SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 720

SUSTAV ZA PREPOZNAVANJE GOVORNIH NAREDBI U STVARNOM VREMENU NA RUBNIM UREĐAJIMA

Luka Balić

Zagreb, 14. veljače 2025.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 30. rujna 2024.

DIPLOMSKI ZADATAK br. 720

Pristupnik:	**Luka Balić (0036513380)**
Studij:	Računarstvo
Profil:	Računalno inženjerstvo
Mentor:	prof. dr. sc. Hrvoje Džapo

Zadatak: **Sustav za prepoznavanje govornih naredbi u stvarnom vremenu na rubnim uređajima**

Opis zadatka:

Proučiti metode i algoritme za analizu i klasifikaciju zvučnih sekvenci u stvarnom vremenu, s naglaskom na pristupe temeljene na dubokom učenju i primjenu u prepoznavanju govornih naredbi. Proučiti programske okvire za razvoj modela prikladnih za prepoznavanje govornih naredbi i implementaciju u ugradbenim računalnim sustavima s ograničenim resursima. Proučiti mogućnosti i ograničenja u primjeni procesora iz porodice ESP32 za implementaciju rješenja za prepoznavanje glasovnih naredbi u stvarnom vremenu te odabrati prikladnu procesorsku porodicu. Implementirati pokazni sustav koji će omogućiti prepoznavanje glasovnih naredbi u stvarnom vremenu. Razviti programsku potporu za mikrokontroler koja će omogućiti snimanje zvuka preko mikrofona, izvođenje modela za prepoznavanje glasovnih naredbi te poduzimanje jednostavnih akcija na svaku prepoznatu naredbu. Detaljno dokumentirati postupak treniranja i prilagodbe modela za izvođenje na ugradbenom računalnom sustavu. Eksperimentalno provesti ispitivanje značajki sustava i uspješnosti prepoznavanja glasovnih naredbi pod različitim uvjetima rada.

Rok za predaju rada: 14. veljače 2025.

Hvala svima...

Sadržaj

1. Uvod

1.1. Tiny ML

Modeli strojnog učenja revolucionarizirali su tehnologiju koju koristimo u svakodnevnom životu tako što su omogućili računalima učenje iz podataka kojima raspolažu u svrhu donošenja odluka u situacijama za koje nisu eksplicitno programirana. Tradicionalno su takvi modeli bili namijenjeni za računala visokih performanci i neograničavajućih resursa, međutim sve bržim razvojem IoT (eng. Internet of Things) područja te zahvaljujući pristupačnoj cijeni mikrokontrolerskih sustava, počela je prilagodba modela strojnog učenja za takve sustave. Na prvi pogled dva nespojiva svijeta su se susrela te pronašla svoju primjenu u raznovrsnim sustavima. Tiny ML (eng. Tiny Machine Learning) je naziv koji se odnosi na implementaciju modela strojnog učenja na uređaje ograničenih resursa kao što su mobilni telefoni i mikrokontroleri. Glavne karakteristike modela namijenjenih za takve sustave su relativno malen memorijski otisak, mogućnost odziva u stvarnom vremenu, smanjenje potrošnje i internetskog prometa te sigurnost [1]. Sustav implementiran kroz ovaj rad ima zadatak prepoznati unaprijed zadane glasovne naredbe te pokrenuti izvršavanje određenog posla vezanog uz specifičnu naredbu.

1.2. Opis problema

Okruženi smo digitalnim glasovnim asistentima kao što su Googleov Assistant, Appleova Siri te Amazonova Alexa. Ovakvi sustavi mogu u vrlo kratkom roku pružiti zatražene informacije te bez ikakvog problema komunicirati s osobom koja ih koristi. Za prepoznavanje i obradu ljudskog govora te dohvaćanje bitnih informacija zaduženi su modeli kojima je potrebna velika procesorska moć i dovoljno prostora za pohranu te se zbog toga taj dio posla odrađuje na serverskim računalima. Takav sustav podrazumijeva

konstantnu internetsku vezu te stabilni i dugotrajni izvor električne energije. Kada bi mobilni uređaji slali konstatan tok audio podataka na server, brzo bi ispraznili bateriju te nepotrebno koristili mobilne podatke za pristup internetu. Zbog toga su takvi sustavi osmišljeni da čekaju naredbu za početak komunikacije, a tek onda uspostave vezu sa serverom. Međutim, i dalje nam ostaje problem konstantne akvizicije ulaznih audio podataka te prepoznavanje naredbe kao što je ”Hey Google” ili nešto slično. U ovoj situaciji savršenu primjenu pronašli su procesori izrazito male potrošnje na kojima je moguće implementirati optimizirane modele strojnog učenja. Takav procesor bi konstantno akvizirao podatke s audio ulaza te lokalno, uz pomoć treniranog modela, čekao ključnu riječ te nakon prepoznavanja dao znak cijelom sustavu da se može ”probuditi” iz stanja niske potrošnje te odraditi svoj posao. Ovakvim pristup, postignuta je efikasnost, niska potrošnja, brz odaziv, smanjena je potrošnja internetskih podataka te možda i najvažnija stvar - privatnost. Naime, nema potrebe za konstantnim slanjem glasovnih podataka na server što omogućuje da na server dospiju samo glasovni isječci u trenucima u kojima želimo.

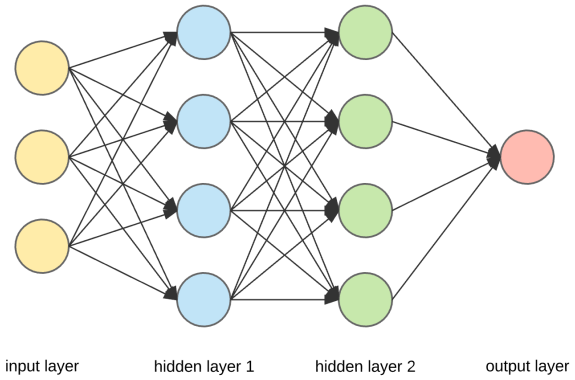
2. Struktura sustava

3. Neuronska mreža

3.1. Općenito

Okosnica cjelokupnog sustava za prepoznavanje izgovorenih naredbi je neuronska mreža. Neuronska mreža ili preciznije umjetna neuronska mreža je računalni model inspiriran biološkom strukturom neurona u mozgu čovjeka. Predstavlja jedan od najkorištenijih modela u dubokom učenju. Sastoji se od čvorova (neurona) i jednosmjernih veza između njih (sinapsa) koji na taj način tvore usmjereni graf. Čvorovi su grupirani u slojeve, a svaki od njih je povezan s čvorovima iz susjednog sloja na određeni način. Način na koji su određeni slojevi međusobno povezani određuje vrstu sloja.

Jednostavan primjer strukture neuronske mreže je potpuno povezani sloj (engl. fully connected layer ili dense layer) koji se često koristi kao osnovni građevni blok u umjetnim neuronskim mrežama [2]. U potpuno povezanom sloju svaki čvor jednog sloja povezan je sa svakim čvorom susjednog sloja. Ovakva struktura omogućava mreži fleksibilno učenje složenih odnosa između ulaznih i izlaznih podataka.



Slika 3.1. Potpuno povezani slojevi [2]

Na slici 3.1. prikazana je struktura neuronske mreže koja se sastoji od ulaznog sloja, dva potpuno povezana sloja te izlaznog sloja. Srednji slojevi (svi osim ulaznog i izlaznog)

se još nazivaju i skriveni slojevi jer kad koristimo model neuronske mreže, gledamo na njega kao na crnu kutiju koja na ulazu prima vrijednosti te na izlazu daje vrijednosti izračunate kroz sve skrivene slojeve [3].

Svaka veza između pojedinih čvorova ima određenu vrijednost koju nazivamo težina, a svaki čvor zapravo predstavlja funkciju koja može aktivirati svoj izlaz i vezu sa sljedećim čvorom.

$$a = f\left(\sum_{i=1}^{n} w_i x_i + b\right) \tag{3.1}$$

Jednadžba (3.1) modelira ponašanje pojedinog čvora u mreži. Aktivacijska funkcija f je vrlo bitna u odvajanju bitnih od nebitnih utjecaja pojedinih čvorova na sljedeći čvor. Također, ona nam omogućava modeliranje složenijih nelinearnih odnosa [4]. Kada bi čvor bio modeliran bez aktivacijske funkcije svako preslikavanje koje bi činio bi bilo linearno, a zbog toga što nam svaka kompozicija linernih funkcija daje opet linearnu funkciju, cjelokupna mreža ne bi bila sposobna modelirati kompleksnije stvari. Sastavnice modela čvora su sljedeće:

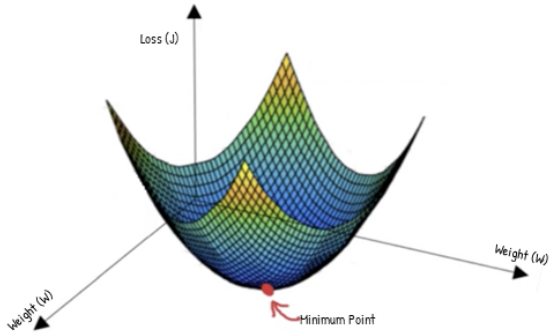
- a: izlazna vrijednost čvora (aktivacija)
- f: aktivacijska funkcija
- w_i: težina i-tog ulaznog čvora (čvor u prijašnjem sloju)
- x_i: vrijednost i-tog ulaznog čvora (njegova aktivacija)
- b: pomak
- n: broj čvorova u prijašnjem sloju koji imaju vezu s modeliranim čvorom

Povezivanjem više ovako definiranih čvorova gradimo neuronske mreže. Ulaz u neuronsku mrežu je informacija na temelju koje će na izlazu iz mreže biti vrijednost izračunata pomoću svih slojeva u mreži. Kako bi vrijednosti na izlazu iz mreže imale smisla, tj. davale korisnu informaciju, potrebno je trenirati mrežu. Treniranje mreže, u općem slučaju nadziranog strojnog učenja, podrazumijeva korištenje označenog skupa podataka koji je istog oblika kao i podaci koji će biti na ulazu u mrežu tijekom korištenja same

mreže. Arhitektura mreže (vrsta, veličina i broj slojeva) određena je prije samog treniranja, dok se težine, pomaci te samim time i razine aktivacija uče, tj. treniraju. Treniranje je proces u kojem se neuronska mreža "hrani" označenim skupom podataka (označeni skup prestavlja podatke za koje znamo što bi mreža trebala dati na izlazu) te provjerava koliko izlazi odstupaju od prave oznake. Na početku su sve težine uglavnom inicijalizirane na nulu. Podatak se predaje ulaznom sloju, prolazi kroz sve skrivene slojeve te na izlazu mreža izbaci određenu vrijednost koja se s očekivanom uspoređuje pomoću funckije gubitka. Takva funkcija predstavlja koliko izlazi iz mreže odstupaju od očekivanih.

Cilj svakog treniranja jest smanjiti vrijednost funkcije gubitka. Stoga se sve težine u mreži ažuriraju na način da njihove promjene pomaknu trenutno stanje mreže u smjeru negativne derivacije funckije gubitka (gradijentni spust). Takvim pristupom, mreža kroz iteracije s novim podacima smanjuje funkciju gubitka (efektivno daje sve točnije predikcije). Težine se ažuriraju od izlaznog sloja prema ulaznom (eng. backpropagation) jer na izlaz pojedinog sloja utječu njegovi ulazi, tj. izlazi prijašnjeg sloja (izlaz svakog sloja je funkcija izlaza prijašenjeg sloja, a kad izračunamo derivaciju funkcije, pomjenom njenih ulaza znamo u kojem smjeru će se mijenjati vrijednost same funkcije).

Bez smanjenja općenitosti, funkcija gubitka prikazana je na trodimenzionalnom grafu na slici 3.2. Složene arhitekture mreža će imati više dimenzija zbog većeg broja težina w_i. Cilj svakog treniranja mreže je doći što bliže minimumu ovakve funckije. Svakom iteracijom pomičemo se sve bliže minimumu, taakva vrsta optimizacije naziva se gradijentni spust.



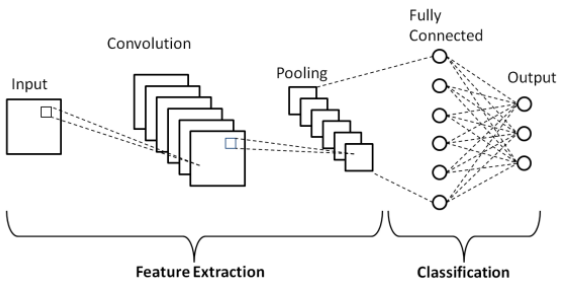
Slika 3.2. Funkcija gubitka [5]

3.2. CNN

Konvolucijska neuronska mreža (eng. Convolutional neural network, CNN) je vrsta umjetne neuronske mreže koja je pogodna je za obradu podataka s rešetkastom topologijom, a najviše se koristi za rješavanje problema iz područja klasifikacije slika te računalnog vida. Inspirirane su načinom funkcioniranja moždanog korteksa zaduženog za vid kod sisavaca [6]. Obrada slike koju vide sisavci funkionira hijerarhijski, tj. u mozgu se ne obrađuje cjelokupna slika odjednom, nego postoje jednostanije stanice koje su zadužene za prepoznavanje osnovnijih oblika koji se nakon toga stapaju u složenije i složenije. Na poslijetku organizam je u mogućnosti prepoznati cjelokupnu sliku koju gleda očima.

Računalni modeli koji koriste strukturu sličnu opisanoj mogu iz podataka koji su u takvom obliku izvući značajke samostalno što znači da nema potrebe za korištenjem metoda koje eksplicitno izvlače bitne značajke iz podataka [7]. Arhitektura najjednostavnije konvolucijske neuronske mreže uključuje:

- *Ulaz* (eng. Input Layer): ulazni sloj modela, prima matrični podatak
- *Konvolucijski sloj*: osnovni sloj modela. Njegov glavni zadatak je ekstrakcija značajki iz ulaznih podataka. Detaljnije je opisan u 3.2.1.
- *Sloj za poduzorkovanje*: smanjuje dimenzionalnost (vidi 3.2.2.)
- *Potpuno povezani sloj*: povezuje značajke s klasifikacijom (vidi 3.2.4.)
- *Izlaz* : izlazni sloj, daje vjerojatnosti klasifikacije (vidi 3.2.5.)



Slika 3.3. Jednostavna CNN [7]

Na slici 3.3. prikazana je opisana struktura. Prva tri sloja grupirana su u dio koji služi za ekstrakciju značajki iz ulaznih podataka, a posljednja dva sloja služe za klasifikaciju. Složenije arhitekture mreže mogu imati veći broj konvolucijskih slojeva (nakon svakog

se nalazi sloj za poduzorkovanje) te veći broj složenijih ili manje složenih potpuno povezanih slojeva.

3.2.1. Konvolucijski sloj

Najbitniji dio konvolucijske neuronske mreže je konvolucijski sloj zbog toga što se u njemu događa konvolucija. Konvolucija (u neuronskim mrežama) je proces kojim iz ulazne matrice podataka (slike) na izlazu dobijemo matricu značajki ili mapu značajki. Neka ulazna matrica bude oblika $x \in M_{mn}(\mathbb{R})$. Umjesto težina (kao kod potpuno povezanog sloja), konvolucijski sloj koristi matricu $\omega \in M_{pr}(\mathbb{R})$ koju nazivamo filtar ili jezgra (eng. kernel) **??** Svaki konvolucijski sloj može imati proizvoljan broj filtara. Izlazna (u ovom slučaju dvodimenzionalna) mapa značajki tada se računa na sljedeći način:

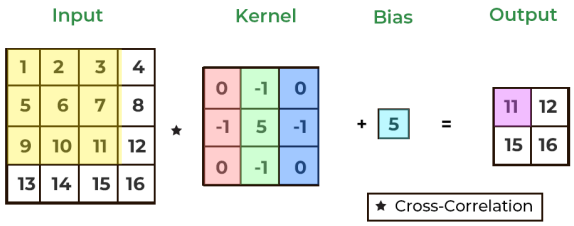
$$h = \omega * x, \tag{3.2}$$

pri čemu $*$ označava operaciju konvolucije te vrijedi:

$$h(i,j) = (\omega * x)(i,j) = \sum_{k=0}^{p-1} \sum_{l=0}^{r-1} x(i+k, j+l)\omega(k,l). \tag{3.3}$$

Formula koja se zapravo koristi naziva se unakrsna korelacija, međutim, zbog sličnosti s formulom konvolucije, mreža nosi takav naziv **??** Na slici 3.4. prikazan je proces koji se događa tijekom prolaska ulaznog podatka kroz konvolucijski sloj. Ulaz (eng. input) se konvolucijski množi s jezgrom kako bismo dobili izlaznu matricu značajki [8]. U slučaju prikazanom na slici, ulazna slika je veličine 4×4, dok je jezgra veličine 3×3. Prvo se podmatrica ulaznog podatka veličine jednake veličini jezgre (3×3) skalarno množi s jezgrom. Izlaz je skalarni umnožak na koji se može dodati konstantna vrijednost (eng. bias). Nakon toga se jezgra pomiče po ulaznoj matrici, tj. sljedeći element izlazne matrice je skalarni umnožak jezgre i sljedeće podmatrice ulaznog podatka. Koliko će se jezgra pomaknuti određuje pomak (eng. stride). U slučaju na slici 3.4. pomak iznosi jedan.

Rezultat opisanog procesa je mapa značajki koja je manja od ulazne, a njema veličina



Slika 3.4. Konvolucija [9]

obrnuto proporcionalno ovisi o veličini jezgre te pomaku [10].

Slično procesu koji se odvija u mozgu čovjeka, opisani struktura omogućava hijerarhijsko učenje. Naime, svaki konvolucijski sloj sadrži jezgre koje su zadužene za lokalno pretraživanje određenih uzoraka (upravo konvolucijom izvlačimo stvari koje su slične između različitih ulaznih slika). Koje jezgre se trebaju koristiti? Vrlo jednostavno, proces učenja (treniranja mreže) će prepoznati lokalne sličnosti između različitih primjera! Ako strukturiramo mrežu tako da se sastoji od više uzastopnih konvolucijskih slojeva, mreža će prvo naučiti najjednostavnije oblike, a zatim u sljedećem sloju takvim oblicima slagati složenije uzorke. Također, još jedna prednost konvolucijskom sloja je u dijeljenju parametara. Takav sloj nema vezu svakog neurona sa svakim ulaznim, nego se težine dijele unutar određene jezgre. Zapravo se cijeli proces učenja svodi na nalaženje odgovarajućih jezgri koje će prepoznati uzorke [6]. Evo uzmimo, na primjer, dvodimenzionalnu ulaznu sliku. Broj pametara takvog dvodimenzionalnog sloja iznosit će:

$$N = (n \cdot m \cdot C_{\text{in}} + 1) \cdot C_{\text{out}} \tag{3.4}$$

Gdje:

- N: ukupni broj parametara
- n i m: visina i širina filtra (jezgre)
- C_{in}: Broj ulaznih kanala (npr. 1 za crno-bijele slike, 3 za RGB slike).
- $+1$: konstanta svakog filtra
- C_{out}: Broj filtara (odnosno mapa izlaznih značajki).

Kako bismo bolje dočarali razliku u broju parametara između ovakog sloja i potpuno povezanog sloja, uzmimo za primjer sliku 3.4. Ulazna slika je veličine 4×4, jezgra 3×3, a izlaz 2×2. Neka je slika jednokanalna, a broj jezgri jedan (sve kao na slici). Konvolucijski sloj će imati 10 parametara, dok će potpuno povezani sloj (16 ulaznih vrijednosti, 4 izlazne te 4 konstante za svaki neuron) imati 68 parametara! Formula za broj parametara u tom slučaju je sljedeća:

$$N = n_{\text{in}} \cdot n_{\text{out}} + n_{\text{out}} \tag{3.5}$$

Gdje:

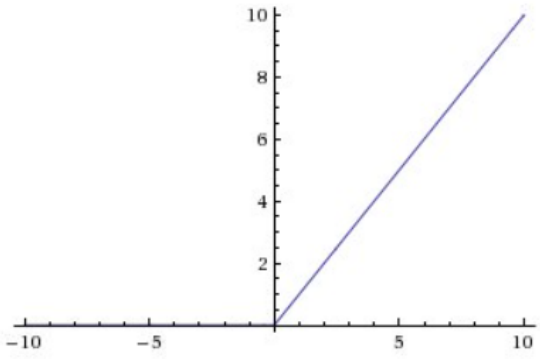
- N: ukupni broj parametara
- n_{in}: broj ulaznih neurona
- n_{out}: broj izlaznih neurona

Također, kao što je opisano u poglavlju 3. vrijednost neurona (čvora) na izlazu it sloja potrebno je provući kroz aktivacijsku funkciju. Postoje različite vrste funkcija koje se koriste u različitim granama strojnog i dubokog učenja [4], a za primjenu u konvolucijskim slojevima, najefektivnija se pokazala ReLu (eng. Rectified linear units) [11]. To je funkcija koja vraća nulu ako joj je ulaz negativan, a za svaki pozitivan ulaz samo prosljeđuje istu vrijednost na izlaz. Modelirana je formulama 3.6 i 3.7, a prikazana je na slici 3.5.

$$f(x) = \max(0, x) \tag{3.6}$$

$$f(x) = \begin{cases} 0, & \text{ako } x < 0, \\ x, & \text{ako } x \geq 0. \end{cases} \tag{3.7}$$

Prednosti ReLu funcije nad nad drugim aktivacijskim funkcijama je u tome za negativne ulaze uopće ne aktivira neuron što izuzetno povećava efikasnost. Druga prednost je u tome što izlaz nelinearno raste s porastom ulazne vrijednosti što znači da nikad neće

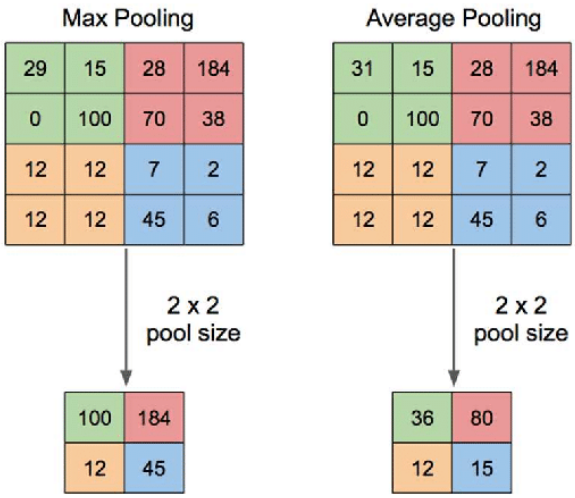


Slika 3.5. ReLu [11]

ući u zasićenje. Ta osobina je važna jer utječe na izgled funkcije gubitka te ubrzava konvergenciju gradijentnog spusta prema minimumu funkcije gubitka [4].

3.2.2. Sloj za poduzorkovanje

Sloj za poduzorkvanje (eng. Pooling layer): služi smanjenju dimenzionalnosti matrice značajki na izlazu iz konvolucijskog sloja. Najčešće korištene tehnike su maksimalno (eng. max pooling) i prosječno poduzorkovanje (eng. average pooling). Radi tako da više susjednih vrijednosti spoji u jednu te tako na svom izlazu da matricu manjih dimenzija [12]. Na taj način postupno smanjuje broj parametara, smanjuje broj operacija potrebnih za daljnje računanje te ono najbitnije, kontrolira prenaučenost [10]. Na slici 3.6. prikazane su obje navedene vrste poduzorkovanja.



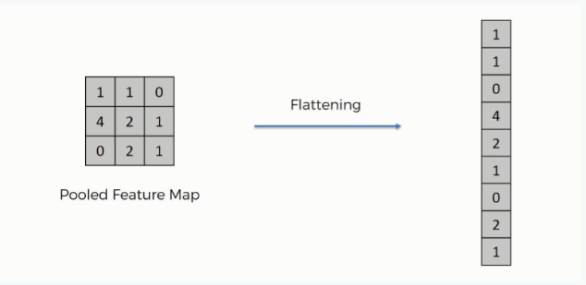
Slika 3.6. Poduzorkovanje [12]

Prosječno poduzorkovanje izglađuje sliku (eng. smoothing). Zbog toga oštri detalji

slike mogu biti izgubljeni što znači da se određene značajke možda neće prepoznati kada se koristi ova metoda. Maksimalno uzorkovanje odabire najsvjetlije piksele iz slike, a oni su se pokazali kao najbitnije značajke jer daju najbolje rezultate [10]. Suprotno tome, minimalno uzorkovanje (eng. min pooling) odabralo bi najtamnije piksele, međutim ono se najrjeđe koristi.

3.2.3. Sloj za poravnavanje

Sloj za poravnavanje (eng. flatten layer) je sloj koji dolazi nakon posjednjeg sloja za poduzorkovanje. Njegova jedina zadaća je poravnati izlaz iz prijašnjeg sloja. Ovaj sloj ništa ne računa, ništa ne uči, jedina zadaća mu je od ulaznih mapa značajki napraviti jedan vektor koji je onda moguće povezati na potpuno povezani sloj. Zbog toga se često izostavi iz skica koje prikazuju strukture CNN-ova kao što je slučaj na slici 3.3. Na slici 3.7. prikazana je uloga ovog sloja.



Slika 3.7. Sloj za poravnjavanje [13]

3.2.4. Potpuno povezani sloj

Potpuno povezani sloj (eng. Fully Connected Layer) detaljnije je pojašnjen u poglavlju 3. Nakon prijašnjih slojeva koji su služili za izvlačenje značajki iz ulaznih podataka, na red dolazi klasifikacija. Budući da je prethodnik prvom ovakvom sloju sloj za poravnavanje, nemamo problem sa spajanje ovog sloja na dosad objašnjenu strukturu. Uloga ovog sloja (ili više ovakvih slojeva) je, najjednostavnije rečeno, klasifikacija. Značajke naučene tijekom konvolucije se ovdje predaju gustoj mreži neurona koja je sposobna odraditi posao do kraja, tj. naučiti kako različite značajke pridonose određenoj izlaznoj klasi.

3.2.5. Izlazni sloj

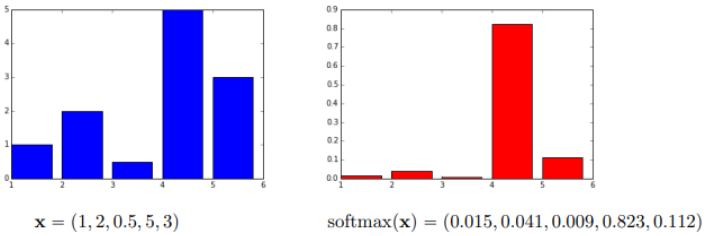
Izlazni (eng. Output Layer) je posljednji sloj potpuno povezanog sloja (a ujedno i cijele neuronske mreže). Ima onoliko neurona koliko želimo imati klasa, pojedina vrijednost neurona predstavlja vjerojatnost pripadnosti klasi. Da bi to stvarno funkcioniralo na takav način, potrebno je odrediti prikladnu aktivacijsku funkciju. Funkcija koja radi baš to naziva se funkcija softmax. Formalno, softmax : $\mathbb{R}^n \to \mathbb{R}^n$, gdje je k-ta komponenta izlaznog vektora definirana kao:

$$\text{softmax}_k(x_1, \ldots, x_n) = \frac{\exp(x_k)}{\sum_j \exp(x_j)} \tag{3.8}$$

Funkcija softmax radi dvije ključne stvari:

- Normalizira sve vrijednosti tako da njihov zbroj bude 1, tj. izlazni vektor predstavlja distribuciju vjerojatnosti.
- Pojačava veće vrijednosti (čini ih dominantnijima) i smanjuje manje vrijednosti.

Funkcija nosi naziv *softmax* jer odgovara funkciji max, ali je "meka" u smislu da je neprekidna i diferencijabilna, za razliku od klasične max funkcije [14].



Slika 3.8. Softmax [14]

3.2.6. Poznate arhitekture konvolucijskih mreža

Arhitektura konvolucijske mreže je ključni faktor koji određuje njene performanse i učinkovitost. Broj konvolucijskih slojeva, izgled istih (broj filtara, njihova veličina, pomak), vrsta slojeva za poduzorkovanje te broj i veličina potpuno povezanih slojeva znatno utječu na brzinu izvođenja i preciznost klasifikacije. Naravno, ne postoji jedan recept koji najbolje radi na svim vrstama ulaznih podataka, nego različite arhitekture daju bolje rezultate u određenim situacijama. Određene arhitekture su kroz povijest ostale zapamćene zbog

toga kako su utjecale na duboko učenje[15]:

- **LeNet-5 (1998):** CNN sa 7 slojeva dizajnirana za klasifikaciju rukom pisanih brojeva na slikama veličine 32×32 piksela u sivim tonovima. Koristila se u bankama za čitanje čekova i bila je prvi značajan korak u korištenju CNN-a u stvarnom svijetu.
- **AlexNet (2012):** Proširena verzija LeNet-a s dubljom arhitekturom (5 konvolucijskih i 3 potpuno povezana sloja). Prva mreža koja je koristila ReLU aktivaciju za brže treniranje. Značajno smanjila stopu pogreške na ILSVRC natjecanju i popularizirala duboko učenje.
- **GoogleNet (Inception V1) (2014):** 22-slojna mreža s inovativnim *inception module*, koji koristi male konvolucije za smanjenje broja parametara (sa 60 milijuna na samo 4 milijuna). Pobjednik ILSVRC 2014 s top-5 pogreškom manjom od 7%. Performanse su usporedive s ljudskim prepoznavanjem slika.
- **VGGNet (2014):** Mreža sa 16 konvolucijskih slojeva koja koristi samo 3×3 konvolucije s povećanim brojem filtara. Iako je jednostavna u dizajnu, ima 138 milijuna parametara, što je čini računalno zahtjevnom za treniranje i implementaciju.

3.3. Dataset

3.4. Struktura mreže

3.5. Treniranje modela

3.6. Vrednovanje modela

3.7. Convert

4. Zaključak

Literatura

[1] K. Pykes, "What is tinyml? an introduction to tiny machine learning", https://www.datacamp.com/blog/what-is-tinyml-tiny-machine-learning, 2023., [Posjećeno: siječanj 2025.].

[2] B. Q. Kevin Ezra, "Introduction to Neural Network", https://iq.opengenus.org/dense-layer-in-tensorflow/, [Posjećeno: siječanj 2025.].

[3] G. for Geeks, "What is Fully Connected Layer in Deep Learning?" https://www.geeksforgeeks.org/what-is-fully-connected-layer-in-deep-learning/, 2024., [Posjećeno: siječanj 2025.].

[4] V. Labs, "Activation functions in neural networks [12 types and use cases]", https://www.v7labs.com/blog/neural-networks-activation-functions, 2021., accessed: January 2025.

[5] M. K. Analyticsvidhya, "Gradient Descent vs. Backpropagation: What's the Difference?" https://www.analyticsvidhya.com/blog/2023/01/gradient-descent-vs-backpropagation-whats-the-difference/, 2023., [Posjećeno: siječanj 2025.].

[6] S. PyCodeMates, https://www.pycodemates.com/2023/06/introduction-to-convolutional-neural-networks.html, [Posjećeno: siječanj 2025.].

[7] V. H. Phung i E. J. Rhee, "A high-accuracy model average ensemble of convolutional neural networks for classification of cloud image patches on small datasets", *Applied Sciences*, sv. 9, br. 4500, 2019. https://doi.org/doi:10.3390/app9214500

[8] J. Brownlee, "How do convolutional layers work in deep learning neural networks?" https://machinelearningmastery.com/convolutional-layers-for-deep-learning-

neural-networks/, 2020., [Posjećeno: siječanj 2025.].

[9] “Apply a 2d convolution operation in pytorch”, https://www.geeksforgeeks.org/apply-a-2d-convolution-operation-in-pytorch/, 2023., [Posjećeno: siječanj 2025.].

[10] D. E. Swapna, “Convolutional neural networks, deep learning”, https://developersbreach.com/convolution-neural-network-deep-learning/, [Posjećeno: siječanj 2025.].

[11] “Rectified linear units (relu) in deep learning”, https://www.kaggle.com/code/dansbecker/rectified-linear-units-relu-in-deep-learning, [Posjećeno: siječanj 2025.].

[12] M. Yani i C. Setianingsih, “Application of transfer learning using convolutional neural network method for early detection of terry’s nail”, *Journal of Physics: Conference Series*, sv. 1201, br. 1, str. 012052, May 2019. https://doi.org/10.1088/1742-6596/1201/1/012052

[13] “Convolutional neural networks (cnn): Step 3 - flattening”, https://www.superdatascience.com/blogs/convolutional-neural-networks-cnn-step-3-flattening, [Posjećeno: siječanj 2025.].

[14] J. Šnajder, “Logistička regresija 2”, Strojno učenje 1, UNIZG FER, ak. god. 2022./2023., predavanja, v1.10, 2023., sveučilište u Zagrebu, Fakultet elektrotehnike i računarstva.

[15] IndianTechWarrior, https://indiantechwarrior.com/convolutional-neural-network-architecture/, [Posjećeno: siječanj 2025.].

Sažetak

Sustav za prepoznavanje govornih naredbi u stvarnom vremenu na rubnim uređajima

Luka Balić

Unesite sažetak na hrvatskom.

Ključne riječi: prva ključna riječ; druga ključna riječ; treća ključna riječ

Abstract

A system for recognizing voice commands in real-time on edge devices

Luka Balić

Enter the abstract in English.

Keywords: the first keyword; the second keyword; the third keyword

Privitak A: The Code