SVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 3945

Vrednovanje raspoznavanja znamenki i slova konvolucijskim neuronskim mrežama

Mislav Larva

Zagreb, lipanj 2015.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA
ODBOR ZA ZAVRŠNI RAD MODULA

Zagreb, 10. ožujka 2015.

ZAVRŠNI ZADATAK br. 3945

Pristupnik: **Mislav Larva (0036466909)**
Studij: Računarstvo
Modul: Računarska znanost

Zadatak: **Vrednovanje raspoznavanja znamenki i slova konvolucijskim neuronskim mrežama**

Opis zadatka:

Područje rada je klasificiranje izdvojenih okana stvarnih slika u različite vrste objekta ili pozadinu. Pretpostavljamo da izdvojena okna sadrže znamenke ili neke od predodređenih kombinacija slova. Automatsko pronalaženje okana za klasificiranje u ulaznim slikama nije predmet interesa ovog rada. Glavni zadatci rada su ručno označavanje okana za učenje, validaciju i testiranje te analiza, eksperimentalno vrednovanje i moguće poboljšanje postojećih izvedbi raspoznavanja temeljenih na konvolucijskim neuronskim mrežama.
U okviru rada, potrebno je iz literature proučiti različite pristupe za ostvarivanje raspoznavanja objekata. Posebnu pažnju posvetiti postupcima koji se temelje na konvolucijskim neuronskim mrežama. Uhodati postupak ručnog označavanja položaja objekata u slikama i zadavanja njihovog točnog razreda. Označavanje se mora moći provesti i u slučaju kad su objekt prizvoljno zakrenuti oko osi projekcije kamere. Ručno označiti skupove slika za učenje, validaciju i testiranje. Analizirati prethodno razvijenu izvedbu raspoznavanja konvolucijskim neuronskim mrežama. Validirati hiperparametre i vrednovati generalizacijsku točnost postupka. Prikazati i ocijeniti ostvarene rezultate.
Radu priložiti izvorni i izvršni kod razvijenih postupaka, ispitne slijedove i rezultate, uz potrebna objašnjenja i dokumentaciju. Citirati korištenu literaturu i navesti dobivenu pomoć.

Zadatak uručen pristupniku: 13. ožujka 2015.
Rok za predaju rada: 12. lipnja 2015.

Mentor:

Izv. prof. dr. sc. Siniša Šegvić

Djelovođa:

Doc. dr. sc. Tomislav Hrkać

Predsjednik odbora za završni rad modula

Prof. dr. sc. Siniša Srbljić

Sadržaj

POPIS SLIKA

1. Uvod

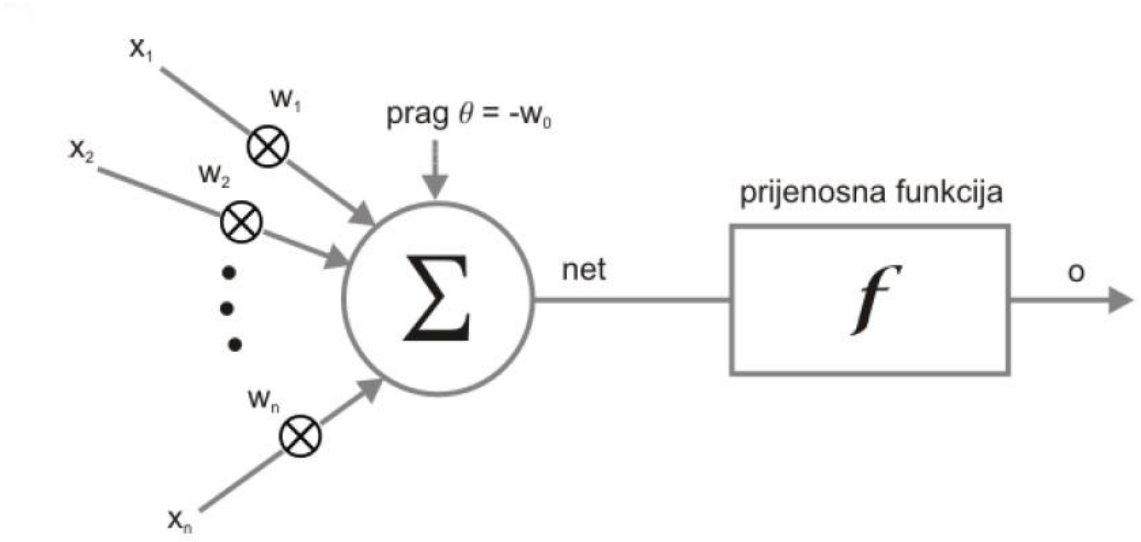
Cilj završnog rada je raspoznavanje znamenki i brojeva konvolucijskim neuronskim mrežama. Kako bi se to ostvarilo, potrebno je znanje i razumijevanje umjetne inteligencije. Umjetna inteligencija je disciplina koja se bavi ostvarivanjem inteligentnih sustava koji implementiraju značajke ljudskog ponašanja koja se smatraju inteligentnima. Može se napraviti podjela na slabu i jaku. Slaba umjetna inteligencija implementira sustave koji imaju samo neke određene značajke ljudskog ponašanja, mogu izvršavati zadatke na ograničenoj razini i nisu svjesna svoga rada. Jaka umjetna inteligencija želi ostvariti sustav koji će moći razmišljati kao ljudi i imati određeni stupanj samosvijesti.

Logičan korak pri pokušaju stvaranja umjetne inteligencije je imitiranje ljudskog mozga te su na toj osnovi razvijene umjetne neuronske mreže. Umjetne neuronske mreže čini skup neurona koji su u interakciji. Logika rješavanja zadataka razlikuje se od klasičnih analitičkih metoda rješavanja. Zadatak se obavlja promjenom i namještanjem parametara neurona. Mreža na ulazu dobiva skup značajki za učenje, prilagođava se problemu i pokušava dati očekivani izlaz. Zbog toga je lako prilagodljiva različitim problemima, a najveću uporabu je pronašla u zadacima raspoznavanja i klasifikacije.

Kako bi rad dao dobre rezultate bilo je potrebno nabaviti veći skup slika brojeva za učenje, testiranje i provjeru. Slike je trebalo prvo pripremitit, a za treniranje mreže koristi se sustav Caffe.

2. Umjetne neuronske mreže

Neuro-računarstvo je područje koje se bavi umjetnim neuronskim mrežama. Početak je vezan uz članak *A Logical Calculus of Ideas Immanent in Nervous Activity* autora Warrena McCullocha i Waltera Pittsa iz 1949. godine. Članak prikazuje prvi matematički model neuronske mreže. Osnovna građevna jedinica je neuron kojeg se često naziva po autorima *McCulloch-Pitts neuron.* Takav pristup razvoju umjetne inteligencije naziva se konektivistički pristup. Temelji se na izgradnji sustava arhitekture slične arhitekturi mozga koji, umjesto da ga se programira, uči samostalno na temelju iskustva. *Frank Rosenblatt* je američki psiholog zaslužan za otkriće jednoslojne neuronske mreže perceptron. Takvoj mreži moguće je uspješno podešavati težinske koefucijente. Prva mreža napravljena je oko 1958.godine. Iako dobar model, autori *Marvin Minsky* i *Seyour Papert* u knjizi *Perceptrons* objavljuju matematički model koji objašnjava kako takva mreže ne može naučiti jednostavnu funkciju XOR. Zatim je nastao kratki zastoj u njihovom razvoju, da bi se opet vratile krajem 80-tih i 90-tih godina uz napretke kao što su backpropagation algoritam i više slojeva unutar neuronske mreže.



Slika 2. 1: Model umjetnog neurona

2.1. Umjetni neuron

Osnovna građevna jedinica živčanog sustava je neuron pa će onda i umjetni neuron biti osnovna građevna jedinica umjetne neuronske mreže. Prirodni neuron čine: soma, dendriti, akson, završni živci, tijelo stanice i sinapse preko kojih se dendriti neurona spajaju s drugim neuronima. Povlače se iduće veze. Dendriti predstavljaju ulaze koje se dovode do neurona. Jakost sinapse predstavlja se s težinskim faktorom ω. Tijelo stanice je zbrajalo. Akson je prijenosna (aktivacijska) funkcija. Ulazi se predstavljaju sa x_1 do x_n kao i težinske funkcije od ω_1 do ω_n koje se množe s njima. Izlaz iz tijela stanice naziva se net koji predstavlja ulaz za prijenosnu funkciju. Model umjetnog neurona prikazan je na slici 2.1.

Ukupni net se računa prema izrazu:

$$net = \omega_1 x_1 + \omega_2 x_2 + \cdots + \omega_n x_n - \theta \tag{2.1}$$

Theta (θ) predstavlja prag paljenja neurona. Prag je ekvivalentan pomicanju aktivacijske funkcije po apscisi. Dodavanjem slobode pomicanja po apscisi proširuje se područje djelovanja neurona. Čime se mogu lakše modelirati ulazi neurona čije težište nije u ishodištu (uz pretpostavku aktivacijske funkcije koja je simetrična oko nule).

Želja je uvijek pojednostavniti postupak računanja. Budući izraz (2.1) koristi puno zbrajanja moguće je ga je oblikovati kao sumu. Kako bi to bilo moguće prvo treba preoblikovati parametar θ. Kao zamjenu za njega uzima se ω_0 , a za ulaz koji se množi s tom prividnom težinom uzima se x_0 koji je postavlja u $x_0 = 1$. Tada se izraz može napisati kao:

$$net = \omega_0 x_0 + \omega_1 x_1 + \omega_2 x_2 + \cdots + \omega_n x_n \tag{2.2}$$

$$net = \sum_{i=0}^{n} \omega_i x_i \tag{2.3}$$

$$net = \vec{\omega}^T * \vec{x} \tag{2.4}$$

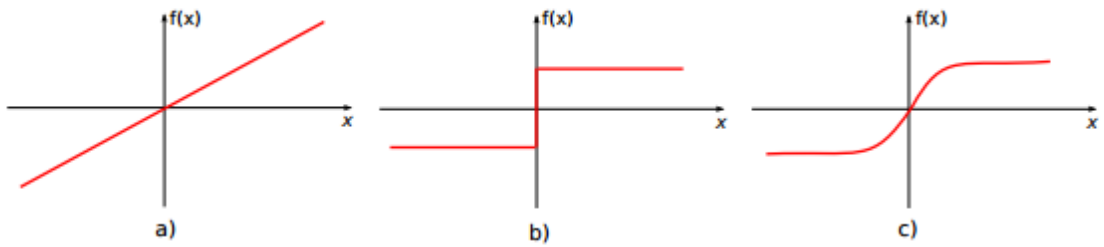
Zatim je još potrebno izračunati izraz za prijenosnu funkciju čiji je izlaz o :

$$o = f(\sum_{i=0}^{n} \omega_i x_i) = f(net) \tag{2.5}$$

Ili:

$$o = f(net) = \left([\omega_n \dots \omega_1 \, \omega_0] * \begin{bmatrix} x_n \\ . \\ . \\ . \\ x_1 \\ 1 \end{bmatrix} \right) \tag{2.6}$$

Kao prijenosne funkcije mogu se odabrati razni oblici kao što su: a) linearna funkcija, b) funkcija praga i c) sigmoidalna funkcija. Kako one izgledaju može se vidjeti na slici 2.2. Različiti oblici funkcija su pogodni za različite vrste zadataka. Linearne funkcije se uglavnom koriste u izlaznim slojevima kada je potreban neograničeni izlaz. Neograničen izlaz je često potreban kod regresijskih problema, no u klasifikacijskim problemima neograničen izlaz samo bi otežavao učenje. Kod klasifikacijskih problema smislenije je ograničavati izlaze neurona na male veličine (klasifikacija se vrši odabirom najvećeg izlaza). Funkcije skoka i sigmoidalne funkcije puno su bolje kod klasifikacijskih problema koje rješavaju neuronske mreže. [4]



Slika 2. 2: Prijenosne funkcije

Učenje neuronske mreže odrađuje se pomoću *backpropagation* algoritma. To je metoda pomoću koje se računa gradijent i određuje greška u svakom sloju te se pomoću tih rezultata ažuriraju težine neurona. Funkcija mora biti derivabilna kako bi to bilo moguće.

Izgled sigmoidalne funkcije:

$$f(net) = \frac{1}{1+e^{-a*net}} \tag{2.7}$$

Parametar a predstavlja strminu krivulje u točki infleksije. Važna je još i derivacija koja je zadana izrazom:

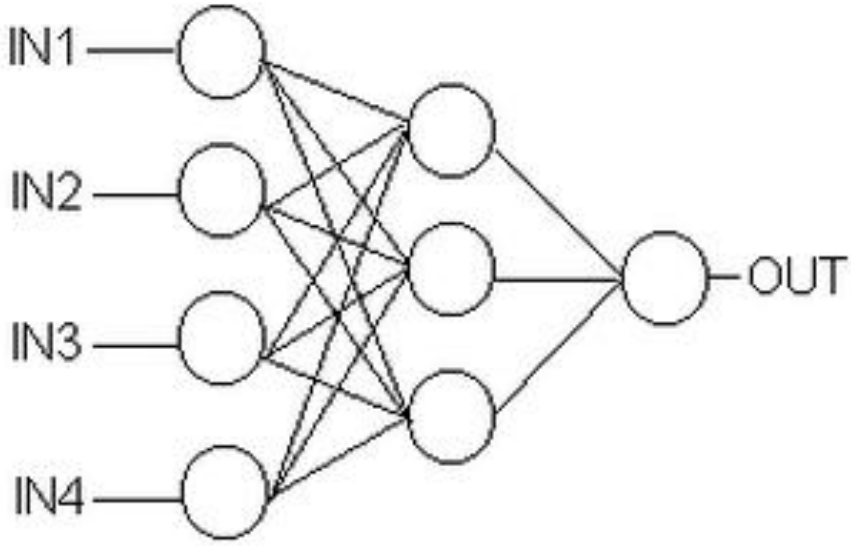
$$df(x)/_x = f(x)\,[1 - f(x)] \tag{2.8}$$

2.2. Model neuronske mreže

Arhitekturu mreže predstavlja posebno uređenje i povezivanje neurona u obliku mreže. Neuronske mreže se razlikuju prema broju neuronskih slojeva. Najčešće, ulazi za svaki sloj su izlazi iz prethodnog sloja, a svoje izlaze šalju idućem sloju. Prvi sloj naziva se ulazni, posljednji sloj naziva se izlazni, a srednji slojevi nazivaju se skrivenim slojevima. Kao jednostavniji primjer, na slici 2.3 vidljiva je neuronska mreža s tri sloja. Prvi sloj (jedini ima dodira s vanjskim svijetom) prima signale iz okruženja te mu je jedini zadatak predati signale skrivenom sloju. Skriveni sloj obrađuje primljene podatke te ih šalje idućem sloju (idući sloj može biti i dalje skriveni ili može biti izlazni sloj). Izlazni sloj prima podatke od skrivenog sloja, a njegovi izlazi su konačni rezultati. Složenije neuronske mreže osim što imaju više slojeva, imaju i veći broj neurona, povratne petlje i elemente za odlaganje vremena.

Bitna razlika među slojevima je to što ulazni i izlazni broj imaju točno definiran broj neurona koji ovisi o dimenziji ulaznog vektora $\vec{x}$ i izlaznog vektora $\vec{y}$. Kod

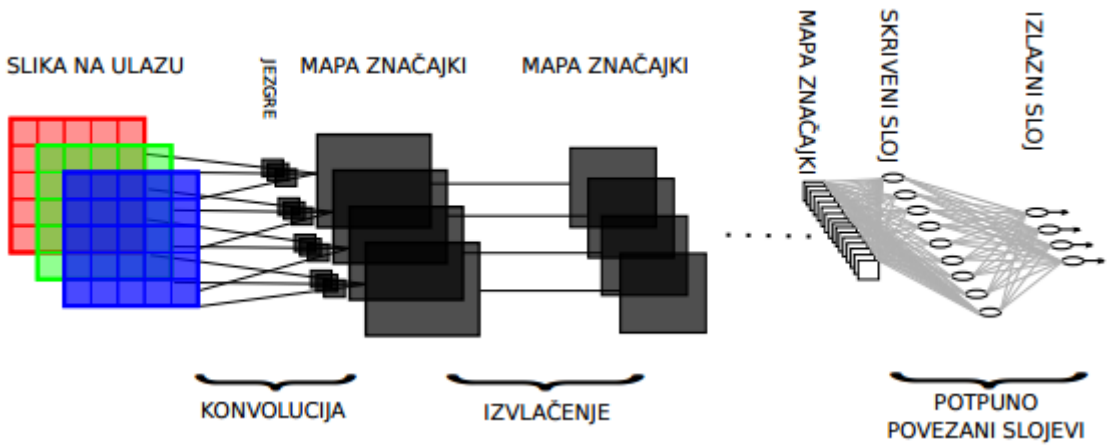
perceptrona, ako učitavamo sliku, tada će svakom neuronu ulaznog sloja biti proslijeđen jedan piksel slike.



Slika 2. 3: Neuronska mreža s tri sloja

3. Konvolucijske neuronske mreže

Konvolucijske neuronske mreže pogodne su za analizu slike odnosno prepoznavanje objekata. Potrebne su određene promjene naspram perceptrona. Izlaz se više neće gledati kao skalar nego se radi proširenje na $\mathbf{R^2}$. Takve izlaze ćemo nazivati mapama značajki Na slici 3.1 vidljiva je struktura konvolucijske neuronske mreže, a umjesto izraza „težina“ koristit će se naziv jezgre. Na ulaz se dovodi slika i zatim naizmjenice slijede konvolucijski slojevi i slojevi sažimanja. Na kraju se nalazi potpuno povezani sloj koji zapravo predstavlja perceptron.



Slika 3. 1: Konvolucijska neuronska mreža

Kao najpogodnija prijelazna funkcija koristi se skalirana sigmoidalna funkcija hiperbolnog tangensa [1]. Funkcija je zadana izrazom :

$$f(x) = 1.7159tanh\left(\frac{2}{3}x\right) \tag{3.1}$$

Derivacija:

$$f'(x) = 1.444\left(1 - tanh^2\left(\frac{2}{3}x\right)\right) \tag{3.2}$$

Konvolucijski slojevi uvijek imaju vezu jedan-na-više s prethodnim slojem i uče težine u jezgrama s kojima obavljaju konvoluciju. Slojevi sažimanja imaju vezu jedan-na-jedan s prethodnim slojem i ne uče nove vrijednosti, jedino mogu pamtiti način propagacije greške.

3.1. Konvolucijski sloj

Sloj na ulaz dobiva sliku (ako se radi o prvom sloju) ili uzorak i radi dvodimenzionalnu konvoluciju s jezgrom. Koristit će se iduće oznake. Mapa značajki označavat će se s M tako što će M^l predstavljati mapu trenutnog sloja, a M^{l-1} mapu prijašnjeg sloja. M_ω predstavlja širinu mape, a M_h visinu mape. S oznakom K umjesto M označavat će se je jezgra s oznakama K_ω *i* K_h za širinu odnosno visinu jezgre. Matrice jezgre najčešće su kvadratne. Još su potrebni pomaci jezgre prilikom konvolucije pa će se tako sa S_ω označavati pomak po širini i sa S_h pomak po visini.

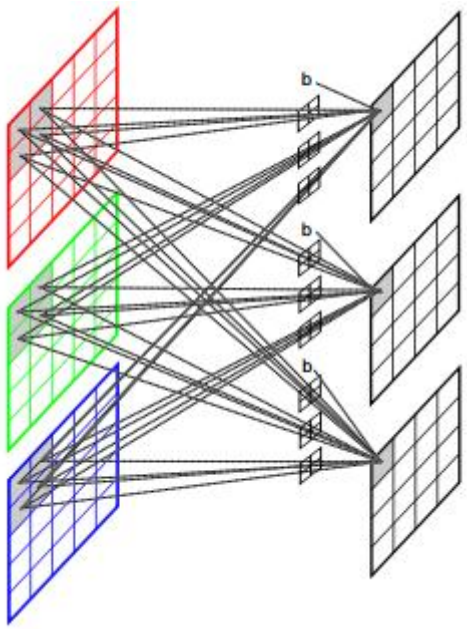
Izrazi za veličine mapi značajki u nekom sloju po širini i visini:

$$M_\omega^l = \frac{M_\omega^{l-1} - K_\omega}{S_\omega} + 1 \tag{3.4}$$

$$M_h^l = \frac{M_h^{l-1} - K_h}{S_h} + 1 \tag{3.5}$$

Način rada sličan je perceptronu. Kroz ulaznu mapu prolazi se s prozorom veličine jezgre. Započinje se s lijevim gornjim kutem. Čitaju se vrijednosti ulaza koje se množe s elementima matrice jezgre. Dobivene vrijednosti se sumiraju, dodaje im se još i vrijednost praga te se rezultat sprema u mapu značajki. Postupak se nastavlja s pomakom u desno za S_ω dok se ne dođe do desnog ruba slika, a zatim se radi i pomak za S_h prema dolje. U ovom slučaju objašnjen je primjer sa samo jednom ulaznom mapom što uglavnom nije slučaj. Ranije je napisano da konvolucijski slojevi imaju vezu jedan na više s prethodnim slojevima. Budući se trenutni sloj

povezuje s prethodnim slojem pomoću jezgre, to znači da treba biti jezgri koliko je ulaznih mapa. Također, svaka mapa značajki ima jedan prag (θ) koji se zbraja sa svim jezgrama za neku lokaciju u mapi značajki. Primjer je vidljiv na slici 3.2.



Slika 3. 2: Konvolucija potpuno povezane mreže

Potrebno je još obaviti normalizaciju. Normalizacija je postupak kojim se svaki element iz izlazne matrice dijeli sa sumom apsolutnih vrijednosti u jezgri. To nam osigurava da su vrijednosti piksela u izlaznoj slici iste relativne veličine kao i oni iz ulazne slike.

Pri postupku konvolucije događaju se situacije u kojima jezgra zahtijeva od ulaza piksele koji se nalaze izvan granice slike. Jedna od metoda rješenja problema je proširivanje (engl. *extand*). Postupak funkcionira tako da se kutni pikseli proširuju s 90^0 stupnjevitim pikselima, odnosno dobivamo površinu. Na taj način pikseli iznad, lijevo i dijagonalno od kutnog piksela u gornjem lijevom kutu poprimaju njegove vrijednosti. Koliko će takvih piksela biti „napravljeno“ ovisi o tome koliko je potrebno piksela izvan slike. Ukoliko pikseli nisu kutni, oni se proširuju u linijama.

3.2. Sloj sažimanja

Prvi takav sloj započinje nakon prvog konvolucijskog sloja koji je za ulaz dobio sliku. Sažimanje (*engl. pooling*) smanjuje rezoluciju mapi značajki. Rezultat je neosjetljivost na manje pomake značajki u uzorku ili slici. Potrebno je poznavati elemente matrice mape značajki nakon čega ih je potrebno grupirati u grupe veličine $S_\omega \times S_h$ i predstaviti jednom vrijednošću. Neke od metoda sažimanja su: **sažimanje usrednjavanjem**, **sažimanje maksimalnom vrijednošću**, **sažimanje Gaussovim usrednjavanjem.**

Sažimanje usrednjavanjem (engl. *Mean-pooling*) grupira vrijednosti te od njih radi aritmetičku sredinu. Tako da za slučaj mape veličine 4×4 metoda uzima četiri grupe veličine 2×2 (ako se pretpostavi da su vrijednosti $S_\omega = 2$ i $S_h = 2$) i računa aritmetičku sredinu. Budući da je postupak odrađen četiri puta, na kraju se dobiva mapa veličine 2×2. Primjer na slici 3.3.

$$\begin{bmatrix} 4 & 2 & 2 & 6 \\ 3 & 3 & 5 & 3 \\ 3 & 2 & 2 & 4 \\ 5 & 2 & 6 & 4 \end{bmatrix} \rightarrow \begin{bmatrix} 3 & 4 \\ 3 & 4 \end{bmatrix}$$

Slika 3. 3: Sažimanje usrednjavanjem

Sažimanje maksimalnom vrijednošću (engl. *Max-pooling*) je metoda koja unutar grupe koju treba sažeti jednostavno uzima najveću vrijednost. Danas se ta metoda najviše koristi. Primjer na slici 3.4.

$$\begin{bmatrix} 4 & 2 & 2 & 6 \\ 3 & 3 & 5 & 3 \\ 3 & 2 & 2 & 4 \\ 5 & 2 & 6 & 4 \end{bmatrix} \rightarrow \begin{bmatrix} 4 & 6 \\ 5 & 6 \end{bmatrix}$$

Slika 3. 4: Sažimanje maksimalnom vrijednošću

Sažimanje Gaussovim usrednjavanjem je metoda slična sažimanju usrednjavanjem. Razlika je u izračunavanju „sredine“ koja se računa pomoću Gaussove jezgre G te se veće težine pridodaju elementima na sredini, a manje elementima koji su udaljeniji od središta.

3.3. Učenje konvolucijske neuronske mreže

Jednu od faza rada konvolucijskih neuronskih mreža čini učenje. Upravo zbog toga se takve mreže razlikuju od bilo kakvih drugih analitičkih metoda rješavanja problema. Razlikujemo dva načina učenja: **učenje s učiteljom** (engl. *supervised learning*) kod kojeg postoje primjeri u obliku (ulaz, izlaz) i **učenje bez učitelja** (engl. *unsupervised learning*) kod kojeg je izlaz a priori nepoznat. U ovom radu se provodi učenje s učiteljem.

Danas najčešće korišten algoritam za učenje je algoritam backpropagation (algoritam širenja greške unatrag) i temelji se na gradijentnom spustu. Rad se odvija u dva koraka. Prvo je potrebno odrediti par (ulaz, izlaz) i zatim se uzorak dovodi na ulaz mreže. Računa se odziv mreže i pogreška, a pogrešku dobivamo kao razliku odziva mreže i očekivanog izlaza. Drugi korak predstavlja širenje greške unazad te se mreža ažurira (mijenjaju se težine neurona) s ciljem smanjenja greške. Algoritam u svakom sloju računa gradijent i grešku. Prvo se izračunavaju greške izlaznog sloja. Zatim se za prethodni sloj određuje koliko je svaki neuron utjecao na greške u idućem sloju te se onda opet računaju njihove greške. Na kraju se određuje gradijent greške po pojedinim težinama te se one ažuriraju.

Radi boljeg razumijevanja prvo je potrebno uvesti oznake. Oznakom i označavat će se prethodni, a oznakom j trenutni sloj. Na taj način y_i označava izlaz prethodno sloja, y_j označava izlaz trenutnog sloja, z_j predstavlja ukupan ulaz trenutnog neurona te ω_{ij} predstavlja težinu koja spaja prethodni sloj s trenutnim.

Grešku trenutnog sloja računamo po formuli:

$$\frac{\partial E}{\partial z_j} = \frac{\partial E}{\partial y_j}\frac{\partial y_j}{\partial z_j} \tag{3.6}$$

Prvi dio desne strane jednadžbe predstavlja parcijalnu derivaciju greške po izlazu neurona, a drugi dio parcijalnu derivaciju izlaza po ulazu neurona. Greška se računa kao:

$$E = \frac{1}{2}\sum_j (t_{j-}\, y_j)^2 \tag{3.7}$$

Oznaka t_j označava očekivanu vrijednost na izlazu neurona j-tog sloja. Parcijalna derivacija greške po izlazu:

$$\frac{\partial E}{\partial y_j} = \frac{1}{2}\frac{\partial}{\partial y_j}\left(t_{j-}\, y_j\right)^2 = \frac{1}{2}\frac{\partial}{\partial y_j} t_j{}^2 - 2t_j\, y_j + y_j{}^2$$

$$= y_j - t_j = -(t_j - y_j) \tag{3.8}$$

Uzme li se na primjer logistička sigmoidalna funkcija $y_j = f(z_j) = \frac{1}{1+e^{-z}}$ desni dio izraza 3.7 izgledao bi:

$$\frac{\partial y_j}{\partial z_j} = \frac{\partial}{\partial z_j}\left(\frac{1}{1+e^{-z_j}}\right) = \frac{\partial}{\partial z_j}(1+e^{-z_j})^{-1} = \frac{-1(-e^{-z_j})}{(1+e^{-z_j})^2} =$$

$$= \frac{1}{1+e^{-z_j}}\,\frac{e^{-z_j}}{1+e^{-z_j}} = y_j\frac{e^{-z_j}}{1+e^{-z_j}} =$$

$$= y_j\frac{\left(1+e^{-z_j}\right)-1}{1+e^{z_j}} = y_j\frac{1+e^{-z_j}}{1+e^{-z_j}}\,\frac{-1}{1+e^{-z_j}} = y_j(1-y_j) \tag{3.9}$$

Nakon računanja greške trenutnog sloja izrazom (3.6), greška se propagira na prethodni sloj sumiranjem utjecaja neurona i na sve neurone trenutnog sloja j te dobivamo izraz:

$$\frac{\partial E}{\partial y_i} = \sum_j \frac{\partial E}{\partial z_j}\frac{\partial z_j}{\partial y_i} = \sum_j \omega_{ij}\frac{\partial E}{\partial z_j} \tag{3.10}$$

Zatim je još potrebno odrediti parcijalne derivacije po težinama:

$$\frac{\partial E}{\partial \omega_{ij}} = \frac{\partial E}{\partial z_j}\frac{\partial z_j}{\partial \omega_{ij}} = \frac{\partial E}{\partial z_j} y_i \tag{3.11}$$

Potrebno je još težine ažurirati u ovisnosti o stopi učenja η:

$$\omega_{ij} \leftarrow \omega_{ij} - \eta \frac{\partial E}{\partial \omega_{ij}} \tag{3.12}$$

3.4. Gradijentni spust

Gradijentni spust može se podijeliti u dvije glavne varijante. To su **standardni (grupni) gradijentni spust** (engl. batch gradient descent) i **stohastički gradijentni spust** (engl. online/stochastic gradient descent)**.** Odabir varijante ovisi o trenutku ažuriranja težina.

Standardni gradijenti spust predstavlja varijantu u kojoj se najprije izračunavaju potrebne promjene težina $\Delta\omega$ koje nastaju svim uzorcima u skupu za učenje ili grupi, a nakon toga se vrši ažuriranje težina. Takav pristup je brži, ali češće zapinje u lokalnim optimumima.

Stohastički gradijenti spust je varijanta kod koje se nakon svakog uzorka ažuriraju težine. Ovakva varijanta više oscilira pa je zbog toga otpornija na zapinjanje u lokalnim optimumima.

3.5. Specifičnosti konvolucijskih neuronskih mreža

Konvolucijske neuronske mreže (za razliku od klasičnih neuronskih mreža), imaju svojstvo dijeljenja težina. To se dešava u konvolucijskim slojevima gdje konvolucijska jezgra utječe na sve neurone u izlaznoj mapi. Zbog toga je prilikom računanja unazadne propagacije potrebno izračunati vrijednosti grešaka, kao i promjenu težina po svim neuronima mape te na kraju sumirati njihov utjecaj.

Kod unazadne propagacije slojeva sažimanja potrebno je propagirati grešku manje mape trenutnog sloja u veću mapu prethodnog sloja. Veza je jedan-na-jedan te nema učenja parametra. Slojevi sažimanja tijekom unazadne propagacije isključivo propagiraju grešku unazad te ne čine ništa drugo. Način propagacije

naravno ovisi o operaciji sažimanja koja se koristila prilikom unaprijedne propagacije.

Za primjer uzimamo metodu sažimanja maksimalnom vrijednošću. Budući da je postupak širenja greške unazad složen, potrebno je za svaku grupu označiti gdje se nalazio element koji je odabran kao maksimum. U primjeru u poglavlju 3.2 apsolutni zapis lokacija maksimalnih vrijednosti izgleda: $\mathbf{L} = \begin{bmatrix}(0,0) & (0,3)\\ (3,0) & (3,2)\end{bmatrix}$. Uzima se za primjer da matrica greške izgleda: $\mathbf{E} = \begin{bmatrix}0.15 & -0.13\\ 0.4 & -0.2\end{bmatrix}$. Unazadnom propagacijom dobiva se greška prethodnog sloja:

$$\mathbf{E} = \begin{bmatrix}0.15 & 0 & 0 & -0.13\\ 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0\\ 0.4 & 0 & -0.2 & 0\end{bmatrix}$$

4. Programska biblioteka Caffe

Caffe[1] je programska biblioteka i *framework* specijalizirana za duboke konvolucijske neuronske mreže. Napravio ju je Yangqing Jia tijekom svog doktorskog studija na Berkeleyu. Izvorni kod je objavljen pod BSD 2-Clause license. Glavne značajke ove biblioteke su definiranje strukture mreže kroz konfiguracijske datoteke te mogućnost treniranja mreža koristeći GPU. Korištena platforma se sastoji od operacijskog sustava Ubuntu 14.04 pokrenutog na virtualnoj mašini (*Vmware Player*) te prijenosnog računala ASUS K555L koje ima NVIDIA GeForce GT 840M grafičku karticu.

Opis postupka instalacije Caffe-a za izvršavanje na procesoru računala (engl. CPU). Moguće je instalirati Caffe i za izvršavanje na grafičkom procesoru (engl. GPU), ali je taj postupak složeniji te je nakon završenog postupka grafičko korisničko sučelje Unity (standardno sučelje operacijskog sustava Ubuntu) postalo neupotrebljivo te zbog navedenih razloga neće biti opisan ovaj postupak. Kako bi započeli rad, potrebno je instalirati potrebne programe i programske pakete bez kojih se ne može instalirati ni pokrenuti Caffe.

```
$ sudo apt-get install git
$ mkdir src; cd src/
$ git clone https :// github.com/BVLC/caffe.git
$ sudo apt-get install libatlas –base –dev libprotobuf –dev \
  libleveldb –dev libsnappy –dev libopencv –dev \
  libboosts –all –dev libhdf5 –serial –dev
$ sudo apt –get install libgflags –dev libgoogle –glog –dev \
  liblmdb –dev protobuf –compiler
```

Prethodne linije su nam potrebne ako želimo koristiti Caffe za CPU. Sljedeći paketi su potrebni ako se želi koristiti sučelje za programski *Python*.

[1] http://caffe.berkeleyvision.org/

```
$ sudo apt –get install python –dev python –numpy \
  Python –skimage python –protobuf ipython \
  Ipython –notebook ipython –qtconsole
```

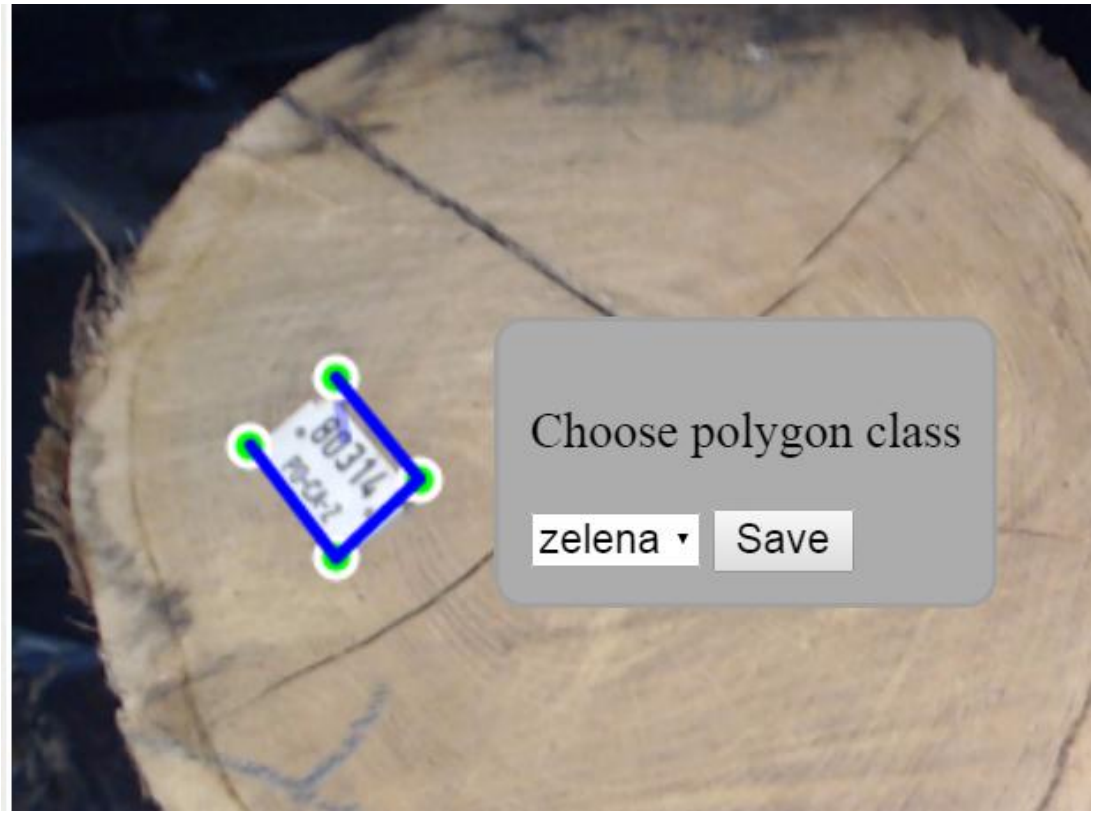
Još je potrebno izvesti kompajliranje izvornog koda.

```
$ cd caffe
$ cp Makefile.config.example Makefile.config
$ sed –i 's/# CPU_ONLY/CPU_ONLY/' Makefile.config
$ make –j4 all
$ make –j4 runtest
$ make –j4 pycaffe
$ make –j4 pycaffe
$ export PYTHONPATH=$CAFFE_ROOT/python
```

CAFFE_ROOT predstavlja putanju do direktorija gdje se nalazi izvorni kod Caffe-a. To nam omogućuje korištenje Python ljuske.

5. Izrezivanje označenih znamenki

5.1. Izrezivanje i rotacija markica



Slika 5. 1: Primjer označavanja markice i odabir klase

DrawMe[2] je Javascript biblioteka za crtanje poligona na slikama čiji je autor Jianxiong Xiao. Nakon instalacije, da bi radili sa slikama, prvo ih treba staviti na lokalni poslužitelj, a onda ih pokrenuti u web pretražitelju. Potrebno je odrediti klase u koje će se svrstavati poligoni. Broj klasa kao i boju po kojoj će se razlikovati potrebno je samostalno odabrati prije početka označavanja. Način rada: lijevom tipkom miša označava se prva točka na slici, zatim ponovno lijevom tipkom miša

[2] http://vision.princeton.edu/pvt/DrawMe/

odabire se druga točka. Pritisne li se nakon toga desna tipka miša otvara se spuštajući prozor u kojem je moguće odabrati jednu od klasa. Uzmu li se samo dvije točke program iscrtava pravokutnik kod kojega su označene dvije točke povezane dijagonalom nastalog pravokutnika. Nakon što je poligon nacrtan moguće je odrediti koordinate njegovih vrhova, što je važno za ovaj rad. Napravljene poligone moguće je brisati i uređivati. Primjer označavanja slike vidljiv je na na slici 5.1.

Za sam rad bilo je potrebno dodati još neke funkcionalnosti koje su nužne za daljnju obradu slike. Dodano je zaključavanje (engl. *mutex*) korisnika koji trenutno radi na slici. Bez toga bi se mogao dogoditi slučaj da dva korisnika istovremeno pristupe slici i pokušavaju raditi promjene (narušena konzistentnost). Dodana je beta verzija uvećavanja (engl. *zoom*) kako bi se omogućilo lakše označavanje markica. Prvotna verzija *DrawMe*-a nije imala mogućnost proizvoljnog dodavanja klasa kao i iskočni prozor na kojem se odabire klasa. Za kraj, omogućeno je i automatsko spremanje koordinata vrhova poligona. [7]

Zadatak je bio odrediti točne položaje markice (poligona kojim je označena etiketa s brojevima). Na označenim markicama brojevi mogu biti proizvoljno zakrenuti oko osi projekcije kamere. Nakon završetka rada, koordinate poligona spremaju se u .txt datoteku. Na primjer, ako se nacrta pravokutnik, koordinate su spremljene u obliku x: (x1, x2, x3, x4), y: (y1, y2, y3, y4). Datoteka također sadrži podatak radi li se o pravokutniku ili poligonu. Prvu koordinatu (x1, y1) predstavlja prva odababrana točka tijekom označavanja slike, drugu druga (x2, y2) i tako dalje. Primjer označene markice na slici 5.2.

Slika 5. 2: Primjer označene markice

Kako bi se mogla naučiti neuronska mreža prvo je trebalo na ulaz dovesti skup slika. Markice dobivene u DrawMe-u potrebno je dalje obraditi. Primjer markica vidljiv je na slici 5.2. Trebalo je odrediti trenutni kut kojeg markica zatvara sa x-osi i y-osi. Nakon toga, potrebno je odrediti u kakvom su odnosi x i y koordinata prve i druge točke. Zamisli li se markica koja stoji u vodoravnom položaju tako da su brojevi uspravno položeni (primjer: 124235), za prvu točku odabire se lijevi donji vrh poligona, a za drugu točku odabire se desni donji kut poligona. Sve markice su na taj način označene neovisno o kutu rotacije kamere. Markice moraju nakon postupka rotiranja biti u takvom položaju u kojem su brojevi uspravni.

Da bi se rotiranje odradilo ispravno bilo je potrebno odrediti međusobni položaj koordinata prve i druge točke. Postoje četiri slučaja, a to su:

- $x1 > x2$ i $y1 > y2$ prvi kvadrant
- $x1 < x2$ i $y1 > y2$ drugi kvadrant
- $x1 < x2$ i $y1 < y2$ treći kvadrant
- $x1 < x2$ i $y1 < y2$ četvrti kvadrant

Kut kojeg markica zatvara s x-osi računa se po formuli 5.1. Poznavanjem takvih ovisnosti i kuta, lako je bilo odrediti za koliki kut i u kojem smjeru treba napraviti rotaciju. Radi pojednostavljenja zadatka, prvo je rotirana cijela slika, a zatim izrezana markica. Za objašnjenu funkcionalnost korištena je OpenCV biblioteka za python.

$$\varphi = arctg\left[\frac{abs(y_2 - y_1)}{abs(x_2 - x_1)}\right] \tag{5.1}$$

Problem predstavlja nemogućnost točno preciznog označavanja markica čiji bi gornji brid trebao biti paralelan sa zamišljnom linijom koja bi bila postavljena na vrh preko svih brojeva u markici. Predstavljene su dvije slike (Slika 5.3) koje pokazuju primjer dobro i loše zarotirane markice. Prethodno opisani problem razlog je loše rotacije.

Slika 5. 3: Prva slika predstavlja dobro, a druga loše zarotiranu markicu

5.2. Izrezivanje brojeva iz markica

Zadnji korak prije nego što je moguće odrediti skup slika koje će predstavljati ulaze za konvolucijske neuronske mreže je izrezivanje brojeva iz markica. Problem predstavlja nemogućnost točno preciznog označavanja markica čiji bi gornji brid trebao biti paralelno sa zamišljenom linijom koja bi bila postavljena na vrh preko svih brojeva u markici (isti problem se javlja kod rotiranja slike koje je objašnjeno u prethodnom poglavlju).

Izrezivanje brojeva podijeljeno je u par koraka. Prvo je trebalo napraviti binarizaciju slike (crno bijela slika). Nakon binarizacije provodi se postupak povezanih komponenti odnosno spajanje susjednih pixela u jednu cjelinu. Budući da se na markici nalazi pet brojeva rezultat mora biti pet povezanih komponenata. Kako bi se znalo što treba izrezati iz markice mora se napraviti zamišljeni pravokutnik oko broja pomoću čijih vrhova će biti izrezani brojevi. Za svaku komponentu pronalaze se pixeli koji se nalaze krajnje lijevo, desno, gore i dolje unutar broja te se pomoću njih određuju vrhovi zamišljenog pravokutnika. Primjer izrezanih brojeva vidljiv je na slici 5.4.

Slika 5. 4: Primjer izrezanih brojeva

6. Ispitni skupovi

6.1. MNIST

Skup MNIST (engl. *Mixed National Institute of Standards and Technology*) [8] sadrži skup rukom pisanih brojeva. Brojevi su podijeljeni u deset klasa od nula do devet. Njegova prvotna svrha bila je testirati ranije sustave automatskog prepoznavanja rukom pisanih brojki kod bankovnih čekova. Slike su 8-bitne te veličine 28×28 točki. Kako bi prepoznavanje bilo optimalnije bilo je potreno napisati veliki broj znakova (70 000) od kojih je 60 000 korišteno za treniranje, a 10 000 za testiranje. Slika 6.1 pokazuje primjer rukom pisanih brojeva skupa MNIST.

Slika 6. 1: Skup rukom pisanih brojeva MNIST

7. Zaključak

Tema rada je vrednovanje raspoznavanja znamenki i slova konvolucijskim neuronskim mrežama. Za rad je trebalo pribaviti veći broj uzoraka (brojeva) kako bi se što točnije moglo naučiti mrežu. Uzorci su podijeljeni u skup za učenje i skup za testiranje.

Kako bi se dobili uzorci za konvolucijsku neuronsku mrežu potrebno je bilo sa slika trupaca ručno označiti markice (na kojima se nalaze slova i znamenke). DrawMe je Javascript biblioteka koja omogućava označavanje markica. Sama funckionalnost biblioteke nije bila dovoljna pa je bilo potrebno dodati nove funkcije koje su bile nužne za „izvlačenje“ potrebnih podataka sa slike. Dodano je automatsko spremanje koordinata markice, zaključavanje (engl. *mutex)* korisnika, odabir proizvoljno velikog skupa klasa u koje se spremaju različite markice (bijela pozadina s crnim slovima i crna pozadina s bijelim slovima), uvećanje slike (engl. *zoom*) te skočni izbornik za odabir klasa (zbog lakšeg i bržeg rada).

Nakon dobivenih markica bilo je potrebno izrezazi ih iz slike te zarotirati ih tako da brojevi budu u uspravnom položaju. To je bilo moguće odraditi pomoću poznatih koordina vrhova markice.

Učenje mreže se izvršavalo pomoću programse biblioteke Caffe, specijalizirane za duboke konvolucijske neuronske mreže.

LITERATURA

[1] Umjetna inteligencija, 29.rujan.2014

URL http://hr.wikipedia.org/wiki/Umjetna_inteligencija

[2] Marija M. Umjetna inteligencija, užujak.2010.

URL http://www.mensa.hr/glavna/misli-21-stoljeca/479-umjetna-inteligencija-

[3] Ilić V.. Neuronske mreže, 1999.

URL http://solair.eunet.rs/~ilicv/neuro.html

[4] Yann LeCun, Léon Bottou, Yoshua Bengio, i Patrick Haffner. Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11):2278– 2324, 1998.

[5] Jurić-Kavelj M. Određivanje smjera gledanja kovolucijskim neuronskim mrežama. Diplomski rad, 2014.

[6] Vukotić V. Raspoznavanje objekata dubokim neuronskim mrežama. Diplomski rad, 2014.

[7] Fabijanić I. Vrednovanje postupka semantičke segmentacije temeljenog na slučajnim šumama. Završni rad, 2015.

Vrednovanje raspoznavanja znamenki i slova konvolucijskim neuronskim mrežama

Sažetak

U radu su objašnjeni osnovni pojmovi potrebni za razumijevanje umjetne inteligencije, neuronskih mreža te konvolucijskih neuronskih mreža. Njihova glavna karakteristika je učenje na danom skupu uzoraka za razliku od ostalih analitičkih metoda rješavanja problema. Predmet rada je ,osim učenja neuronskih mreža, sakupljanje uzoraka koji se dovode na ulaz mreže. Korištena je Javascript biblioteka DrawMe za označavanje markica s brojevima te su razvijene funkcije za rotaciju i izrezivanje markica i brojeva. Za treniranje konvolucijske neuronske mreže koristi gotova biblioteka Caffe.

Ključne riječi: umjetna inteligencija, neuron, umjetne neuronske mreže, konvolucijske neuronske mreže, DrawMe, Caffe

Experimental evaluation of recognizing digits and letters by convolutional neural networks

Abstract

The paper explains the basic concepts necessary to understand artificial intelligence, neural networks and convolution neural networks. Their main caracteristic is learning on the given set of samples, unlike other analytical methods to solve problems. The subject of the paper is, besides learning neural networks, collecting samples which are being brought on network entrance. Javascript library DrawMe was used for tagging stamps with numbers and addition functions were made for rotation and cropping marks and numbers have been developed. For training on convolution neural networks is used library Caffe.

Key words: artificial intelligence, neuron, artificial neural network, convolution neural network, DrawMe, Caffe