12 / 24 mjeseci (zaokružite odgovarajući broj mjeseci).

**U slučaju potrebe dodatnog ograničavanja pristupa Vašem ocjenskom radu, podnosi se obrazloženi zahtjev nadležnom tijelu Ustanove.*

__Osijek, 15.10.2023._____ _____

(mjesto i datum) (vlastoručni potpis studenta/ice)

SADRŽAJ

1. UVOD

Strojno učenje je sastavni dio područja umjetne inteligencije. To polje koristi metode učenja, algoritme i statističke modele kako bi omogućilo računalima da obavljaju specifične zadatke bez potrebe za izravnom ljudskom intervencijom. Dok umjetna inteligencija omogućava računalima da izvršavaju zadatke na način sličan ljudima, strojno učenje koristi velike količine podataka za generiranje predikcija putem statističkih algoritama. Idealno, ti modeli mogu nadmašiti i same zadatke za koje su originalno kreirani.

Modeli temeljeni na strojnom učenju su efikasni u prepoznavanju uzoraka i već se koriste u svakodnevnom životu. Njihova primjena je široka; mnoga softverska rješenja koriste strojno učenje u kombinaciji s računalnim vidom, omogućujući im da putem kamera u stvarnom vremenu analiziraju i interpretiraju podatke.

Podaci korišteni za treniranje ovih modela mogu biti raznoliki. Međutim, za efikasnu pripremu i upotrebu tih podataka potrebno je imati duboko razumijevanje kako ih najbolje iskoristiti. U tom kontekstu, većina podataka se može klasificirati u jednu od četiri osnovne kategorije:

1. Numerički - podatci koji su isključivo brojčane prirode.
2. Kategorički podatci - predstavljaju karakteristike
3. Vremenski podatci - podatci prikupljani određenom frekvencijom kroz definirani vremenski rok.
4. Tekstualni podatci - jednostavno, tekst.

Oko nas se nalazi ogromna količina podataka, no njihovo prikupljanje i klasifikacija ostaju problem. Ipak postoje pojedine baze koje je moguće koristiti (Google, Microsoft, Amazon).

Korištenje fontova je uobičajena praksa u današnjem modernom svijetu. Fontovi se široko koriste u web dizajnu, reklamnim kampanjama, tiskanim materijalima i drugim medijima. Svaki font donosi svoj specifični stil i dizajn kako bi se komunicirala određena poruka. Zadatak prepoznavanja fontova može biti koristan u domenama kao što je grafički dizajn, alati temeljeni na strojnom učenju olakšavaju identifikaciju željenih fontova. Problem prepoznavanja može biti

kompliciran zbog sličnosti među fontovima ili prisutnih šumova u analiziranim podacima. Također, isti font može biti podebljan ili stiliziran na drugi način, dodatno kompliciraući zadatak.

Prije primjene strojnog učenja, ovaj problem često se rješavao metodama poput vizualnog uspoređivanja sa poznatim uzorcima fontova, što je proces koji može biti dugotrajan i težak za prilagodbu. Većina neuronskih mreža koje se koriste za identifikaciju fontova operiraju na principu nadgledanog učenja, gdje su podaci na kojima se trenira neuronska mreža prethodno označeni odgovarajućim fontom. Konvolucijske neuronske mreže, specijalizirane za detekciju uzoraka u podacima, pokazale su se kao izuzetno korisne u klasifikaciji fontova. Upotrebom sve naprednijih modela neuronskih mreža u budućnosti, očekuje se još preciznija i efikasnija identifikacija željenih fontova.

1.1 Zadatak završnog rad

Zadatak ovog rada fokusira se na razvoj softverske aplikacije koja će koristiti ulazne datoteke u TrueType Font (kratica: TTF) formatu kako bi generirala slike s tekstualnim podacima u različitim fontovima. Ovi generirani podaci potom će biti klasificirani, što je nužno za drugi dio ovog zadatka. Nakon uspostave pripremljenog podatkovnog seta, dobiveni set implementirat će se u jednostavnu neuronsku mrežu s ciljem prepoznavanja fontova na slikovnim datotekama.

2. PREGLED TRENUTNIH RIJEŠENJA

Prepoznavanje fontova predmet je interesa u području računalnog vida i strojnog učenja već neko vrijeme. Primarni cilj je identificirati ili klasificirati fontove iz dane slike teksta. Za rješavanje ovog problema korištene su različite metode, u rasponu od tradicionalnih algoritama strojnog učenja do naprednijih arhitektura neuronskih mreža.

2.1 Tradicionalne metode

Raniji pristupi često su se za klasifikaciju oslanjali na tehnike izdvajanja značajki, kao što su histogram usmjerenih gradijenata[1] (engl. Histogram of Oriented Gradients), metoda potpornih vektora[2] (engl. Support vector machines) ili metoda k-najbližih sujeda[3]. Histogram usmjerenih gradijenata je tehnika koja dijeli slike na manje dijelove, te proučava promijene na tim dijelovima. Stvara histogram promijene te ga uspoređuje sa već poznatim histogramima koje već poznaje. Metoda potpornih vektora pokušava razdvojiti podatke po kategorijama, razdvaja ih te pomoću vektora koji su najbliži granici kategorija pokušava izdovjiti podatke. Metoda k-najbližih susjeda koristi udaljenost podataka od danog primjera kako bi odlučila kojoj klasi podatak pripada. Ove metode su, iako u određenoj mjeri učinkovite, ograničene svojom nemogućnošću rukovanja složenim varijacijama fontova i stilova.

2.2 Korištenje strojnog učenja za primjenu identifikacije fonta

S pojavom strojnog učenja, neuronske mreže s povratnom vezom[3] su pokazale obećavajuće rezultate u području prepoznavanja fontova. Neuronske mreže s povratnom vezom isprva su korištene zbog svoje sposobnosti rukovanja sekvencijalnim podacima, što ih je činilo prikladnima za tekstualne zadatke. Međutim, neuronske mreže s povratnom vezom su često zahtjevne nad hardwareom i manje učinkovite u hvatanju detalja podataka. Iduća rješenja za zadatak prepoznavanja fontova očituje se u konvolucijskim neuronskim mrežama (eng. *Convolutional Neural Networks*). Konvolucijske neuronske mreže posebno su prikladne za zadatke temeljene na slikama jer mogu automatski naučiti bitne značajke iz podataka. Takve mreže vrsta su umjetnih

neuronskih mreža posebno dizajnirana za obradu slika. One su ključne u mnogim aplikacijama računalnog vida kao što su prepoznavanje slika, detekcija objekata i analiza videozapisa. Konvolucijske neuronske mreže također su korisne u drugim vrstama podataka koji imaju prostornu strukturu, kao što su zvučni zapisi i vremenske serije. Prednosti takvih mreža su:

- Visoka točnost u obradi slika
- Automatsko učenje značajki, što smanjuje potrebu za ručnim inženjeringom značajki

No, konvolucijske neuronske mreže također zahtijevaju veliku količinu podataka za treniranje, te su računalno intenzivne, posebno za velike slike.

Dosadašnji doseg rješenja za klasifikaciju fonta su napredni modeli dubokog učenja koji imaju milijune parametara, te koriste kompleksne algoritme i arhitekture neuronskih mreža. Neki od najupečatljivijih modela su:

- HENet (kratica, engl. *Hide and Enhance Network*) [4]
- DeepFont[5]

HENet model bavi se problemom klasificiranja sličnih fontova. Predloženo rješenje prepoznavanja fontova koristeći HENet ne ovisi o informacijama rastavljanih znakova ili prepoznavanju znakova. Model ne zahtjeva nikakve dodatne ulazne podatke osim slike koja se klasificira. Koristi modul nazvan HE blok koji povećava točnost modela. HE blok služi za potiskivanje istaknutih značajki, te tjera model da pronađe kompleksnije značajke. Model je treniran nad dva podatkovna seta koji sadrže veliki broj sintetičkih podataka i podataka iz stvarnoga svijeta. Explor_all koji sadrži tisuće različitih klasa fontova, koristi se za evaluaciju točnosti modela na znakovnoj razini i Adobe_VFR: Set podataka na razini riječi koji se koristi za evaluaciju performansi modela u prepoznavanju fontova u kontekstu cijelih riječi. Model ima točnost prepoznavanja od 98% nad sintetičkim podatcima, dok na podatcima iz stvarnoga svijeta točnost naglo otpada te ona iznosi 47%.

DeepFont model za prepoznavanje fontova temeljen je na konvolucijskim neuronskim mrežama. Sustav je osmišljen kako bi automatski identificirao i predložio slične fontove sa slike ili fotografije. Kao i HENet ovaj model ne zahtjeva rastavljene znakove i kontekst teksta koji klasificira. Za treniranje modela koristio se podatkovni set AdobeVFR koji sadrži označene i neoznačene slikovne podatke. Model koristi tehnike prilagođavanja podataka kako bi smanjio razlike između sintetičkih i

podataka iz stvarnoga svijeta. Točnost ovoga modela prelazi vrijednost od 80%. Ovaj model također koristi kompresiju za smanjivanje veličine modela bez gubitka performansi, no unatoč tome njegova veličina čini ga nepogodnim za korištenje u programskim rješenjima.

Oba prethodno navedena modela zahtijevaju veliku kompleksnost mreže, veliki broj podataka i zahtjev za računalni hardware te nisu pogodni za korištenje u svakodnevnoj primjeni.

3. ALFANUMERIČKI GENERATOR

Jedan od primarnih izazova u procesu treniranja modela strojnog učenja jest prikupljanje podataka. Ovaj proces može biti resursno intenzivan i vremenski zahtjevan. U kontekstu modela koji zahtijevaju grafičke podatke, izazovi se mogu pojaviti u pogledu pristupa podacima (npr., autorska prava), kvalitete (npr., nejasne slike), točnosti (npr., netočna klasifikacija) i cjelovitosti (npr., nepotpuni prikazi objekata).

3.1 Kreiranje podataka

Da bismo olakšali proces prikupljanja podataka, planiramo generirati vlastite podatke koji uključuju objekte relevantne za prepoznavanje.

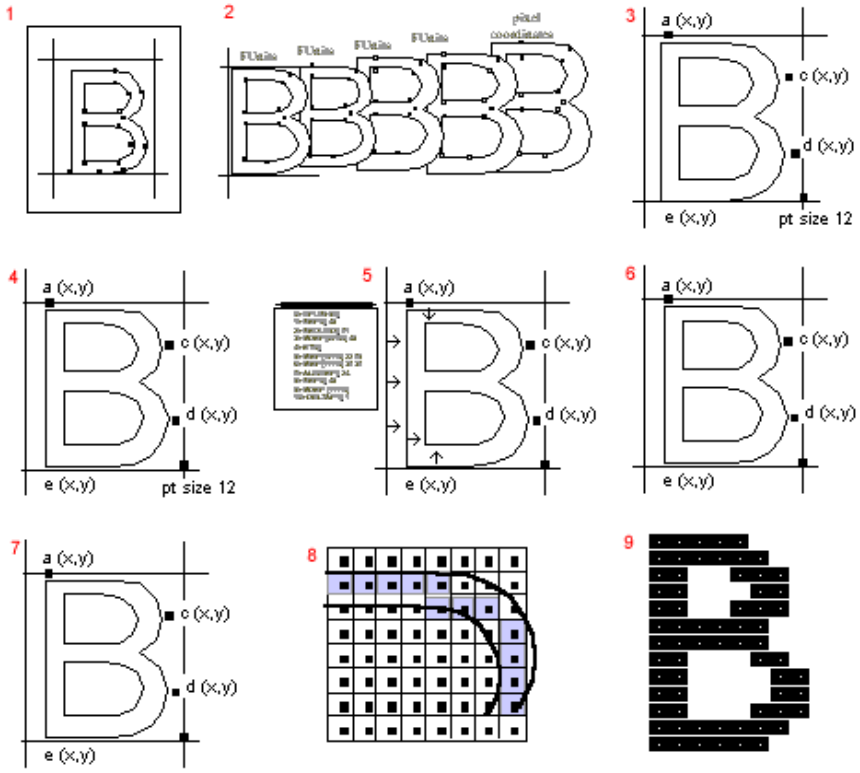
3.2 TTF format

TTF[6] ili **TrueType font format** razvila je kompanija Apple tijekom 1980-ih godina i postao je dominantan format za fontove na Windows i macOS operativnim sustavima. Konture znakova (engl. *Glyph,* vizualna reprezentacija simbola) u TrueType fontu matematički su definirane i sastoje se od pravocrtnih segmenata i Bézierovih krivulja. Osim toga, moguće je uključiti dodatne informacije (engl. *Hints* – metoda određivanja najvažnijih podataka za prikazivanje) u datoteku. Zbog svoje vektorske osnove, TrueType fontovi omogućuju skalabilnost bez gubitka kvalitete. Oni su kompatibilni s digitalizacijom i mogu se konvertirati iz drugih formata. Tehnologija TTF datoteka sastoji se od dva osnovna segmenta:

- TrueType Rasterizer
- TrueType Fonts

Rasterizer je softverski modul integriran u Windows i macOS operativne sustave koji sakuplja informacije o veličini, boji, orijentaciji i lokaciji TrueType fontova, konvertirajući te podatke u bitmap format radi kompatibilnosti s grafičkim komponentama. Slika *Sl. 3.1.* Prikazuje proces prikazivanja znaka. Gdje se obrisu znaka mijenja veličina, te se obris opisuje koordinatama te dolazi do odluke koji će se pikseli paliti za prikaz znaka. Piksel (engl. *Pixel* – skraćenica za element slike)

predstavlja najmanji blok koji čini sliku ekrana. Obično se sastoji od tri pod-piksela koji predstavljaju tri osnovne boje: Crvenu, zelenu i plavu.



Sl 3.1. Rasteriziranje TrueType datoteke. Slika preuzeta iz [1]

3.3 Python skripta

Da bismo iskoristili prednosti TrueType formata za generiranje podatkovnog seta, koristili smo programski jezik Python. Osim jednostavnosti kodiranja, Python nudi i širok spektar alatki za generiranje grafičkih podataka. Za razvoj algoritama generiranja podataka u ovom istraživanju koristili smo biblioteku Pillow. Pillow, koji se temelji na PIL-u (engl. *Python Imaging Library*), omogućuje Pythonu obradu slika i podržava različite datotečne formate. Za dodatno uređivanje i manipulaciju slika koristili smo OpenCV i NumPy.

Skripta prvo provjerava valjanost datoteke korištene za renderiranje. Tijekom otvaranja TTF datoteke, skripta provjerava jesu li znakovi u skladu s ASCII tablicom. Ako je datoteka valjana, koristi se kao zadani font za renderiranje alfanumeričkih znakova. Skripta iterira kroz ASCII tablicu i stvara direktorije za svaki znak na temelju odabranih parametara. Kako je broj generiranih slika ograničen brojem TTF datoteka, koristi se mogućnost augmentacije slika (Objašnjeno u poglavlju 2.4).

Generiranje slika se odvija u tri koraka. Prvo, koristeći Pillow, kreira se pozadina u RGB formatu sa prethodno definiranim dimenzijama slike. U ovom slučaju, dimenzije su 224x28 piksela. Drugi korak čini "crtanje" teksta na pozadinu odabranim fontom. Treći i posljednji korak kreiranja podataka je njihovo pretvaranje u red (engl. *array*) i spremanje u csv (engl. *comma-separated values* -vrijednosti razdvojene zarezom) datoteku. Csv datoteka sastoji se od prvog stupca koji definira etiketu (engl.*label*) slikovne datoteke. Ostatak stupaca označava redni piksel slikovne datoteke. Kako imamo prethodno definiranu veličinu slikovne datoteke. Matricu A možemo prikazati na način:

$$A[x, y] = [(x + 1) * y] = [(224 + 1) * 28] = 6272 + 28 \qquad (3-3)$$

Gdje x i y predstavljaju dimenzije slike, na os x dodajemo još jedan element koji predstavlja stupac etikete. Etiketa se nalazi samo u jednoj ćeliji, no prilikom pisanja ostataka elemenata kreiraju se i prazne ćelije. Dio koda prikazan na slici *Sl. 3.2.* predstavlja prethodno

```
for c in range(33, 127, 1):  # range is 33 to 127 for all chars

    # Char we want to draw on image,
    drawnChar = chr(c)
    # Saving on conversions
    stringOfChar = str(c)

    # Checks for the char in .ttf file, skips iteration if there is no current char. This is used as we can
    # create some chars from font files even tho not all of them are defined in .ttf
    if char_in_font(drawnChar, checkFont) == False:
        print('There is no char('+drawnChar +
              ') defined in '+fileNameJoin)
        continue

    # Make an Image x*x size with transparent background, alpha is 0. Note that changes for alpha here wont affect
    # cropped images as they are transformed in imageCropp() function
    image = Image.new(
        "RGB", (224, 28), (255, 255, 255))
    draw = ImageDraw.Draw(image)

    # Adds the text over image, with indexed font. And saves the image under the name of font + ASCII value of char
    draw.text((randomXDraw, 0), drawnChar,
              font=font, fill=color)

    # NameOfImage = str(str(filename[:-4]) + stringOfChar + strRandomFontSize +'.png') #Needs to have '.png' to work
    # Needs to have '.png' to work
    NameOfImage = str(
        str(filename) + str(num_of_image) + '.png')
    num_of_image += 1

    print(NameOfImage)

    # All of given variables from above have the same name as variables in imageCropp() function

    cv_image = cv.cvtColor(
        np.array(image), cv.COLOR_RGB2BGR)
    cv_image_gs = cv.cvtColor(cv_image, cv.COLOR_BGR2GRAY)
    cv_image_gs = ~cv_image_gs
    #cv.imwrite("/home/zule/anaconda3/envs/AlpNum/AlpNums/Renders/" + NameOfImage, cv_image_gs)

    image_array = np.asarray(cv_image_gs)

    oneDimensionArray = image_array.flatten()

    row_info = [numerical_label]
    for value in oneDimensionArray:
        row_info.append(value)
```

Sl. 3.2. Dio skripte za kreiranje slike s znakom

Osim generiranja individualnih znakova, skripta također podržava stvaranje slika s tekstualnim podacima. Postupak je jednak kao i za znakove. Skripta čita tekstualne datoteke iz određenog direktorija, jednu po jednu. Tekstualni sadržaj sprema se kao niz znakova, iz kojeg se nasumično izabire segment varijabilne duljine. Nakon toga, slika se generira, tekst se dodaje na nju, i slika se zatim sprema u CSV format, prikaz podataka u ovom format može se vidjeti na slici *Sl. 2.3.* Ako je to omogućeno, generirana slika također se može sačuvati u .png formatu.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	label	pixel0	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	pixel9	pixel10	pixel11	pixel12	pixel13	pixel14
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
17	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
18	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
20	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
21	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
22	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
23	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
24	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
25	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
26	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
27	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
28	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
29	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
30	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
31	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
32	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
33	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
34	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Sl. 3.3. Izgled slikovnih podataka u numeričkom obliku

Sve korištene fontove skripta također sprema u zasebnu .csv datoteku, prikazano na slici *Sl. 3.4*. Datoteka sadrži dva stupca. Numeričku etiketu korištenog fonta (Numerički prikaz za neuronsku mrežu) i tekstualno ime korištenog fonta.

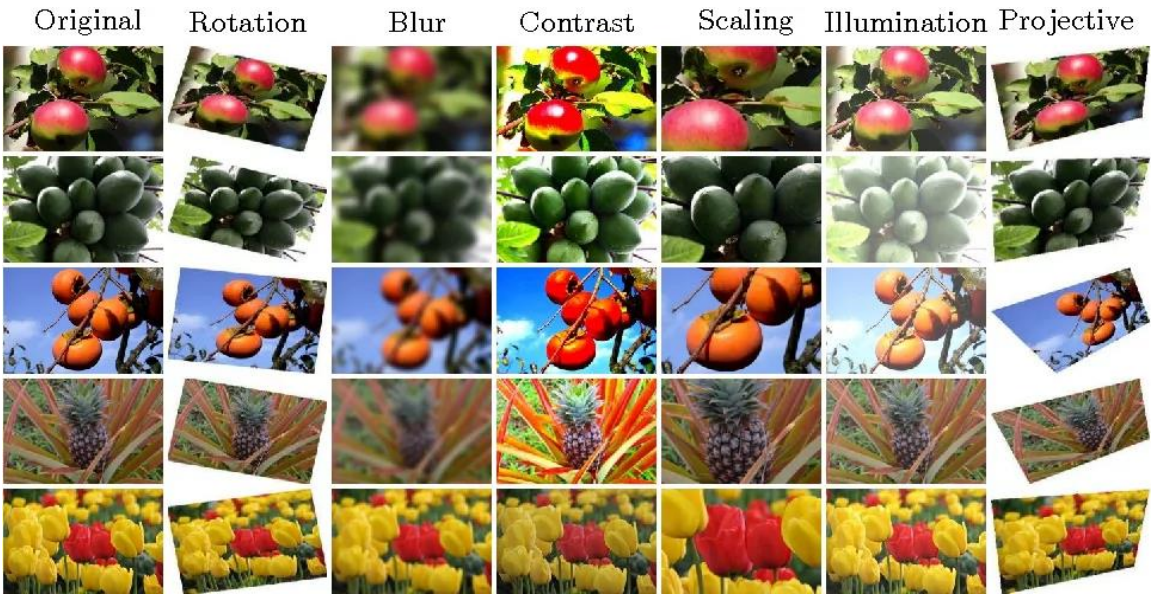
	A	B
1	consolaz	0
2	segoeprb	1
3	ntailub	2
4	comic	3
5	ebrima	4
6	constan	5
7	Candara	6
8	arial	7
9	calibrii	8
10	mvboli	9

Sl. 3.4. Izgled datoteke s etiketama

3.4 Augmentacija

Performanse većine algoritama strojnog i dubokog učenja su uvelike ovisne o kvaliteti, količini i relevantnosti podataka koji se koriste za trening. Prikupljanje visokokvalitetnih podataka često je vremenski zahtjevan i naporan proces. U situacijama gdje postojeći podaci nisu dostatni za efikasno treniranje modela, augmentacija podataka postaje iznimno korisna tehnika za poboljšanje performansi i ishoda modela.

Postupak augmentacije[6] podataka uključuje primjenu različitih tehnika izmjene na postojećem podatkovnom setu. U kontekstu obrade slika, te tehnike mogu obuhvaćati ponovno skaliranje, nasumične rotacije, translacije slika, vertikalna i horizontalna zrcaljenja, obrezivanje, zumiranje, zatamnjenje i promjene u boji slike. Iako može izgledati da se, na primjer, rotirana ili zumirana slika ne smatra "novom" u konvencionalnom smislu, u kontekstu konvolucijskih neuronskih mreža ovo nije slučaj. Tri varijante slike broja "3" će se tretirati kao tri različita podatka. Ako je model treniran koristeći podatkovni skup s manjkavim ili nereprezentativnim primjerima (npr. sve instance broja "6" su u uspravnom položaju), to može dovesti do izazova u prepoznavanju tog broja kada je rotiran ili promijenjen na neki drugi način, kao što je rotacija za 90 stupnjeva. Augmentirani podaci ne samo da pomažu u slučajevima gdje postoji deficit kvalitetnih podataka, već mogu biti od velike pomoći i kada model ima problema sa velikim i raznolikim skupom podataka. Na primjer, ako model konzistentno prepoznaje broj "9" kao broj "6" na temelju originalnog podatkovnog seta, augmentacija može pomoći u rješavanju tog problema. Na slici *Sl. 3.5.* Prikazani su načini na koji se slikovni podatci mogu augmentirati.



Sl. 3.5. Augmentacija slikovnih podataka. Slika preuzeta iz [6]

Napisana python skripta može raditi augmentaciju ponovnim-skaliranjem i translacijom. Što smanjuje mogućnost krive detekcije modela za prepoznavanje. Tako da kod kreiranja znakovnih podataka, gdje jedna slika predstavlja jedan znak. Taj znak bit će prikazan na n različitih veličina fonta i m različitih pozicija unutar pozadine. Slika *Sl. 3.6.* Prikazuje znak "!" agumentira po položaju na slici.

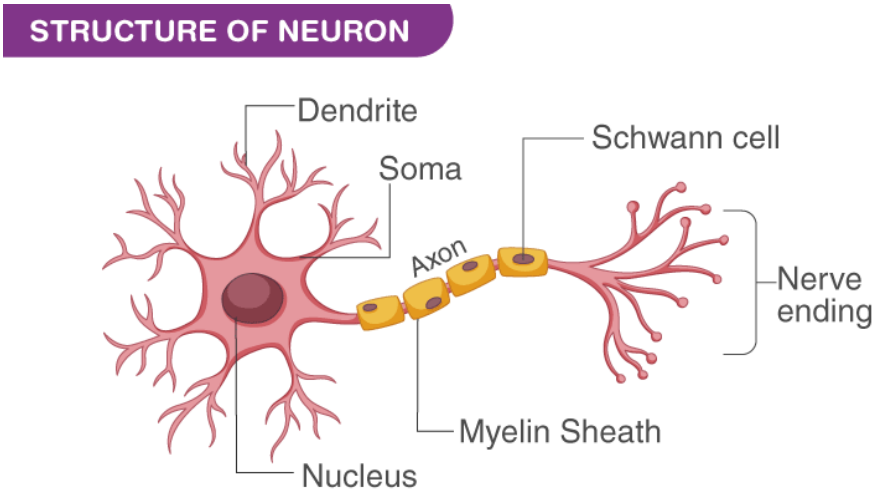
Sl. 3.6. Augmentirani sintetički podatci

4. NEURONSKE MREŽE

Neuronske mreže, također poznate pod nazivom “Umjetne neuronske mreže” (kratica: *ANNs*, engl. *Artificial neural networks.*) pripadaju grani strojnog učenja. One čine suštinu algoritama dubokog učenja.

4.1 Neuron

Stvaranje umjetnih neuronskih mreža inspirirano je građom ljudskoga mozga. Neuronske mreže[7] sastoje se od mnoštva umjetnih neurona, koji pokušavaju replicirati rad neurona nađenih u ljudskom mozgu, model neurona prikazan je na slici *Sl. 4.1*. Umjetni neuroni uzimaju dijelove prirodnih kao što su dendridi koji predstavljaju ulazne podatke i aksone koji čine podatkovni izlaz.



Sl. 4.1. Neuron ljudskog mozga. Slika preuzeta iz [6]

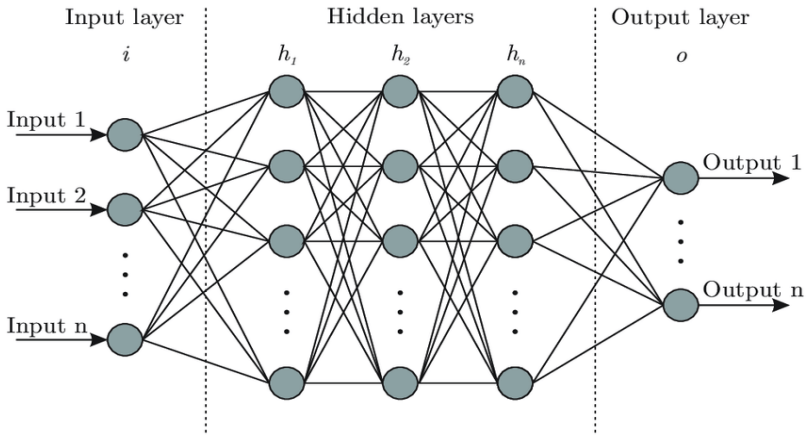
Umjetni neuron predstavlja matematičku funkciju koja uzima ulazne podatke (gdje svaki ulaz ima vrijednost težine), te izvršava nad njima aktivacijsku funkciju kako bi dobili izlazne podatke. Funkcija izgleda ovako:

$$y_k = \varphi\left(\sum_{j=0}^{m} w_{kj} * x_j\right) \qquad (4-1)$$

- y_k - umjetni neuron k
- x - ulazni podatak
- w - težina
- φ - aktivacijska funkcija

4.2 Građa neuronskih mreža

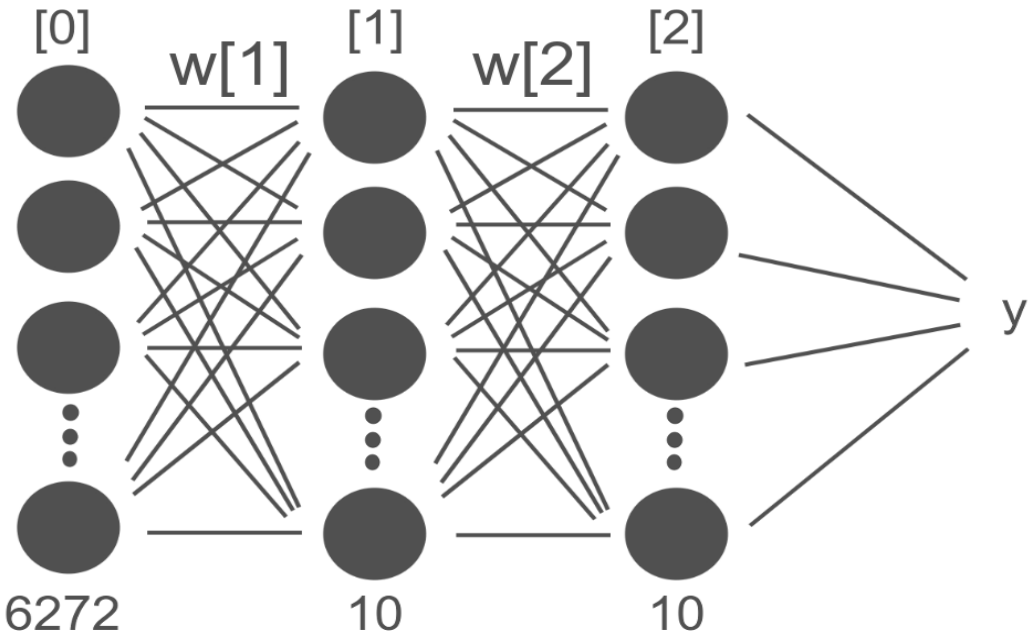
Umjetne neuronske mreže[8, 9] sastoje se od slojeva čvorova (neuroni). Slojeve klasificiramo u 3 klase: Ulazni sloj, skriveni sloj (jedan ili više) i izlazni sloj. Svaki neuron spojen je na druge neurone te ima svoju vrijednost težina i aktivacijsku funkciju. Slojevi I građa opisane mreže prikazani su na slici *Sl. 4. 2.* Ako je izlazna vrijednost nekog neurona veća od određene granične vrijednosti aktivacijske funkcije, neuron se aktivira i šalje podatke u idući sloj. Suprotno, podatci ne prolaze dalje.



Sl. 4.2. Građa neuronske mreže. Slika preuzeta iz [6]

4.3 Arhitektura mreže

Arhitektura neuronske mreže koja je izrađena u ovom istraživanju prilagođena je specifičnostima korištenog skupa podataka. Ulazni sloj mreže sadrži 6272 neurona prikazan kao prvi stupac na slici *Sl. 3.2.* Što odgovara broju piksela u svakoj slici iz našeg seta podataka. Skriveni sloj mreže obuhvaća 10 neurona, prikazanih kao drugi stupac na slici *Sl. 4.3.* Izlazni sloj također ima 10 neurona, te je prikazan kao zadnji stupac sa slike. Ulazni sloj prilagođen je tako da odgovara broju elemenata u matrici koja predstavlja svaku sliku u setu podataka. Na sličan način, izlazni sloj mreže prilagođen je broju različitih TTF datoteka koje koristimo za generiranje znakova i tekstualnih podataka. Ovo omogućava mreži da precizno klasificira i prepoznaje različite stilove pisanja i oblike koje generiraju različite TTF datoteke.



Sl. 4.3. Prikaz neuronske mreže korištene u radu

5. RAD NEURONSKE MREŽE

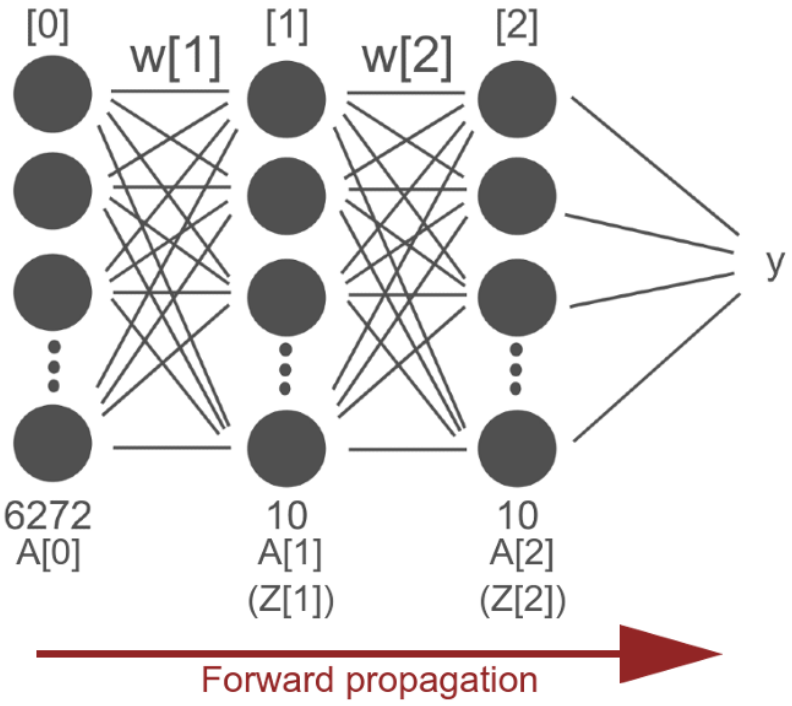
Treniranje kreirane neuronske mreže izvodit će se u tri djela: Propagiranje unaprijed, propagiranje unatrag te modifikacija parametara neuronske mreže.

5.1 Algoritam propagiranja unaprijed

Algoritam propagiranja unaprijed odnosi se na proces uzimanja podataka te njihovo procesiranje u neuronskoj mreži. U odnosu na arhitekturu mreže, smjer vidimo na slici *Sl. 5.1.* Rezultat procesa propagiranja unaprijed je vjerojatnost dobivenog fonta. Prvotno kreiramo varijablu $A^{[0]}$. Varijabla $A^{[0]}$ jednaka je ulaznom sloju.

$$A^{[0]} = X(6272 * m) \qquad (5-1)$$

- X - ulazni sloj
- m - broj podataka (slika)



Sl. 5.1. Smjer propagiranja unaprijed

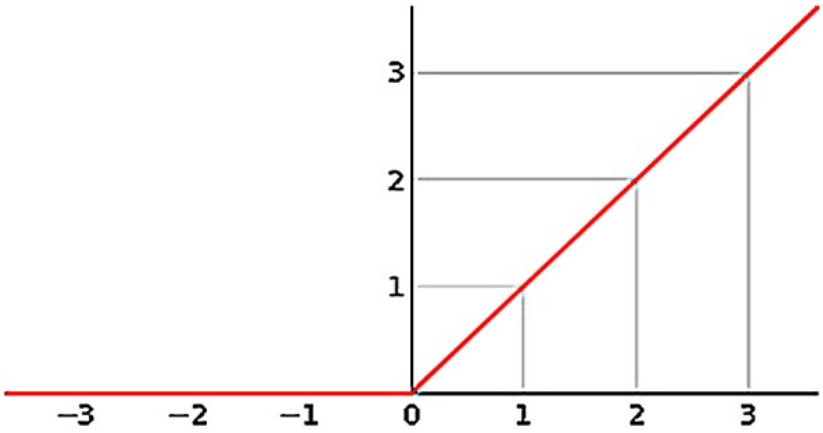
Nadalje kreiramo varijablu $Z^{[1]}$ (Z predstavlja neaktivirani sloj $A^{[1]}$) čija je vrijednost linearna kombinacija ulaznog sloja. Nju dobivamo množenjem matrice vrijednosti težina $w^{[1]}$ s početnim vrijednostima koje su u varijabli $A^{[0]}$. Tom produktu dodajemo vrijednost pristranosti (engl. *Bias).* Vrijednost pristranosti pomaže nam dobiti točna predviđanja na način da model linearne regresije ne prolazi uvijek kroz ishodište.

$$Z^{[1]} = w^{[1]} * A^{[0]} + b^{[1]} \qquad (5-2)$$

- w - matrica s vrijednostima težina
- $A^{[0]}$ - ulazni podaci
- $b^{[1]}$ – vrijednost pristranosti

Za idući korak nad neaktiviranim slojem $Z^{[1]}$ iskorištavamo aktivacijsku funkciju. U ovom slučaju to je funkcija ispravljena linearna jedinica, skraćeno ReLU (engl. *Rectified linear unit*). Funkcija radi na način da za vrijednosti koje su veće od 0, imaju istu vrijednost kao i ulazni parametar, ako su manje. Vrijednost im je 0. Slika *Sl. 5.2* prikazuje nam graf funkcije ReLu i odnos vrijednosti na grafu.

$$ReLU(x) = f(x) = (0, x) \qquad (5-3)$$



Sl. 5.2. Graf funkcije ReLU

Iskorištavanjem aktivacijske funkcije ReLU nad svakoj vrijednosti u $Z^{[1]}$ dobivamo drugi sloj, $A^{[1]}$.

Kako bi dobili vrijednosti za sloj $A^{[2]}$. Radimo sličan proces kao u prvom koraku. Kreiramo neaktivirani sloj $Z^{[2]}$. On je jednak produktu vrijednosti težina $w^{[2]}$ i aktiviranog prvog sloja $A^{[1]}$ kojem dodamo vrijednost pristranosti $b^{[2]}$.

$$Z^{[2]} = w^{[2]} * A^{[1]} + b^{[2]} \qquad (5-3)$$

Nad neaktiviranim slojem $Z^{[2]}$ izvodimo aktivacijsku funkciju zvanu softmax. Fukcija softmax omogućava nam da izlazne vrijednosti predstavimo kao vjerojatnost predviđanja fonta. Uzima vrijednosti ulaznog sloja i dijeli ih sa sumom svih vrijednosti:

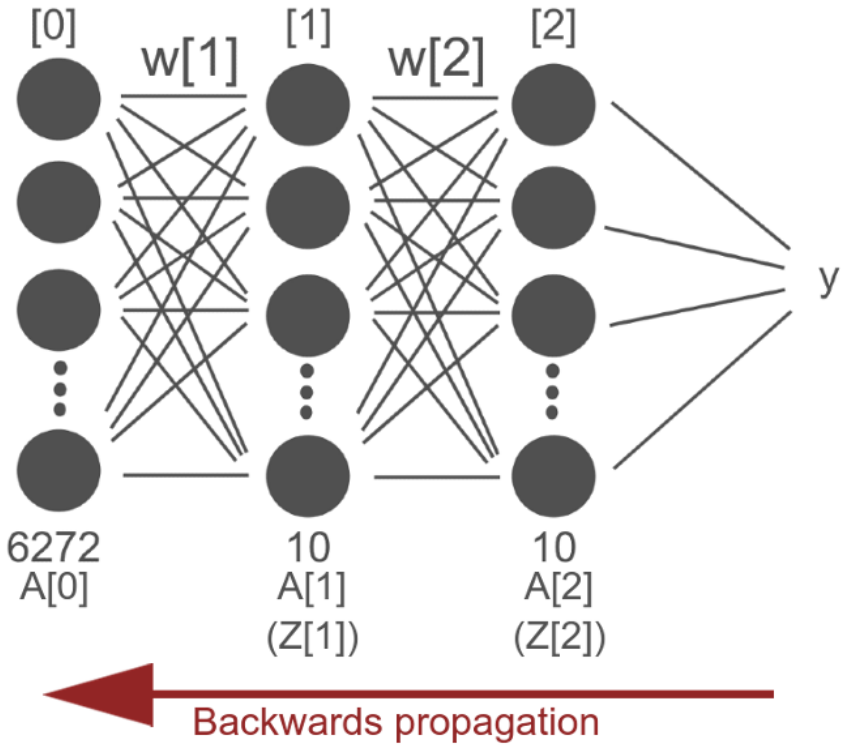
$$\sigma_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_i}} \qquad (5-4)$$

- σ - softmax
- e^{z_i} - vrijednost čvora iz ulaznog sloja
- $\sum_{j=1}^{K} e^{z_i}$ - suma svih čvorova ulaznog sloja

Primjenom aktivacijske funkcije na neaktivirani sloj $Z^{[2]}$ dobili smo sloj $A^{[2]}$, on predstavlja vjerojatnosti fonta iz predane slike.

5.2 Algoritam propagiranja unatrag

Propagiranje unatrag proces je koji radimo kako bi optimizirali vrijednosti težina i vrijednost pristranosti. Propagiranje unatrag odvija se suprotnim smjerom od propagiranja unaprijed,, te započinje izlaznim vrijednostima. Smjer propagiranja u odnosu na arhitekturu mreže vidimo na slici *Sl. 5.3.* Koristeći propagiranje unatrag dolazimo do vrijednosti koja upućuje koliko je daleka vrijednost dobivenog predviđanja od etikete datoteke, te koliko su utjecaj na pogrešku imali utezi i vrijednost pristranosti iz propagiranja unaprijed.



Sl. 5.3. Smjer propagiranja unatrag

Prvi korak je računanje pogreške zadnjeg sloja $A^{[2]}$. Od dobivenih vjerojatnosti oduzmemo njihove etikete $Y^{[E]}$. Ovdje koristimo tehniku kodiranja One hot. Tehnika se izvodi na način da se iskaže važnost etikete slike (koja će imati vrijednost 1), dok se ostale zanemaruju (poprimaju vrijednost 0). Primjer vidimo na slici *Sl. 5.4.* Gdje nam je bitna vrijednost, vrijednost druge etikete te je ona postavljena kodiranjem na “1”.

$$e_{rr}Z = \begin{bmatrix} 0.08 \\ 0.2 \\ 0.12 \\ 0.05 \\ 0.025 \\ 0.05 \\ 0.075 \\ 0.1 \\ 0.15 \\ 0.15 \end{bmatrix} - \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Sl. 5.4. Primjer matrice kodirane One hot tehnikom

Vrijednost pogreške drugog sloja $\partial Z^{[2]}$ prikazuje se na sljedeći način:

$$\partial Z^{[2]} = A^{[2]} - Y^{[E]} \qquad (5-5)$$

Vrijednost pogreške težina između $A^{[1]}$ i $A^{[2]}$ dobiva se tako da se produkt pogreške drugog sloja $\partial Z^{[2]}$ i transponirane vrijednosti sloja $A^{[1]}$ podijeli s ukupnim brojem elemenata.

$$\partial w^{[2]} = \frac{\partial Z^{[2]} * A^{[1]T}}{m} \qquad (5-6)$$

Vrijednost pristranosti između $A^{[1]}$ i $A^{[2]}$ jednaka je srednjoj vrijednosti pogreške drugog sloja $\partial Z^{[2]}$.

$$\partial b^{[2]} = \frac{\sum \partial Z^{[2]}}{m} \qquad (5-7)$$

Nakon računanja vrijednosti izlaznog sloja, iste vrijednosti računaju se za skriveni sloj. Uzima se vrijednosti pogreške drugog sloja, i množi s transponiranim utezima drugog sloja $w^{[2]}$. Također uklanjamo utjecaja aktivacijske funkcije prvog sloja njenim deriviranjem.

$$\partial Z^{[1]} = w^{[2]T} * \partial T^{[2]} * g'\left(Z^{[1]}\right) \qquad (5-8)$$

Vrijednosti pogreške $\partial w^{[1]}$ težina i vrijednost pristranosti $\partial b^{[1]}$ između ulaznoj sloja $A^{[0]}$ i skrivenog sloja $A^{[1]}$ dobiva se kao u prethodnim koracima.

$$\partial w^{[1]} = \frac{\partial Z^{[1]} * A^{[0]T}}{m} \qquad (5-8)$$

$$\partial b^{[1]} = \frac{\sum \partial Z^{[1]}}{m} \qquad (5-9)$$

Ovim kalkulacijama dobivaju se vrijednosti koje iskazuju koliko su pojedini utezi i vrijednost pristranosti pridonijeli pogrešci predviđanja fonta.

5.3 Podešavanje mreže

Posljednji dio procesa odnosi se na osvježavanje dosadašnjih vrijednosti neuronske mreže. Dobivenim pogreškama podešavaju se vrijednosti težina I vrijednost pristranosti.

$$w^{[1]} = w^{[1]} - \alpha * \partial w^{[2]} \qquad (5-10)$$

$$b^{[1]} = b^{[1]} - \alpha * \partial b^{[2]} \qquad (5-11)$$

$$w^{[2]} = w^{[2]} - \alpha * \partial w^{[1]} \qquad (5-12)$$

$$b^{[2]} = b^{[2]} - \alpha * \partial b^{[1]} \qquad (5-13)$$

α - Stopa učenja, parametar koji određuje koliko često se osvježavaju podatci učenja

6. TRENIRANJE MODELA NEURONSKE MREŽE

U ovome segmentu rada opisan je proces treniranja prethodno objašnjenog model nad sintetičkim podatcima kreiranim skriptom za kreiranje slikovnih podataka.

6.1 Priprema podataka

U ovom radu koristimo skriptu napisanu u Pythonu za sintetiziranje baze podataka koja će poslužiti za treniranje modela. Skripta generira slike na kojima su pojedinačni znakovi prikazani u različitim fontovima i pozicijama na slici. Svaki znak augmentiran je više puta, transformacije uključuju promjenu veličine fonta i položaja na slici. Osim toga, skripta uzima tekstualne datoteke i stvara slike s nasumičnim rečenicama, također prikazanim u raznim fontovima i pozicijama. Po svakom fontu generira se n broj slika.

Treba napomenuti da broj može varirati s obzirom na to da neki znakovi možda nisu definirani u korištenim TTF datotekama.

6.2 Treniranje i rezultati

Nakon što su podatci pripremljeni, pokreće se skripta za treniranje neuronske mreže. Važno je provjeriti format trenutno napravljenih podataka. Format dobivamo pozivanjem funkcije head vidljive na slici *Sl. 6.1*, koja nam vraća zaglavlje podatkovnog seta.

```
data.head()
```

	label	pixel0	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	...
0	0	0	0	0	0	0	0	0	0	0	...
1	0	0	0	0	0	0	0	0	0	0	...
2	0	0	0	0	0	0	0	0	0	0	...
3	0	0	0	0	0	0	0	0	0	0	...
4	0	0	0	0	0	0	0	0	0	0	...

5 rows × 6273 columns

Sl. 6.1. Pregled priređenih podataka u prvih 5 redaka

Podatci su potom razvrstani slučajnim odabirom i raspodijelih u dvije grupe. Podatke korištene za treniranje, i podatke koje korištene za provjeravanje točnosti mreže. Podatci su podijeljeni u omjeru 8:2. Matrica slikovnih podataka se transponira, tako da svaki red predstavlja jednu sliku, svaki stupac predstavlja piksel. Proces treniranja pokrene se s predanim parametrima za stopu učenja α i broj iteracija učenja.

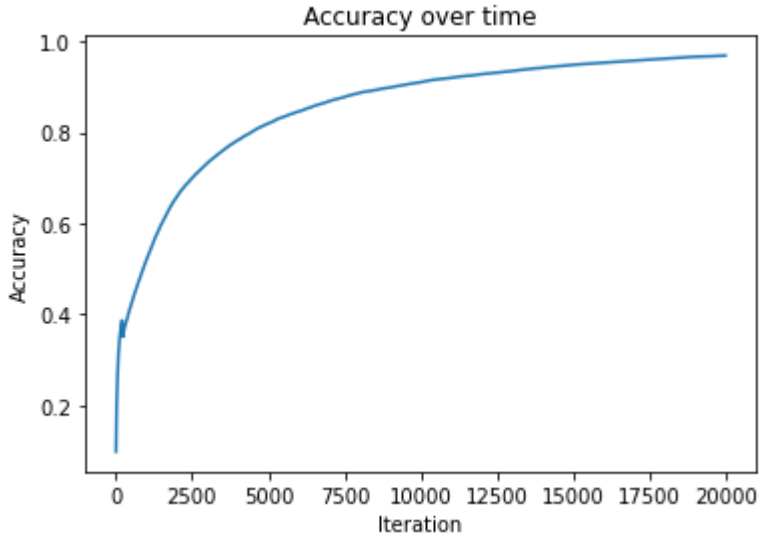
Model inicijalizira početne parametre $w^{[1]}$, $b^{[1]}$, $w^{[2]}$, $b^{[2]}$. Njima daje slučajno odabrane vrijednosti između 0 i 1.

Tijekom procesa treniranja prvo se odrađuje propagiranje unaprijed, krenuvši s ulaznim podatcima do izlaznih podataka drugog sloja neuronske mreže. Model uspoređuje predviđene podatke i stvarne podatke koji kreiraju pogrešku. Kako bi smanjili pogrešku iskorištava se proces propagiranja unatrag. Derivacija funkcije ReLU iskorištava se na prvom sloju, grešku drugog sloja propagira na prvi.

Parametri w[1], b[1], w[2], b[2] poprimaju nove vrijednosti nakon svake iteracije propagiranja unatrag. Svaki proces odrađuje se određeni broj iteracija, točnost neuronske mreže ispisuje se na konzoli.

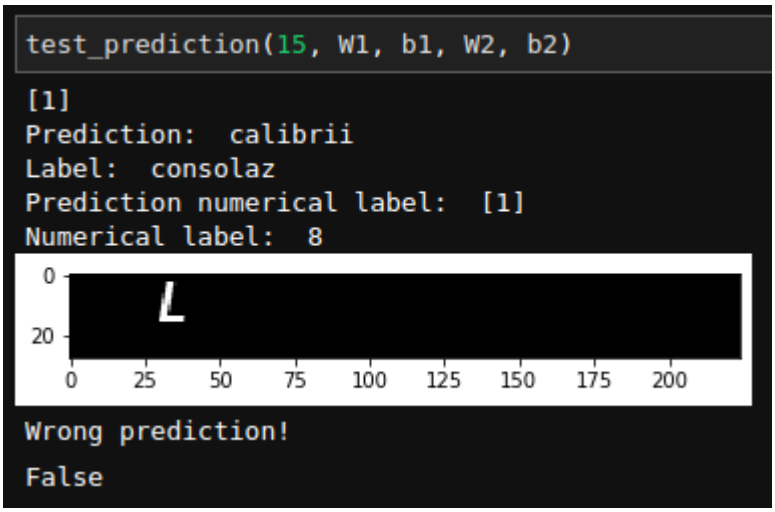
Nakon zadnje iteracije parametri se spremaju pomoću python biblioteke “Pickle” u file pod nazivom “params.pkl”. Svi parametri mogu se ponovno učitati u neuronsku mrežu kako se model ne bi morao trenirati kod svakog korištenja.

Napredak točnosti neuronske mreže grafički je prikazan na grafu. Izgled možemo pratiti na grafu *Graf 6.2.* Gdje x-os predstavlja broj iteracija učenja, y-os točnost neuronske mreže. Točnost modela varirat će o danim podatcima, njihovoj kvaliteti i količini. Točnost mreže također se provjerava na slučajno odabranim podatcima iz danog podatkovnog seta namijenjenog provjeravanju istih.



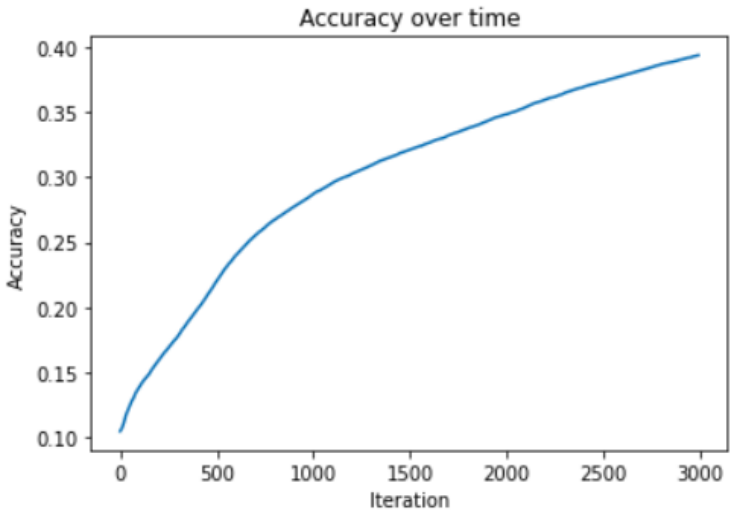
Graf 6.2. Ovisnost točnosti o iteracijama učenja

Kod kreiranja modela neuronske mreže cilj je kreirati sistem koji iz “znanja” stečenih treniranjem može točno predvidjeti rezultate iz podataka koji su prvi puta predstavljeni neuronskoj mreži. Kod treniranja posebnu pažnja treba biti usmjerena prema problemu pretreniranosti.



Sl. 6.3. Netočno predviđanje fonta nad novim podacima

Problem pretreniranosti je problem koji se manifestira kada model postigne visoku točnost na trening setu podataka, ali ima loše performanse na novim, neviđenim podacima. Na primjer, u ovom istraživanju model je u jednoj iteraciji treniranja postigao točnost od 100% na trening setu, ali točnost na validacijskom setu bila je znatno niža, jedan od primjera je na slici *Sl. 6.3* gdje nam mreža vraća krivo predviđanje na novom neviđenom podatku. Ovaj fenomen može biti rezultat modela koji "uči" šum i nebitne detalje iz trening seta, što ga čini manje generaliziranim.
Jedan od načina identifikacije pretreniranosti je praćenje grafa točnosti kroz iteracije: ako točnost na trening setu raste dok točnost na validacijskom setu stagnira ili opada, to je indikator pretreniranosti. U ovom radu, primijenili smo nekoliko tehnika kako bismo se nosili s problemom pretreniranosti. Povećali smo veličinu i raznovrsnost seta podataka putem augmentacije. Ove mjere poboljšale su modelovu sposobnost generalizacije na novim podacima. Iako je točnost modela sada zadovoljavajuća, daljnjim razvijanjem kompleksnijeg modela i kvalitetnijeg seta podataka možemo postići još bolje rezultate.
Osim pretreniranosti, do problema možemo doći i kod treniranja neuronske mreže gdje nemamo dovoljno iteracija treniranja kao što je to prikazano na grafu *Graf 6.4.* Iako mreža napreduje u učenju ne izvršimo dovoljno etapa te preciznost mreže nije dovoljna za prihvatljive rezultate.



Graf 6.4: Ovisnost točnosti o iteracijama učenja s premalim brojem iteracija

Razvijena neuronska mreža koristi pristup multi klasne klasifikacije. Kako koristi 10 različitih kategorija za klasifikaciju fontova. Svaka klasa predstavlja jedinstveni font koji je numerički predstavljen mreži kao vrijednosti od 0 do 9. Za razliku od binarnih klasifikatora koji imaju 2 outputa ova mreža ima ih 10. Koristimo balansirani set podataka, što znači da imamo isti broj svih podataka svake klase.

Rezultati treniranja neuronske mreže dobiveni na podatkovnom setu od 124000 slika. Od kojih 24800 čine validacijski set, dok je ostatak dio seta za treniranje. S parametrom stope učenja od 0.05 i odrađenih 300 000 epoha. Daju preciznost mreže od 62.70%. Ta preciznost daje generalan uvid u rad modela. Iz analize podataka, može se vidjeti da model ima problema s prepoznavanjem fonta pojedinih znakova. Kako model koristi multi klasnu klasifikaciju, valja prikazati preciznost mreže na svakom odrađenom fontu. Preciznost za svaki jedinstveni font može se prikazati pomoću matrice zabune. Matrica zabune je tablica koja detaljno prikazuje performanse klasifikacijskog modela. Prikazuje prave i lažne pozitivne i negativne rezultate za svaku klasu. Omogućuje da se vizualiziraju greške i tipove grešaka, te daje mogućnost da se odredi točnost, preciznost i opoziv mreže za svaku klasu.

- Točnost - predstavlja omjer točnih predviđanja i broj svih predviđanja.
- Preciznost - predstavlja omjer istinskih pozitivnih predviđanja i sume istinskih i lažnih predviđanja. Govori nam koliko je predviđenih podataka jedne klase zapravo dio te iste klase.
- Opoziv - predstavlja omjer točnih istinskih pozitivnih predviđanja i ukupni broj istinskih pozitivnih. Govori nam koliki je broj lažnih negativnih predviđanja modela.

Naša matrica zabune izgleda na slijedeći način *Tablica 6.5*. U prvom retku imamo predviđene klase, a u prvom stupcu točne klase.

	Prd Class 0	Prd Class 1	Prd Class 2	Prd Class 3	Prd Class 4	Prd Class 5	Prd Class 6	Prd Class 7	Prd Class 8	Prd Class 9
Act Class 0	5649	708	147	1529	858	393	266	53	91	187
Act Class 1	570	6180	4	813	2047	34	49	10	38	141
Act Class 2	197	80	5879	267	37	198	389	1167	719	1042
Act Class 3	1716	814	113	5917	634	229	144	23	130	150
Act Class 4	628	2086	19	851	5960	124	136	2	32	72
Act Class 5	386	162	146	421	194	7231	857	116	247	151
Act Class 6	300	145	304	365	182	1094	6640	180	195	344
Act Class 7	106	79	791	196	45	194	289	6780	1006	407
Act Class 8	111	63	901	242	83	264	506	1727	5360	665
Act Class 9	216	146	1114	322	81	135	446	308	629	6563

Tablica 6.5: Matrica zabune

Analizom matrice konfuzije utvrđeno je kako model pokazuje varijabilne performanse u klasifikaciji različitih klasa fontova. Na primjer, klasa 5 ima najvišu točnost prepoznavanja sa 73.10%, dok je klasa 3 najslabije prepoznata s preciznošću od samo 54.10%. Ovaj niski postotak točnosti za klasu 3 implicira da model često pogrešno klasificira podatke kao pripadnike ove klase, što dovodi do velikog broja lažnih pozitivnih predviđanja. Što se tiče klase 8, njen niski opoziv ukazuje na to da model često lažno klasificira podatke iz drugih klasa kao pripadnike klase 8, ali također propušta prepoznati velik broj podataka koji stvarno pripadaju toj klasi. To ukazuje na potrebu za dodatnim optimizacijama modela za ovu specifičnu klasu. Klase 0, 4 i 9 pokazale su dosljedne performanse, što znači da model sa sličnom efikasnošću klasificira podatke iz ovih klasa. Ovo može biti dobar temelj za daljnje istraživanje i optimizaciju, jer dosljednost u ovim klasama može pružiti uvid u što model dobro radi i kako se slični principi mogu primijeniti na slabije performirajuće klase. Sve rezultate prikazujemo tablicom *Tablica 6.6.*

Class	Precision	Recall
0	0.572	0.572
1	0.591	0.622
2	0.621	0.589
3	0.541	0.599
4	0.589	0.601
5	0.731	0.73
6	0.683	0.667
7	0.654	0.685
8	0.62	0.54
9	0.675	0.659

Tablica 6.6: Točnost i opozivna vrijednost po klasi

7. ZAKLJUČAK

U nedostatku podataka za treniranje neuronskih mreža mogu se kreirati sintetički podatci. Problem sintetičkih podataka očituje kod vrijednost pristranosti nad sintetičkim podatcima.. Nedostatak u treniranju nad tim podatcima je što ne priliče onima iz stvarnog svijeta koji dolaze s nedostatcima. No, ipak. Model treniran and sintetičkim podatcima može raditi i na onima iz stvarnoga svijeta. Cilj rada za treniranje neuronske mreže je ispunjen, te nakon njenog treniranja kreirana neuronska mreža može s velikom sigurnošću odrediti font korišten na slici. Model I njegova preciznost ograničeni su brojem fontova zbog velikih zahtjeva za računalnom opremom na podatkovnim setovima.

LITERATURA

[1] Peter Constable, Mike Jackobs, True Type fundamentals, Microsoft 12.09.2021
https://docs.microsoft.com/en-us/typography/opentype/spec/ttch01 28.06.2022

[2] scikit-learn, 1.4. Support Vector Machines
https://scikit-learn.org/stable/modules/svm.html 18.09.2023

[3] Geeksforgeeks, K-Nearest Neighbor(KNN) Algorithm 05.05.2023
https://www.geeksforgeeks.org/k-nearest-neighbours/ 19.09.2023

[4] J.Chen, S.Mu, S.Xu, Y.Ding, HENet: Forcing a Network to Think More for Font Recognition, 2021 3rd International Conference on Advanced Information Science and System (AISS 2021), Članak No.: 8, str. 1-5. Studeni 2021

[5] Z.Wang, J.Yang, H.Jin, E.Shechtman, A.Agarwala, J.Brandt, T.S.Huang, MM '15: Proceedings of the 23rd ACM international conference on Multimedia, str. 451–459, Brisbane Australia Listopad 2015

[6] Amey Gondhalekar 24.02.2020
https://medium.com/analytics-vidhya/data-augmentation-is-it-really-necessary-b3cb12ab3c3f
https://medium.datadriveninvestor.com/enhance-artificial-intelligence-eac894e23a7e 16.09.2023

[7] IBM Cloud Education, Neural Networks 17.08.2020
https://www.ibm.com/cloud/learn/neural-networks 16.09.2022

[8] Martin T. Hagan, Neural network design. PWS Pub. Co. 01.01.1995

[9] Bojana Dalbelo Bašić, Umjetne neuronske mreže Svibanj 2008

SAŽETAK

Glavni zadatak završnog rada je treniranje modela za prepoznavanje fonta datoteke koristeći napisani generator Alfa numeričkih slikovnih podataka. Za generator podataka korišten je programski jezik Python koji daje podršku mnogim paketima, uključujući i onima za obradu slike kao što su PIL i OpenCV, korišteni kod kreacije generatora podataka. Drugi dio zadatka bilo je kreiranje Neuralne mreže za prepoznavanje fonta slikovne datoteke.

Ključne riječi: Generator podataka, Python, font, TTF.

ŽIVOTOPIS

Ivan Zubčić, rođen 8.10.1998. U Našicama. Pohađao je Opću gimnaziju Stjpan Ivšić u Orahovici. 2017. Godine upisuje Računarstvo stručni smijer na Fakultetu Elektrotehnike, računarstva i informacijskih tehnologija. 2022. Godine odlazi na Erasmus praksu u Valenciju gdje dolazi do teme za rad Prepoznvanje fonta datoteke.

ABSTRACT

The main task of this paper is the creation of a neural network that is able to recognize font of a file. Data used for training the model was synthetically made using a python script for Image generating. The language used for creating both neural network and data generator is Python, chosen for its great support and libraries such as PIL and OpenCV.

Key words: Data generator, Python, font, TTF.

POPIS SLIKA

POPIS GRAFOVA

POPIS TABLICA: