Univerzitet u Beogradu
Matematički fakultet

Master Rad

FONTOOL alat za klasifikaciju i generisanje računarskih fontova

Autor:
Nemanja Mićović

Mentor:
dr Mladen Nikolić

Članovi komsije:

prof. dr Predrag Janičić
doc. dr Aleksandar Kartelj
doc. dr Mladen Nikolić

Beograd, 2018

Zahvalnica

Veliku zahvalnost dugujem mentoru doc. dr Mladenu Nikoliću na velikoj količini znanja koje je nesebično delio, na velikoj podršci, trudu, savetima i idejama tokom izrade master teze i kursevima na kojima mi je držao nastavu.

Veliku zahvalnost dugujem i prof. dr Predragu Janičiću na velikoj inspiraciji da se počnem baviti naučno-istraživačkim radom kao i na divnom uvodu u oblast veštačke inteligencije i svim kursevima na kojima mi je držao nastavu.

Veliku zahvalnost dugujem i mom prijatelju i kolegi Lazaru Rankoviću na sugestijama, idejama i podršci tokom izrade teze.

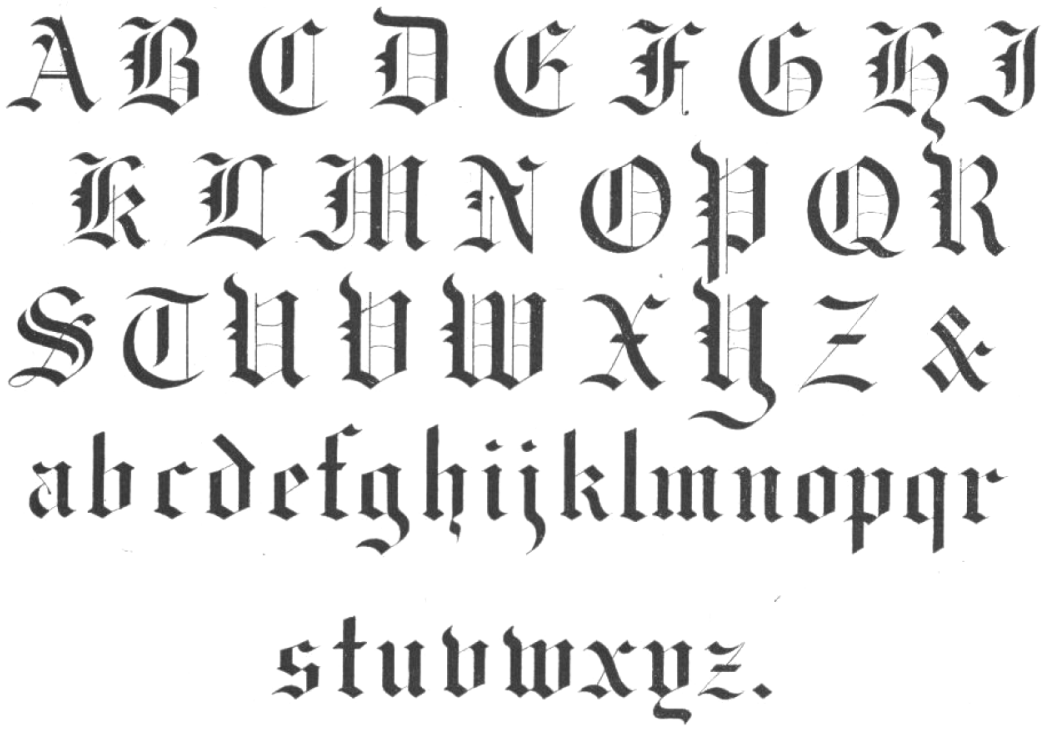
Sadržaj

Mojim dragim roditeljima...

Glava 1

Uvod

Johan Gutenberg [1] smatra se pronalazačem tehnike štampanja pokretnim slovima. Pripisuje mu se otkrivanje prese za štampanje, preteče modernog štampača koja je pokrenula revoluciju štampanja u Evropi. Do tog trenutka su knjige pisane rukom što je zahtevalo veliku količinu vremena i truda od strane tekstopisaca za njihovu izradu, a cena im je usled toga bila visoka. Gutenberg je takođe i razvio prvi font sa nazivom `Black Letter` prikazan na slici 1.1 koji je kasnije korišćen pri štampanju. Godine 1501. Aldo Manucio[2] izumeo je kurziv (eng. *italic*) ilustrovan na slici 1.2. Iako se danas kurziv prvenstveno koristi za isticanje važnijih delova teksta ili citiranje, zanimljivo je napomenuti da je kurziv nastao motivisan uštedom novca time što je omogućio da na stranu stane više teksta pri štampanju.



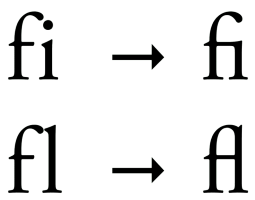
Slika 1.1: Prvi font - `Black Letter`.

Razvoj tehnike štampanje uticao je na razvoj oblasti tipografije. Tipografija (poznata još i kao slovoslagarstvo) je veština i oblast koja se bavi grafičkim uređenjem

[1] Johannes Gutenberg (1398 - 1468) - nemački pronalazač.
[2] Aldus Manutius (1449 - 1515) - italijanski štampar

Slika 1.2: Prvi font koji je imao kurziv (eng. *italic*).

teksta, slova i strane, kao i aspektima koji utiču na vizuelnu percepciju tekstuelnog sadržaja kao što su razmaci između redova i slova, broj redova na strani i slično. Tipografija se bavi i dizajnom fontova za zapis teksta. Kako je tekst ranije pisan rukom, iskusni tekstopisci su radi bržeg pisanja često isto slovo pisali drugačije (ali ipak slično) u zavisnosti od slova koje mu prethodi. Ovakva pojava poznata je kao tipografska ligatura (eng. *ligature*). Glif (eng. *glyph*) je grafički oblik koji predstavlja znak u sistemu pisanja i može biti slovo, cifra, interpunkcijski ili specijalni znak. Font (eng. *font*) predstavlja skup glifova. Prikaz teksta na računaru vrši se prikazivanjem glifova na osnovu kodova karaktera koji su prisutni u binarnoj reprezentaciji teksta. Treba napomenuti da karakter ne mora uvek biti prikazan istim glifom usled prethodno pomenutih ligatura. Na slici 1.3 ilustrovane su dve verzije reči `fi`, sa i bez ligature. Serif (eng. *serif*) predstavlja grafički dodatak na ivicama slova prikazan na slici 1.4. Font čija slova sadrže serife naziva se serifni font. Ukoliko font ne koristi ligature i svaki karakter zauzima jednako prostora, font se naziva *font sa jednakim razmakom* (eng. *monospace font*). Ovakvi fontovi su posebno popularni u editorima i razvojnim okruženjima za programiranje jer omogućavaju ravnomerno slaganje programskog koda. Na slici 1.5 prikazana je razlika između fonta koji koristi ligature i fonta sa jednakim razmakom.



Slika 1.3: Reč levo - bez ligature i reč desno - sa ligaturom.

Slika 1.4: Slovo levo poseduje serife dok slovo desno ne.

Slika 1.5: Donji font predstavlja font sa jednakim razmakom. Gornja reč zauzima isto prostora koliko i donja uprkos tome što poseduje 3 slova više.

Tipografija je važan deo oblasti grafičkog dizajna, a font koji se koristi može drastično da promeni vizuelnu percepciju kreiranog grafičkog sadržaja. Usled toga u grafičkom dizajnu se velika pažnja poklanja odabiru fonta koji se koristi za prikaz sadržaja. Izdvajaju se tri suštinska problema u oblasti: vizuelno prepoznavanje fontova (eng. *visual font recognition - VFR*), vizuelno generisanje fontova (eng. *visual font generation - VFG*) i vizuelizacija fontova. Kako su navedeni fontovi rešavani metodama mašinskog učenja, poglavlje 2 posvećeno je pregledu oblasti i uvođenjem osnovne terminologije, modela i metoda.

Vizuelno prepoznavanje fontova je problem u kojem je potrebno na slici odrediti koji je font korišćen za kreiranje tekstualnog sadržaja. Motivacija za rešavanje ovog problema je u tome što nije retka situacija da dizajner prilikom rada naiđe na neku sliku na kojoj vidi font koji se svidi njemu ili klijentu, ali nema informaciju o tome koji je font korišćen. U poglavlju 3 detaljnije je izložen problem, prethodni pristupi u njegovom rešavanju i predloženo rešenje u okviru teze. Problem je rešavan korišćenjem konvolutivne neuronske mreže (eng. *convolutional neural network*) nad generisanim skupom podataka `ftW`. Skup `ftW` poseduje 120000 instanci koje predstavljaju slike na kojima je koristeći neki od fontova za predikciju ispisana reč engleskog jezika srednje dužine. Isprobano je nekoliko modela od kojih je izabran model koji je pokazao najveću preciznost pri klasifikaciji. Model je postigao preciznost od 0.993 na skupu za testiranje. Dalji rad bi uključio obučavanje modela na većem skupu podataka, njegovo objavljivanje na vebu i eventualnu integraciju u neki od softvera otvorenog koda za obradu slika.

Kreiranje fontova zahteva domenskog eksperta u oblasti tipografije i predstavlja složen i zahtevan proces. Vizuelno generisanje fontova za cilj ima da olakša neki od delova ovog posla, bilo generisanjem rasterske ili vektorske verzije simbola za font. U poglavlju 4 opisan je problem, prethodni pristupi i predloženo rešenje zasnovano na korišćenju konvolutivnog variacionog autoenkodera (eng. *convolutional variational autoencoder - covae*). Autoenkoder je obučavan na generisanom skupu podataka `ftS` koji sadrži 2000 instanci koje predstavljaju slike slova `b`, `a`, `s` i `q` za 500 odabranih fontova. Model je postigao zadovoljavajuće rezultate, a na generisanim slikama slova su prepoznatljiva i odgovarajuće oštrine. Dalji rad na ovom problemu uključuje isprobavanje generativnih suparničkih mreža i uslovnih varijacionih autoenkodera.

Vizuelizacija fontova predstavlja problem u kojem je potrebno izvršiti grafički prikaz fontova u zavisnosti od njihove međusobne sličnosti. Usled velikog broja fontova dostupnih na vebu (po nekim procenama preko 500000) ili računaru dizajnera javlja

se potreba za njihovim adekvatnijom organizacijom u odnosu na sortiranje po nazivu. Prvi korak ka tome je organizacija fontova po međusobnoj sličnosti tako da se korisniku omogući da za izabrani font pronađe fontove koji su mu *bliski*. U poglavlju 5 detaljnije je opisan navedeni problem, njegova ranija rešenja i predloženo rešenje u okviru teze. Rešenje je bazirano na korišćenju konvolutivnog variacionog autoenkodera razvijenog za problem vizuelnog generisanja fontova. Preciznije, koristi se latentni prostor u koji enkoder autoenkodera vrši preslikavanje da se dobiju niskodimenzione reprezentacije fontova i time omogući vizuelizacija korisniku. Dalji rad na ovom problemu uključuje precizniju formulaciju termina *sličnosti fontova* i detaljniji rad na razvijenom modelu. Dobijeni sistem bi se potencijalno mogao ugraditi u softver za editovanje dokumenata ili slika tako da korisniku olakšava traženje sličnih fontova.

Razvijeni sistemi za prepoznavanje i generisanja fontova dostupni su za korišćenje kroz veb aplikaciju `Fontool` opisanu u poglavlju 6 koja kroz responzivni grafički korisnički interfejs korisniku omogućava interaktivno korišćenje razvijenih sistema sa računara i prenosivih uređaja.

Glava 2

Relevantni pojmovi mašinskog učenja

U ovom poglavlju dat je kratak pregled delova oblasti mašinskog učenja koji su relevantni za tezu. Opisana je osnovna terminologija koja se koristi u oblasti kao i osnove nadledanog i nenadgledanog učenja. Potom su opisane neuronske mreže sa propagacijom unapred, kao i konvolutivne neuronske mreže. Iz oblasti nenadledanog učenja prikazani su autoenkoderi i varijacioni autoenkoderi.

2.1 Osnovni pojmovi mašinskog učenja

Mašinsko učenje je oblast u okviru veštačke inteligencije koja se bavi formalizacijom problema učenja i dizajnom algoritama koji su u stanju da uče i donose odluke na osnovu iskustva. Ime *mašinsko učenje* dato je 1959. godine od strane Artur Semjuela[1] koji se smatra jednim od osnivača oblasti. Zanimljivo je da mnoge ideje i algoritmi zastupljeni u oblasti postoje već dugi niz godina. Na primer, rad na neuronskim mrežama uopšte nije nov već je (idejno) započet još daleke 1943. godine radom Vorena Mekkuloka[2] i Valtera Pitsa[3] [1], dok je glavni deo algoritma za obučavanje neuronskih mreža - propagacija unazad (eng. *backpropagation*) formalno opisan još osamdesetih godina XX veka [2, 3, 4].

Artur Semjuel mašinsko učenje definiše na sledeći način: *„Mašinsko učenje je oblast koja računarima daje sposobnost da uče bez toga da budu eksplicitno isprogramirani."*

Jedna od čestih definicija koja se pominje u oblasti i obuhvata formalniji opis učenja je data u knjizi Toma Mičela[4] [5].

Definicija 1. Računarski program uči iz iskustva E u odnosu na neku klasu zadataka T i meru performansi P ako se njegove performanse u zadacima T koje meri P popravljaju u odnosu na veličinu iskustva E.

Mašinsko učenje je pre svega matematička disciplina koja u sebi sadrži koncepte iz oblasti kao što su linearna algebra, verovatnoća, statistika, analiza, optimizacija, numerička matematika i topologija. Kako je jedan od ciljeva oblasti konstrukcija algoritama i modela koji uče, oblast obuhvata i dosta algoritmike, softverskih alata i znanja iz softverskog inženjerstva. Određeni algoritmi mašinskog učenja su izuzetno

[1] Arthur Samuel (1901-1990), američki matematičar
[2] Warren McCulloch (1898-1969) - američki matematičar
[3] Walter Pitts (1923 - 1969) - američki matemičar
[4] Tom Mitchell (1951-) - američki matematičar

računski intenzivni, te se javlja i ozbiljna potreba za paralelizacijom poslova i izračunavanja.

Jedan od razloga popularnosti oblasti leži u velikom uspehu rešavanja problema koji su teško rešivi determinističkim egzaktnim algoritmima, ali treba istaći da oblast mašinskog učenja poseduje i ozbiljnu teorijsku potporu koja za cilj ima formalizaciju teorije učenja [6]. Kako logika predstavlja formalizaciju koncepta *dedukcije*, teorija mašinskog učenja predstavlja formalizaciju teorije *indukcije*. Cilj je doći do modela koji generalizuju - razumeju suštinu problema. To ne znači da modeli neće praviti greške, ali ono što se očekuje je da ako model greši, onda ne greši često ili puno.

U literaturi se najčešće učenje deli na:

- nadgledano učenje (eng. *Supervised learning*),
- nenadgledano učenje (eng. *Unsupervised learning*),
- učenje uslovljavanjem (eng. *Reinforcement learning*).

Često se pominje i oblast polunadgledanog učenja (eng. *Semi-Supervised learning*) [7].

2.1.1 Nadgledano učenje

Učenje se zove nadgledano kada su u podacima dostupne informacije o pravoj vrednosti ciljne promenljive (eng. *ground truth*). Podatak se najčešće naziva instanca (eng. *instances*), a instancu čine njeni atributi (eng. *feature*) koji mogu imati razne vrednosti. Standardni problemi nadgledanog učenja su:

- klasifikacija - koja se karakteriše time da je ciljna promenljiva kategorička i
- regresija - koja se karakteriše time da je ciljna promenljiva neprekidna.

Tabela 2.1 ilustruje jedan skup podataka (eng. *dataset*). Instance su *(160, 65)*, *(172, 75)*, *(180, 82)*, *(165, 70)*, *(171, 81)* i *(193, 94)*, a atributi su *visina* i *težina*.

Generalno, nije striktno određeno šta je ciljna promenljiva, već se to određuje u zavisnosti od dostupnih podataka i problema koji se rešava. U daljoj analizi i izlaganju, kao ciljna promenljiva biće izabran atribut *težina*. Kao problem se postavlja konstrukcija modela koji na osnovu vrednosti atributa *visina* pokušava da odredi vrednost za *težinu*.

U praktičnim primenama, često skup podataka ima desetine, stotine, a nekada čak i hiljade atributa.

visina	**težina**
160	65
172	75
180	82
165	70
171	81
193	94

Tabela 2.1: Primer podataka

Osnovni koraci u problemu nadgledanog učenje

Rešavanje problema nadgledanog učenja najčešće sadrži sledeće korake:

- pretprocesiranje podataka,
- formulacija problema,
- smanjivanje dimenzionalnosti,
- odabir modela,
- obučavanje modela,
- evaluacija rezultata i ocena greške.

Pretprocesiranje uključuje standardizaciju ili normalizaciju podataka, procenu da li se neki atribut uzima u razmatranje ili ne, eliminisanje nedostajućih vrednosti u podacima i slično.

Formulacija problema obuhvata da je dobijen inicijalni skup podataka za rad, da je određena ciljna promenljiva i da je problem formulisan kao problem klasifikacije ili regresije.

Visoka dimenzionalnost podataka može poremetiti rad nekih algoritama mašinskog učenja i dovesti do prokletstva dimenzionalnosti [8, 6]. Za slučaj kada je u skupu podataka pristutan *veliki*[5] broj atributa često se pribegava nekim od metodama za smanjivanje dimenzionalnosti kao što su analiza glavnih komponenti (eng. *PCA - Principal component analysis*) [9, 10] i autoenkoderi (eng. *Autoencoder*) (deo 2.4).

Nakon što je problem formulisan i podaci pripremljeni, potrebno je odabrati model koji će biti obučavan da nauči zakonitosti u podacima. Često se u praksi umesto jednog uzima više modela iz kojih se nakon koraka evaluacije rezultata i ocene greške bira jedan ili manje njih koji čine ansambl [11, 6]. Osim odabira modela, potrebno je odabrati/formulisati funkciju greške nad kojom će potom biti formulisan minimizacioni problem.

Obučavanje modela obuhvata odabir parametara modela tako da se minimizuje greška koju model pravi nad podacima nad kojima se obučava. Minimizacija se najšešće izvodi nekom varijacijom metode gradijentnog spusta [12, 8].

Korak evaluacije rezultata i ocene greške je od fundamentalnog značaja u mašinskom učenju. Ukoliko se sprovede loše, može dovesti do optimistične procene i lošeg ponašanja dobijenog modela u kasnijem korišćenju nad podacima koje nikada nije video. Jedno od glavnih pravila koje treba pratiti u ovom delu je podacima korišćeni u fazi obučavanja modela ni na koji način ne smeju biti korišćeni u fazi evaluacije modela [8].

Linearna regresija

Linearna regresija je jedan od algoritama mašinskog učenja koji se koristi za rešavanje problema regresije u nadgledanom učenju. Model linearne regresije zadat je na sledeći način:

[5] Nije jednostavno reći koliko veliki, ali hiljade atributa već mogu praviti probleme.

$$f_w(x) = w_0 + \sum_{i=1}^{m} w_i x_i \tag{2.1}$$

gde je N broj instanci u podacima, m broj atributa, w_i parametar modela, w_0 slobodni član i x_i i-ta instanca iz skupa podataka. Izračunavanje funkcije $f_w(x)$ daje vrednost ciljne promenjive za prosleđenu instancu x.

Kao funkcija gubitka se često uzima srednjekvadratna greška (eng. *MSE - mean squared error*):

$$MSE = \mathcal{L}(w) = \frac{1}{N} \sum_{i=1}^{N} (f_w(x_i) - y_i)^2 \tag{2.2}$$

gde je x_i vektor dimenzije m, a y_i vrednost ciljne promenljive za i-tu instancu.

Neka je dostupno 90 instanci pri čemu je za svaku instancu poznata vrednost *visina* i *težina* (slično tabeli 2.1).

Pretprocesiranje podataka u ovom jednostavnom slučaju nije neophodno jer će biti korišćen samo jedan atribut - visina i ne postoje nedostajuće vrednosti. U pitanju je problem regresije jer je potrebno predvideti težinu na osnovu visine. Nije potrebno vršiti smanjivanje dimenzionalnosti podataka usled toga što se koristi jedan atribut.

Obučavanje modela se vrši metodom gradijentnog spusta rešavanjem sledećeg minimizacionong problema:

$$\min_{w} \mathcal{L}(w) \tag{2.3}$$

Izačunavanje gradijenata daje:

$$\frac{\partial \mathcal{L}}{\partial w_0}(w) = \frac{1}{2N} \sum_{i=1}^{N} (f_w(x_i) - y_i)$$

$$\frac{\partial \mathcal{L}}{\partial w_1}(w) = \frac{1}{2N} \sum_{i=1}^{N} (f_w(x_i) - y_i) x_i$$

Odnosno gradijent:

$$\nabla \mathcal{L} = \left(\frac{\partial \mathcal{L}}{\partial w_0}(w), \frac{\partial \mathcal{L}}{\partial w_1}(w) \right)$$

Optimizacija se može vršiti metodom gradijentnog spusta koja koristi gradijent da vrši odabir koeficijenata w tako da se vrši minimizacija greške.

$$w_{k+1} = w_k - \alpha \nabla \mathcal{L}(w_k) \tag{2.4}$$

gde je α parametar učenja (eng. *learning rate*) koji kontroliše koliko gradijenti utiču na odabir koeficijenata modela, $k \geq 0$, a w_k k-ti odabrani vektor koeficijenata w. Tipično je α neka mala vrednost, na primer 0.1 ili 0.01, ili se menja tokom procesa optimizacije [8].

Nakon pokretanja optimizacije dobijeni su koeficijenti $w_0 = 1.3$ i $w_1 = 0.46$. Na slici 2.1 su prikazani podaci, pri čemu je na x osi atribut *visina*, a na y osi atribut *težina*. Može se primetiti da crvena prava prati trend koji postoji u podacima i da iako na mnogim mestima greši (za fiksiranu vrednost visine se kao predviđanje uzima vrednost težine na pravoj), ne odstupa drastično od većine prisutnih podataka.

Jednačina dobijene prave je:

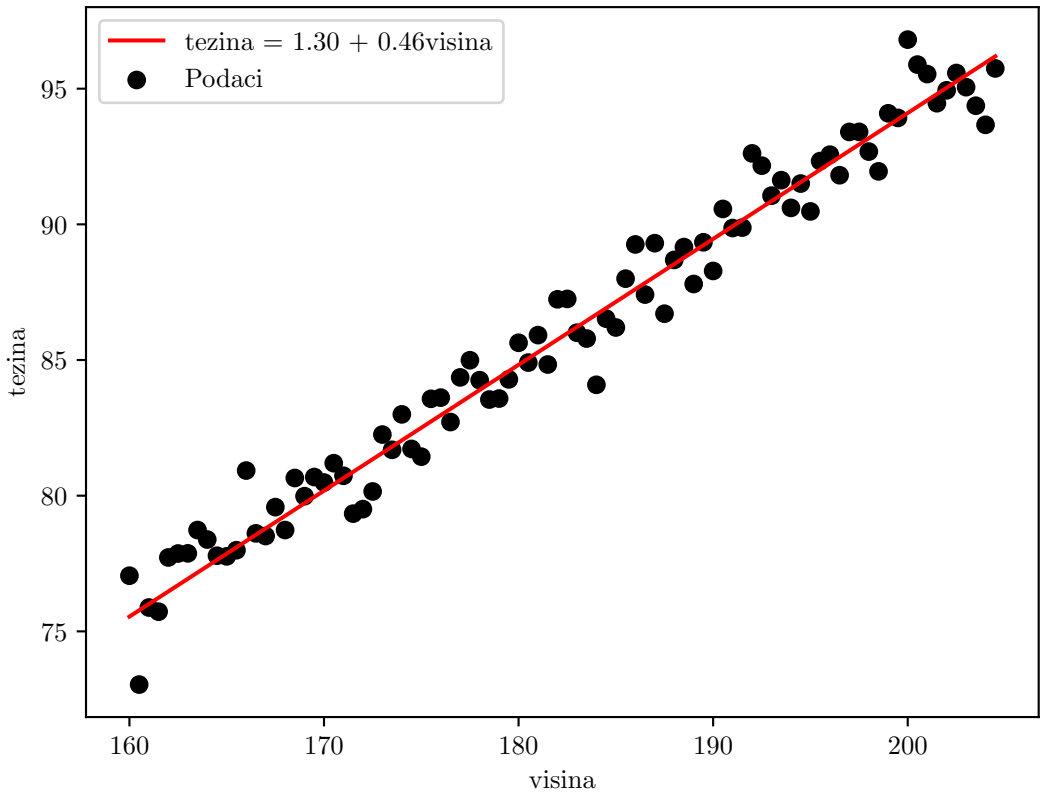
$$w_0 = 1.3$$

$$w_1 = 0.46$$

$$f_w(x) = w_0 + w_1 \cdot x$$

Odnosno:

$$tezina(visina) = 1.3 + 0.46 \cdot visina$$



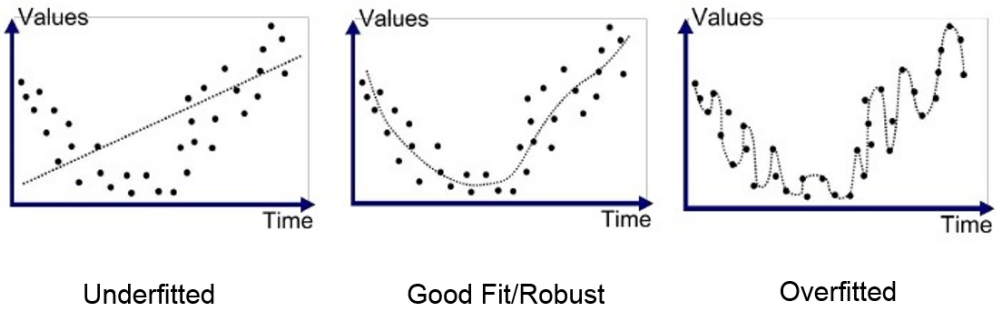
Slika 2.1: Primer linearne regresije

Potprilagođavanje i preprilagođavanje

Potprilagođavanje (eng. *underfitting*) i preprilagođavanje (eng. *overfitting*) su situacije u kojima model nije adekvatno naučio da generalizuje i manifestuje se lošim rezultatima nad podacima koje model nije video.

Potprilagođavanje se dešava u situaciji kada model nije dovoljno izražajan i nije u stanju da izrazi pravilnost koja važi u podacima. Karakteriše se velikom greškom i na podacima nad kojim se model obučava i na podacima nad kojima se testira.

Preprilagođavanje se dešava kada se model previše prilagodi podacima nad kojima se obučava. Karakteriše se niskom greškom nad podacima za obučavanje, ali velikom greškom nad podacima za testiranje. Preprilagođavanje se može otkloniti korišćenjem manje fleksibilnih modela ili korišćenjem regularizacije kojom se vrši kontrola fleksibilnosti modela.



Slika 2.2: Primeri za potprilagođavanje i preprilagođavanje

Regularizacija

Regularizacija predstavlja modifikaciju minimizacionog problema 2.3 dodavanjem parametra $\Omega(w)$ kojim se vrši određena kontrola fleksibilnosti modela i time kontroliše preprilagođavanje. Minimizacioni problem time postaje:

$$\min_w \mathcal{L}(w) + \lambda \cdot \Omega(w),\ \lambda \geq 0 \tag{2.5}$$

gde je λ regularizacioni parametar kojim se kontroliše jačina regularizacije, a $\Omega(w)$ regularizacioni član koji je najčešće ℓ_1 norma (formula 2.6) parametara modela ili kvadrat ℓ_2 norme parametara modela (formula 2.7).

$$\Omega(w) = \sum_{i=1}^{m} |w_i|,\ \lambda \geq 0 \tag{2.6}$$

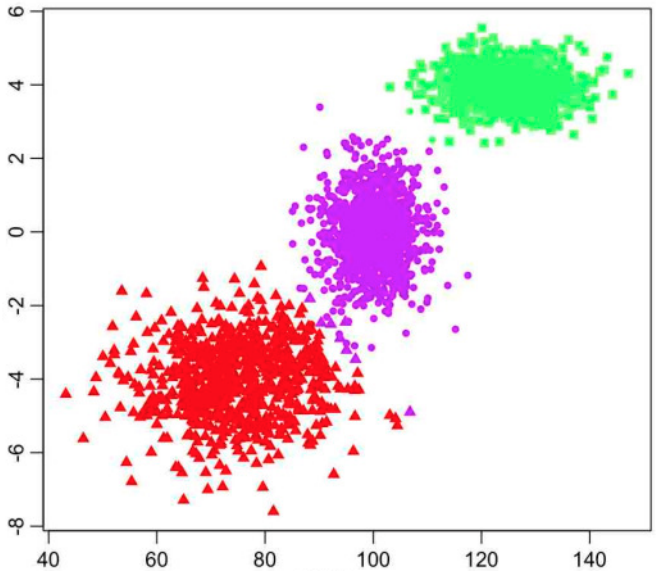
$$\Omega(w) = \sum_{i=1}^{m} w_i^2,\ \lambda \geq 0 \tag{2.7}$$

Regularizacija se skoro uvek koristi u procesu obučavanja modela mašinskog učenja i predstavlja jednu od značajnijih tema u oblasti. Za odabir parametra λ ne postoji striktno pravilo, ali se najčešće uzimaju vrednosti bliske nuli ili se pokušava sa više različitih vrednosti parametra koristeći naprednije metode evaluacije modela [8].

2.1.2 Nenadgledano učenje

U nenadgledanom učenju nije poznata ciljna promenljiva, a zadatak je pronaći (naučiti) određenu zakonitost koja važi u podacima. Na slici 2.3 je prikazan jedan primer nenadgledanog učenja. Algoritam koji je pokrenut je izdvojio tri grupe koje

se razlikuju po nekim parametrima i ovakav proces se naziva *klasterovanje*. Važno je napomenuti da nije jednostavno evaluirati kvalitet dobijenog rešenja. Na primer, na slici 2.3 neko bi možda rekao da podaci čine jedan klaster.



Slika 2.3: Primer jednog algoritma nenadgledanog učenja

Autoenkoderi [13] su jedan od osnovnih modela u nenadgledanom učenju. Omogućavaju da se pronađu nelinearne zakonitosti u podacima i preslikaju u prostor manje dimenzije. Detaljnije su opisani u delu 2.4.

2.1.3 Oblasti primene

Oblasti primene su široke i to predstavlja jedan od razloga za visok nivo popularnosti oblasti. Neke od primena su:

- Klasifikacija slika [14, 15, 16] (poglavlje 3)
- Igranje video igara [17, 18, 19]
- Obrada prirodnog jezika [20, 21, 22]
- Generisanje slika [23, 24] (poglavlje 4)
- Generisanje zvuka/muzike [25]
- Algoritamski portfolio [26]
- Autonomna vožnja
- Bioinformatika [27, 28]
- Društvene mreže [29]

2.2 Klasifikacija

Klasifikacija je problem nadgledanog učenja u kojem je ciljna promenljiva kategorička. Neki od primera klasifikacije su određivanje da li je novinarski članak računarski

ili sportski, ili određivanje toga koji font se koristi na tekstu prisutnom na slici (poglavlje 3).

Neki od poznatih algoritama i modela mašinskog učenja [12, 6, 8] za klasifikaciju su:

- logistička regresija (eng. *Logistic Regression*),
- metod potpornih vektora (eng. *Support Vector Machine*),
- naivni bajesov klasifikator (eng. *Naive Bayes Classifier*),
- neuronske mreže (eng. *Neural Networks*),
- k najbližih suseda (eng. *K Nearest Neighbours*).

2.2.1 Evaluacija modela klasifikacije

Kada je dobijen model pomoću kojeg se vrši klasifikacija, potrebno je oceniti koliko greši. Postoji nekoliko mera koje se često koriste u problemima klasifikacije.

Tačnost klasifikacije

Tačnost klasifikacije (eng. *accuracy*) je udeo tačno klasifikovanih instanci u odnosu na ukupan broj instanci [8].

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.8}$$

Pri čemu važi:

- TP - stvarno pozitivne instance (eng. *true positive*),
- FP - lažno pozitivne instance (eng. *false positive*),
- TN - stvarno negativne instance (eng. *true negative*),
- FN - lažno negativne instance (eng. *false negative*).

Stvarno pozitivne (TP) instance su instance koje su pozitivne i klasifikator im je dodelio oznaku da su pozitivne. Lažno pozitivne (FP) instance su instance koje su negativne, ali im je klasifikator dodelio oznaku da su pozitivne. Slično, stvarno negativne (TN) instance su negativne instance za koje je klasifikator rekao da su negativne i lažno negativne (FN) instance su pozitivne instance za koje je klasifikator odredio da su negativne.

Često se koriste i mere preciznost (eng. *precision*) i odziv (eng. *recall*) [8]. Preciznost je udeo pozitivno klasifikovanih instanci koje i jesu pozitivne, a odziv je udeo instanci za koje je klasifikator rekao da su pozitivne u odnosu na sve pozitivne instance.

$$Precision = \frac{TP}{TP + FP} \tag{2.9}$$

$$Recall = \frac{TP}{TP + FN} \tag{2.10}$$

Harmonijska sredina za preciznost i odziv naziva se $F1$ mera.

$$F_1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \tag{2.11}$$

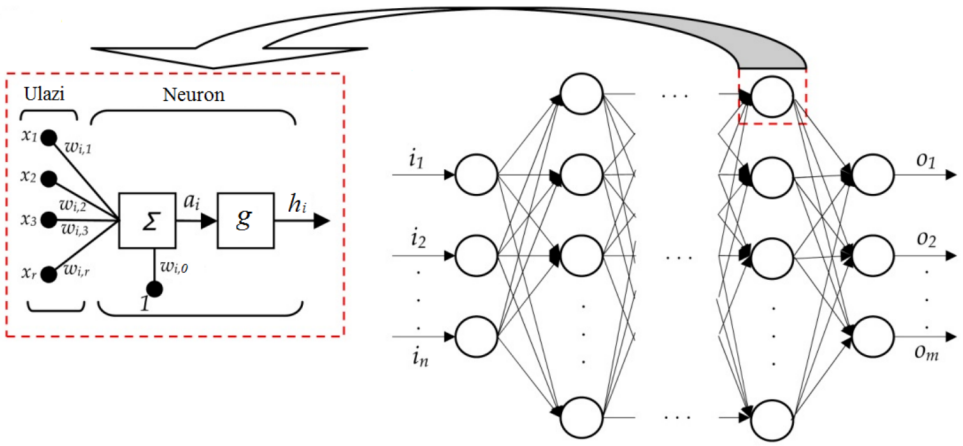
2.3 Neuronske mreže

Neuronske mreže su jedna od metoda mašinskog učenja koja je u poslednjoj deceniji postigla značajne rezultate [14, 30, 25]. Značajni rezultati su doveli do velike popularnosti njihove primene u industriji i nije retkost da se (pogrešno) smatra da su neuronske mreže glavna komponentna oblasti mašinskog učenja. Dostpunost grafičkih kartica sa velikim brojem procesora je u velikoj meri doprineli uspešnosti primene neuronskih mreža jer njihovo obučavanje predstavlja računski intenzivan proces.

Neuronska mreža predstavlja graf koja omogućava aproksimaciju drugih funkcija. Čvorovi ovog grafa se nazivaju *neuroni* koji su međusobno povezani granama. Svaki neuron ima svoje parametre i obučavanje mreža utiče na odabir parametara tako da se vrši minimizacija greške. Metode optimizacije zasnovane na gradijentima su najšešći vid obučavanja neuronskih mreža. Mogućnost paralelizacije algoritma propagacije unazad [2, 3, 4] je jedan od centralnih uzroka za uspešnu primenu neuronskih mreža u poslednjoj deceniji.

2.3.1 Neuronske mreže sa propagacijom unapred

Neuronske mreže sa propagacijom unapred se sastoje od nanizanih *slojeva* neurona pri čemu iz i-tog sloja ne postoje grane ka $(i-1)$-om sloju, a postoje ka $(i+1)$-om sloju. *Skriveni slojevi* su slojevi između prvog i poslednjeg sloja. *Ulazi mreže* su ulazi u prvi sloj, a *izlazi mreže* su izlazi poslednjeg sloja. Česta je upotreba termina *duboka neuronska mreža* i *duboko učenje*. Ne postoji formalni opis koliko skrivenih slojeva mreža treba da poseduje, no u literaturi se najčešće uzima da su potrebna barem dva [31, 8]. Na slici 2.4 su prikazani struktura neurona i mreže sa propagacijom unapred.



Slika 2.4: Struktura neurona i mreže sa propagacijom unapred

Model i struktura

Model se definiše kao:

$$\begin{aligned} h_0 &= x \\ h_i &= g(W_i h_{i-1} + w_{i0}) \quad i = 1, 2, ..., L \end{aligned} \tag{2.12}$$

gde je x vektor koji predstavlja ulaz mreže, W_i matrica čija je j-ta vrsta vektor vrednosti parametara jedinice j u sloju i, L broj slojeva, g nelinearna aktivaciona funkcija i w_{i0} vektor slobodnih članova linearnih kombinacija koje jedinice i-tog sloja izračunavaju. Ako je x vektor dimenzije m onda je:

$$g(x) = (g(x_1), g(x_2), ..., g(x_n))^T$$

Vrednost $f_w(x)$ se izračunava kao:

$$\begin{aligned} f_w(x) &= h_L \\ h_L &= g(W_L h_{L-1} + w_{L0}) \end{aligned} \tag{2.13}$$

Teorema u nastavku predstavlja teorijsko opravdanje za izražajnost neuronskih mreža i govori o tome da neuronska mreža sa jednim skrivenim slojem i konačnim brojem neurona može proizvoljno dobro aproksimirati svaku neprekidnu funkciju [8].
Teorema 1 (Teorema o univerzalnoj aproksimaciji)
Neka je g ograničena i strogo rastuća neprekidna funkcija. Tada za svaku funkciju f $\in C[0,1]^n$ *i svako* $\varepsilon > 0$*, postoji broj* $m \in \mathbb{N}$*, matrica* $W \in R^{m \times n}$*, vektor* $w_0 \in R^m$ *i vektor* $v \in R^m$*, tako da za svako* $x \in [0,1]^n$ *važi*

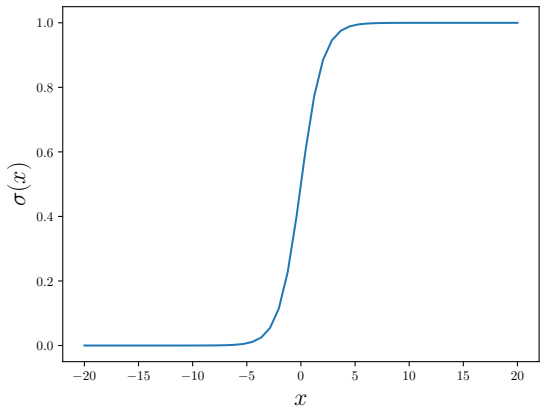
$$|v^T g(Wx + w_0) - f(x)| < \varepsilon$$

Aktivacione funkcije

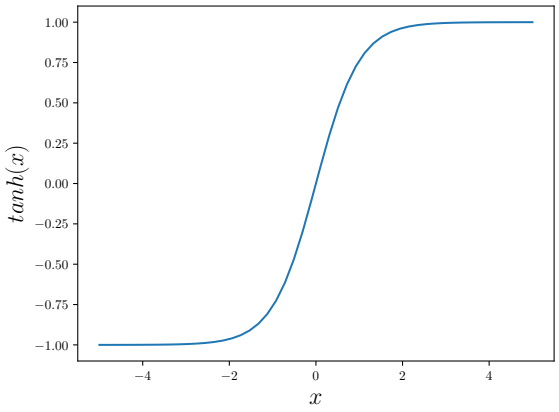
U prethodno datoj formulaciji nije navedena konkretna aktivaciona funkcija g, a najčešće korišćene su sigmoidna funkcija σ, tangens hiperbolički *tanh* i ispravljena linearna jedinica *relu* (eng. *Rectified linear unit - Relu*). Njihovi grafici su prikazani na slikama 2.5, 2.6 i 2.7 i definicije date jednačinama 2.14.

$$\begin{aligned} \sigma(x) &= \frac{1}{1 + e^{-x}} \\ \tanh(x) &= \frac{e^{2x} - 1}{e^{2x} + 1} \\ relu(x) &= max(0, x) \end{aligned} \tag{2.14}$$

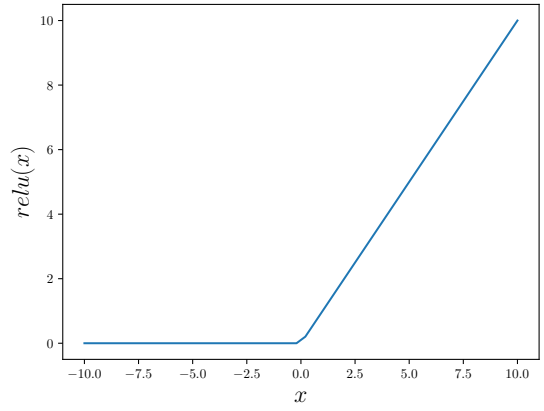
Ispravljena linearna jedinica predstavlja trenutno najkorišćeniju aktivacionu funkciju pri optimizaciji neuronskih mreža. Pokazano je da često ubrzava konvergenciju gradijentnih metoda optimizacije u odnosu na prethodne dve - u radu [14] čak 6 puta. Neki od razloga za to jesu jednostavnost izračunavanja i što je izvod u linearnom delu konstantan i iznosi jedan [8].



Slika 2.5: Sigmoidna funkcija



Slika 2.6: Tangens hiperbolički



Slika 2.7: Ispravljena linearna jedinica

Neuronske mreže u klasifikaciji i regresiji

Neuronske mreže se mogu koristiti i u problemima regresije i klasifikacije. Ukoliko je u pitanju problem regresije, izlaz mreže predstavlja vrednost ciljne promenljive čije

se predviđanje vrši uz napomenu da se ne primenjuje aktivaciona funkcija. Optimizacioni problem koji se rešava je:

$$\min_w \sum_{i=1}^{N} (f_w(x_i) - y_i)^2 + \lambda \cdot \Omega(w)$$

gde je λ regularizacioni parametar i Ω regularizacioni član. Regularizacija teorijski nije neophodna, ali se u praksi praktično uvek koristi jer su neuronske mreže izuzetno prilagodljivi modeli koji su podložni preprilagođavanju. **Jedan od čestih vidova regularizacije koji se koristi u obučavanju neuronskih mreža je i *regularizacija izostavljanjem* (eng. *dropout*). Osnovna ideja je oceniti očekivano predviđanje neuronskih mreža koje se mogu dobiti obučavanjem tako što se prilikom optimizacionog procesa neke jedinice u mreži izostavljaju sa verovatnoćom koja predstavlja metaparametar regularizacije [32, 8].**

Ukoliko se neuronska mreža koristi za klasifikaciju, potrebno je prilagoditi poslednji sloj tako da se vrši predviđanje klasa. To se može učiniti koristeći funkciju *mekog maksimuma* (eng. *softmax*):

$$softmax(x) = \left(\frac{e^{x_1}}{\sum_{i=1}^{C} e^{x_i}}, ..., \frac{e^{x_C}}{\sum_{i=1}^{C} e^{x_i}} \right) \tag{2.15}$$

gde je $x \in \mathbb{R}^C$, a C broj klasa u problemu klasifikacije. Vrednost $softmax(x)$ se može posmatrati kao raspodela verovatnoće jer se vrednost sumiraju na 1. Za verovatnoću jednog podatka važi:

$$P_w(y_i|x_i) = \prod_{j=1}^{C} \left(\frac{e^{a_i}}{\sum_{k=1}^{C} e^{a_i}} \right)^{t_{ij}}$$

gde je t_{ij} promenljiva koja ima vrednost 1 ako y_i odgovara klasi j, a 0 u suprotnom. Minimizacioni problem postaje:

$$\min_w - \sum_{i=1}^{N} \log P_w(y_i|x_i) + \lambda \cdot \Omega(w) \tag{2.16}$$

2.3.2 Konvolutivne neuronske mreže

Konvolutivne neuronske mreže su specijalizovane neuronske mreže namenjene obradi signala koji ima topologiju u obliku mreže [33, 31]. To uključuje signale poput slika, zvuka, videa i vremenskih serija. Zovu se konvolutivne usled matematičke operacije konvolucije koja se primenjuje nad podacima u okviru mreže. Do pojave konvolutivnih mreža bilo je često neophodno ručno definisati atribute nad kojima bi bila primenjena neka od metoda mašinskog učenja. Pomoću konvolucije konvolutivne mreže su u stanju da uče filtere koji predstavljaju drugačiju reprezentaciju signala sa ulaza koji može biti povoljniji za dalju obradu i analizu.

Konvolucija

Konvolucija je matematička operacija nad dve funkcije koja definiše treću funkciju koja ilustruje kako se oblik prve funkcije menja od strane druge fukcije. Postoji više

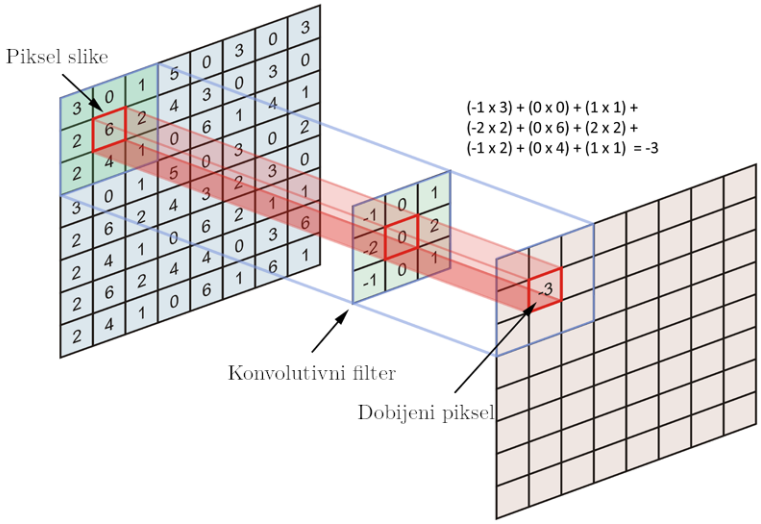
formulacije konvolucije [31], a biće prikazana formulacija koja se često koristi u obradi dvodimenzionog signala kod konvolutivnih neuronskih mreža.

Neka je f matrica dimenzija $m \times n$ i g matrica dimenzija $p \times q$. Konvolucija se definiše na sledeći način:

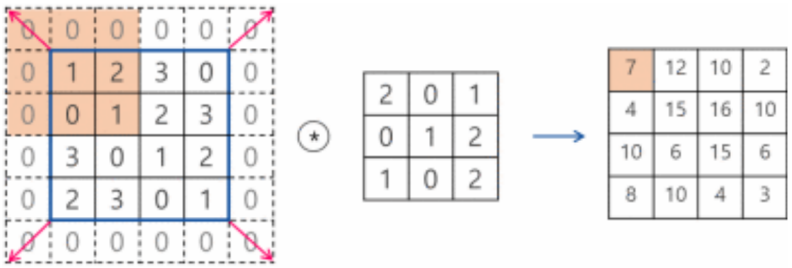
$$(f * g)_{ij} = \sum_{k=0}^{p-1} \sum_{l=0}^{q-1} f_{i-k,i-l} g_{k,l} \tag{2.17}$$

U navedenoj formulaciji može se primetiti da konvolucija nije definisana za neke vrednosti i i j na ivicama slike, odnosno rezultujuća slika će biti manje dimenzije. Često se početna slika proširi nekim vrednosti, tipično nulama kako bi rezultujuća slika ostala iste dimenzije što ilustruje slika 2.9.

Na slici 2.8 prikazana je operacija konvolucije primenjena nad matricom piksela slike i konvolutivnim filterom koji se još naziva i kernel.



Slika 2.8: Operacija konvolucije nad slikama



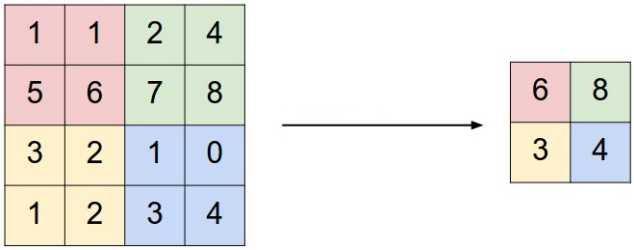
Slika 2.9: Proširivanje slike nulama

Agregacija

Agregacija je operacija koja se koristi da ukrupni informacije i smanji kompleksnost izračunavanja. Obično je ulaz dimenzija 2×2 ili 3×3 i primenom funkcije maksimuma ili proseka nad vrednosti agregacija kao rezultat daje jednu vrednost. Na slici 2.10 je prikazana agregacija nad dimenzijom 3×3 sa funkcijom maksimuma.

Primena agregacije uvodi određeni nivo gubitka informacije, no ono što se gubi je precizna pozicija pronađenog šablona, ali ne i informacija o tome da je šablon pronađen. Ukoliko je poznato da na slici postoji staklo, kvaka, volan i automobilska guma, velika je verovatnoća da se na slici nalazi automobil.

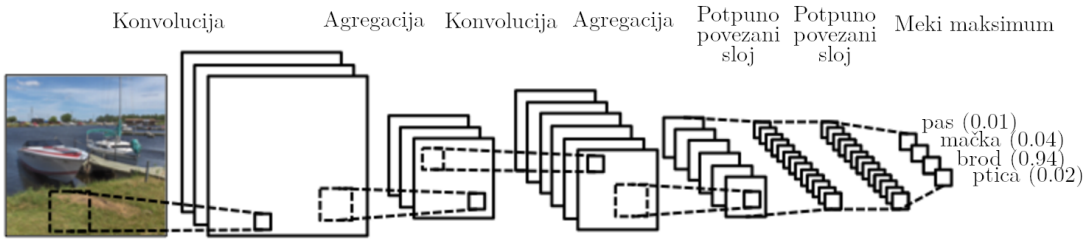
Postoje situacije u kojima nije povoljna primena agregacije. To su najšešće situacije u kojima je neophodno zadržati informaciju o poziciji detektovanog objekta što je karakteristično za problem detekcije objekata i igranja igara [19], mada postoje i pristupi u detekciji objekata koji koriste agregaciju [30].



Slika 2.10: Agregacija sa funkcijom maksimuma

Arhitektura konvolutivne mreže

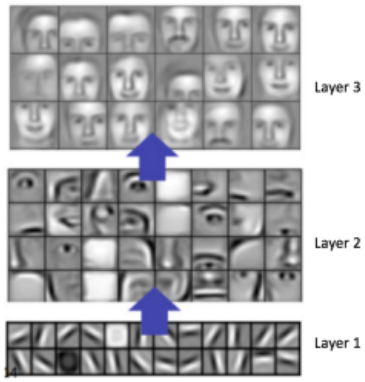
Konvolutivne mreže su postigle značajne rezultate u raznim problemima, no najpoznatija je njihova primena u klasifikaciji slika i detekciji objekata [14, 30]. Razlozi za uspeh konvolutivnih mreža leže u automatskoj konstrukciji atributa nad kojima se vrši nadgledano učenje. Konvolutivne mreže u procesu obučavanja uče filtere koji se obučavaju da prepoznaju određene oblike i koncepte u slučaju klasifikacije slika. Na slici 2.11 je prikazana jedna arhitektura mreže. Često se smenjuju slojevi konvolucije i agregacije kako se ide dublje u arhitekturu mreže. Filteri koji se nalaze na početku mreže uče osnovne oblike kao što su horizontalne, vertikalne i kose linije, a filter dublje u mreži kompleksnije stvari kao što su teksture ili čak i koncepte kao što su jarbol, oči, krila i slično. Na slici 2.12 ilustrovani su naučeni filteru u jednom problemu klasifikacije slika [34].



Slika 2.11: Shema arhitektura konvolutivne mreže

2.4 Autoenkoderi

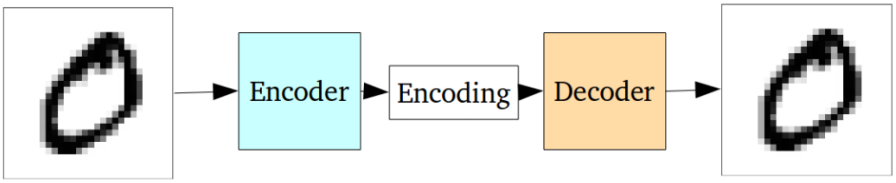
Autoenkoderi su neuronske mreže koje za cilj imaju da nauče kompresovanu reprezentaciju podataka sa ulaza [13, 31]. Kako bi se to ostvarilo, potrebno je da se obuče da podatke sa ulaza kopiraju na izlaz, ali tako što prvo nauče funkciju enkodiranja ulaza u reprezentaciju koja je obično manje dimenzionalnosti, a potom i



Slika 2.12: Primer Vizuelizacije konvolutivne mreže

funkciju dekodiranja koja tu reprezentaciju dekodira u nešto što sličnije originalnom ulazu [31]. Iako je u pitanju jedna neuronska mreža, ona se sastoji iz dve mreže - enkodera i dekodera (slika 2.13). Enkoder ulaz preslikava u kod $h = f(x)$, a dekoder vrši rekonstrukciju ulaza sa $r = g(h)$. Na ovaj način se vrši nelinearno smanjenje dimenzionalnosti što je upravo jedna od glavnih i originalnih primena autoenkodera u nenadgledanom učenju. Reprezentacija manje dimenzionalnosti, naziva se latentni prostor. Ukoliko bi se koristile linearne aktivacione funkcije, autoenkoder bi tražio linearne potprostore što odgovara metodi analize glavnih komponenti.

Kako je autoenkoder neuronska mreža sa propagacijom unapred, proces obučavanja modela se izvodi gradijentnim metodama i algoritmom propagacije unazad kao što je diskutovano u delu 2.3.



Slika 2.13: Shema autoenkodera

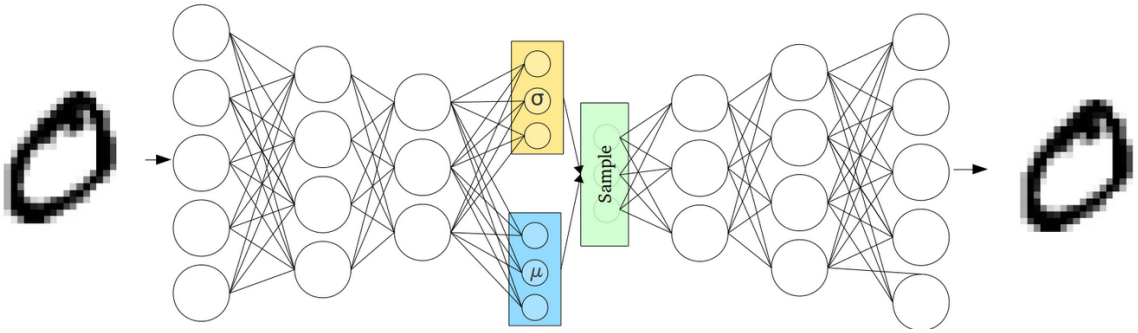
2.5 Varijacioni autoenkoderi

Varijacioni autoenkoder (eng. *Variational autoencoder - VAE*), koji je ilustrovan na slici 2.14 je autoenkoder koji uči rasporedelu verovatnoće koja predstavlja podatke, odnosno uči da modeluje podatke [31]. Usled ovoga, moguće je vršiti uzorkovanje nad obučenim autoenkoderom i koristiti ga kao generativni model.

Funkcija greške za instancu x_i data je jednačinom 2.18.

$$l_i(\theta, \phi) = -E_{z \sim q_\theta(z|x_i)}[\log p_\phi(x_i|z)] + KL(q_\theta(z|x_i)||p(z)) \tag{2.18}$$

gde $q_\theta(z|x)$ predstavlja enkoder, a $p_\phi(x|z)$ dekoder. Prvi sabirak kvantifikuje grešku rekonstrukcije i u procesu optimizacije utiče na varijacioni autoenkoder tako da uči rekonstrukciju ulaznih podataka. Drugi sabirak je vrsta regularizacije i predstavlja Kulbak-Lajbler divergenciju (eng. *Kullback-Leibler divergence - KL*) između raspodele enkodera $q_\theta(z|x)$ i $p(z)$. Ukupna funkcija greška se izračunava sumiranjem člana l_i



Slika 2.14: Shema variacionog autoenkodera

nad skupom instanci.

$$\mathcal{L}(\theta, \phi) = \sum_{i=1}^{N} l_i \tag{2.19}$$

2.6 Biblioteke za mašinsko učenje

Popularnost mašinskog učenja je propraćena i potpomognuta i sa velikim izborom dostupnih kvalitetnih biblioteka koje olakšavaju implementaciju mnogih metoda iz oblasti i drastično ubrzavaju izvođenje eksperimenata jer dozvoljavaju brzo iteriranje kroz eksperimente. U ostatku ovog dela biće prokomentarisane neke od popularnijih biblioteka.

2.6.1 TensorFlow

`TensorFlow` je softverska biblioteka otvorenog koda za efikasna numerička izračunavanja koju razvija kompanija `Google` [35]. Biblioteka je originalno razvijena za interne potrebe u okviru kompanije, a kasnije je doneta odluka da se javno objavi i da doprinos zajednici otvorenog koda. Slika 2.15 prikazuje logo biblioteke. U pitanju je biblioteka niskog nivoa sa odličnom dokumentacijom koja korisniku nudi veliku dozu fleksibilnosti pri korišćenju i definisanju modela mašinskog učenja. Osnovna ideja je na definisanju grafa numeričkih izračunavanja koje biblioteka može da paralelizuje nad dostupnih hardverom u okruženju u kojem se kod izvršava, a najšešće su to grafičke kartice. Jezgro je razvijeno u jeziku `C++` ali postoji omotač koji omogućava da se biblioteka koristi kroz jezik `Python`. Nad tim mehanizmima izgrađena je podrška za implementaciju širokog spektra metoda mašinskog učenja i njihove primene.

2.6.2 Keras

Keras [36] je biblioteka visokog nivoa za programski jezik `Python` prvenstveno namenjeno za rad sa neuronskim mrežama. Biblioteka se oslanja u pozadini na korišćenje drugih biblioteka kao što su `TensorFlow`, `Theano` i `CNTK`. Kada se biblioteka instalira, potrebno je konfigurisati koja će se biblioteka koristiti u zadnjem delu (eng. *backend*). Osnovna motivacija za bibliotekom je olakšavanje rada sa neuronskim mrežama i eliminisanje velikoj broja tehničkih detalja koje nude biblioteke kao što je `TensorFlow`. Na slici 2.16 prikazan je logo biblioteke. Autor biblioteke, Fransoa Šole (eng. *Francois*

Slika 2.15: Logo biblioteke `TensorFlow`

Chollet) objavio je i knjigu koja se bavi dubokim učenjem i intenzivno koristi `Keras` [37].

Slika 2.16: Logo biblioteke `Keras`

2.6.3 SciKit-Learn

Biblioteka `SciKit-Learn` je biblioteka za programski jezik `Python` koja pruža podršku za klasifikaciju, regresiju, klasterovanje, smanjenje dimenzionalnosti i preprocesiranje podataka. Poseduje vrlo kvalitetno i ažurnu dokumentaciju i predstavlja jednu od popularnijih biblioteka u oblasti mašinskog učenja. Na slici 2.17 prikazan je logo biblioteke.

Slika 2.17: Logo biblioteke `SciKit-Learn`

2.6.4 Dodatne korisne biblioteke

Dodatne korisne biblioteke su:

- `SciPy` - podrška za linearnu algebru, numeričku matematiku i slično,
- `NumPy` - rad sa N dimenzionalnim nizovima,

- `Pandas` - manipulacija podataka,
- `Matplotlib` - vizuelizacija podataka,
- `Seaborn` - vizuelizacija podataka,
- `Scrapy` - parsiranje Veb stranica.

Glava 3

Vizuelno prepoznavanje fontova

Vizuelno prepoznavanje fontova predstavlja problem u kojem je potrebno za tekst sa slike odrediti kojim je fontom napisan. Jedan od razloga za rešavanje ovog problema je mogućnost da grafički dizajner odredi koji je font korišćen za ispis teksta na slici za koju nije dostupna takva informacija. U daljem tekstu je dat precizniji opis problema, prethodna rešenja i radovi na temu, a potom je opisano predloženo rešenje i dalji rad na ovoj temi.

3.1 Postavka problema

Vizuelno prepoznavanje fontova (eng. *Visual font recognition - VFR*) je problem u kojem je potrebno odrediti font koji je korišćen da se napiše tekst prisutan na slici. Problem se može svesti na problem klasifikacije u okviru nadgledanog učenja. Instance predstavljaju sliku, a ciljna promenljiva predstavlja oznaku fonta. Ono što problem razlikuje u odnosu na opštu klasifikaciju slika [14] i skupova podataka kao što su `ImageNet` i `MNIST` [38, 39] je veliki broj klasa uzrokovan velikim brojem dostupnih fontova, nepostojanje referentnog skupa podataka kao što su `ImageNet` i `MNIST` i potreba da se na fotografijama obradi dodatna pažnja na detalje kao što su serifi.

Efikasno i kvalitetno rešavanje `VFR` problema može doprineti kvalitetu grafičkog dizajna jer rešava probleme određivanja nepoznatog fonta na slici i olakšava pronalaženje sličnih fontova.

Izdvajaju se dve varijante `VFR` problema. U jednom su slike striktno nastale na računaru kao rezultat rada nekog softvera (označimo takav problem sa `VFRa`), a u drugom, `VFRb`, pored takvih slika dozvoljene su i fotografije iz realnog sveta. Iako ne postoje referentni skupovi podataka za `VFR` problem, za `VFRa` nije problem napraviti skup podataka ukoliko su dostupni fontovi jer se pomoću njih može generisati veliki broj instanci. Postoji dosta fotografija za problem `VFRb` ali je veliki problem u tome što one nemaju poznatu informaciju o tome koji se font nalazi na fotografiji, a određivanje toga je proces koji zahteva eksperta u tipografiji i vremenski može izuzetno da potraje. Usled tog ograničenja, za rešavanje je izabran problem `VFRa`, pristup koji je korišćen je izložen u delu 3.3, a eksperimentalni rezultati su prikazani u delu 3.4.

3.2 Prethodni pristupi

Neki od prethodnih pristupa se oslanjaju na statističku analizu dostupnih fontova, analizu lokalnih delova slova i teksturei ručno konstruisanje atributa [40, 41, 42, 43,

44, 45]. Ovakvi pristupi su primenjeni nad malim brojem fontova i nisu otporni na šum, zamućivanje, afine transformacije, promene perspektive i kompleksnu pozadinu. Rad [45] kao poslednji pronađeni sa okvakvim pristupom obučava model potpornih vektora na skupu od 216 instanci i 6 fontova i ostvaruje prosečnu preciznost od 93.54%.

Pristup korišćen u radu [46] koristi duboko učenje, konvolutivnu mrežu i autoenkoder da nauči klasifikator fontova za 2383 različita fonta. Model je ostvario preciznost od preko 80% na problemu `VFRb` za proveru da li je ciljni font u prvih 5 rezultata mreže. Ostvareni rezultat je impresivan usled činjenice da su dopuštene i fotografije iz realnog sveta. Sistem je razvijen od strane kompanije `Adobe` i ugrađen je u njihov softverski proizvod `Adobe Photoshop` 2016. godine.

3.3 Predloženi pristup problemu

Načelno je razmatrano i oprobano da se za svaki font generiše slika jednog slova i da se vrši klasifikacija nad jednim slovom. Time bi za svaki font bilo prisutno k odabranih slova, na primer 26 ili 52 (mala ili mala i velika slova engleskog alfabeta) ali je uočen ključni problem koji ograničava klasifikator u preciznosti, a to je da su neka slova izuzetno slična u određenim fontovima i da ukoliko se razmatraju individualno teško je uočiti razliku. Rešenje ovog problema jeste da se ne razmatra jedno slovo, već više slova koje čine reč čime podatak onda poseduje određenu informaciju o ligaturama fonta i sadrži više pristunih slova koje nose drastično više informacije.

Za razvijanje neuronskih mreža korišćena je biblioteka `Keras` sa podrškom za biblioteku `TensorFlow`.

3.3.1 Skup podataka `ftW`

Sa veba je preuzeto oko 18000 fontova i 10000 engleskih reči srednje dužine (oko 5 slova). Preuzeti fontovi su dobijeni u `TrueType` formatu tako da su varijante fonta koje imaju kurziv ili podebljana slova i slično odvojene datoteke te su u klasifikaciji tumačeni kao i različiti fontovi. Usled hardverskih ograničenja bilo je nužno ograničiti se na podskup fontova te je odabrano 40 fontova trudeći se da odabrani fontovi budu što razvnovrsniji. U daljem radu na ovom problemu, u planu je koristiti pun skup podataka i podržati prepoznavanje svih dostupnih fontova.

Skup svih dostupnih fontova će biti označen sa $\mathcal{F}$, skup svih dostupnih reči sa $\mathcal{W}$, 40 odabranih fontova sa $\mathcal{F}'$, a odabrane reči kao skup $\mathcal{W}'$. Za svaki font $f \in \mathcal{F}'$ i za svaku reč $w \in \mathcal{W}'$ je generisana slika dimenzija 104×104 piksela na kojoj je reč w napisana fontom f. Time je za svaki font dobijeno 3000 instanci i ukupno $40 \cdot 3000 = 120000$ instanci. Skup podataka je dalje podeljen na tri podskupa:

- skup za obučavanje od 60000 instanci - `ftWtrain`,
- skup za testiranje od 30000 instanci - `ftWtest`,
- skup za validaciju od 30000 instanci - `ftWval`.

Pri kreiranju navedenih skupova uzeto je podjednako instanci iz svake od klasa, a pre generisanja slika za font lista sa rečima je permutovana.

Na slici 3.2 prikazano je nekoliko instanci iz skupa podataka. Konstruisani skup je namenjen rešavanju problema `VFRa`.

Slika 3.1: Svi fontovi iz skupa $\mathcal{F}'$

Slika 3.2: Uzorak iz skupa `ftW`

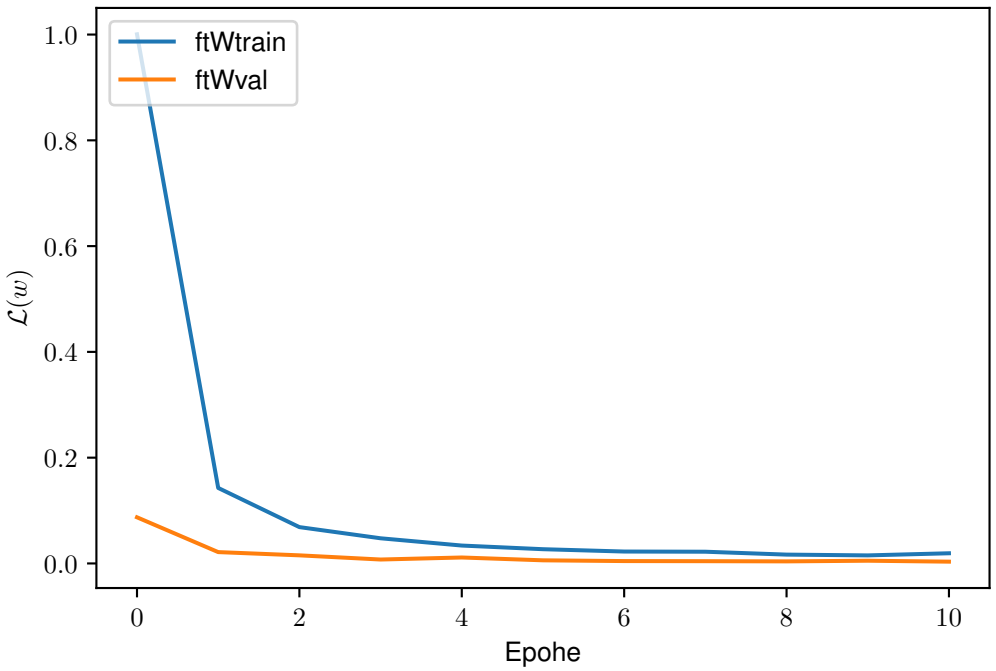
3.4 Eksperimentalni rezultati

Isprobano je nekoliko arhitektura koje su obučavane na skupu podataka `ftWtrain` i testirane na skupu podataka `ftWval`. Inicijalno su testirane arhitekture koje se koriste u zvaničnim primerima biblioteke `keras` za skupove podataka `MNIST` i `CIFAR-10` [47], a potom su na osnovu `CIFAR-10` arhitekture konstruisane još dve arhitekture dodavanjem jednog (`FONTOOLV1`) i dva (`FONTOOLV2`) dodatna sloja konvolucije i agregacije. Obučavanje modela je sprovedeno sa korišćenjem ranog zaustavljanja koje je konfigurisano tako da zaustavi optimizaciju ukoliko se greška na validacionom skupu ne smanji više od 0.001 u 7 epoha za redom. Tokom obučavanja su čuvane težine modela ukoliko bi se smanjila greška na validacionom skupu. Nakon što je završeno obučavanje, za svaki model su uzete težine za koje je greška na validacionom skupu bila najmanja. U tabeli 3.1 prikazani su rezultati testiranja. U koloni `model` nalazi se ime korišćenog modela, kolona `ftWval` sadrži preciznost na skupu `ftWval` i kolona `min` sadrži broj epohe nakon koje je postignut minimum greške na validacionom skupu `ftWval`. Za dalji rad je odabran model `FONTOOLV1` usled preciznosti od 0.9992. Nakon evaluacije, njegova preciznost na skupu za testiranje `ftWtest` iznosi 0.9993. Na slici 3.3 prikazan je grafikon koji ilustruje menjanje greške na trening i validacionom skupu tokom obučavanja. Obučavanje navedenih modela sprovedeno je na hardveru koji poseduje grafičku kartu `Nvidia GeForce 1060 GTX` i procesor `Intel i7 7700k`.

Na dijagramu 3.4 je prikazan graf koji predstavlja arhitekturu odabrane mreže `FONTOOLV1`. Prikazano je kako se dimenzije podataka menjaju njihovom propagacijom

model	ftWval	min
MNIST	0.9979	18
CIFAR-10	0.9989	9
FONTOOLV1	0.9992	10
FONTOOLV2	0.9987	14

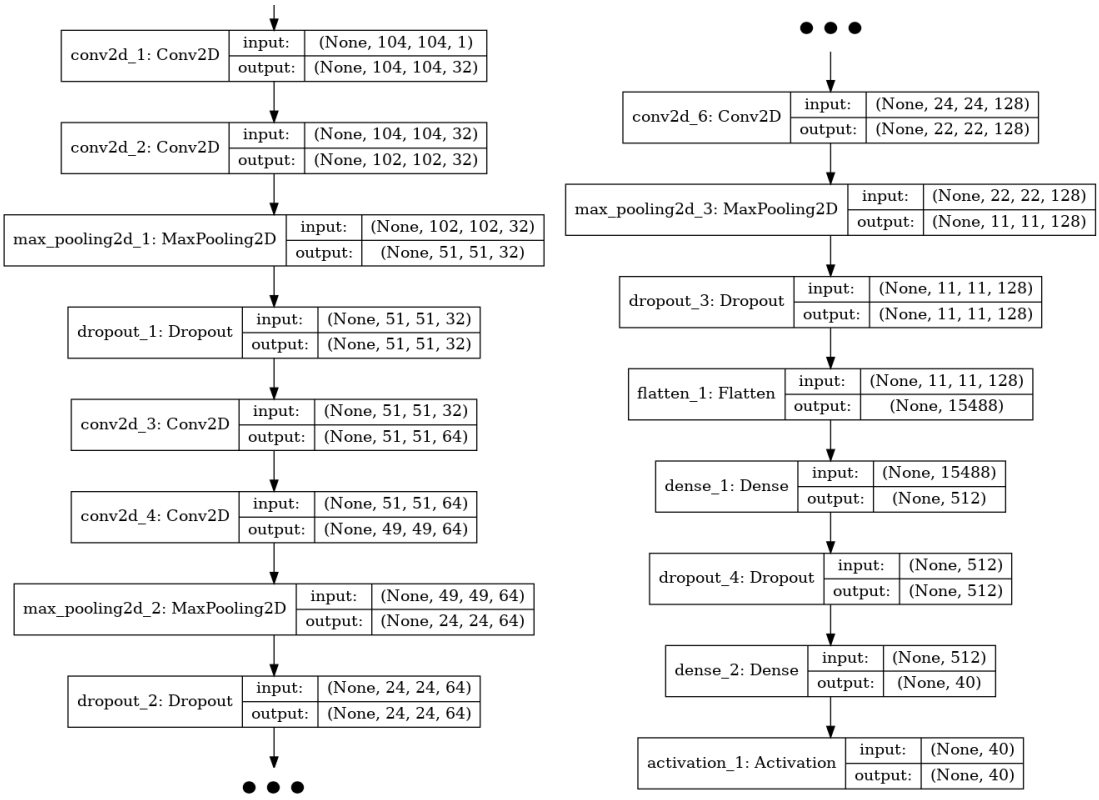
TABELA 3.1: Preciznost obučavanih modela na skupu podataka `ftWval`.



SLIKA 3.3: Greška tokom obučavanja.

kroz mrežu. Na početku, ulazna dimenzija je `(None, 104, 104, 1)` što označava da se na ulaz dovodi k slika (oznaka za nepoznatu dimenziju u biblioteci `TensorFlow` je `None`) dimenzije 104×104 sa jednim kanalom jer su slike crno bele. Na izlazu mreže je dimenzija `(None, 40)` i predstavlja ocene pripadnosti klasama koristeći funkciju mekog maksimuma (eng. *softmax*). Veličina kernela sa kojima je vršena konvolucija postavljena je na 3×3. Oznaka:

- `Conv2D` - označava operaciju konvolucije,
- `MaxPooling2D` - označava operaciju agregacije sa funkcijom maksimuma,
- `Dropout` - označava tehniku *regularizacije izostavljanjem* [8, 31],
- `Dense` - označava potpuno povezani sloj neurona,
- `Activation` - označava aktivacionu funkciju, a u konretnom slučaju, funkciju mekog maksimuma.



Slika 3.4: Odabrana arhitektura konvolutivne mreže

3.5 Interpretabilnost modela

Neuronske mreže u većini slučajeva ne daju intepretabilne rezultate što je jedna od njihovih glavnih mana. Kako su u pitanju izuzetno fleksibilni modeli postoji velika opasnost od preprilagođavanja pri obučavanju. Kao dodatna ilustracija kvaliteta dobijenog modela, u editoru rasterskih slika `Gimp`[1] napravljeno je 16 slika sličnih onima iz skupa podataka `ftW` na kojima je korišćen neki font iz skupa $\mathcal{F}'$ koji je prethodno bio instaliran na sistemu. U svih 16 situacija model je tačno klasifikovao font sa ocenom oko 0.99, odnosno ocena funkcije mekog maksimuma u poslednjem sloju je u raspodeli verovatnoće za tačnu labelu dala vrednost oko 0.99. Na slici 3.5 prikazane se slike koje su korišćene.

Koristeći pristup iz [34] izvršena je vizuelizacija naučenih filtera u modelu koja nekada ume da signalizira na koje aspekte slike se neuronska mreža fokusira. Iz dobijenih slika takav zaključak nije mogao biti izveden, mada se mogu primetiti neki filteri koji se bave teksturama pristunim na slici. Na slikama 3.6 i 3.7 prikazani su neki filteri iz konvolutivnih slojeva.

3.6 Moguća poboljšanja i dalji rad

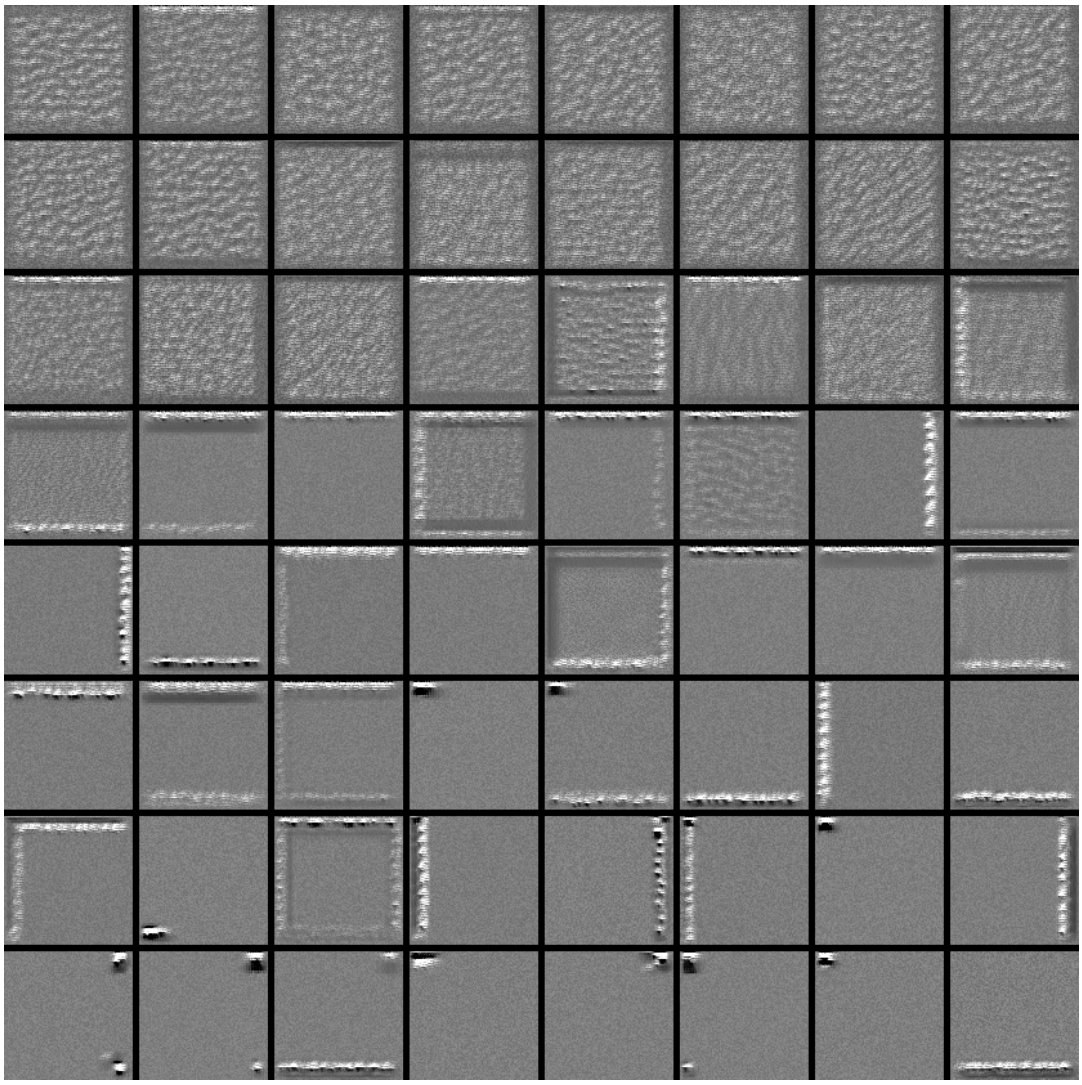
Jedna od tehnika koja se može iskoristiti da se poboljša robusnost modela [14] je dodavanje novih instanci nad skupom podataka tako što se nad instancama primenjuju transformacije kao što su:

[1] Popularni softver za obradu rasterskih slika otvorenog koda.

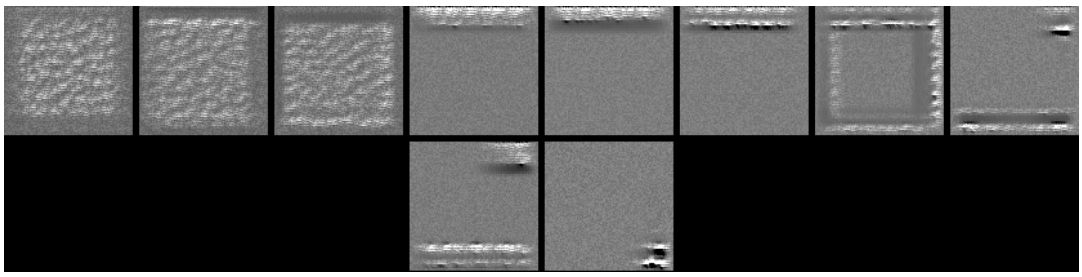
Slika 3.5: Dodatne slike za evaluaciju

- translacija,
- smicanje,
- rotacija,
- zumiranje,
- zamućivanje (eng. *blurring*),
- zaoštravanje (eng. *sharpening*).

Usled specifičnosti problema `VFRa` neke transformacije nisu neophodne. Naime translacija nije potrebna jer se od korisnika očekuje da reč na slici iseče adekvatno tako da sama reč nema više prostora za transliranje. Smicanje nije potrebno jer se u tipografiji ne koristi za transformaciju postojećeg fonta, a zakošena slova nekog fonta (eng. *italics*) su tumačena kao zaseban font. Zumiranje nije neophodno usled maksimalnog prostiranja slova unutar slike, dok rotacija može imati smisla za slučaj da je u ulaznom podatku dizajner želeo rotiranjem teksta da postigne određeni efekat. Korisnik vrlo lako može sam rotirati sliku pre nego što je prosledi klasifikatoru. Može



Slika 3.6: Sloj `conv2d_5`



Slika 3.7: Sloj `conv2d_6`

delovati kao nepotrebno da se od korisnika očekuje neka vrsta pretprocesiranja podataka, ali treba uzeti u obzir da je u pitanju najčešće dizajner ili neko ko koristi program za obradu slika tako da rotacije teksta na slici za takvu osobu nije veliki posao, dok bi forsiranje mreže da uči različite uglove osetno otežalo proces učenja.

Transformacije koje imaju smisla da budu primenjene jesu zamućivanje i zaoštavanje i to iz razloga kompresije slika. Naime dešava se da pri izmeni veličine slike dođe do gubljenja na kvalitetu, a kako su instance u skupu `ftW` generisane na isti način

koristeći jezik `Python` i biblioteku `Pillow` može se reći da sve instance imaju istu količinu izgubljenog kvaliteta. Ukoliko bi se prosledila slika manje rezolucije, na primer 50×50 koja bi bila povećana na 104×104 moglo bi doći do potencijalnog problema jer u podacima nije postojala *mutna* instanca. Ipak ovo se retko dešava jer se slika konstruiše na osnovu dokumenta u kojem je tekst vektorski prikazan ili ukoliko je već pristutna slika rasterska, dizajner često ima sliku u dovoljno visokoj rezoluciji.

Jedno od poboljšanja je ažuriranje skupa podataka za obučavanje tako da koristi sve fontove koji su prikupljeni. Time bi klasifikator postao praktično upotrebljiv jer bi bio u stanju da prepozna veliki broj fontova. Iako može delovati da nekoliko hiljada klasa na izlazu može poremetiti klasifikaciju, rad [46] ilustruje da to ne predstavlja nikakav problem za mrežu. Osim toga, skup podataka `ImageNet` poseduje 20000 klasa i postignuti su impresivni rezultati [14, 16].

Jedan od mogućih daljih pravaca je i korišćenje mehanizma *pažnje* (eng. *attention*) iz rada [16]. Specifičnost problema klasifikacije fonta potencijalno može biti pogodna za primenu mehanizma *pažnje* usled potrebe da se dodatno obrati pažnja na detalje poput serifa.

Za slučaj da se razvijeni sistem bude ugrađivao u već neki postojeći softver poput editora rasterskih slika ili slično, potrebno je pojednostaviti model i smanjiti neophodno memorijsko zauzeće. Kompresija neuronskih mreža je poznat problem i postoje već adekvatna rešenja [48, 46].

Glava 4

Vizuelno generisanje fontova

Kreiranje fontova je kompleksan proces u kojem je potrebno definisati vektorski svaki glif pri čemu je poželjno da glifovi budu definisani tako da poseduju isti stil. Usled ovoga, javlja se potreba za sistemom koji je u stanju da automatizuje neki deo opisanog procesa. Postoje fontovi koji nemaju definisane simbole za sve glifove te se javlja i problem generisanja novih reprezentacija glifova koje sadrže stil prisutan u fontu [11]. Ovaj problem je posebno izražen u kineskom jeziku [49] usled velikog broja simbola[1] koje jezik poseduje. U tekstu koji sledi detaljnije je opisan problem vizuelnog generisanja fontova, prethodnih pristupa u rešavanju ovog problema, a potom je dat predlog rešenja koje koristi varijacioni konvolutivni autoenkoder. Korišćeni model se pokazao kao adekvatan za rešavanje problema a u ostatku teksta su prikazani generisani simboli i diskutovan dalji rad.

4.1 Postavka problema

Vizuelno generisanje fontova (eng. *visual font generation - VFG*) je problem u kojem je potrebno izvršiti automatsko generisanje jednog ili više simbola za font bez informacije o tome kako izgledaju ostali simboli. Rešavanje problema omogućava da se generiše potpuno nov font ili neki njegov simbol. Jedna od primena je i olakšavanje početnog koraka dizajna fontova u tipografiji kroz koje prolazi dizajner tako što od sistema može dobiti skice nekih slova koje mu mogu dati ideju za dalji razvoj fonta. U daljem tekstu rešavan je ovaj problem sa ciljem da se generiše reprezentacija jednog simbola. Postoji i sličan problem u kojem su unapred poznati neki simboli, te je moguće generisati ostale simbole u istom stilu [11].

4.2 Prethodni pristupi

Postoji nekoliko pristupa koji se bave varijantama problema `VFG`. Pristup [50] simbole fonta posmatra kao Bezijerove krive [51], pravi parametrizaciju nad simbolima tako da se poklapa u što više tačaka između istih slova i u odnosu na prostor svih dostupnih fontova traži površ na kojoj *leže* dostupni fontovi. Traženje površi se vrši korišćenjem Gausovih procesa [52]. Neki od modernijih pristupa [24, 11] koriste generativne suparničke mreže (eng. *generative adverserial network - GAN*) [23, 24, 53] za generisanje potrebnih simbola.

[1] Nije jednostavno odgovoriti koliko kineski jezik poseduje simbola, ali zvanični test *Hànyǔ Shuǐpíng Kǎoshì* za kineski jezik zahteva poznavanje 2600 simbola

4.3 Predloženi pristup problemu

Razvijen je duboki varijacioni konvolutivni autoenkoder (eng. *convolutional variational autoencoder - covae*) [54] koji za cilj ima da nauči funkciju interpolacije između fontova koji su u skupu podataka. Varijacioni autoenkoder omogućava generisanje podataka koristeći dekoder obučenog autoenkodera. Podaci se generišu uzimanjem različitih vektora iz latentnog prostora, pri čemu se očekuje da male promene u nekoj od vrednosti vektora uzrokuju malu promenu u generisanom izlazu, odnosno preslikavanje bi trebalo da bude neprekidno. U stranoj literaturi ovakva pojava se naziva interpolacija [31] i u daljem tekstu će imati ovo značenje.

4.3.1 Skup podataka `ftS`

Neka je $\mathcal{S}$ skup svih simbola koje je potrebno generisati, $\mathcal{S}' \subseteq \mathcal{S}$ i $k = |\mathcal{S}'|$.

Za rad je razvijen skup podataka `ftS` koji se sastoji od 500 fontova iz skupa $\mathcal{F}$ iz dela 3.3.1. Za svaki font je generisana slika rezolucije 64×64 na kojoj je napisano slovo iz skupa dostupnih slova $\mathcal{S}'$. Time je dobijen skup podataka sa $500 \times k$. U konkretnom eksperimentu uzeto je $\mathcal{S}' = \{b, a, s, q\}$ čime je dobijeno 2000 instanci. Na slici 4.1 prikazano je nekoliko instanci iz skupa podataka.

4.3.2 Arhitektura razvijenog modela

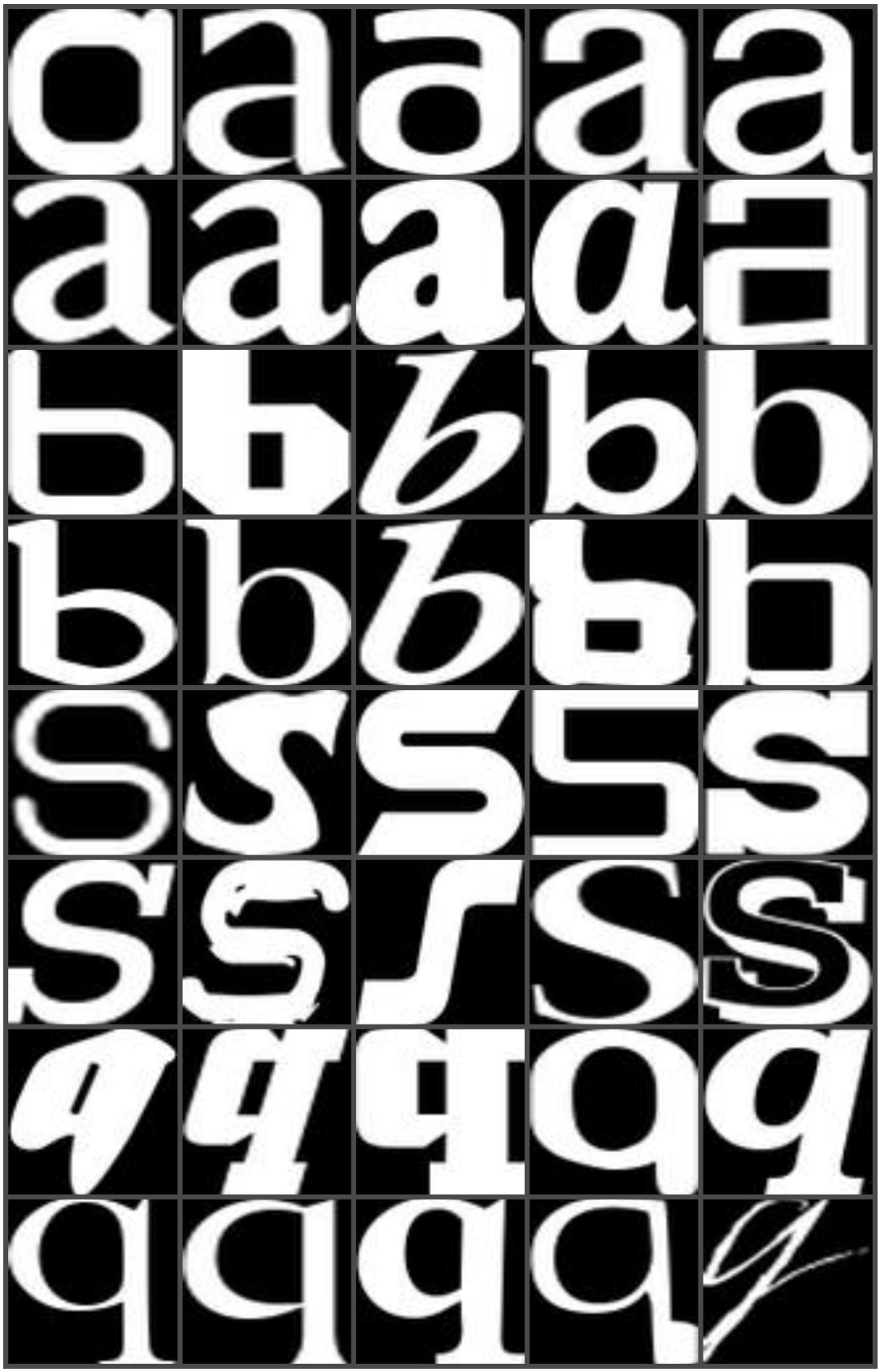
Na slici 4.2 je prikazana arhitektura razvijenog autoenkodera, dok su na slikama 4.3 i 4.4 ilustrovane arhitekture enkodera i dekodera. Arhitektura je preuzeta iz zvaničnog primera za biblioteku `keras` [36] sa veba koji koristi varijacioni konvolutivni autoenkoder da generiše `MNIST` cifre [39].

4.4 Eksperimentalni rezultati

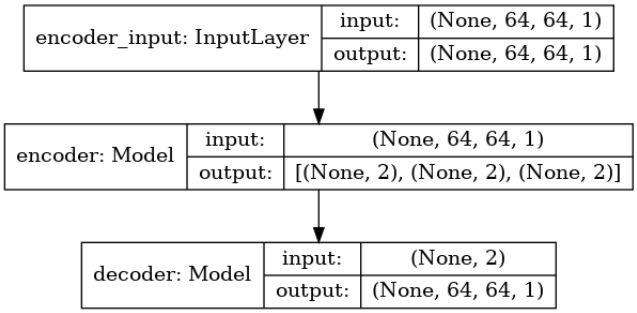
Obučavanje modela zahtevalo je više iteracija usled dobijanja inicijalnih loših rezultata prikazanih na slici 4.5. Loši rezultati su dobijeni usled korišćenja više fontova za koje nije postojala adekvatno definisana reprezentacija za neke glifove te je dobijana slika koja sadrži beli okvir umesto slova. To je drastično uticalo na autoenkoder da generiše mutne slike i da veliku pažnju posveti generisanju belih okvira.

Nakon što su pažljivije odabrani fontovi (njih 500) tako da nemaju nedostajuće reprezentacije za glifove, dobijeni su bolji i oštriji rezultati. Fontovi su birani tako da podjednako budu zastupljeni tipični i egzotični fontovi. Za dimenziju latentnog prostora (prostora u koji enkoder preslikava ulaz) isprobane su vrednosti 2, 3, 5 i 30. Na slikama 4.6, 4.7, 4.8, 4.9 su prikazani uzorci dobijenihm modela.

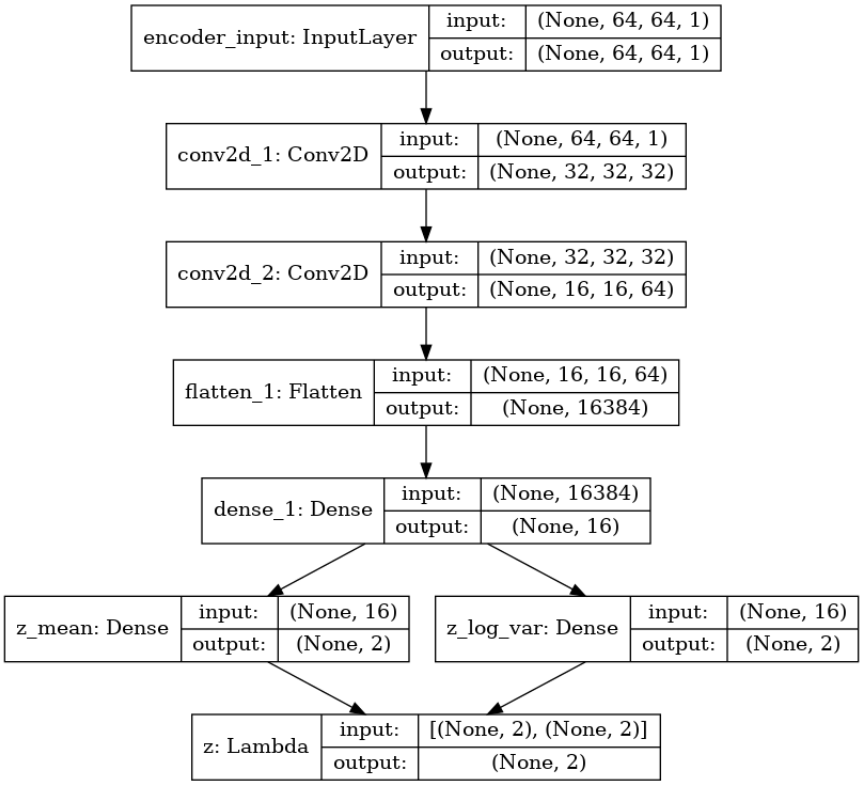
Može se zaključiti da su dobijena slova uglavnom prepoznatljiva, a primeri poseduju više različitih stilova koje je autoenkoder uspeo da nauči iz skupa za obučavanje.



Slika 4.1: Uzorak iz skupa `ftS`



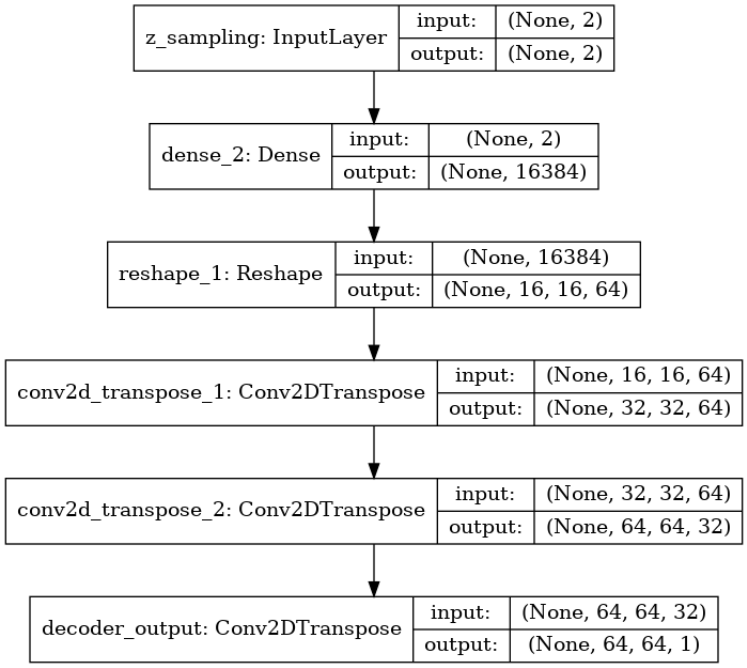
Slika 4.2: Arhitektura autoenkodera



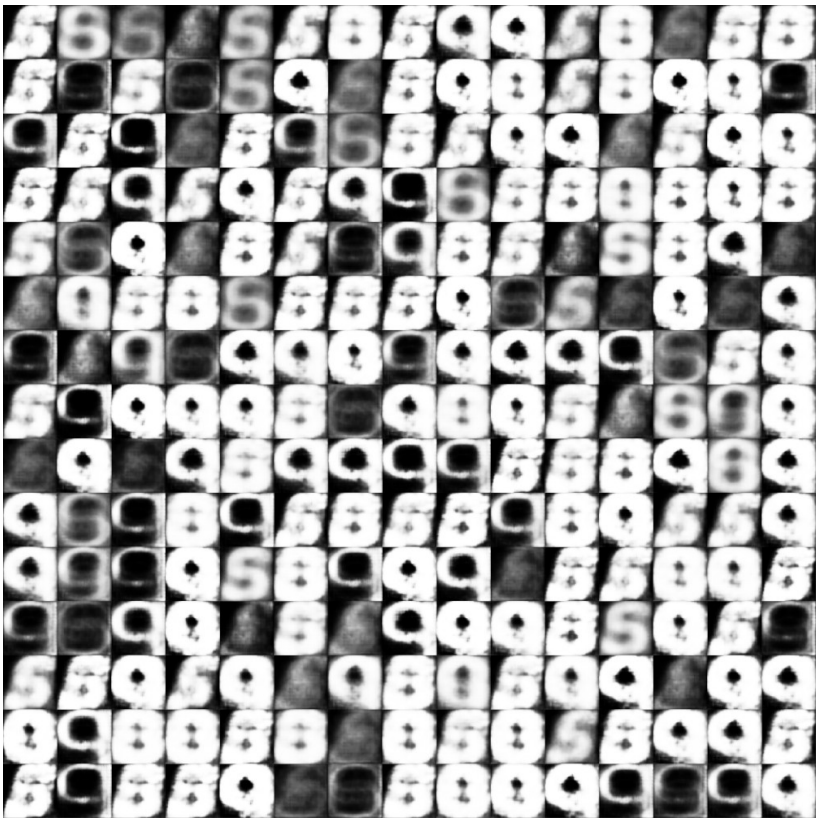
Slika 4.3: Arhitektura enkodera

4.5 Interpretabilnost modela

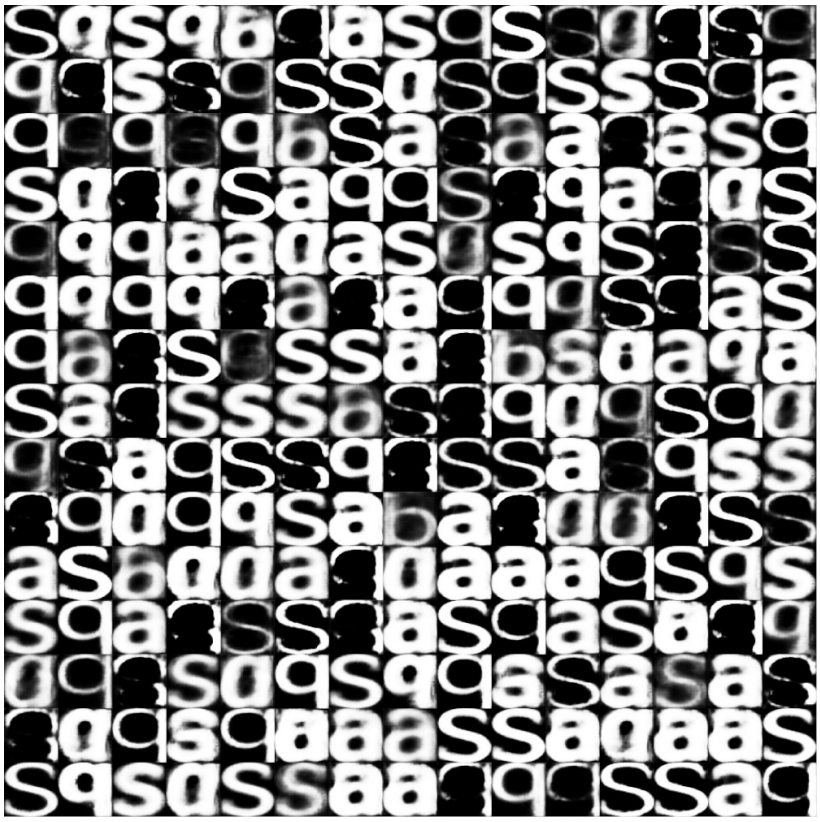
Interpretabilnost rezultata može delovati trivijalno usled mogućnosti da se slika vizuelno oceni, ali treba biti pažljiv jer se može desiti da se autoenkoder preprilagodio podacima i da su slike koje generiše zapravo replike slika iz skupa za obučavanje. Jedan od načina da se proveri da li je došlo do preprilagođavanja je generisanjem slika, ali tako što se za vrednosti u latentnom prostoru uzimaju bliske vrednosti parametara i proverava da li su dobijene slike koje liče, odnosno da li dolazi do interpolacije ili se dobijaju neprepoznatljivi oblici. Na slici 4.10 je prikazana dobijena slika kada se variraju vrednosti za vektor latentnog prostora. U konkretnom primeru dimenzija latentnog prostora postavljena je na 2 usled efektnije vizuelizacije, a na x i y osi su vrednosti iz intervala $[-200, 200]$ koje su korišćene da se dobiju generisane slike za slovo `a`. Na slici se jasno vidi da je autoenkoder naučio da interpolira između slova iz skupa za obučavanje.



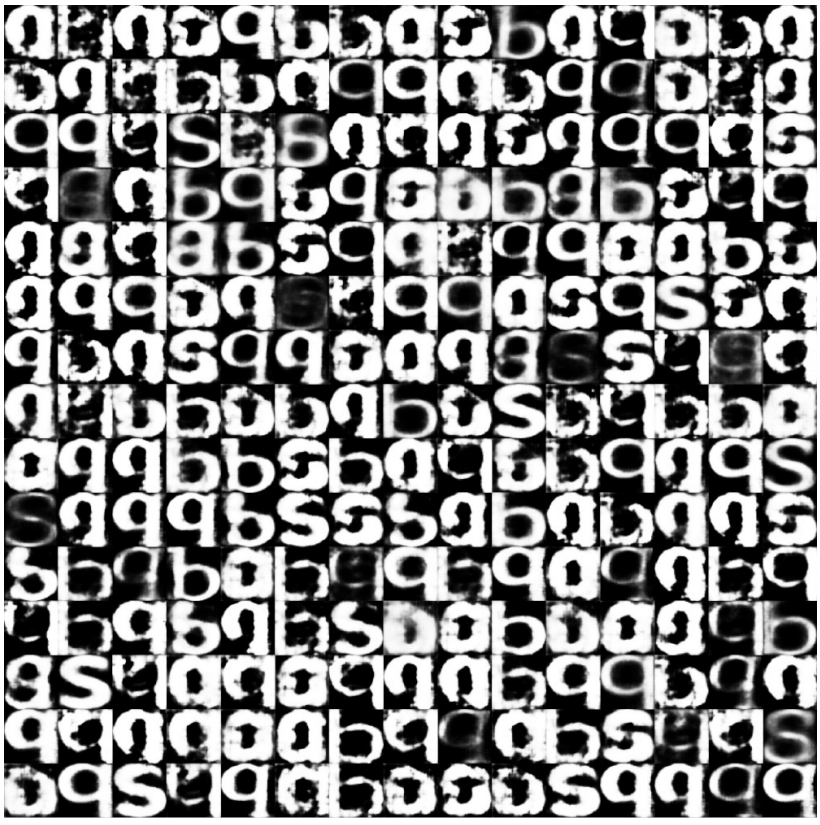
Slika 4.4: Arhitektura dekodera



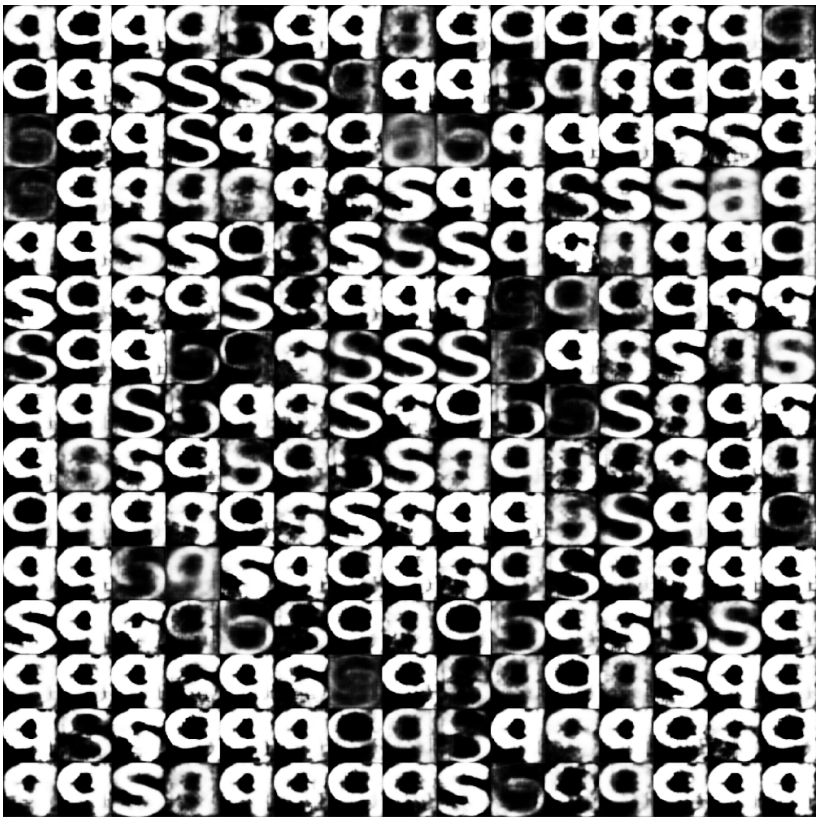
Slika 4.5: Rezultati dobijeni korišćenjem nekoliko loših fontova u skupu podataka



Slika 4.6: Generisani rezultati za latentnu dimenziju 2

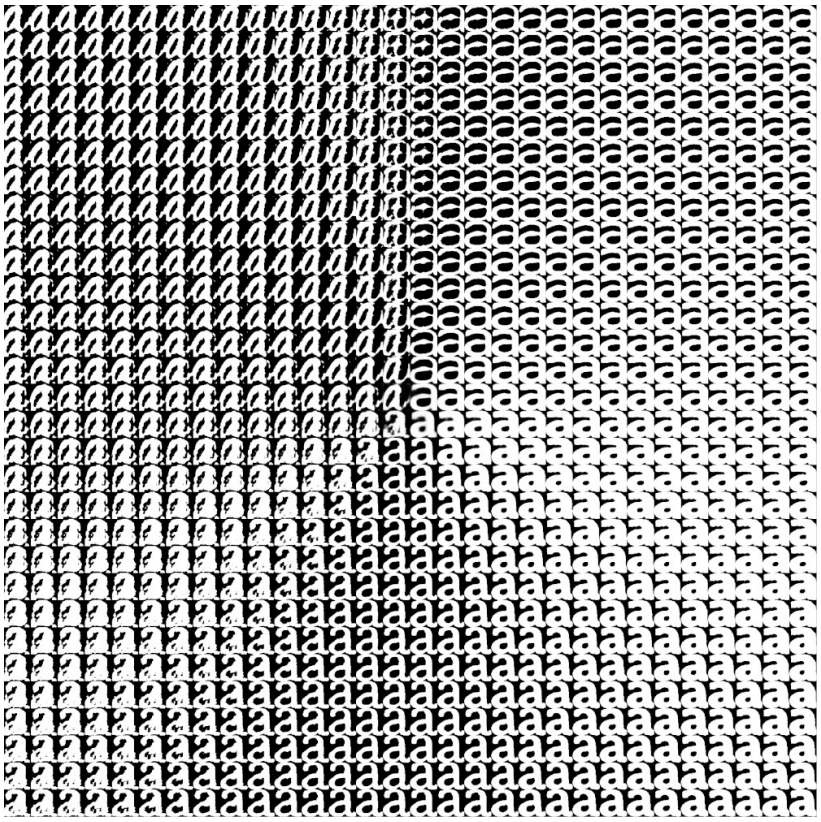


Slika 4.7: Generisani rezultati za latentnu dimenziju 3



Slika 4.8: Generisani rezultati za latentnu dimenziju 5

Slika 4.9: Generisani rezultati za latentnu dimenziju 30



Slika 4.10: Interpolacija slova a

4.6 Moguća poboljšanja i dalji rad

Problem generisanja fontova nije toliko zastupljen u oblasti mašinskog učenja i još uvek ne postoji neko rešenje koje je dovoljno kvalitetno i stabilno da bi se moglo iskoristiti u softveru.

Eksperiment je pokazao da je moguće generisati dovoljno oštre slike koristeći varijacioni autoenkoder i da je moguće vršiti interpolaciju između fontova što su bili i postavljeni ciljevi sa početka rada na problemu.

Jedan od daljih pravaca bio bi nastavak rada na autoenkoderu, ali ovaj put koristeći uslovni konvolucioni varijacioni autoenkoder. Time bi se omogućilo da se u autoenkoder doda informacija o tome koje slovo je potrebno generisati čime bi teorijski bilo moguće vršiti generisanje različitih slova u istom stilu.

Generativne suparničke mreže su poznate među generativnim modelima mašinskog učenja ali su i poznate kao neprijatne za obučavanje usled simultanog obučavanja dve neuronske mreže (generator i diskriminator). Pristup koji koristi generativne suparničke mreže je definitivno jedan od daljih pravaca koje vredi istraživati.

Pristup koji takođe treba razmotriti je korišćenje reprezentacije fonta u obliku Bezijerove krive umesto reprezentacije u obliku slike što je takođe jedan od daljih koraka u istraživanju. Reprezentacija u obliku slike je pogodna jer se mogu primeniti razne metode za rad na slikama, ali sa sobom donosi gubitak informacije i zahtev sa velikom količinom memorije prilikom generisanja skupa podataka i obučavanja modela.

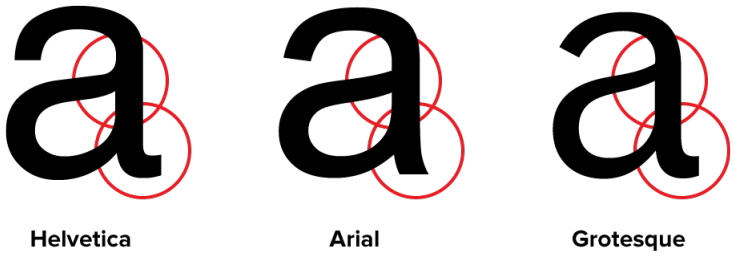
Glava 5

Vizuelizacija fontova

Vizuelizacija fontova predstavlja problem u kojem je potrebno izvršiti grafički prikaz fontova tako da su fontovi grupisani po svojoj sličnosti. Uređivanje fontova u zavisnosti od njihove sličnosti omogućava da se korisniku olakša odabir i pretraga fontova. Problem vizuelizacije fontova opisan je u ostatku teksta zajedno sa nekim od prethodnih rešenja na temu. Nakon toga je dat predlog rešenja zasnovan na korišćenju konvolutivnog variacionog autoenkodera, kao i dobijena vizuelizacija iz skupa za obučavanje.

5.1 Postavka problema

Procenjuje se da postoji preko 500000 fontova dostpunih na vebu[1], a nije retkost da grafički dizajner ima hiljade fontova instaliranih na sistemu na kojem se kreira grafički sadržaj. Velika količina fontova donosi izazove u odabiru fontova pri procesu izrade grafičkog sadržaja jer su fontovi u programima koji se koriste skoro uvek sortirani po imenu. Time je dizajner praktično ograničen na linearnu pretragu fontova za slučaj da nije siguran koji font da iskoristi ili kako se zove font koji želi da koristi. Javlja se potreba za određenom organizacijom fontova kako bi se olakšala pretraga. Na slici 5.1 prikazane su verzije slova `a` za fontove `Helvetica`, `Arial` i `Grotesque`. Može se primeniti da prikazana slova izuzetno liče, pogotovo verzija za fontove `Helvetica` i `Arial` i zaista, pri grafičkom dizajnu, dešava se da dizajner koristi neke od ovih fontova kao zamenu za drugi. Može se zaključiti da je pri organizaciji fontova potrebno obezbediti da slične reprezentacije slova budu *blizu* jedno drugom.



Slika 5.1: Nekoliko sličnih varijanti slova `a`.

[1]Procena preuzeta sa veb strane `www.quora.com`.

Potrebno je skup fontova vizuelizovati u nekom prostoru V tako da vizuelna sličnost fontova utiče na rastojanje reprezentacija tih fontova u tom prostoru. Ukoliko su fontovi vizuelno slični, onda će njihova reprezentacije biti na manjem rastojanju u odnosu na fontove koji su vizuelno različiti. Ukoliko je poznat ovakav prostor reprezentacija fontova, onda je moguće drugačije organizovati pregledanje instaliranih fontova u računarskom softveru kao što je na primer `Gimp` ili `Inkscape`[2]. Na primer jedna od ideja je da korisnik kroz softver može da odabere font f, a potom da se ostali fontovi sortiraju po rastojanju od fonta f u prostoru V.

5.2 Prethodni pristupi

Ukoliko se koristi reprezentacija u vidu slike, fontovi su podaci visoke dimenzionalnosti. Na primer ukoliko je rezolucija slike 64×64, ona se može posmatrati kao vektor iz prostora $\mathbb{R}^{4096}$. Usled dimenzionalnosti podataka u ovoj reprezentaciji njihova vizuelizacija postaje praktično nemoguća. Jedan od pristupa problemu prikazan je u radu [55] gde je osnovna ideja primeniti neki od algoritama za smanjivanje dimenzionalnosti podataka kao što su analiza glavnih komponenti i `Isomap` [56] kako bi se za podatke dobila reprezentacija podataka u prostoru koji se može vizuelizovati.

5.3 Predloženi pristup problemu

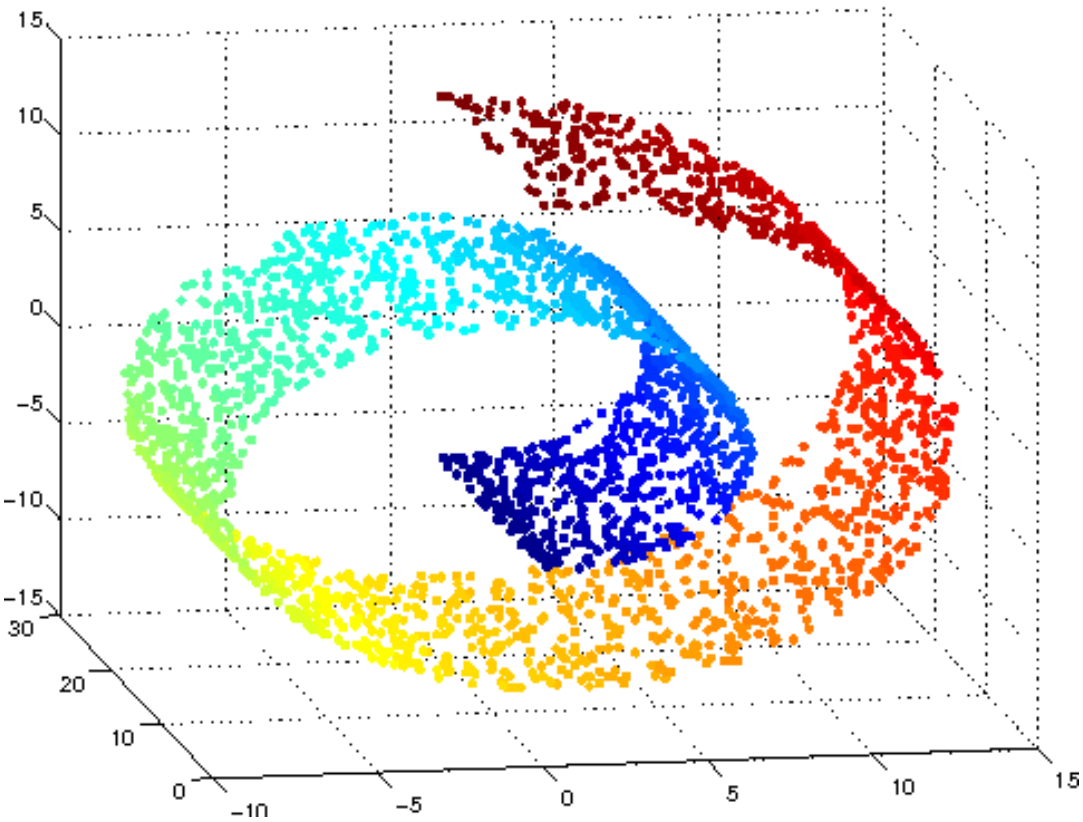
Kao što je pomenuto, podaci koji reprezentuju fontove su visoke dimenzionalnosti, a česta je situacija da podaci visoke dimenzionalnosti poseduju nelinearne zavisnosti koje metode koje traže linearne zavisnosti (poput metoda glavnih komponenti) nisu u stanju da pronađu. Usleg toga, javlja se problem nelinearnog smanjivanja dimenzionalnosti podataka u literaturi poznat kao *učenje mnogostrukosti* (eng. *manifold learning*) [10, 31]. Na slici 5.2 prikazani su podaci u kojima postoji nelinearna pravilnost.

Jedan od vodećih pristupa u rešavanju problema učenja mnogostrukosti je korišćenje autoenkodera koji usled nelinearnih aktivacionih funkcija ima mogućnost da pronađe nelinearne zavisnosti u podacima [31]. Latentni prostor u koji enkoder preslikava ulaz se može iskoristiti kao prostor reprezentacije V pomenut u delu 5.1. U poglavlju 4 je opisan razvijeni varijacioni autoenkoder za generisanje fontova i on se može iskoristiti za vizuelizaciju fontova. Korišćena je varijanta čija je vrednost dimenzije latentnog prostora postavljena na 2 kako bi se dobile dvodimenzione reprezentacije ulaznih podataka. Iz skupa podataka su uzete reprezentacije slova `a` za fontove iz skupa $\mathcal{F}'$ i izračunate njihove reprezentacije $z \in \mathbb{R}^2$ koristeći enkoder razvijenog enkodera. Rastojanje nad fontovima se meri izračunavanjem funkcije rastojanja između dobijenih vektora z.

5.4 Eksperimentalni rezultati

Na slici 5.3 prikazana je vizuelizacija fontova iz skupa $\mathcal{F}'$ u latentnom prostoru variacionog autoenkodera. Koordinate podataka predstavljaju vrednosti vektora $z \in \mathbb{R}^2$, a prikazane slike su slike iz skupa podataka koji je korišćen da se obuči autoenkoder.

[2]Softver otvorenog koda za kreiranje vektorskih slika.

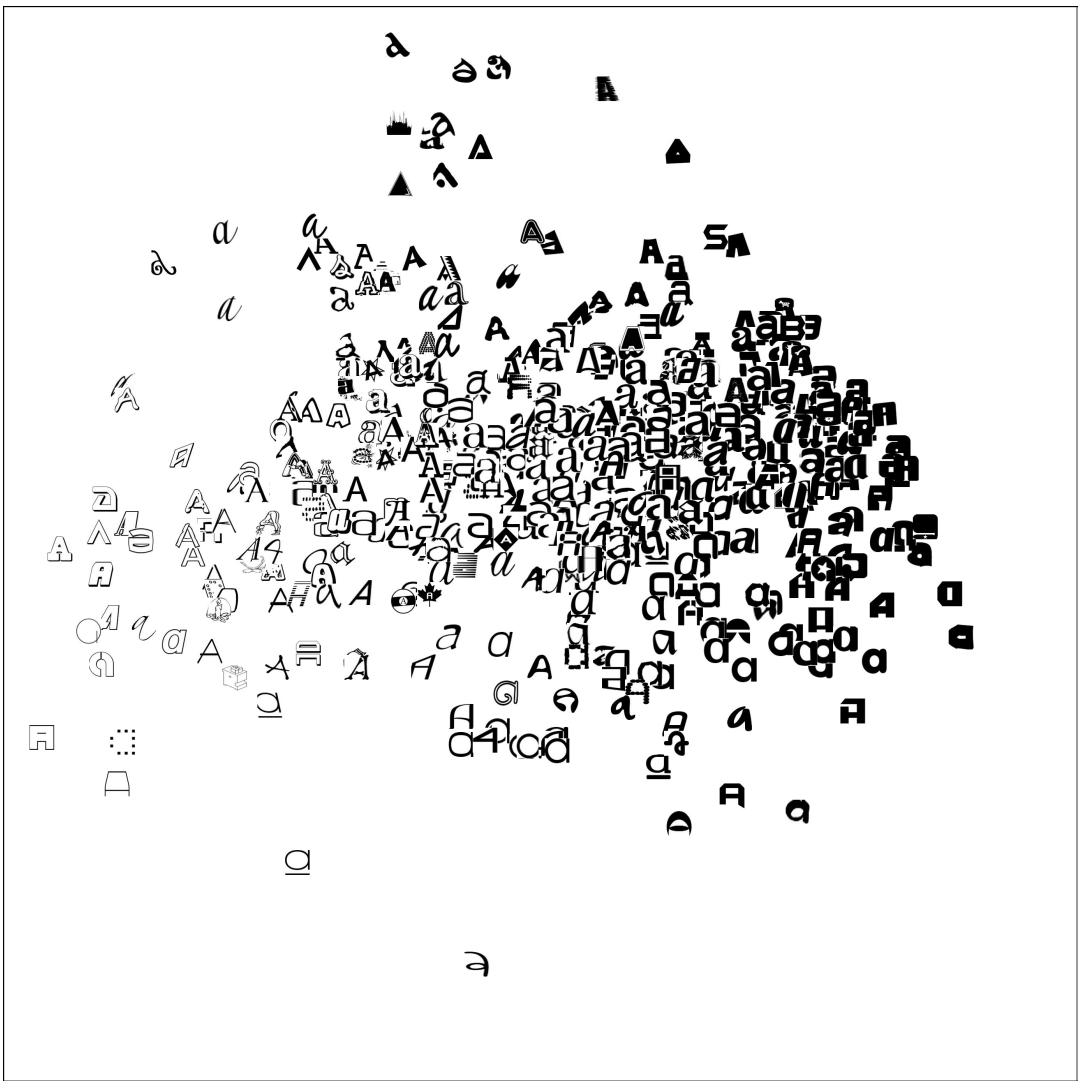


Slika 5.2: Ilustracija podataka u kojima postoji nelinearna pravilnost.

Može se primetiti da su slike sa tankim linijama grupisane u levoj polovini slike, a slike sa debljim linijama u desnoj polovini slike. Mogu se uočiti instance koje poseduju neku netipičnu reprezentaciju slova `a` bliske ivicama slike oko kojih nema puno suseda. Posebno je zaniljivo primetiti da su reprezentacije slova `a` u obliku lista i kruga kao vrlo retke u tipografiji završile pri sredini prostora ali međusobno blisko, verovatno usled karakteristike da slovo sadrže unutar nekog predmeta.

5.5 Moguća poboljšanja i dalji rad

Dalji razvoj prikazanog pristupa tesno je vezan sa razvojem sistema iz poglavlja 4. Poboljšanja autoenkodera direktno bi doprinela poboljšanju kvaliteta vizuelizacije. Korišćenje više fontova i bogatijeg skupa podataka za obučavanje autoenkodera takođe bi doprineli kvalitetu dobijenih rezultata. Jedan od daljih pravaca razvoja sistema je u njegovoj integraciji u neki od programa koji se koriste u oblasti grafičkog dizajna kao što su `Gimp` i `InkScape`, ili integraciji u program za uređivanje teksta kao što je `Libre Office`.



Slika 5.3: Ilustracija fontova.

Glava 6

Veb aplikacija

Za potrebe demonstracije modela opisanih u poglavljima 3 i 4 razvijena je veb aplikacija `Fontool` koja omogućava interaktivno korišćenje razvijenih sistema. Dat je kratak pregled korišćenih biblioteka, osnovnih delova veb aplikacije, kao i opis funkcionalnosti koje aplikacija pruža.

6.1 Korišćene biblioteke

Za razvoj aplikacije `Fontool` korišćene su biblioteke:

- `Flask` (serverski deo aplikacije - jezik `Python`) [57],
- `Bootstrap` (vizuelni deo aplikacije - `JavaScript`, `CSS` i `HTML`) [58],
- `JQuery` (klijentski deo aplikacije - `JavaScript`) [59],
- `List.js` (klijentski deo aplikacije - `JavaScript`) [60].

6.1.1 Biblioteka `Flask`

`Flask` je obljavljen 2010. godine i intenzivno se razvija i koristi od tada. U pitanju je mikro radni okvir (eng. *micro framework*) koji korisniku pruža ograničen skup funkcionalnosti u odnosu na popularne biblioteke i radne okvire za veb kao što su `Django`, `Laravel` i `Symphony`. `Flask` koristi radni okvir `Jinja2` za pisanje šablona (eng. *template engine*). Za slučaj da su korisniku potrebne dodatne funkcionalnosti poput objektno relacionog mapiranja podataka, rada sa bazama podataka i slično, potrebno je instalirati i koristiti neku dodatnu biblioteku poput `SQLAlchemy` ili `psycopg2`. Logo biblioteke prikazan je na slici 6.1.

O kvalitetu biblioteke svedoči i to što je koriste servisi kao što su `LinkedIn`[1] i `Pinterest`[2].

6.1.2 Biblioteka `Bootstrap`

Biblioteka `Bootstrap` inicijalno je objavljena 2011. godine u sa ciljem da olakša kreiranje responzivnih veb sadržaja. Njen razvoj od tada intenzivno napreduje i trenutno predstavlja jednu od važnijih biblioteka za razvoj veb stranica i veb aplikacija.

[1] Poslovna društvena mreža u vlasništvu kompanije `Microsoft`.
[2] Mreža posvećena pretrazi i deljenju medijskih sadržaja, prvenstevno slika.

Slika 6.1: Logo biblioteke `Flask`.

Razlozi za njenu široku primenu u industriji su kvalitet, jednostavnost, kao i odlična zajednica i dokumentacija. Na slici 6.2 prikazan je logo biblioteke.

Slika 6.2: Logo biblioteke `Bootstrap`.

6.1.3 Biblioteka `JQuery`

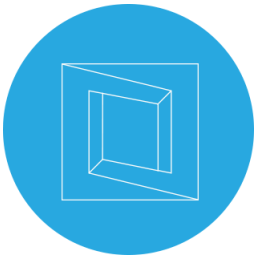
Biblioteka `JQuery` objavljena 2007. godine već dugi niz godina je izuzetno popularna za razvoj klijentskih delova veb aplikacija i stranica. Po statistikama[3] za 2018. godinu, od 10 miliona popularnih veb stranica preko 70% koristi `JQuery` na neki način. Originalnu popularnost biblioteka je zaslužila dizajnom interfejsa sa korišćenje koji je drastično smanjio i pojednostavio potrebnu količinu `JavaScript` koda za tipične klijentske skriptove. Na primer, biblioteka je omogućila da se bilo koji `HTML` element dohvati direktno iz jezika `JavaScript` koristeći `CSS` selektore. Jezik `JavaScript` je u poslednjih nekoliko godina uneo veliki broj novina i dodataka tako da je biblioteka `JQuery` izgubila na značaju u profesionalnom razvoju klijentskih delova veb aplikacija da bi je danas zamenile biblioteke `Angular` i `React`. U implementaciji izabrana je biblioteka `JQuery` usled toga što nije bilo potrebe za većinom funkcionalnosti koje nude `Angular` i `React`. Ukoliko aplikacija bude dalje razvijana sa naprednijim mogućnostima verovatno bi bila korišćena neka od navedene dve biblioteke. Na slici 6.3 prikazan je logo biblioteke `JQuery`.

Slika 6.3: Logo biblioteke `JQuery`.

[3]Preuzeto sa: `https://w3techs.com/technologies/overview/javascript_library/all`.

6.1.4 Biblioteka `List.js`

U razvoju softvera česta je praksa da se prikaz podataka jasno odvoji od logike koja obrađuje podatke. Na ovaj način se postiže da za podatke postoji nekoliko različitih pogleda. Tipičan primer je program za obradu tabelarnih podataka `Libreoffice Calc` koji omogućava da za tabelarno zadate podatke korisnik vidi više različitih vizuelizacija. Osim prikaza, pogled nad podacima često treba da omogući i sortiranje, pretragu i inkrementalno izlistavanje podataka pri čemu te operacije ne vrše promenu stvarnih podataka. Biblioteka `List.js` je minimalistična biblioteka koja omogućava upravo ovakav način implementacije logike. Na slici 6.4 prikazan je logo biblioteke.



Slika 6.4: Logo biblioteke `List.js`.

6.2 Funkcionalnosti veb aplikacije

Aplikacija korisniku omogućava da interaktivno koristi razvijene sisteme. Kako je izračunavanje neophodno za neuronske mreže smešteno na server, aplikacija se može integrisati u aplikaciju za mobilne uređaje bez posebnih hardverskih zahteva za same uređaje. Kako je grafički interfejs responzivan, aplikacija se može direktno koristiti sa prenosivih uređaja kroz veb pregledač ili kroz nativnu aplikaciju uz podršku komponente `WebView`[4]. Ovakav način rada zahteva da je omogućena mrežna konekcija.

Na slici 6.6 prikazan je početni ekran, a na slici 6.7 ekran koji omogućava korisniku da pretraži skup fontova koji su korišćeni u obučavanju modela. Slike 6.8 i 6.9 ilustruju proces vizuelnog prepoznavanja fonta, dok slika 6.10 prikazuje vizuelno generisanje fonta.

6.2.1 Neki detalji implementacije

Kako su neuronske mreže i serverski deo veb aplikacija razvijeni u jeziku `Python`, korišćenje funkcionalnosti neuronskih mreža u okviru serverskog dela aplikacija nije zahtevalo puno posla. Bilo je neophodno implementirati klase omotače `wrapper class` koje predstavljaju apstrakciju nad razvijenim modelima i olakšavaju njihovo korišćenje iz veb aplikacije. Implementirane su dve klase, `FontClassifier` i `FontGenerator` čijim instanciranjem se dobija objekat kroz koji je moguće vršiti klasifikaciju i generisanje fontova. U daljem razvoju dovoljno je povezati serverski deo aplikacije sa navedenim klasama i time izbeći pisanje bilo kakve logike na serverskom delu koja ima veze sa mašinskim učenjem.

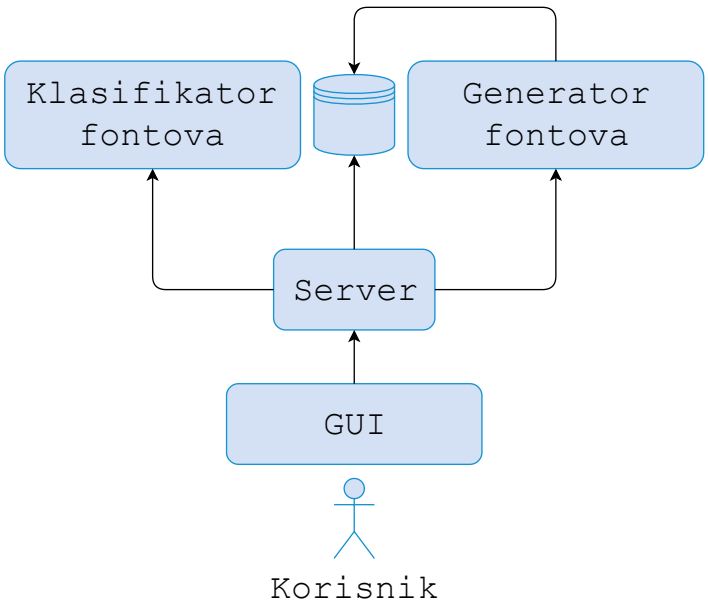
[4]Komponenta koja omogućava prikazivanje veb stranica unutar softvera, bez korišćenja zasebnog veb pregledača.

Klasa `FontClassifier` nudi metod `predict(path_to_image)` koji učitava obučeni model, učitava sliku sa putanje `path_to_image`, vrši klasifikaciju i vraća odgovor kao rečnik koji ima sledeće ključeve:

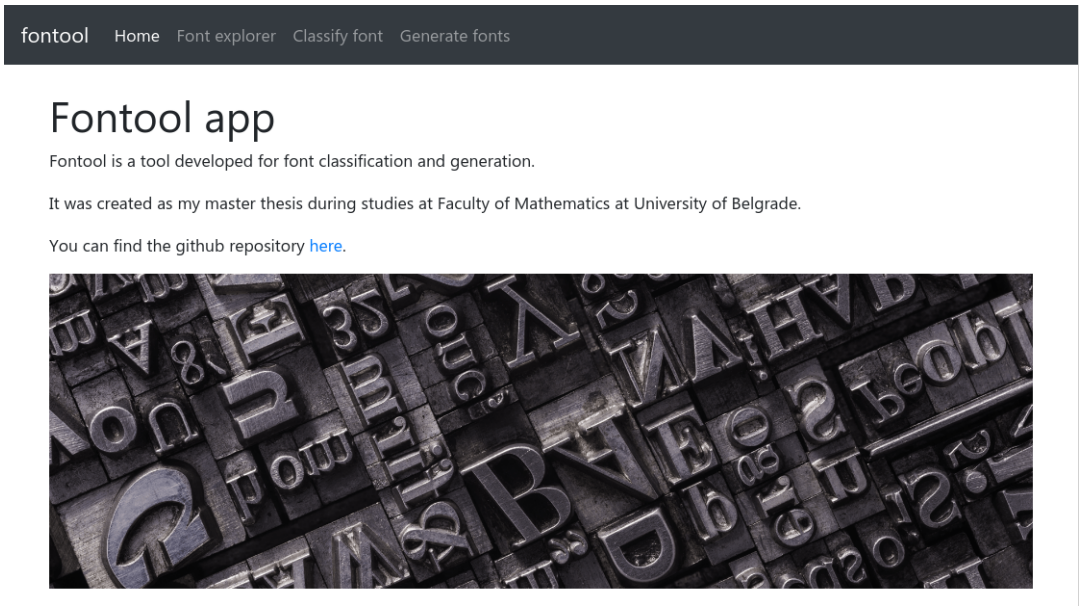
- `label`: broj koji označava indeks fonta u poslednjem sloju mreže,
- `name`: ime fonta dobijenog klasifikacijom,
- `results`: niz koji sadrži imena svih fontova i ocene verovatnoće,
- `img_path`: putanju do slike koja je proseđena kao arugment.

Klasa `FontGenerator` nudi metod `generate(z_vector)` koji učitava obučeni model i vrši generisanje slike koristeći kao ulaz za dekoder vektor `z_vector`. Nakon što se generiše slika, ona se čuva na serveru i kao odgovor se vraća putanja do generisane slike.

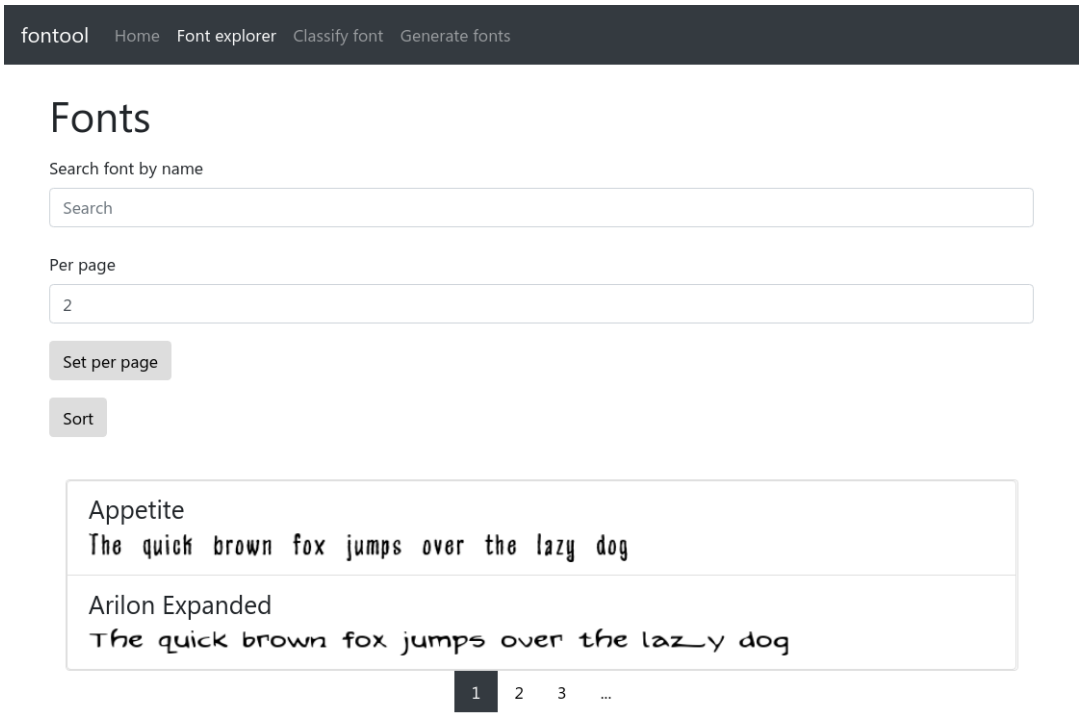
Na slici 6.5 prikazana je arhitektura veb aplikacije. Moduli *Klasifikator fontova* i *Generator fontova* su moduli koji kroz prethodno navedene klase omogućavaju serverskom delu koda da koristi funkcionalnosti razvijenih modela. Sa GUI (eng. *graphical user interface*) je označen grafički korisniki interfejs dostupan korisniku.



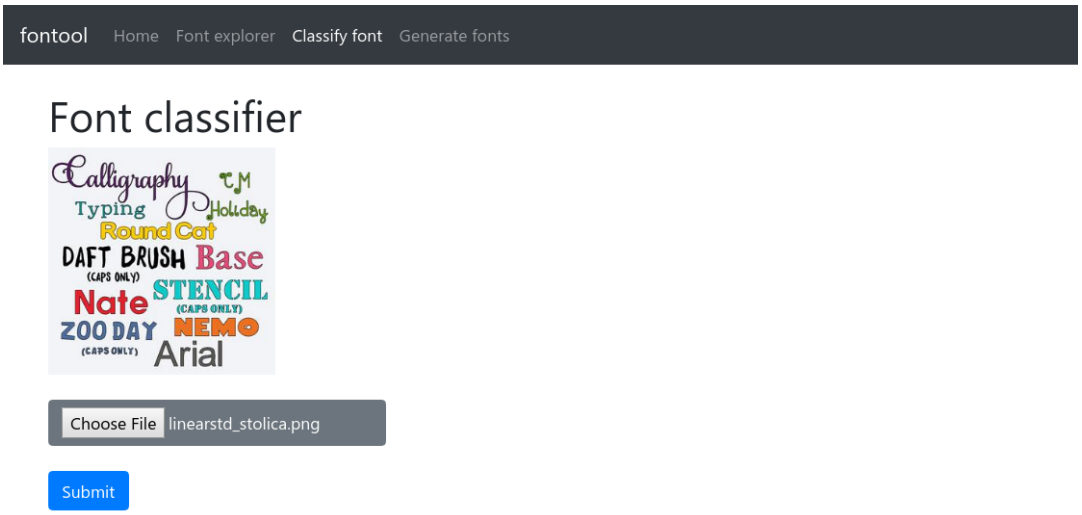
Slika 6.5: Dijagram arhitekture veb aplikacije.



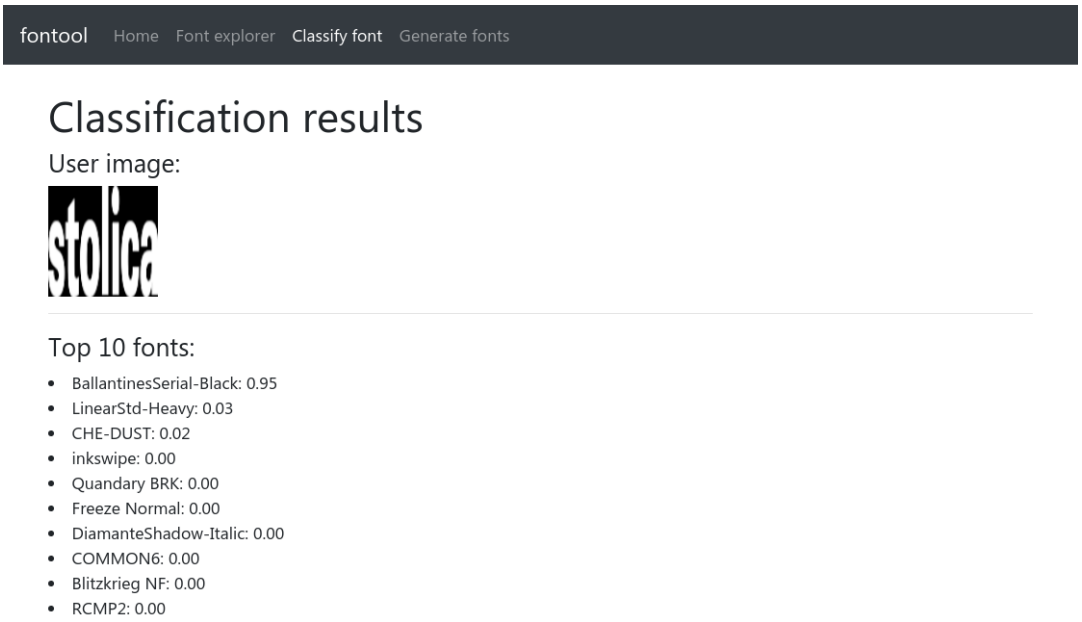
Slika 6.6: Početna stranica veb aplikacije.



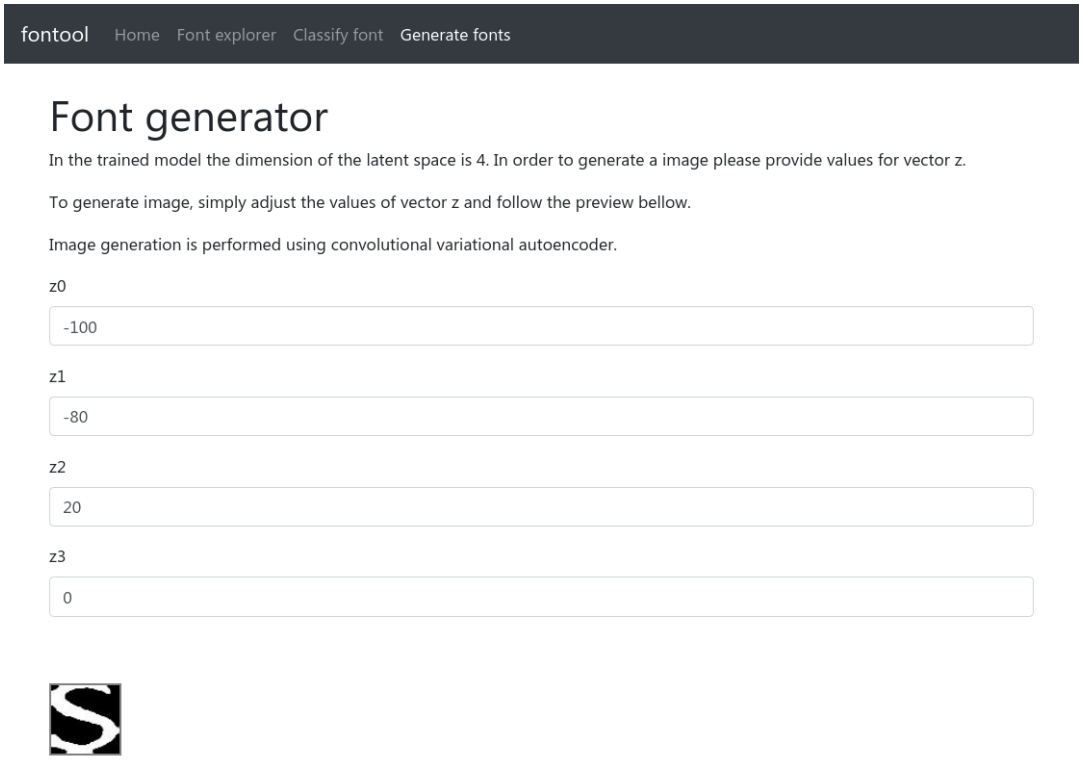
Slika 6.7: Pregled i pretraga fontova iz korišćenog skupa podataka.



Slika 6.8: Vizuelno prepoznavanje fontova - odabir slike.



Slika 6.9: Vizuelno prepoznavanje fontova - dobijeni rezultati.



Slika 6.10: Forma za generisanje slova.

Glava 7

Zaključak

Oblast mašinskog učenja pronalazi primenu u raznim oblastima, a između ostalog i u oblastima tipografije i grafičkog dizajna. Iako su trenutna dostignuća oblasti daleko od toga da zamene ljude u kreativnom procesu, ipak su dovoljno dobra da ljudima pruže podršku i olakšaju neke delove posla.

U tezi su razmatrana tri problema, vizuelno prepoznavanje fontova `VFR`, vizuelno generisanje fontova `VFG` i vizuelizacija fontova za koje su prikazana ukratko prethodna dostignuća, predložena rešenja i dobijeni rezultati. Razvijena je i veb aplikacija `Fontool` koja omogućava interaktivno korišćenje razvijenih modela.

Problem `VFR` sveden je na problem klasifikacije slika i rešavan koristeći konvolutivnu neuronsku mrežu. Mreža se pokazala kao odličan način za rešavanje problema jer je uspela da postigne impresivnu preciznost 0.993. Konvolutivna mreža se dalje potencijalno može unaprediti korišćenjem i evaluacijom naprednijih arhitektura za klasifikaciju slika kao i obučavanjem modela na većem skupu podataka.

Problem `VFG` sveden je na problem generisanja slika čije je rešenje traženo uz pomoć konvolutivnog variacionog autoenkodera. Dobijene slike nisu savršene ali su ipak slova na njima prepoznatljiva i oštra. Problem vizuelizacije fontova je prirodno naveo na rešavanje pomoću razvijenog autoenkodera usled potrebe za smanjenjem dimenzionalnosti podataka. U dobijenoj ilustraciji fontova može se zaključiti da su fontovi preslikani u latentni prostor u kojem se primećuju da su neki vizuelno slični fontovi međusobno blizu. Kako je u pitanju i dalje otvoren problem, postoji više smerova u kojima se istraživanje može nastaviti. Uslovni varijacioni autoenkoder bi potencijalno omogućio da se dodatno nametne ograničenje o tome koje slovo će model generisati čime bi se omogućilo da mreža generiše veliki broj različitih slova u istom stilu. Osim toga, generativne suparničke mreže su pokazale odlične rezultate u generativnim problemima, pa potencijalno mogu doprineti poboljšanju rešenja za problem `VFG`.

Implementirana je veb aplikacija `Fontool` koja omogućava interaktivno korišćenje razvijenih sistema. Kako je aplikacija responzivna, može se bez mnogo posla prevesti i u mobilnu aplikaciju za korišćenje na prenosivim računarima i mobilnim uređajima.

Može se zaključiti da su predložena rešenja dala zadovoljavauće rezultate, ali i stvorila prostor za dalje istraživanje i poboljšanja.

Bibliografija

[1] Warren S. McCulloch and Walter Pitts. "Neurocomputing: Foundations of Research". In: ed. by James A. Anderson and Edward Rosenfeld. Cambridge, MA, USA: MIT Press, 1988. Chap. A Logical Calculus of the Ideas Immanent in Nervous Activity, pp. 15–27. ISBN: 0-262-01097-6. URL: http://dl.acm.org/citation.cfm?id=65669.104377.

[2] P. J. Werbos. "Applications of Advances in Nonlinear Sensitivity Analysis". In: *Proceedings of the 10th IFIP Conference, 31.8 - 4.9, NYC*. 1981, pp. 762–770.

[3] Y. LeCun. "Learning Processes in an Asymmetric Threshold Network". In: *Disordered systems and biological organization*. Ed. by E. Bienenstock, F. Fogelman-Soulié, and G. Weisbuch. Les Houches, France: Springer-Verlag, 1986, pp. 233–240.

[4] Y. LeCun. "A theoretical framework for Back-Propagation". In: *Proceedings of the 1988 Connectionist Models Summer School*. Ed. by D. Touretzky, G. Hinton, and T. Sejnowski. CMU, Pittsburgh, Pa: Morgan Kaufmann, 1988, pp. 21–28.

[5] Tom M. Mitchell. *Machine Learning*. New York: McGraw-Hill, 1997. ISBN: 978-0-07-042807-2.

[6] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. New York, NY, USA: Springer New York Inc., 2001.

[7] Olivier Chapelle, Bernhard Schlkopf, and Alexander Zien. *Semi-Supervised Learning*. 1st. The MIT Press, 2010. ISBN: 0262514125, 9780262514125.

[8] Anđelka Zečević Mladen Nikolić. *Mašinsko učenje*. http://ml.matf.bg.ac.rs/readings/ml.pdf. 2017.

[9] Anđelka Zečević Mladen Nikolić. *Naučno izračunavanje*. http://ni.matf.bg.ac.rs/materijali/ni.pdf. 2016.

[10] Yunqian Ma and Yun Fu. *Manifold Learning Theory and Applications*. 1st. Boca Raton, FL, USA: CRC Press, Inc., 2011. ISBN: 1439871094, 9781439871096.

[11] Shumeet Baluja. "Learning Typographic Style". In: *CoRR* abs/1603.04000 (2016). arXiv: 1603.04000. URL: http://arxiv.org/abs/1603.04000.

[12] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006. ISBN: 0387310738.

[13] Pierre Baldi. "Autoencoders, Unsupervised Learning and Deep Architectures". In: *Proceedings of the 2011 International Conference on Unsupervised and Transfer Learning Workshop - Volume 27*. UTLW'11. Washington, USA: JMLR.org, 2011, pp. 37–50. URL: http://dl.acm.org/citation.cfm?id=3045796.3045801.

[14] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems 25*. Ed. by F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger. Curran Associates, Inc., 2012, pp. 1097–1105. URL:

http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

[15] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. "DeepFace: Closing the Gap to Human-Level Performance in Face Verification". In: *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition*. CVPR '14. Washington, DC, USA: IEEE Computer Society, 2014, pp. 1701–1708. ISBN: 978-1-4799-5118-5. DOI: 10.1109/CVPR.2014.220. URL: https://doi.org/10.1109/CVPR.2014.220.

[16] Fei Wang, Mengqing Jiang, Chen Qian, Shuo Yang, Cheng Li, Honggang Zhang, Xiaogang Wang, and Xiaoou Tang. "Residual Attention Network for Image Classification". In: *CoRR* abs/1704.06904 (2017). arXiv: 1704.06904. URL: http://arxiv.org/abs/1704.06904.

[17] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. "Mastering the Game of Go with Deep Neural Networks and Tree Search". In: *Nature* 529.7587 (Jan. 2016), pp. 484–489. DOI: 10.1038/nature16961.

[18] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy P. Lillicrap, Karen Simonyan, and Demis Hassabis. "Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm". In: *CoRR* abs/1712.01815 (2017). arXiv: 1712.01815. URL: http://arxiv.org/abs/1712.01815.

[19] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. *Playing Atari with Deep Reinforcement Learning*. cite arxiv:1312.5602Comment: NIPS Deep Learning Workshop 2013. 2013. URL: http://arxiv.org/abs/1312.5602.

[20] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. "Distributed Representations of Words and Phrases and their Compositionality". In: *Advances in Neural Information Processing Systems 26*. Ed. by C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger. Curran Associates, Inc., 2013, pp. 3111–3119. URL: http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf.

[21] Alex Graves. "Generating Sequences With Recurrent Neural Networks." In: *CoRR* abs/1308.0850 (2013). URL: http://dblp.uni-trier.de/db/journals/corr/corr1308.html#Graves13.

[22] Thang Luong, Hieu Pham, and Christopher D. Manning. "Effective Approaches to Attention-based Neural Machine Translation". In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal: Association for Computational Linguistics, 2015, pp. 1412–1421. DOI: 10.18653/v1/D15-1166. URL: http://www.aclweb.org/anthology/D15-1166.

[23] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. "Generative Adversarial Nets". In: *Advances in Neural Information Processing Systems 27*. Ed. by Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger. Curran Associates, Inc., 2014, pp. 2672–2680. URL: http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf.

[24] Alec Radford, Luke Metz, and Soumith Chintala. "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks". In: *CoRR* abs/1511.06434 (2015). arXiv: 1511.06434. URL: http://arxiv.org/abs/1511.06434.

[25] Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alexander Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. "WaveNet: A Generative Model for Raw Audio". In: *Arxiv*. 2016. URL: https://arxiv.org/abs/1609.03499.

[26] Mladen Nikolic, Filip Maric, and Predrag Janicic. "Simple Algorithm Portfolio for SAT". In: *CoRR* abs/1107.0268 (2011). arXiv: 1107.0268. URL: http://arxiv.org/abs/1107.0268.

[27] Pierre Baldi and Søren Brunak. *Bioinformatics: The Machine Learning Approach*. 2nd. Cambridge, MA, USA: MIT Press, 2001. ISBN: 0-262-02506-X.

[28] Pedro Larrañaga, Borja Calvo, Roberto Santana, Concha Bielza, Josu Galdiano, Iñaki Inza, José A. Lozano, Rubén Armañanzas, Guzmán Santafé, Aritz Pérez, and Victor Robles. "Machine learning in bioinformatics". In: *Briefings in Bioinformatics* 7.1 (2006), pp. 86–112. DOI: 10.1093/bib/bbk007. eprint: /oup/backfile/content_public/journal/bib/7/1/10.1093_bib_bbk007/2/bbk007.pdf. URL: http://dx.doi.org/10.1093/bib/bbk007.

[29] Sagar S. De, Satchidananda Dehuri, and Gi-Nam Wang. "Machine Learning for Social Network Analysis: A Systematic Literature Review". In: 8 (Jan. 2012), pp. 30–51.

[30] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. "You Only Look Once: Unified, Real-Time Object Detection". In: *CoRR* abs/1506.02640 (2015). arXiv: 1506.02640. URL: http://arxiv.org/abs/1506.02640.

[31] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[32] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". In: *J. Mach. Learn. Res.* 15.1 (Jan. 2014), pp. 1929–1958. ISSN: 1532-4435. URL: http://dl.acm.org/citation.cfm?id=2627435.2670313.

[33] Yann Lecun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE*. 1998, pp. 2278–2324.

[34] Matthew D. Zeiler and Rob Fergus. "Visualizing and Understanding Convolutional Networks". In: *CoRR* abs/1311.2901 (2013). arXiv: 1311.2901. URL: http://arxiv.org/abs/1311.2901.

[35] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: https://www.tensorflow.org/.

[36] François Chollet et al. *Keras*. https://keras.io. 2015.

[37] Francois Chollet. *Deep Learning with Python*. 1st. Greenwich, CT, USA: Manning Publications Co., 2017. ISBN: 1617294438, 9781617294433.

[38] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. "ImageNet: A Large-Scale Hierarchical Image Database". In: *CVPR09*. 2009.

[39] Yann LeCun and Corinna Cortes. "MNIST handwritten digit database". In: (2010). URL: http://yann.lecun.com/exdb/mnist/.

[40] Carlos Avilés-Cruz, Risto Rangel-Kuoppa, Mario Reyes-Ayala, Edgar Andrade González, and Rafael Escarela-Perez. "High-order statistical texture analysis - Font recognition applied". In: 26 (Jan. 2005), pp. 135–145.

[41] Min-Chul Jung, Yong-Chul Shin, and Sargur N. Srihari. "Multifont Classification using Typographical Attributes". In: *ICDAR*. IEEE Computer Society, 1999, pp. 353–356.

[42] Huanfeng Ma and David Doermann. "Gabor Filter Based Multi-class Classifier for Scanned Document Images". In: *Proceedings of the Seventh International Conference on Document Analysis and Recognition - Volume 2*. ICDAR '03. Washington, DC, USA: IEEE Computer Society, 2003, pp. 968–. ISBN: 0-7695-1960-1. URL: http://dl.acm.org/citation.cfm?id=938980.939479.

[43] R. Ramanathan, K. P. Soman, L. Thaneshwaran, V. Viknesh, T. Arunkumar, and P. Yuvaraj. "A Novel Technique for English Font Recognition Using Support Vector Machines". In: *2009 International Conference on Advances in Recent Technologies in Communication and Computing*. 2009, pp. 766–769. DOI: 10.1109/ARTCom.2009.89.

[44] Hung-Ming Sun. "Multi-Linguistic Optical Font Recognition Using Stroke Templates". In: *18th International Conference on Pattern Recognition (ICPR'06)*. Vol. 2. 2006, pp. 889–892. DOI: 10.1109/ICPR.2006.824.

[45] Yong Zhu, Tieniu Tan, and Yunhong Wang. "Font Recognition Based on Global Texture Analysis". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 23.10 (Oct. 2001), pp. 1192–1200. ISSN: 0162-8828. DOI: 10.1109/34.954608. URL: http://dx.doi.org/10.1109/34.954608.

[46] Zhangyang Wang, Jianchao Yang, Hailin Jin, Jonathan Brandt, Eli Shechtman, Aseem Agarwala, Zhaowen Wang, Yuyan Song, Joseph Hsieh, Sarah Kong, and Thomas Huang. "DeepFont: A System for Font Recognition and Similarity". In: *Proceedings of the 23rd ACM International Conference on Multimedia*. MM '15. Brisbane, Australia: ACM, 2015, pp. 813–814. ISBN: 978-1-4503-3459-4. DOI: 10.1145/2733373.2807988. URL: http://doi.acm.org/10.1145/2733373.2807988.

[47] *The CIFAR-10 dataset*. https://www.cs.toronto.edu/~kriz/cifar.html.

[48] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications". In: *CoRR* abs/1704.04861 (2017).

[49] Yue Jiang, Zhouhui Lian, Yingmin Tang, and Jianguo Xiao. "DCFont: an end-to-end deep chinese font generation system". In: *SIGGRAPH Asia Technical Briefs*. ACM, 2017, 22:1–22:4.

[50] Neill D. F. Campbell and Jan Kautz. "Learning a Manifold of Fonts". In: *ACM Trans. Graph.* 33.4 (July 2014), 91:1–91:11. ISSN: 0730-0301. DOI: 10.1145/2601097.2601212. URL: http://doi.acm.org/10.1145/2601097.2601212.

[51] Tijana Šukilović Srđan Vukmirović. *Geometrija za informatičare*. http://poincare.matf.bg.ac.rs/~tijana/geometrija/Knjiga.pdf. 2015.

[52] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005. ISBN: 026218253X.

[53] Kotaro Abe, Brian Kenji Iwana, Viktor Gösta Holmér, and Seiichi Uchida. "Font Creation Using Class Discriminative Deep Convolutional Generative Adversarial Networks". In: ().

[54] Yunchen Pu, Zhe Gan, Ricardo Henao, Xin Yuan, Chunyuan Li, Andrew Stevens, and Lawrence Carin. "Variational Autoencoder for Deep Learning of Images, Labels and Captions". In: *Advances in Neural Information Processing Systems 29*. Ed. by D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett. Curran Associates, Inc., 2016, pp. 2352–2360. URL: http://papers.nips.cc/paper/6528-variational-autoencoder-for-deep-learning-of-images-labels-and-captions.pdf.

[55] Martin Solli and Reiner Lenz. "Visualization of large font databases". In: (Jan. 2009), p. 10.

[56] Joshua B. Tenenbaum, Vin de Silva, and John C. Langford. "A Global Geometric Framework for Nonlinear Dimensionality Reduction". In: *Science* 290.5500 (2000), pp. 2319–2323. ISSN: 0036-8075. DOI: 10.1126/science.290.5500.2319. eprint: http://science.sciencemag.org/content/290/5500/2319.full.pdf. URL: http://science.sciencemag.org/content/290/5500/2319.

[57] *Flask library*. http://flask.pocoo.org/.

[58] *Bootstrap library*. http://getbootstrap.com/.

[59] *JQuery library*. https://jquery.com/.

[60] *List.js library*. http://listjs.com/.