SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 5702

Klasifikacija slika kućnih brojeva dubokim konvolucijskim modelima

Ivan Šego

Zagreb, srpanj 2018.

Umjesto ove stranice umetnite izvornik Vašeg rada.
Da bi ste uklonili ovu stranicu obrišite naredbu `\izvornik`.

Zahvaljujem mentoru prof. dr. sc. Siniši Šegviću, obitelji, prijateljima te Zagrebačkom računalnom savezu na bezuvjetnoj potpori tokom cijelog studiranja

SADRŽAJ

1. Uvod

Sve od ranih razvoja računarstva pa do danas čovjekova velika želja je napraviti stroj koji će dostići čovjekovu inteligenciju ili je čak nadmašiti. Postoje mnogi kognitivni zadaci koji su čovjeku poprilično laki dok su za računalo vrlo teški. Područje računarstva koje su bavi upravo ovakvim problemima se zove umjetna inteligencija (engl. *artificial intelligence*). Vrlo zanimljiva grana umjetne inteligencije je strojno učenje (engl. *machine learning*). Algoritmi strojnog učenja pokušavaju naučiti računalo da obavlja razne zadatke bez da ga eksplicitno programira na temelju dostupnih podataka.

Duboko učenje (engl. *deep learning*) je grana strojnog učenja čiji glavni fokus su umjetne neuronske mreže te slaganje višeslojevite (duboke) neuronske mreže. Predmet ovog rada će biti duboko učenje i njegova primjena. Duboko učenje crpi inspiraciju u živčanom sustavu čovjeku tako pokušavajući imitirati čovjekov mozak. Umjetne neuronske mreže svoje začetke imaju tokom 40-tih godina 20. stoljeća, ali tek razvojem GPU-a (engl. *graphics processing unit*) doživljavaju procvat. Prvi veliki uspjeh umjetnih neuronskih mreža je opisan u [9] kada je skupina istraživača trenirala umjetnu neuronsku mrežu za prepoznavanje slika 1000 različitih objekta (skup ImageNet) i postigla tada u svijetu najbolji rezultat (engl. *state-of-the-art*). Danas, duboko učenje je našlo svoju primjenu u širokom spektru poput prepoznavanja pisanih znakova, prepoznavanje uzoraka, obrada prirodnog jezika, financije, robotika, autonomna vozila i mnoga druga zanimljiva područja.

Predmet ovog rada će biti prepoznavanje slika kućnih brojeva na skupu SVHN (engl. *The Street View House Numbers*), te prepoznavanje ručno pisanih znamenki na skupu MNIST (engl. *Modified National Institute of Standards and Technology database)*). U radu će biti opisane neuronske mreže, posebice konvolucijske, prikazat će se rezultati te dati programski kod.

2. Umjetna neuronska mreža

2.1. Inspiracija

Umjetna neuronska mreža (engl. artificial neural network) je inspirirana biološkom neuronskom mrežom (živčani sustav). Kako bi razumjeli umjetne neuronske mreže vrlo je bitno promotriti mozak kao središnji dio živčanog sustava i uspostaviti analogiju s umjetnim neuronskim mrežama.

Mozak je najsloženiji organ ljudskog tijela koji je odgovoran za sve aktivnosti koji su neophodne za preživljavanje te mnoge druge od kojih su za ovaj rad najbitnije sposobnosti učenja i pamćenja. Upravo umjetne neuronske mreže pokušavaju doseći, pa čak i nadmašiti, ljudsku inteligenciju u raznima zadacima kao što je opisano u poglavlju 1

Mozak se sastoji od međusobno povezanih neurona (engl. neurons) koji tvore informacijsku mrežu. U prosjeku, mozak je sastavljen od 86 milijardi neurona. Postoji više od 100 vrsta neurona od kojih svaki ima točno određenu zadaću i smješten je na točno određenom mjestu unutar mozga. Neuroni su međusobno povezani s 10^{14} - 10^{15} sinapsi (engl. synapses). U prosjeku, svaki neuron je povezan s 10^4 drugih neurona. Vrlo je važno istaknuti ove brojke, jer umjetna neuronska mreža nastoji nadmašiti ovakav sustav.

2.2. Neuron

Neuron je osnovna građevna jedinica živčanog sustava. Tipično se dijeli na 4 osnovna dijela:

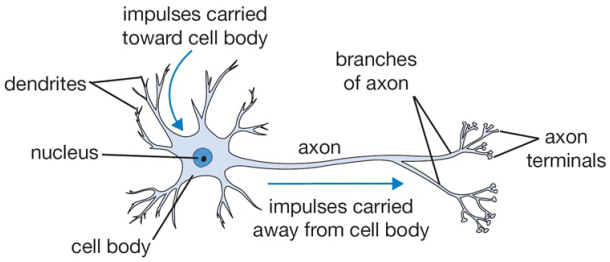
1. tijelo (engl. cell body) - sadrži informaciju u obliku razlike električkog potencijala između unutarnjeg i vanjskog dijela stanice

2. dendriti (engl. dendrites) - ogranci preko kojeg tijelo prima električki impuls

3. akson (engl. axon) - provodi impulse prema drugim neuronima

4. niz završnih članaka (engl. axon terminals) - spajaju se pomoću sinapse na dendrite drugih neurona

Iz redoslijeda navođenja dijelova neurona se nazire i princip rada biološkog neurona. Promjena električkog potencijala unutar tijela, nastala usred bilo kakvog podražaja, dovodi do aktivacije te nastaje elektrokemijski impuls koji se preko aksona, putem sinapse, širi na dendrite te nadalje prenosi informaciju idućem neuronu. Češći prijenos između dva neurona uzrokuje jaču sinaptičku vezu između ta dva neurona. Biološki neuron je prikazan na slici 2.1



Slika 2.1: Model biološkog neurona

Analogno, možemo konstruirati matematički model neurona. Dendriti postaju ulazne vrijednosti $x_1, x_2, ..., x_n$ koji tvore ulazni vektor X. Ulazni vektor može biti ili izlaz iz prethodnog neurona ili stvarni ulaz. Za svaku ulaznu vrijednost (dendrit) x_i definirana je vrijednost w_i (nazivamo ju težina (engl. *weight*)) koja predstavlja jačinu sinaptičke veze tj. koliko i_{ta} vrijednost djeluje na neuron što je dano izrazom $w_i \cdot x_i$. Dodatno, definirana je prag (engl. *bias*) označen s w_0 koji označava prag aktivacije funkcije. Tijelo stanice akumulira djelovanje svih ulaza prema izrazu 2.1

$$net = \sum_{i=1}^{n} w_i \cdot x_i + w_0 \tag{2.1}$$

Dogovorno se najčeće uvodi $x_0 = 1$ pa je izlaz iz neurona jednostavnije dan izrazom 2.2

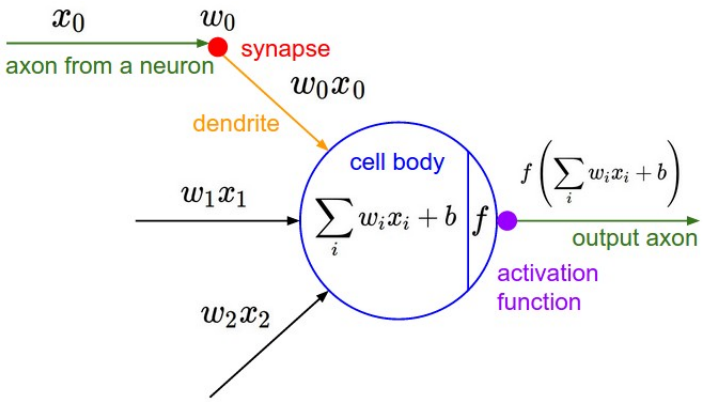
$$net = \sum_{i=0}^{n} w_i \cdot x_i \tag{2.2}$$

Težine predstavljaju značajku neuronske mreže koju učimo i koja se prilagođava dosad viđenim i na temelju toga generalizira neko znanje. Nakon akumuliranja dolazi do aktivacije koju modelira aktivacijska funkcija f (engl. *activation function*) koja predstavlja informaciju koja se prenosi aksonom. Konačno, izlaz iz neurona je dan

izrazom:

$$y = f(net) = f(\sum_{i=0}^{n} w_i \cdot x_i) \tag{2.3}$$

Gdje je aktivacijska funkcija f najčešće sigmoidalna (engl. sigmoid), tangens hiperbolni (engl. tanh), zglobnica (engl. Rectified Linear Unit, kratica ReLU) ili propusna zglobnica (engl. Leaky ReLU)



Slika 2.2: Model matematičkog neurona

2.2.1. Aktivacijske funkcije

Kao što je opisano u 2.2 vrlo je važan odabir prikladne aktivacijske funkcije. U nastavku poglavlja slijedi opis trenutno najvažnijih aktivacijskih funkcija. Za svaku funkciju je naveden njen izraz, derivacija (važna zbog procesa učenja) te prednosti i nedostaci.

Sigmoidalna funkcija (logistička) je dana izrazom 2.4

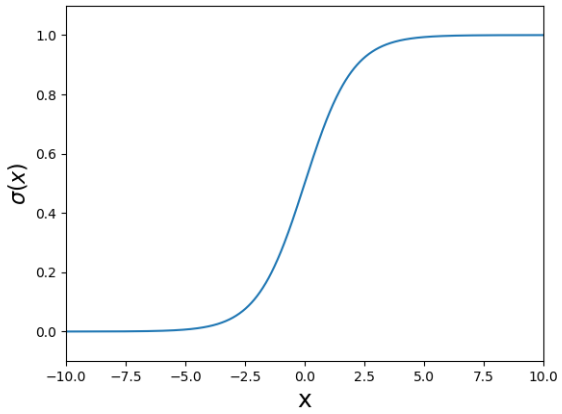
$$sigma(x) = 1/(1 + e^{-x}). \tag{2.4}$$

Njena derivacija je dana izrazom 2.5

$$\frac{d\sigma}{dx} = \sigma(x) \cdot (1 - \sigma(x)) \tag{2.5}$$

Sigmoidalna funkcija je prikazana na slici 2.3

Najvažnije prednosti sigmoidalne funkcije su derivabilnost, pretvaranje realnog broj u interval [0, 1] te njena nelinearnost. Povijesno gledano, sigmoidalna je najvažnija aktivacijska funkcija. Njeni glavni nedostaci su što je gradijent približno 0 prije $x = 0$ i nakon $x = 1$ (nepogodno za postupak učenja propagacije pogreške unatrag, problem zasićenja) te što izlaz nije centriran oko 0 (problemi s učenjem)



Slika 2.3: Prikaz sigmoidalne funkcije na intervalu [-10, 10]

Tangens hiperbolni je dan izrazom 2.6

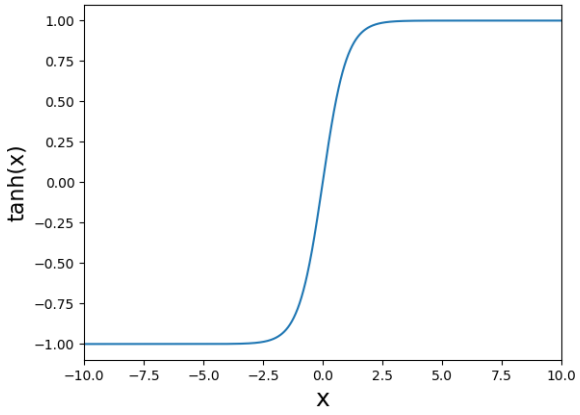
$$tanh(x) = 2 \cdot \sigma(2 \cdot x) - 1 \tag{2.6}$$

Njegova derivacija je dana izrazom 2.7

$$\frac{dtanh(x)}{dx} = 1 - tanh^2(x) \tag{2.7}$$

Tangens hiperbolni je sličan sigmoidalnoj ali pretvara brojeve u interval [-1, 1] (izlaz centriran oko 0) što mu je glavna prednost u odnosu na sigmoidalnu funkciju.

Tangens hiperbolni je prikazan na slici 2.4



Slika 2.4: Prikaz tangens hiperbolne funkcije na intervalu [-10, 10]

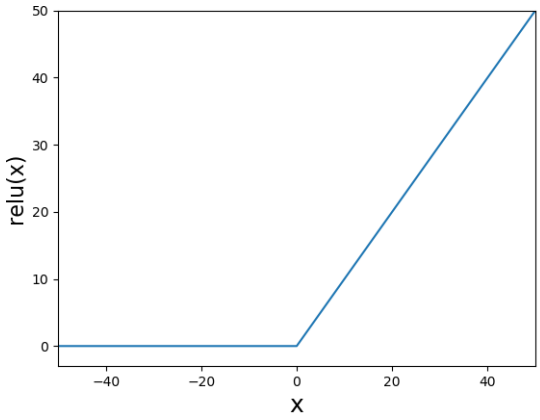
ReLU je dana izrazom 2.8

$$relu(x) = max(0, x) \tag{2.8}$$

Dok je njena derivacija dana izrazom 2.9

$$\frac{drelu}{dx} = \begin{cases} 0 & x \leq 0 \\ 1 & x > 0 \end{cases} \tag{2.9}$$

ReLU je prikazan na slici 2.5



Slika 2.5: Prikaz funkcije ReLU na intervalu [-50, 50]

Pokazano je da ReLU ubrzava do 6 puta [9] konvergenciju algoritma učenja, vrlo je jednostavna za implementirati, nema problema sa zasićenjem s pozitivne strane (nakon $x = 1$ je derivacija 1 u odnosu na 2.5 i 2.7 gdje je 0). Glavni nedostatak je *umirući* (engl. *dying*) ReLU tj. ako je distribucija ulaza zakrivljena udesno onda algoritam propagacije pogreške unatrag lošije uči jer je gradijent 0.

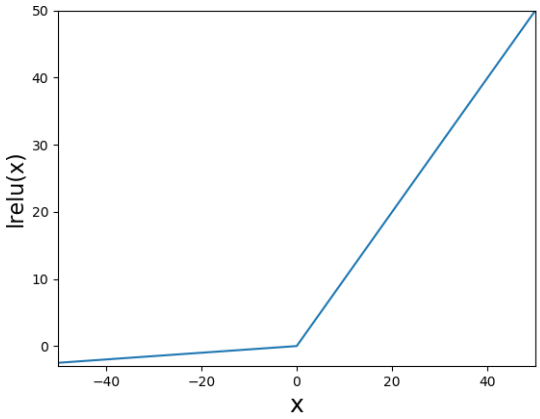
Leaky ReLU je dan izrazom 2.10

$$lrelu(x) = \begin{cases} \alpha \cdot x & x \leq 0 \\ x & x > 0 \end{cases} \tag{2.10}$$

Derivacije je dana izrazom 2.11

$$\frac{dlrelu}{dx} = \begin{cases} \alpha & x \leq 0 \\ 1 & x > 0 \end{cases} \tag{2.11}$$

Leaky ReLU je prikazan na slici 2.6

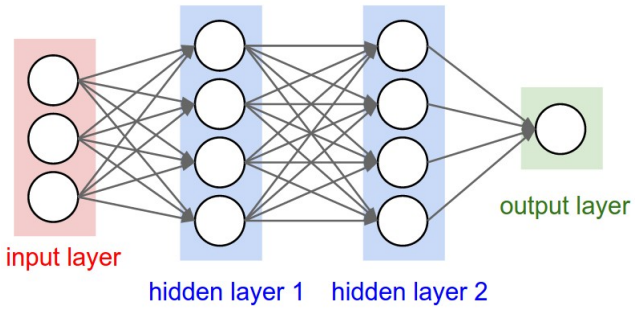


Slika 2.6: Prikaz funkcije leaky ReLU na intervalu [-50, 50] s α = 0.05

Glavna prednost je rješavanje problema *umirućeg* ReLU na način da se omogući da gradijent ipak ne bude 0 za $x_i \leq 0$ nego neka vrlo mala konstanta α tipično 0.01.

2.3. Arhitektura umjetne neuronske mreže

Nakon što smo uveli matematički model jednog neurona možemo definirati cjelokupnu arhitekturu neuronske mreže. Umjetna neuronska mreža se sastoji od skupa međusobno povezanih matematičkih neurona koje najlakše možemo predstaviti pomoću teorije grafova. Prema slici 2.7 vidimo da je neuronska mreža skup neurona koji tvore usmjereni aciklički graf. Naravno, postoje i drugačije arhitekture koje ne uključuju svojstvo acikličnosti grafa, ali neće biti proučavane u ovom radu. Neuronske mreže se najčešće sastoje od 3 različita sloja: ulazni sloj (engl. input layer), izlazni sloj(engl. output layer) koji su prisutni te skriveni slojevi (engl. hidden layers) koji ne moraju postojati, ali za teže probleme upravo skriveni slojevi omogućuju njihovo efikasno rješavanje.



Slika 2.7: Arhitektura neuronske mreže

2.4. Učenje umjetne neuronske mreže

Učenje neuronske mreže se ugrubo može podijeliti na dvije faze: faza učenja i faza evaluacije. U fazi evaluacije evaluiramo dosadašnji naučeni model i testiramo na neuronsku mrežu na dosad neviđenim podacima. Faza evaluacije je vrlo jednostavna za izvesti dok je faza učenja ključ neuronskih mreža.

Postoje 3 vrste učenja (na temelju predočavanja uzorka):

1. pojedinačno učenje (engl. on-line) - učenje nakon svakog uzorka,
2. učenje s mini-grupama (engl. mini-batches) - učenje se događa nakon određen grupe uzoraka.
3. grupno učenje (engl. batch) - učenje nakon svih predočenih uzoraka.

U radu će biti korišteno učenje s mini-grupama koje se pokazalo najisplativijim s obzirom na vrijeme i memoriju. Iteracija je naziv za predočavanje jednog uzorka mreži. Dok epoha označava jedno predočavanje svih uzoraka. Također, možemo učenje neuronskih mreža podijeliti na 3 vrste (na temelju podataka) učenja-

Vrste učenja s obzirom na oblik podataka su:

1. nadzirano učenje (engl. supervised) - ulazi su oblika $[x_1, f(x_1)], [x_2, f(x_2)], ...$ - za svaki ulaz postoji njegov očekivani izlaz
2. nenadzirano učenje (engl. unsupervised) - ulazi su oblika $x_1, x_2, x_3, ...$ - ne postoji očekivani izlaz, najčešće se traži grupiranje
3. podržano učenje(engl. reinforcement) - učenje na temelju obavljenih akcija u nekom okruženju

U radu će biti korišteno nadzirano učenje. Nadzirano učenje tipično rješava probleme klasifikacije i regresije. Klasifikacija je oblik u kojem se ulazni podaci pokušavaju svrstati u određene diskretne klase. Dok regresija pokušava svrstati podatke u kontinuirane vrijednosti. U radu se koristi klasifikacija.

2.4.1. Funkcija gubitka

Funkcija gubitka je mjera u nadziranom učenju koja predstavlja koliko dobro je neuronska mreža naučila na temelju podataka. Cilj je minimizirati funkciju gubitka tj. pronaći takve težine za koje je funkcija gubitka najmanja.

Tipične funkcija gubitka koja se koristi je L^2 norma. Označimo s Y stvarne vrijednosti izlaza dok s Y' predviđene vrijednosti te s n broj podataka, tada je L^2 norma dana izrazom 2.12

$$E(Y, Y') = \frac{1}{2}\|\mathbf{Y} - \mathbf{Y}'\|^2 = \frac{1}{2}\sum_{i=1}^{n}(y_i - y_{i'})^2 \tag{2.12}$$

L^2 norma je ipak više korištena u regresijskim problemima dok se u klasifikacijskim problemima pretežito koristi unakrsna entropija (engl. cross entropy) koja je dana izrazom 2.13

$$L = \sum_{i=1}^{n} y_i \cdot log(y_{i'}) \tag{2.13}$$

2.4.2. Algoritam propagacije pogreške unatrag (engl. backpropagation)

Algoritam propagacije pogreške unatrag je središnji algoritam u treniranju neuronskih mreža. Algoritam nam daje način kako smanjiti funkciju gubitka opisanu u 2.4.1 na temelju gradijenta funkcije.

Gradijent funkcije više varijabli je vektor koji se sastoji od parcijalnih derivacija funkcija s obzirom na svaku varijablu.

Primjerice, gradijent funkcije od 3 varijable $f(x_1, x_2, x_3)$ je dan izrazom 2.14

$$\nabla f = (\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \frac{\partial f}{\partial x_3}) \tag{2.14}$$

Algoritam propagacije pogreške unatrag se temelji na jednostavnom pravilo ulančavanja koje nam daje način izračunavanja gradijenta. Označimo sa $z = f(y)$, te $y = g(x)$ (z ovisi o y dok y ovisi o x), tada je pravilo ulančavanja dano izrazom 2.15

$$\frac{dz}{dx} = \frac{dz}{dy}\frac{dy}{dx} \tag{2.15}$$

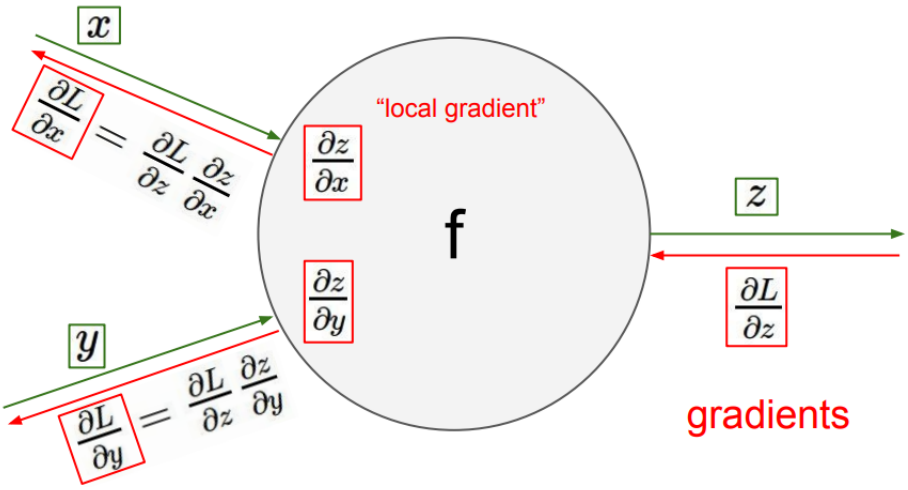
Ukoliko funkcija ovisi o više varijabli, primjerice $z = f(x(t), y(t))$ tada je prema pravilu ulančanja derivacija funkcije z po t-u dano izrazom 2.16

$$\frac{dz}{dt} = \frac{\partial z}{\partial x}\frac{\partial x}{\partial t} + \frac{\partial z}{\partial y}\frac{\partial y}{\partial t} \tag{2.16}$$

Algoritam propagacije pogreške unatrag kreće od funkcije gubitka i pomoću pravila ulančavanja propagira pogrešku unatrag do ulaznog sloja.

Pogledajmo primjenu pravila derivacije kompozicije funkcije u modelu neurona na slici 2.8. Neka su x i y ulazi u neuron dok je z izlaz. Funkcija gubitka je označena s

L. $\frac{\partial L}{\partial z}$ dobijemo direktnim deriviranjem funkcije gubitka po varijabli z dok primjerice $\frac{\partial L}{\partial x}$ dobijemo pomoću pravila 2.15 koristeći prethodno izračunato $\frac{\partial L}{\partial x}$ - zato moramo ići unatrag. Analogno, računamo i $\frac{\partial L}{\partial y}$.



Slika 2.8: Algoritam propagacije pogreške unatrag u jednostavnom neuronu

Smjer vektora gradijenta nam daje smjer rasta funkcije. Ako težine neuronskih mreža ažuriramo u negativnom smjeru gradijenta funkcije gubitka onda se krećemo prema najmanjoj vrijednosti funkcije gubitka što je cilj neuronskih mreža. Takav algoritam koji ažurira težine za određeni iznos u negativnom smjeru gradijenta se zove zove gradijentni spust (engl. *gradient descent*). Ažuriranje težine w_i je dano izrazom 2.17

$$w_i = w_i - \eta \frac{\partial E}{\partial w_i} \tag{2.17}$$

U izrazu 2.17 je dan parametar η koji se naziva stopa učenja (engl. *learning-rate*), a označava koliko brzo želimo da učenje ide. Ako je stopa učenja prevelika moguće je da algoritam učenja divergira i tako ne pronalazi globalni minimum dok u slučaju da je premala stopa ulazimo u opasnost sporog učenja. Tipične vrijednosti stope učenja su intervalu $[0.01, 0.001]$

U radu će biti korištena inačica gradijentnog spusta zvana Adam opisana u [7]

2.4.3. Normalizacija nad grupom (engl. *batch-normalization*)

Normalizacija nad grupom se pokazalo kao vrlo efikasna metoda koja poboljšava točnost neuronske mreže [6]. Možemo je smatrati metodom regularizacije. Ideja je da izlaz iz svakog sloja normalizira. Normalizirani ulazi/izlazi omogućavaju gradijentnom spustu da brže konvergira. Bez normalizacije promjena u jednom sloju neuronske

mreže utječe na drugi sloj, dok se s normalizacijom podataka omogućava svojevrsna neovisnost između slojeva i tako se omogućava brže konvergiranje. U radu će biti korištena normalizacija nad mini grupom prema sljedećim izrazima 2.18, 2.19, 2.20

Označimo s mini grupu s B čiji se podaci normaliziraju s $B = x_{1..m}$

Aritmetičku sredinu mini grupe nalazimo prema izrazu 2.18

$$\mu_B := \frac{1}{m} \sum_{i=1}^{m} x_i \tag{2.18}$$

Varijancu mini grupe nalazimo prema izrazu 2.19

$$\sigma_B^2 := \frac{1}{m} \sum_{i=1}^{m} (x_i - \mu_B)^2 \tag{2.19}$$

Tada, normalizaciju možemo napraviti prema izrazu 2.20

$$\overline{x_i} := \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} \tag{2.20}$$

2.4.4. Neprijateljski primjeri (engl. *adversarial examples*)

Neprijateljski primjeri su slike koje napadač namjerno dizajnira kako bi prouzročio da model krivo klasificira. Predstavljaju vrlo ozbiljan problem u svijetu neuronskih mreža odnosno modela.

Jedna od efikasnih metoda za generiranje neprijateljskih primjera se zove brza predznačna metoda temeljena na gradijentu (engl. *fast gradient sign method*). Uvedimo η kao novu sliku izračunatu prema izrazu 2.21

$$\eta = \epsilon sign(\nabla_{\boldsymbol{x}} J(\theta, \boldsymbol{x}, y)) \tag{2.21}$$

Gdje je ϵ proizvoljno odabrana konstanta, najčešće 0.01. Dok θ označava parametre neuronske mreže, y predstavlja izlaz, a x originalnu sliku. Gradijent dobivamo koristeći algoritam propagacije pogreške unatrag opisan u 2.4.2

3. Konvolucijska neuronska mreža

Konvolucijske neuronske mreže se pojavljuju u [11]. Procvat doživljavaju u [9] u kojem po prvi puta duboke (one s više skrivenih slojeva) konvolucijske mreža postižu tada najbolji rezultat na području prepoznavanja objekata na slici.

Namjena konvolucijske neuronske mreže je obrada i učenje na temelju podataka koji dolaze u obliku više dimenzionalnih polja poput tri 2D polja koji predstavljaju jednu sliku u RGB obliku (svako 2D polje za jednu boju) koja je ujedno i korištena u ovom radu. Drugi primjeri uključuju 1D signale, 1D nizove riječi, 3D videe i mnoge druge. Konvolucijska neuronska mreža se esencijalno ne razlikuje od potpuno povezane (obične) neuronske mreže opisane u poglavlju 2. Sastavljena je od neurona, težina i pragova koje je cilj naučiti na temelju ulaznog vektora. Njena moć leži u činjenici da različitim vrstama slojeva i prikladnom arhitekturom (slaganjem) slojeva postiže bolje rezultate u odnosu na potpuno povezanu neuronsku mrežu.

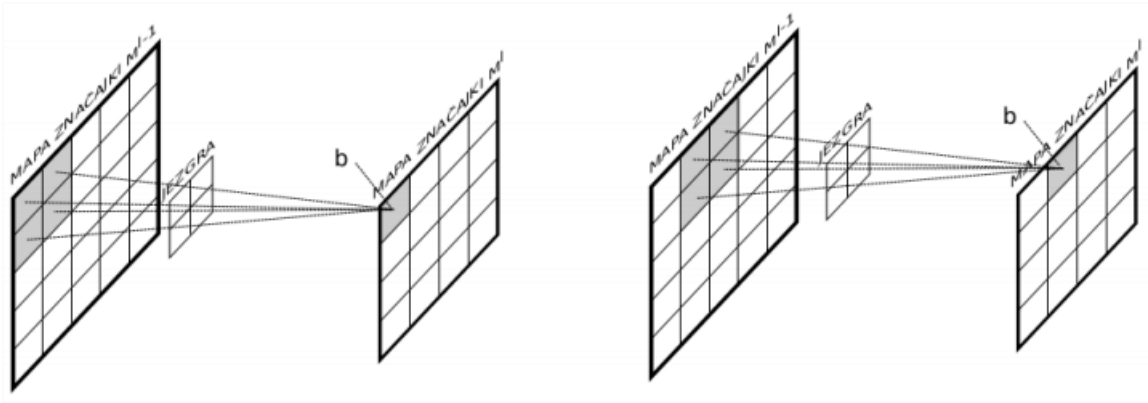
Arhitektura konvolucijskih mreža se sastoji od sljedećih slojeva: konvolucijski sloj (engl. *convolutional layer*), sloj sažimanja (engl. *polling layer*) i potpuni povezani sloj engl. Fully-Connected layer koji je identičan potpuno povezanom sloju upisanom u 2.3

3.1. Konvolucijski sloj

Konvolucijski sloj je najbitniji sloj u arhitekturi konvolucijske neuronske mreže. Ime je dobio prema operatoru konvolucije. Konvolucijski sloj se sastoji od filtara koji sadrže težine koje je potrebno naučiti. Filter je 2D matrica težina (u ovom radu) tipično puno manjih dimenzija od ulaza. Filter možemo zamisliti kao pomični prozor koji se kreće od početka do kraja stupca, pa prelazi u novi red i nastavlja tako sve dok ne obiđe svaku polje ulaza barem jednom. U svakom trenutku (pomaku) se sve težine iz pomičnog prozora množe s odgovarajućim dijelom kojeg pokrivaju. Na slici 3.1 možemo vidjeti dva koraka konvolucijskog sloja za određenu jezgru.

Ovakvim načinom želimo naučiti filtre da prepoznaju određene prostorne značajke,

primjerice rubove.



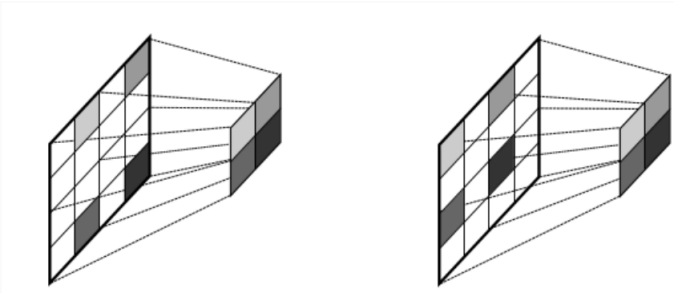
Slika 3.1: Dva koraka konvolucijskog sloja

Svaki konvolucijski sloj ima 3 vrlo bitna parametra koju su unaprijed određeni, a to su:

1. dubina (engl. *depth*) - broj filtera
2. pomak (engl. *stride*) - za koliko će se filter pomicati
3. nadopunjavanje (engl. *padding*) - broj stupaca i redaka koji se dodaju kako se ne bi gubila informacija na krajevima

3.2. Sloj sažimanja

Sloj sažimanja smanjuje dimenzionalnost ulaza tako da prostorno bliske značajke (najčešće kvadrat dimenzija 2×2) mapira u jednu značajku primjenom odgovarajuće matematičke operacije (najčešće uzimanje najvećeg (engl. *max pooling*) ili srednje vrijednosti). Primjer je prikazan na slici 3.2 koja označava uzimanje najvećeg elementa iz pomičnog prozora dimenzija 2×2. Najveći elementi su označeni sivim bojama.



Slika 3.2: Sloj sažimanja

4. Programski okvir PyTorch

PyTorch je programski okvir zasnovan u okviru programskog jezika Python. Primarno ga je napravio i dizajnirao Facebook te je trenutno otvorenog koda (engl. *open source code*). U trenutku pisanja ovog završnog rada PyTorch slavi oko 1 godinu i 6 mjeseci od svog izlaska [3]. Programski kod u ovom radu je pisan u verziji 0.4.0. Glavna karakteristika PyTorcha je njegov $tensor$ koji je sličan Pythonovom numpy polju ali se pokreće na GPU što mu daje veliku brzinu izvođenja. Jedan od najkorištenijih paketa u PyTorchu je *Autograd* koji omogućava automatsko praćenje gradijenta u te tako omogućava vrlo jednostavnu implementaciju algoritma propagacije pogreške unatrag.

Pogledajmo jednostavan primjer uporabe paketa $Autograd$

```
x = torch.ones(1, requires_grad = True) #tensor od jednog broja 1
y = torch.ones(1, requires_grad = True) #tensor od jednog broja 1

z = 2 * x * x + y * y
w = z * z + 4
w.backward() #pozivamo racunanje gradijenta unazad

print(x.grad) #dobivamo dw/dx = 24
print(y.grad) #dobivamo dw/dy = 12
```

5. Ispitni skupovi

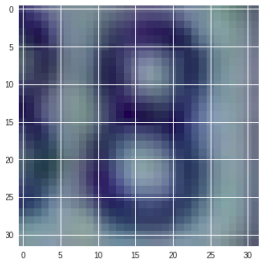
5.1. SVHN

SVHN (engl. *The Street View House Numbers*) [12] je skup slika koji su dobiven iz Google Street Viewa - tehnologija koja omogućava panoramski pogled iz većine pozicija na svijetu. Sastoji se od 10 klasa (svaki jednoznamenkasti broj od 0-9) gdje je 0 označena klasom 10, dok su ostali označeni sukladno svom pravom broju. Skup se sastoji od ukupno 73257 znamenaka za treniranje te 26032 znamenaka za testiranje i dodatnih 531131 znamenaka za treniranje. Skup podataka dolazi u 2 forme:

1. originalni skup - na slikama je dodatno označen oko svake znamenke njen pripadajući pravokutnik

2. podrezani (engl. *cropped*) skup - na svakoj slici je jedna znamenka dimenzija 32×32 piksela.

   U radu će biti predstavljeno rješenje za podrezani skup, dok se zainteresiranog čitatelja upućuje na [5] gdje je opisano rješenje za originalni skup. Na slici 5.1 vidimo primjer slike iz skupa MNIST.



Slika 5.1: Kućni broj 8

5.2. MNIST

MNIST je skup ručno pisanih znamenaka [10]. Ima 60000 slika za treniranje te 10000 slika za testiranje. Znamenke na slikama su u centru te je slika dimenzija 28×28. Na slici 5.2 vidimo prije slike iz skupa MNIST.

Slika 5.2: Ručno napisan broj 8

MNIST skup je vrlo lakši od skupa SVHN jer su sve znamenke crno-bijele te nema popratnih teškoća prepoznavanja zbog pozadine slike.

6. Eksperimentalni rezultati

6.1. Rezultati na skupu SVHN

6.1.1. Arhitektura korištene mreže

Za klasifikaciju skupa SVHN korištena je sljedeća arhitektura:

1. ulazni sloj - slika dimenzija $32 \times 32 \times 3$ - visina i širina slike su 32, dok treća dimenzija predstavlja 3 2D matrice koje označavaju redom količine crvene, zelene i plave (engl. *RGB*) boje za svaki piksel.

2. Konvolucijski sloj - 32 filtra dimenzija 5×5, nakon slijedi normalizacija nad grupom pa prijenosna funkcija ReLU, pomak je 1 dok je nadopunjavanje 4. Izlaz: 32 matrice dimenzija 36×36

3. Sloj sažimanja - filter veličine 2×2. Izlaz: 32 matrice dimenzija 18×18

4. Konvolucijski sloj - 64 filtra dimenzija 3×3, nakon slijedi normalizacija nad grupom pa koji prijenosna funkcija ReLU, pomak je 1 dok je nadopunjavanje 2. Izlaz 64 matrice dimenzija 20×20

5. Sloj sažimanja - filter veličine 2×2. Izlaz: 64 matrice dimenzija 10×10

6. Konvolucijski sloj - 128 filtra dimenzija 3×3, nakon slijedi normalizacija pa prijenosna funkcija ReLU. Pomak je 1 dok je nadopunjavanje 2. Izlaz: 128 matrica dimenzija 12×12

7. Sloj sažimanja - filter veličine 2×2. Izlaz: 128 matrica dimenzija 6×6

8. Potpuno povezani sloj - Ulaz je 128 matrica dimenzija 6×6 koji se poredaju u 4608 slijednih neurona. Izlaz: 256 neurona

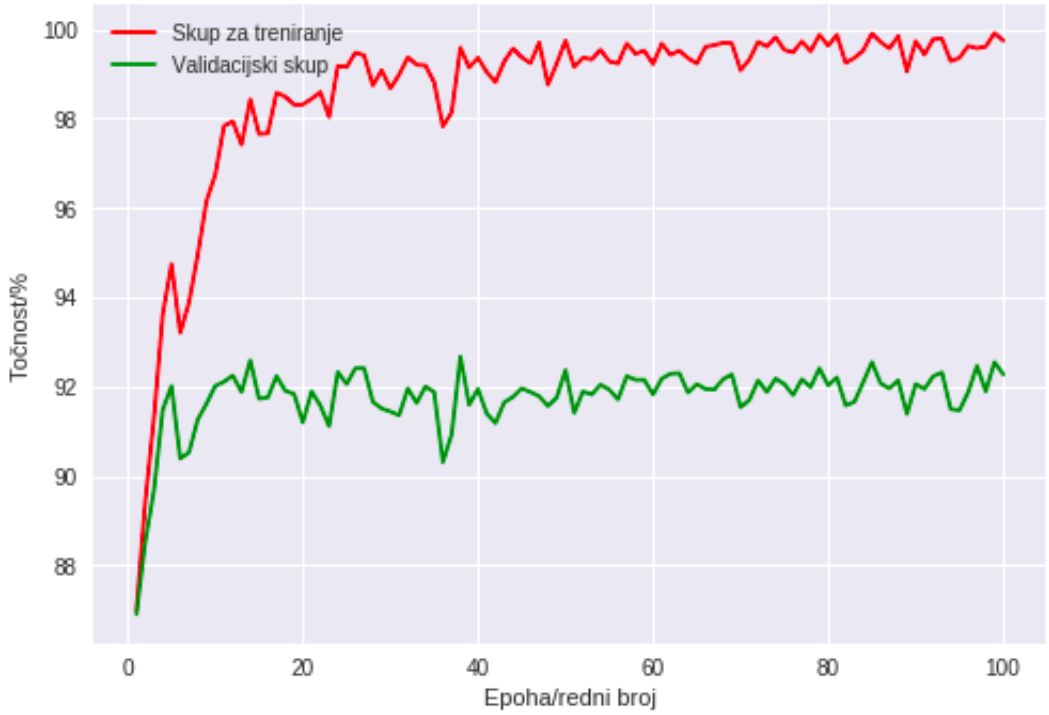
9. Potpuno povezani sloj - 128 neurona, nakon slijedi ReLU

10. Potpuno povezani sloj - 10 neurona

6.1.2. Prikaz rezultata

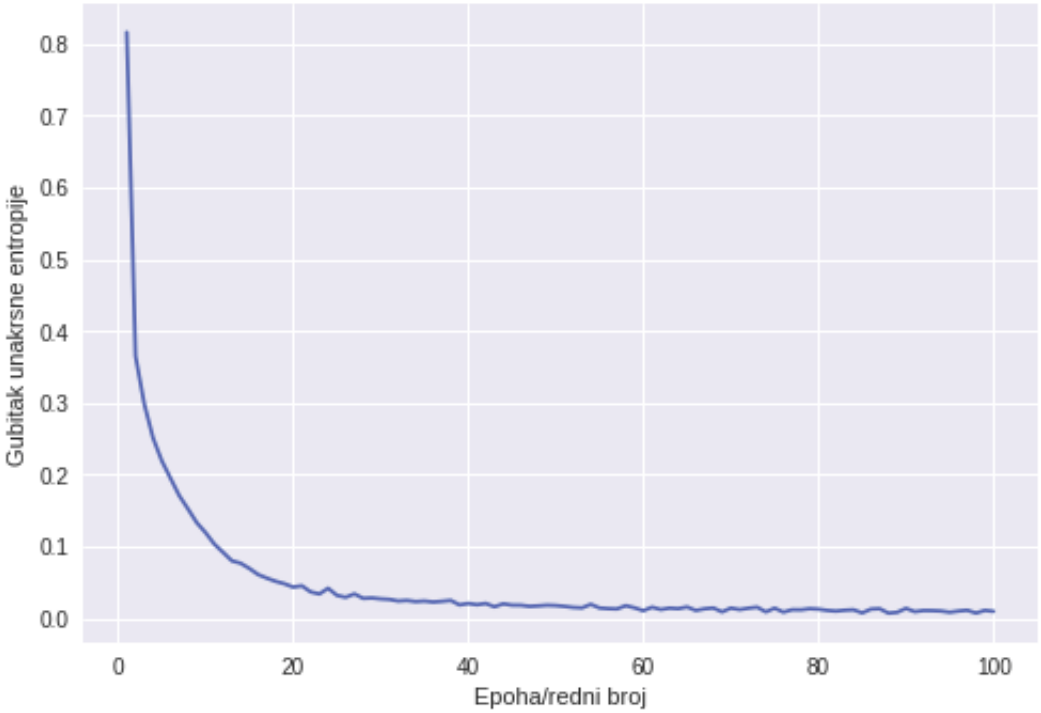
Na slici 6.1 vidimo prikaz dvije krivulje točnosti. Crvenom je označena točnost na skupu za treniranje, dok je zelenom na validacijskom skupu. Prikazan je točnost na ukupno 100 epoha za učenje. Stopa učenja je bila 0.001 kroz sve epohe.

Najveća točnost (engl. *Accuracy*) na validacijskom skupu je bila na epohi rednog broja 38 i iznosila je 92.67% dok je točnost na skupu za testiranje 91.80%.



Slika 6.1: Prikaz točnosti kroz epohe na skupu za treniranje i validacijskom skupu

Na slici 6.2 se nalazi graf koji prikazuje gubitak unakrsne entropije (funkcija gubitka) kroz epohe. Možemo vidjeti da je očekivano ponašanje gubitka kroz epohe. U prvih 20 epoha je nagli pad dok kasnije nastupa vrlo blagi pad.



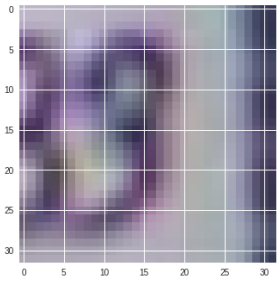
Slika 6.2: Prikaz gubitka unakrsne entropije kroz epohe

Matrica zabune se nalazi na slici 6.1. U matrici zabune svaki redak predstavlja točnu klasu (točan kućni broj) dok svaki stupac predstavlja predviđeni broj. Tako primjericu u 1.retku (klasa 0) u 7. stupcu (klasa 6) predstavlja postotak slika na kojima je bio broj 0 a predviđen je broj 6. Očekivano, na glavnoj dijagonali su najveći postotci što predstavlja točnu klasifikaciju kućnih brojeva. Najlošiji rezultat se nalazi za broj 8 (klasa 8) gdje primjerice u 9. retku (klasa 8) i 7. stupcu(klasa 6) se nalazi najveći postotak koji nije na glavnoj dijagonali i iznosi 5.48% i predstavlja ukupan postotak slika na kojima je broj 8, a klasificiran je 6.

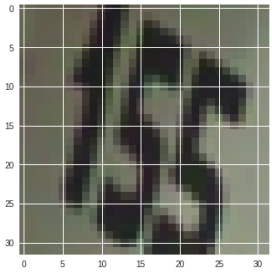
Matrica zabune										
Klase	0	1	2	3	4	5	6	7	8	9
0	**93.12**	0.69	0.46	0.34	0.40	0.17	1.55	0.52	0.46	2.29
1	0.92	**93.78**	0.88	0.55	1.82	0.10	0.31	1.22	0.22	0.20
2	0.34	0.72	**94.38**	1.04	0.89	0.31	0.39	0.94	0.14	0.84
3	0.42	2.08	1.49	**86.09**	0.45	3.71	0.59	0.49	0.87	3.82
4	0.55	2.26	0.87	0.4	**93.74**	0.36	0.52	0.36	0.28	0.67
5	0.17	0.5	0.42	1.93	0.63	**91.99**	2.64	0.17	0.34	1.22
6	1.77	0.71	0.1	0.76	0.61	2.23	**91.1**	0.35	1.57	0.81
7	0.25	4.26	1.93	0.74	0.25	0.15	0.1	**91.58**	0.0	0.74
8	1.14	0.96	0.48	1.45	0.6	1.45	5.48	0.12	**84.46**	3.86
9	1.76	0.56	1.5	0.44	0.5	0.63	0.63	0.25	0.69	**93.04**

Tablica 6.1: Matrica zabune

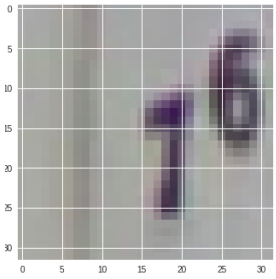
U nastavku se nalazi prikaz 12 slika kućnih brojeva koji su krivo klasificirani.



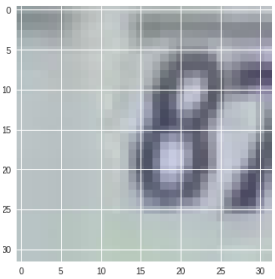
(a) Predviđeno: 9 Vjerojatnost:1.0 Točno: 3



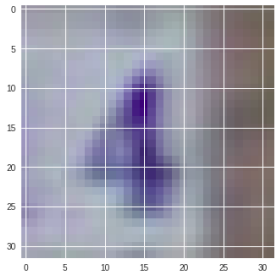
(b) Predviđeno: 3 Vjerojatnost:1.0 Točno: 5



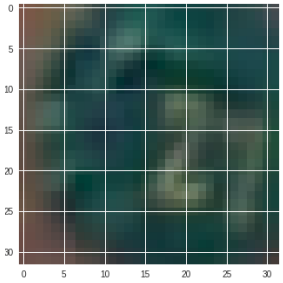
(c) Predviđeno: 1 Vjerojatnost:0.99 Točno: 7



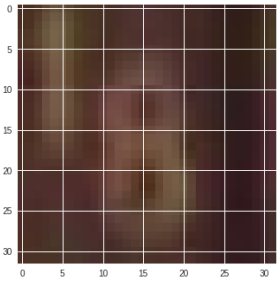
(d) Predviđeno: 1 Vjerojatnost:0.98 Točno: 8



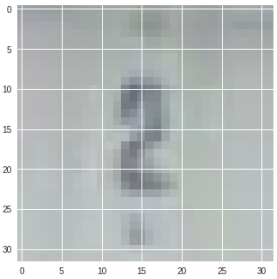
(e) Predviđeno: 3 Vjerojatnost:0.98 Točno: 4



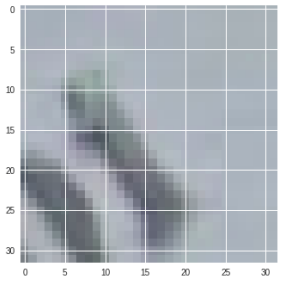
(f) Predviđeno: 6 Vjerojatnost:0.90 Točno: 2



(g) Predviđeno: 6 Vjerojatnost:0.86 Točno: 8



(h) Predviđeno: 3 Vjerojatnost:0.82 Točno: 2



(i) Predviđeno: 0 Vjerojatnost:0.48 Točno: 1

Slika 6.3: Prikaz netočno klasificiranih slika brojeva

6.2. Rezultati na skupu MNIST

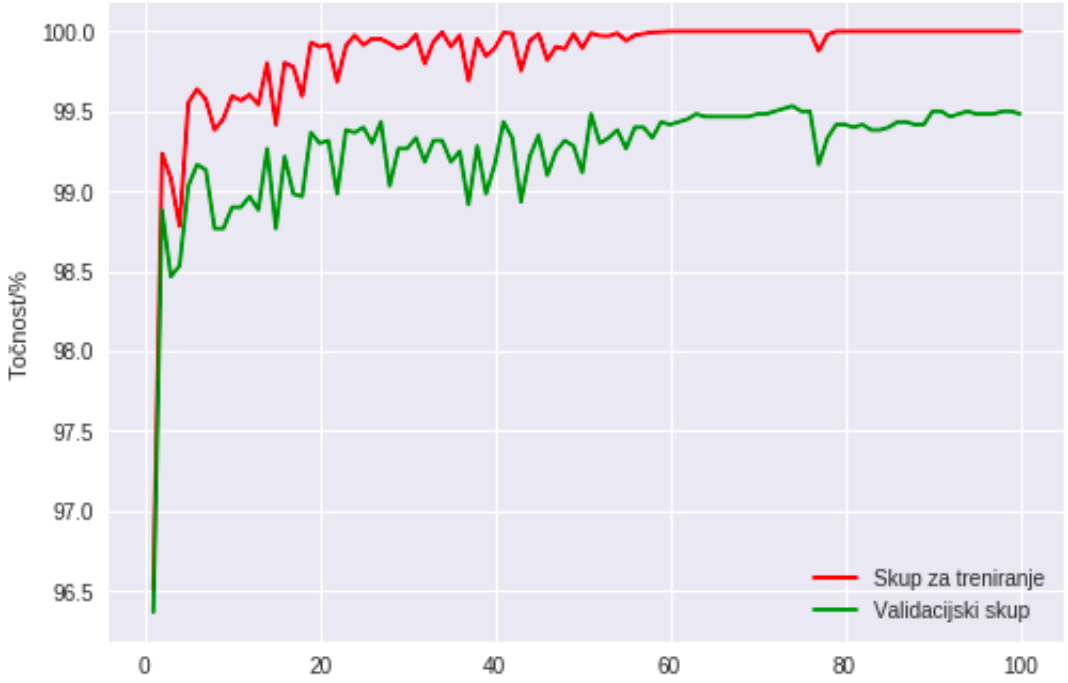
6.2.1. Arhitektura mreže

Za MNIST je korištena ista arhitektura kao i za SVHN opisana u 6.1.1, uz naravno manju promjenu da je umjesto ulaza $32 \times 32 \times 3$ koji je predstavljao SVHN sliku

dovedena MNIST slika koja je dimenzija $28 \times 28 \times 1$.

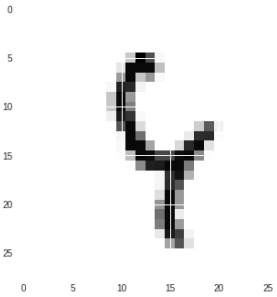
6.2.2. Prikaz rezultata

Na slici 6.4 su prikazane dvije krivulje točnosti na skupu za treniranje i validacijskom skupu. Iz grafa je vidljivo da se krivulje ponašaju vrlo slično do oko 10 epohe te se onda krivulja točnosti na skupu za treniranje naglo poraste te do kraja učenja (100. epohe) se ponašaju vrlo slično.
Najveća točnost na validacijskom skupu je 99.53% koja je postignuta u 74. epohi. Dok je točnost na skupu za testiranje visokih 99.52%

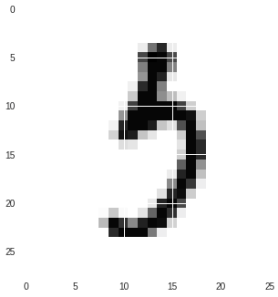


Slika 6.4: Prikaz točnosti kroz epohe na skupu za treniranje i validacijskom skupu

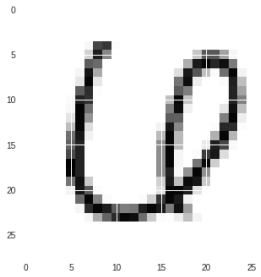
U nastavku na slici 6.5 se nalazi prikaz 6 slika primjera krive klasifikacije te ujedno iznos vjerojatnosti te krive klasifikacije.



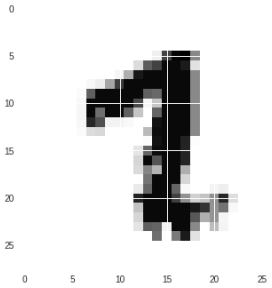
(a) Predviđeno: 4 Vjerojatnost:1.0 Točno: 9



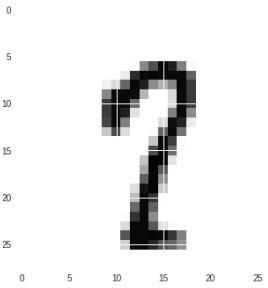
(b) Predviđeno: 5 Vjerojatnost:1.0 Točno: 3



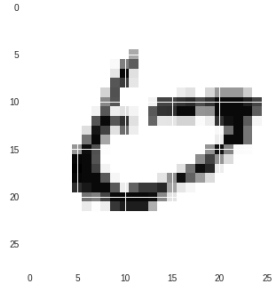
(c) Predviđeno: 0 Vjerojatnost:0.99 Točno: 6



(d) Predviđeno: 1 Vjerojatnost:0.99 Točno: 2



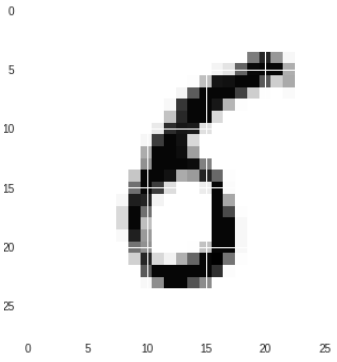
(e) Predviđeno: 2 Vjerojatnost:0.99 Točno: 7



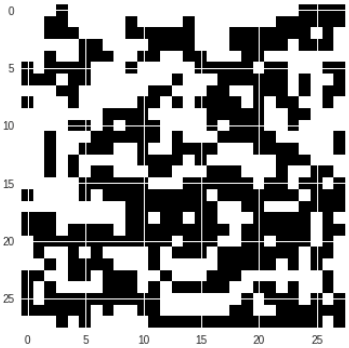
(f) Predviđeno: 0 Vjerojatnost:0.99 Točno: 6

Slika 6.5: Prikaz netočno klasificiranih slika brojeva

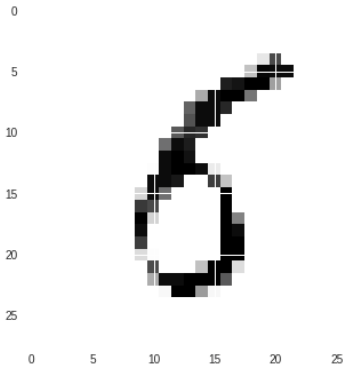
U nastavku je dan primjer generiranog neprijateljskog primjera prema 2.4.4



Slika 6.6: Predividio 6 s vjerojatnosi 0.98



Slika 6.7: Preturbacija η



Slika 6.8: Zbrojene slike 6.6 i 6.7. Predividio 5 s 0.99

7. Programska izvedba

Cjelokupni programski kod za ispitni skup SVHN (slično i za MNIST) je organiziran kao jedna IPython bilježnica *.ipynb* gdje su najbitniji dijelovi grupirani po ćelijama. Cijeli programski kod je pokretan na GPU (engl. *Graphics processing unit*)radi bržeg izvođenja na servisu Google colab.
U nastavku su izdvojene 2 najbitnije komponente i dan je njihov programski kod u PyTorch-u, a to su: definicija mreže, i treniranje Zainteresirani čitatelj može kontaktirati autora za detaljniju implementaciju (faza evaluacije, priprema podataka, grafovi i slično).
Važno je napomenuti da je sav kod pisan u PyTorch-u verziji 0.4.0.

7.1. Definicija konvolucijske mreže

U nastavku je dan programski kod koji definira konvolucijsku neuronsku mrežu i njene slojeve te konceptualno odgovara arhitekturi 6.1.1

```
class convNet(nn.Module):
  def __init__(self):
    super(convNet, self).__init__()
    self.layer1 = nn.Sequential(
        nn.Conv2d(3, 32, kernel_size = 5, stride = 1, padding = 4),
        nn.BatchNorm2d(32),
        nn.ReLU(),
        nn.MaxPool2d(2, 2)
    )
    self.layer2 = nn.Sequential(
        nn.Conv2d(32, 64, 3, padding = 2),
        nn.BatchNorm2d(64),
        nn.ReLU(),
        nn.MaxPool2d(2, 2)
    )
    self.layer3 = nn.Sequential(
```

```
17        nn.Conv2d(64, 128, 3, padding = 2),
18        nn.BatchNorm2d(128),
19        nn.ReLU(),
20        nn.MaxPool2d(2, 2)
21
22      )
23      self.layer4 = nn.Sequential(
24          nn.Linear(4608, 256),
25          nn.ReLU()
26      )
27      self.layer5 = nn.Sequential(
28          nn.Linear(256, 128),
29          nn.ReLU()
30      )
31      self.layer6 = nn.Sequential(
32          nn.Linear(128, 10)
33      )
34
35    def forward(self, x):
36      out = self.layer1(x)
37      out = self.layer2(out)
38      out = self.layer3(out)
39      out = out.reshape(out.size(0), -1)
40      out = self.layer4(out)
41      out = self.layer5(out)
42      out = self.layer6(out)
43      return out
```

Klasa $convNet$ predstavlja opisanu arhitekturu konvolucijske neuronske mreže. Izvedena je iz osnovne klase $torch.nn.Module$ koja je baza za sve modele neuronskih mreža. U konstruktoru se definraju svi potrebni slojevi. Korišten je kontenjer $torch.nn.Sequential$ radi preglednosti. U metodi $forward$ se odvija unaprijedni prolaz podataka kroz slojeve neuronske mreže.

7.2. Treniranje

U nastavku se nalazi programski kod za treniranje konvolucijske neuronske mreže definirane programskim kodom u 7.1

```
1 accelerator = 'cuda' if path exists('/opt/bin/nvidia-smi')else 'cpu'
2 model = convNet().to(accelerator)
3 criterion = nn.CrossEntropyLoss()
```

```
optimizer = torch.optim.Adam(model.parameters(), lr = 0.001)

for epoch in range(NUM_EPOCH):
    for i, (images, labels) in enumerate(trainloader):
        images = images.to(accelerator)
        labels = labels.to(accelerator)
        outputs = model(images)
        loss = criterion(outputs, labels)
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()
```

$accelerator$ predstavlja ili CPU ili GPU (u ovisnosti što postoji). Nadalje, definiramo $criterion$ koji označava funkciju gubitka opisanu u 2.4.1. $optimizer$ označava Adamov gradijentni spust opisan u [7]. Stopa učenja je 0.001. Pozive funkcije $optimizer.zero\_grad()$ čisti prijašnje izračunate gradijente za svaki parametar dok $loss.backward()$ računa nove gradijente za svaki parametar algoritmom unaprijedne propagacije unatrag opisanim u 2.4.2. Poziv $optimizer.step()$ ažurira težine.

8. Zaključak

Klasifikacija slika je problem računalnog vida s mnogim zanimljivim primjenama. U posljednje vrijeme najbolji rezultati u tom području postižu se pristupima utemeljenima na dubokim konvolucijskim modelima. Ovaj rad usredotočuje se na nadzirane pristupe za strojno očitavanje znamenki. Dan je opis općenito neurona, neuronskih mreža i nadalje konvolucijskih neuronskih mreža koje su korištene pri klasifikaciji slika kućnih brojeva što je i glavna tema ovog rada. Dan je opis konvolucijske neuronske mreže koja je bila korištena na skupovima SVHN i MNIST. Prikazani su relevantni rezultati brojčano i grafički. Na skupu SVHN prema [4] trenutno najbolji rezultat je greška od 1.69% što je ukupno 98.31% točnosti. U radu je dobiveno 91.80% na tom skupu. Na skupu MNIST prema [4] je trenutno najbolji rezultat greška od 0.21% što je ukupno 99.79% točnosti dok je u radu dobiveno autoru fascinantnih 99.52% te se tako svrstalo u sam vrh najboljih rezultata.

Možemo zaključiti da duboki konvolucijski modeli jako dobro rješavaju probleme klasifikacije slika kućnih znamenki te rukom pisanih znamenki. Također, važno je istaknuti da trenutno najbolji rezultati u svijetu u području klasificiranja objekata ponekad nadmašuju ljude, pa je tako primjerice na skupu SVHN procijenjeno da je ljudska točnost oko 98% dok najbolji duboki konvolucijski modeli na tom skupu nadmašuju 98%.

LITERATURA

[1] *CS231n: Convolutional Neural Networks for Visual Recognition.* URL http://cs231n.stanford.edu/.

[2] *Umjetna inteligencija, Umjetne neuronske mreže.* URL http://java.zemris.fer.hr/nastava/ui/ann/ann-20180604.pdf.

[3] *PyTorch Tutorials.* URL https://pytorch.org/tutorials/.

[4] *What is the class of this image?* URL http://rodrigob.github.io/are_we_there_yet/build/classification_datasets_results.html.

[5] Ian J. Goodfellow, Yaroslav Bulatov, Julian Ibarz, Sacha Arnoud, i Vinay Shet. Multi-digit number recognition from street view imagery using deep convolutional neural networks, 2013.

[6] Sergey Ioffe i Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift, 2015.

[7] Diederik P. Kingma i Jimmy Ba. Adam: A method for stochastic optimization, 2014.

[8] K. Kralj. *Klasifikacija slika dubokim konvolucijskim modelima.* URL http://www.zemris.fer.hr/~ssegvic/project/pubs/kralj17bs.pdf.

[9] Alex Krizhevsky, Ilya Sutskever, i Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. U F. Pereira, C. J. C. Burges, L. Bottou, i K. Q. Weinberger, urednici, *Advances in Neural Information Processing Systems 25*, stranice 1097–1105. Curran Associates, Inc., 2012. URL http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-netwo
pdf.

[10] Y. LeCunn, C. Cortes, i C. Burges. *The MNIST Database*, . URL http://yann.lecun.com/exdb/mnist/.

[11] Y. LeCunn, P. Haffner, L. Bottou, i Y. Bengio. *Object Recognition with Gradient Based Learning*, . URL http://yann.lecun.com/exdb/publis/pdf/lecun-99.pdf.

[12] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, i Ng A. *Reading Digits in Natural Images with Unsupervised Feature Learning*. URL http://ufldl.stanford.edu/housenumbers/nips2011_housenumbers.pdf.

[13] D. Smolčić. *Raspoznavanje objekata konvolucijskim neuronskim mrežama*. URL http://www.zemris.fer.hr/~ssegvic/project/pubs/smolcic15bs.pdf.

[14] V. Vukotić. *Raspoznavanje objekata dubokim neuronskim mrežama*. URL http://www.zemris.fer.hr/~ssegvic/project/pubs/vukotic14ms.pdf.

[15] S. Šegvić. *Predavanja iz predmeta duboko učenje*.

Klasifikacija slika kućnih brojeva dubokim konvolucijskim modelima

Sažetak

Rad započinje opisom općenito neuronskih mreža, neurona te njihove inspiracije. Nastavlja se opisom konvolucijskih neuronskih mreža te njihove primjene na klasifikaciju objekta. U nastavku je dan pregled programskog okvira PyTorch. Nadalje, dani je prikaz rezultat implementirane konvolucijske neuronske mreža skupovima SVHN, MNIST te njihova programska implementacija. Dodatno, objašnjen je koncept neprijateljskih primjera te prikazan jedan konkretan neprijateljski primjer

Ključne riječi: Umjetne neuronske mreže, duboke neuronske mreže, konvolucijske neuronske mreže, duboko učenje, strojno učenje, klasifikacija, kućni brojevi, SVHN

Classification of Street-View House Numbers With Deep Convolutional Models

Abstract

This works starts with a general description of artificial neural networks, neurons and their inspiration. It continues with a description of convolutional neural network and their application to a classification of objects. Then, it is given an overview of library PyTroch. Then, it is given an overview of the results implementing convolutional neural network on datasets SVHN, MNIST and their code in PyTorch. Additionally, concept of adversarial examples is shown and one concrete adversarial examples is given.

Keywords: Artifical neural networks, deep neural network, convolutional neural network, deep learning, machine learning, classification, house numbers, SVHN