Klasifikacija tipova vozila korištenjem dubokih neuronskih mreža

Bakarić, Dominik

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 552

KLASIFIKACIJA TIPOVA VOZILA KORIŠTENJEM DUBOKIH NEURONSKIH MREŽA

Dominik Bakarić

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 552

KLASIFIKACIJA TIPOVA VOZILA KORIŠTENJEM DUBOKIH NEURONSKIH MREŽA

Dominik Bakarić

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 4. ožujka 2024.

DIPLOMSKI ZADATAK br. 552

Pristupnik:	**Dominik Bakarić (0036516132)**
Studij:	Računarstvo
Profil:	Računarska znanost
Mentor:	prof. dr. sc. Vedran Mornar

Zadatak: **Klasifikacija tipova vozila korištenjem dubokih neuronskih mreža**

Opis zadatka:

U današnje vrijeme, napretkom u području prepoznavanja objekata i računalnom vidu, jasno se vidi mogućnost primjene tih tehnologija u prometu i transportu. Te tehnologije mogu se primjenjivati na razne aspekte prometa kao što su regulacija prometa, sigurnost u prometu te razvoj autonomnih vozila. U sklopu ovog rada treba ostvariti model koji je će na slici prepoznavati vozila te ih klasificirati u kategorije kao što su automobil, motocikl, autobus, kamion ili kombi. Za realizaciju tog zadatka koristiti biblioteku Tensorflow 2 programskog jezika Python s ciljem izgradnje dubokog modela koji je baziran na konvolucijskim neuronskim mrežama te treniran korištenjem metoda prenesenog učenja.

Rok za predaju rada: 28. lipnja 2024.

Sadržaj

Uvod

Porastom raznolikosti motornih vozila u prometu, njihovih oblika i veličina, te funkcionalnosti, dosadašnje metode klasifikacije vozila nisu adekvatne za određivanje potrebnih karakteristika kako bi se uspješno moglo klasificirati sve potrebne tipove vozila. Pojava dubokih neuronskih mreža potaknula je napredak računalnog vida i strojnog učenja, što je rezultiralo učinkovitijim pristupom rješavanju navedenog problema. Duboke neuronske mreže, dizajnirane da oponašaju mogućnost ljudskog mozga da razazna raznolike obrasce, pokazale su se kao adekvatan pristup automatskom prepoznavanju i kategorizaciji vozila na temelju vizualnih informacija dobivenih sa slika prometa.

Mogućnost precizne klasifikacije tipova vozila postaje ključan faktor u rješavanju trenutnih i budućih izazova s prometom zbog ubrzanog razvoja urbanih sredina. Precizna klasifikacija tipova vozila ima dalekosežne implikacije u nekoliko važnih domena modernog života. U kontekstu inteligentnog upravljanja prometom, ona omogućuje optimizaciju protoka prometa i smanjenje prometnih zagušenja, dok u urbanističkom planiranju pomaže u dizajniranju infrastrukture prilagođene specifičnim potrebama različitih vrsta vozila. Također, bitno je spomenuti doprinos ispravne klasifikacije vozila kod nadzora prometa i provedbe zakona, osiguravajući ispravno identificiranje vozila u prekršaju i unaprjeđenje sigurnosnih mjera. Konačno, ovakav sustav ima široku primjenu u kontekstu praćenja utjecaja različitih tipova vozila na okoliš što olakšava analizu djelovanja motornih vozila na emisiju štetnih plinova.

Ovaj rad fokusira se na zadatak prepoznavanja tipa vozila u prometu, s ciljem pružanja robusnog rješenja koje može automatski klasificirati različite tipove vozila s visokom preciznošću i učinkovitošću. U okviru ovog rada, istražit će se osnove neuronskih mreža, objasniti neke od njihovih osnovnih arhitektura, te analizirati prednosti, primjene i ograničenja svake od njih. Posebna pozornost bit će posvećena konvolucijskim neuronskim mrežama, njihovoj strukturi i razlozima zašto su one prikladna metoda za zadatak klasifikacije vozila. Nadalje, proučit će se koncept prijenosa učenja i njegove prednosti. Konačno, bit će prikazan primjer realizacije takvog zadatka u sklopu jednostavne web aplikacije.

1. Neuronske mreže

Neuronske mreže, koje su grana strojnog učenja, građene su korištenjem principa neuronske organizacije koji su otkriveni proučavanjem konekcija u biološkim neuronskim mrežama kao što je to u životinjskim mozgovima. Neuronske mreže osnova su u izgradnji algoritama dubokog učenja, a sastoje se od skupa čvorova koji su podijeljeni u slojeve. Slojevi od kojih je neuronska mreža najčešće sastavljana su ulazni sloj, izlazni sloj i jedan ili više skrivenih slojeva, koji se nalaze između tih dvaju slojeva. Svaki čvor, odnosno neuron, povezan je s idućim neuronom u mreži. Svaki neuron u mreži ima vlastiti prag i aktivacijsku funkciju, ako dođe do toga da vrijednost aktivacijske funkcije neurona dosegne taj prag ili ga prekorači, neuron se aktivira te prosljeđuje podatke na idući neuron. U slučaju kada vrijednost aktivacijske funkcije nije dosegla potrebnu vrijednost neuron se ne aktivira, te se podaci ne prosljeđuju na iduće neurone. Svaki neuron može se promatrati kao instancu linearne regresije, što znači da svaki neuron ima svoje težine, ograničenje i izlaz. To se može izrazit idućom formulom:

$$y_i = f\left(\sum_{j=1}^{N} w_{ij} \cdot x_j + b_i\right)$$

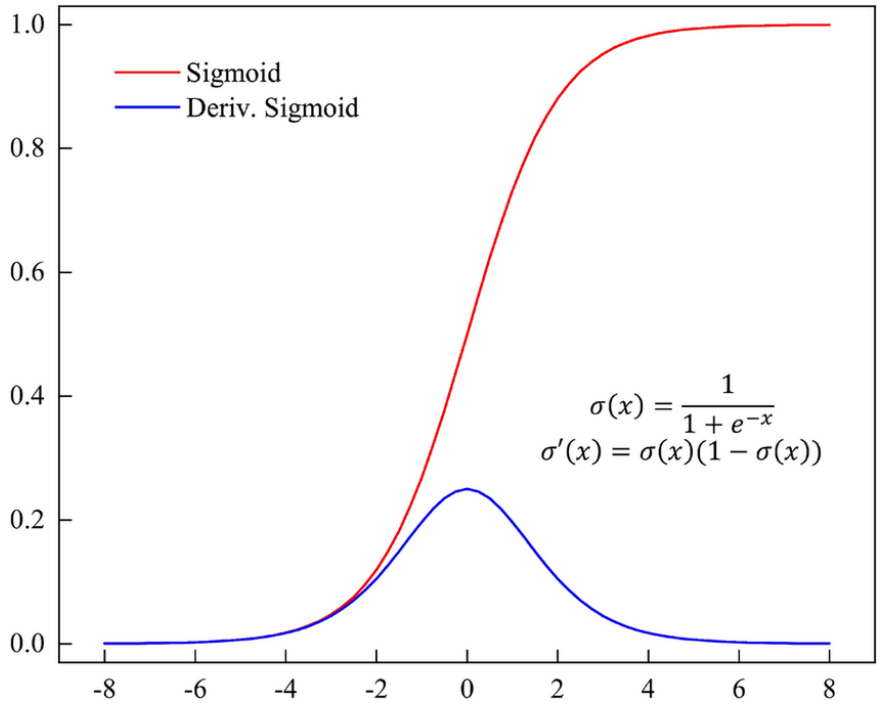
Svim konekcijama između neurona, kao i na ulazima, pridijeljena je težina. Težine su numeričke vrijednosti karakteristične za veze među neuronima, koje određuju značaj pojedine varijable ulaza, odnosno, predstavljaju važnost izlaza prethodnog neurona za trenutno gledani neuron. Inicijalne vrijednosti težina postavljaju se slučajnim odabirom ili na osnovu intuicije, a one kasnijim prolascima kroz mrežu mijenjaju implementacijom različitih algoritama kako bi se postigla veća točnost predikcije. Povećanje točnosti predikcije ono je što se naziva učenjem neuronske mreže, a ono se provodi prolaskom kroz dani skup podataka za koji su vrijednosti izlaza poznate i ovisno o tome prilagođavaju svoje težine te tako povećavaju točnost modela.

1.1. Aktivacijske funkcije

Tijekom prolaska podatka kroz neuron, prije nego što neuron proslijedi svoje podatke dalje u mrežu, na podatak će se primijeniti aktivacijska funkcija. Vrijednost neurona nakon primjene aktivacijske funkcije određuje je li se neuron aktivirao ili nije. Neke od najučestalijih aktivacijskih funkcija su sigmoida, tangens hiperbolni, softmax te RELU. Sigmoida je funkcija koja je najzastupljenija u starijim arhitekturama neuronskih mreža, ali se koristi i danas. Njena svrha jest mapiranje ulaznih podataka na interval [0,1], te se takvim mapiranjem podataka postiže više funkcionalnosti kao što su poboljšana konvergencija za vrijeme treniranja, povećana numerička stabilnost, podjednaka distribucija značajki, itd. Formula sigmoidalne funkcije jest sljedeća:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

Iz formule za sigmoidalnu funkciju možemo zaključiti svojstvo simetričnosti funkcije, te da za bilo koju ulaznu vrijednost, izlazna vrijednost funkcije biti će između nula i jedan. Također, može se iščitati kako krivulja ima najveći nagib za vrijednosti u okolini x=0 , a izlazna vrijdnost funkcije u toj okolini iznosi približno 0.5. Također jasno je da će krivulja biti simetrična. Grafički prikaz navedene funkcije možemo vidjeti na Slici 1.



Slika 1 Graf funkcije sigmoide i njene derivacije

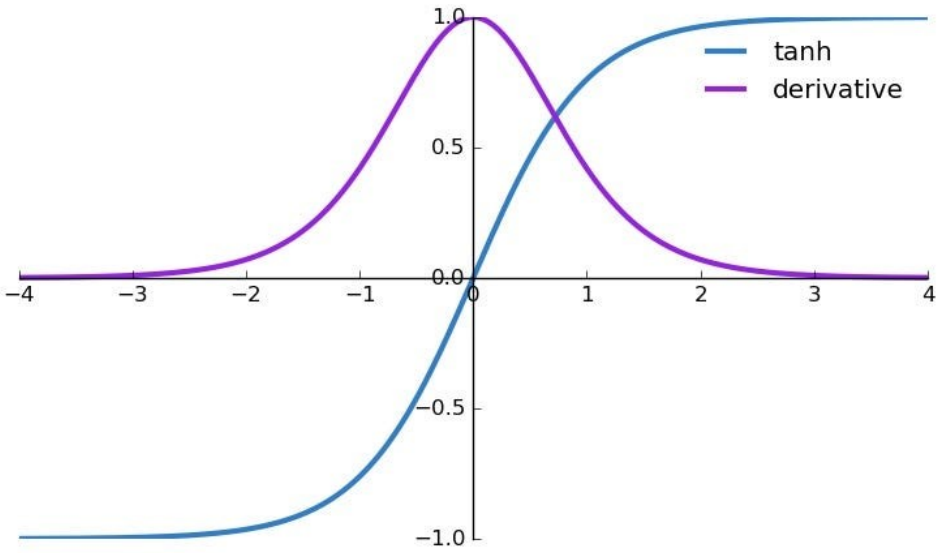
Jedam od bitnih razloga zašto se sigmoida pokazuje kao vrlo adekvatan izbor aktivacijske funkcije jest njena derivacija. Derivacije je ključan aspekt na koji se mora misliti tokom unatražnog prolaza mrežom. Za sada bitno je znati da će se tokom unatražnog prolaza računati gradijent funkcije što zahtjeva uporabu derivacije. Derivacija sigmoide jest zadana sljedećom formulom.

$$\sigma'(x) = \sigma(x) * (1 - (x))$$

Ono što je jasno iz formule za derivaciju sigmoide jest, da je ona ekstremno pogodna za izračun gradijenta radi toga što je glatka te jednostavna za računanje. No iako se možda na prvi pogled čini kap da je sigmoida apsolutno savršeno aktivacijska funkcija za svaku neuronsku mrežu, to baš i nije tako. Naime za izgradnju duboke neuronske mreže ili rješavanje kompleksnijih zadataka često je se zamjenjuje nekom drugom aktivacijskom funkcijom, to je zato što sigmioda sporo konvergira, a u tim situacijama to to je nepoželjno.

Nadalje kod sigoide se javlja problem nastajućeg gradijenta, a više o toj pojavi biti će objašnjeno u kasnijim poglavljima.

Iduća funkcija koja se često koristi jest tangens hiperbolni. Tangens hiperbolni je zapravo dosta sličan sigmoidi što se jasno može vidjeti s grafa funkcije.



Slika 2 Graf funkcije tangens hiperbolni i njene derivacije

Ono što se prvo može primijetiti na grafu funkcije jest da je oblik krivulje isti kao i kod sigmoide, no izlazne vrijednosti funkcije nalaze se unutar intervala [-1,1]. Radi toga korištenjem ove funkcije postiže se to da su podaci centrirani oko nule što rezultira balansiranijim gradijentima i efikasnijim ažuriranjem težina. Sama formula za tangens hiperbolni glasi ovako:

$$tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

Iz navedene formule jasno se može vidjeti da je funkcija neprekinuta i glatka. Pošto vrijednosti izlaza ove funkcije mogu biti negativne, tijekom unatražnog prolaza, gradijenti će biti balansiraniji, to je što gradijent neće uvijek težiti pozitivnoj vrijednosti kao što je to slučaj sa sigmoidom. Kao što je naglašeno kod sigmoide, derivacija je vrlo bitan aspekt kod odabira aktivacijske funkcije. Formula za derivaciju tangensa hiperbolong zadana je na idući način:

$$\tanh'(\mathrm{x}) = 1 - tanh^2(\mathrm{x})$$

Iz prikazane formule i grafa derivacija tangensa hiperbolnog, prikazanog na Slici 2, jasno je da je i derivacija slična onoj sigmoide što znači da ima jednaka pogodna svojstva prilikom računanja gradijenta, ali za razliku od sigmoide izlazne vrijednosti poprimaju vrijednosti iz intervala [-1,1]. Ono što dobivamo korištenjem ove aktivacijske funkcije jest brža konvergencije te bolji protok gradijenta za vrijeme unatražnog prolaza uz naravno prije navedenu prednost balansiranijeg gradijenta. Unatoč poboljšanim performansama u gore navedenim kategorijama problem nestajućeg gradijenta i dalje je prisutan kao i kod sigmoide radi čaga funkcija može naići na probleme u situacijama kada je ulaz x malen a vrijednost gradijenta velika. Kada se dogodi takva situacija dolazi do saturacije i sporijeg treniranja modela.

Odabir Aktivacijske funkcije uglavnom ovisi o tipu problema koji se rješava, ako recimo zadatak zahtjeva binarnu klasifikaciju koristiti će se sigmoida, dok kod izgradnje duboke mreže veću funkcionalnost dobivamo koristeći RELU i softmax. Te dvije funkcije su ujedno i funkcije koje se koriste za odabrani zadatak te će se njihova svojstva, prednosti i mane analizirati u kasnijim poglavljima.

1.2. Unatražni prolaz

Kada se razmatra proces učenja neuronske mreže unatražni prolaz javlja se kao srž procesa treniranja mreže. Unatražni prolaz pokreće se nakon svake propagacije ulaznog podatak kroz mrežu. Kada se obradom tog podatka u mreži generira neki rezultat, kao prvi korak unatražnog prolaza, računa se ukupni gubitak. U slučaju da zadataka zahtijeva binarnu klasifikaciju taj gubitak računa se po idućoj formuli.

$$Loss = \frac{1}{2}(y_{true} - y_{pred})^2$$

Naravno ako zadatak ipak zahtijeva više klasna klasifikacija kao funkcija gubitka koristiti će se gubitak unakrsne entropije. Gubitak unakrsna entropija matematički je zapisana na idući način:

$$L = -\sum_{i=1}^{N} y_i \log(\mathrm{p}_i)$$

Nakon što je gubitak izračunat može se krenuti sa algoritam unatražnog prolaza koji računa gradijente funkcije gubitka u odnosu na težine primjenjujući pritom lančano pravilo. To uključuje računanje parcijalnih derivacija gubitka u odnosu na svaki parametar mreže. Na izlazu mreže računa se sljedeće

$$\delta_2 = \frac{\partial Loss}{\partial y_{pred}}$$

Kada je taj gradijent izračunat on se propagira unazad po skrivenim slojevima mreže te se tako podešavaju gradijenti težina i gradijenti pristranosti. Takvo podešavanje matematički se prikazuje na sljedeći način.

$$\delta_1 = \delta_2 * \sigma'(z_2) * W_2$$

U prikazanoj formuli δ_1 predstavlja izraz greške za prvi skriveni sloj, a $\sigma'(z_2)$ označava derivaciju aktivacijske funkcije drugog sloja. Nakon što se izračuna greška, korištenjem gradijenata koji su izračunati prilikom unatražnog prolaza, ažuriraju se vrijednosti težina i pristranosti kako bi si smanjila pogreška sustava. Ažuriranje vrijednosti težina i pristranosti najčešće se odrađuje korištenjem algoritma gradijentnog spusta. Matematički taj postupak zapisan je na sljedeći način.

$$W_1 = W_1 - \eta \frac{\partial Loss}{\partial W_1}$$

$$b_1 = W_1 - \eta \frac{\partial Loss}{\partial b_1}$$

$$W_2 = W_1 - \eta \frac{\partial Loss}{\partial W_2}$$

$$b_2 = W_1 - \eta \frac{\partial Loss}{\partial b_2}$$

U iznad navedenim formulama η predstavlja stopu učenja, W_n predstavlja težine određenog sloja n, dok b_n predstavlja pristranost tog sloja. Stopa učenja kontrolira veličinu koraka u

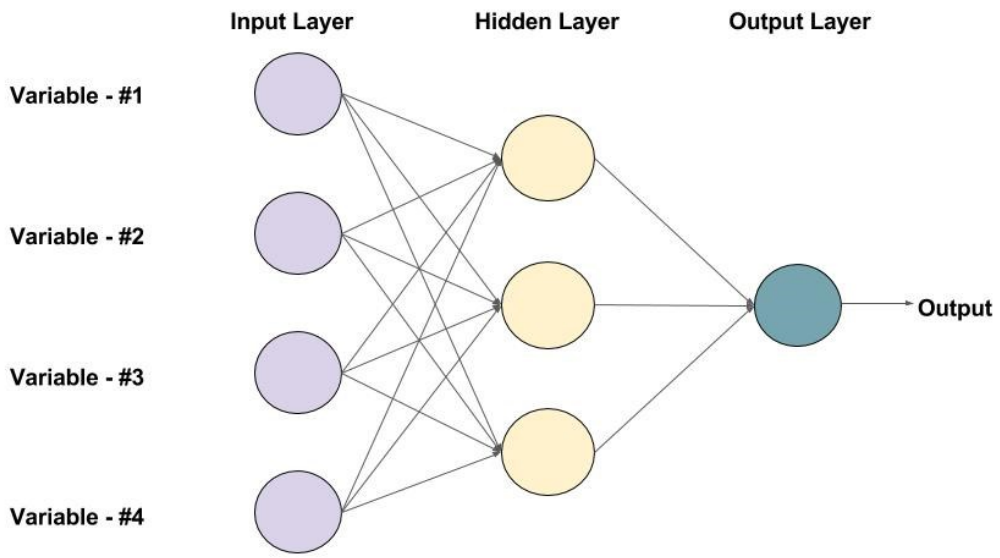
svakoj iteraciji učenja tokom cijelog vremenma kretanja vrijednosti funkcije gubitka prema minimumu. U ovom slučaju stopa učenja određuje koliko se parametri modela prilagođavaju u odnosu na gradijent funkcije gubitka. Stopom učenja kontrolira se koliko brzo ili sporo model konvergira, pa je radi toga izbore pravilne stope ključan za optimalno funkcioniranje modela. Biranjem prevelike stope učenja moguće je dovesti model do toga da prebrzo konvergira ili u još gorem slučaju da divergira, jer ako je korak koje model napravi preveliki može se dogoditi da premaši minimum funkcije. Ako pak odaberemo premalu stopu učenja konvergencija može biti prespora što će rezultirati predugačkim učenjem. Kao što se može vidjeti odabir ispravne stope učenja je vrlo bitan s obzirom na to da ona pospješuje stabilnost modela te smanjuje oscilacije i mogućnost divergiranja. Stopu učenja može se odabrati na mnogo načina kao na primjer kaljene stope učenja, metoda prilagodljive stope učenja, pretraživanje mreže, nasumično pretraživanje mreže, strategija zagrijavanja, itd. Svaka od tih metoda je dobra i validna te se odabir metode treba izvršiti s obzirom na dani zadatak. Svaka od metoda bit će dobra ako zadatak za koji ju primjenjujemo podliježe njenim prednostima.

1.3. Osnovni tipovi neuronskih mreža

U sklopu ovog poglavlja biti će objašnjeno kako funkcioniraju neki od osnovnih tipova neuronskih mreža. Analizirat će se njihove prednosti i mane te će se elaborirati na njihove primjene na određene zadatke. Mreže koje će se istražiti u ovom poglavlju su Feedforward Neural Network (FNN), Recurrent Neural Network (RNN), Generative Adversarial Networks (GAN), Convolutional Neural Network (CNN).

1.3.1. Feedforward Neural Network (FNN)

Feedforward Neural Network (FNN) najjednostavniji su oblik neuronskih mreža. Karakteriziraju ih jednostavnost i jednosmjeran tok podataka kroz mrežu koji prolaze od ulaza prema izlazu bez mogućnosti povratnih veza. Posljedica ovakve konfiguracije mreže je jednostavna implementacija, povećana fleksibilnost zbog velike prilagodljivosti na široki spektar zadataka i u teoriji, može aproksimirati bilo koju neprekidnu funkciju, uz uvjet da su mreža i računalna snaga, potrebna za podržavanje te mreže, dovoljno veliki. Neki od problema ovakvih mreža su skalabilnost na zadatak, sklonost prenaučenosti i pojava nestajući gradijent.



Slika 3 Prikaz Feedforward arhitekture

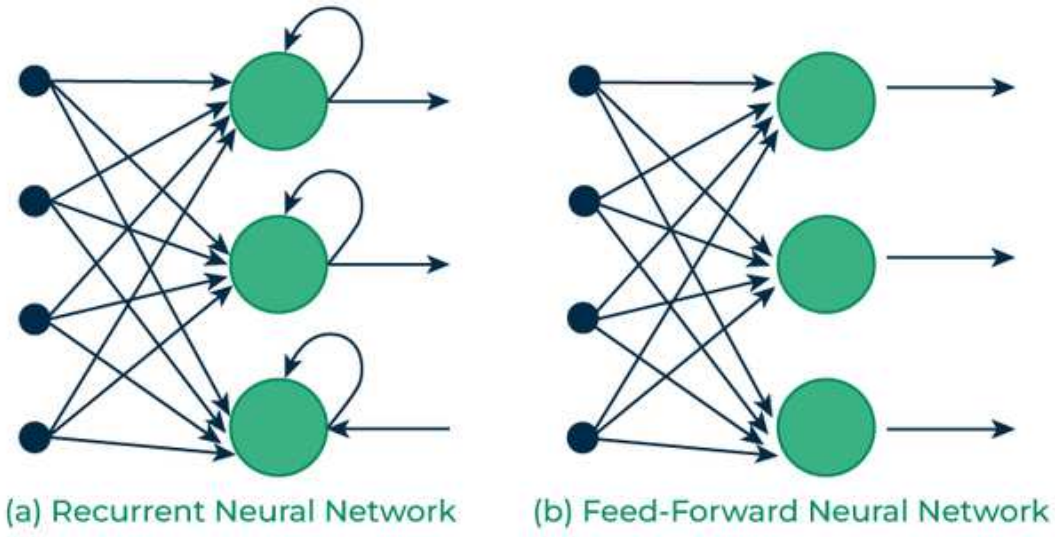
Slika 3 prikazuje jedan jednostavan primjer Feedforward neuronske mreže koja ima 4 ulaza, jedan skriveni sloj u kojemu se nalaze 3 neurona i izlazni sloj. Kao što se jasno vidi Slike 3 povratne veze ne postoje te podaci idu isključivo od ulaza prema izlazu nakon čega se izvršava unatražni prolaz te se ažuriraju težine kako bi finalni model davao što točnije predikcije.

1.3.2. Recurrent Neural Network (RNN)

Recurrent Neural Networks (RNN) su tip neuronskih mreža koji je dizajniran sa namjerom da funkcionira na sekvencijalnim podacima gdje je redoslijed ulaznih podataka bitan. Za razliku od FeedForward neuronskih mreža ovaj tip mreža ima povratne veze koje omogućavaju podacima da ostaju zapamćeni, što ih čini idealnim izborom za zadatke koji uključuju vremenske raspone, jezike i sekvencijalne podatke. Recurrent Neural Network strukturiran je koristeći tri komponente ulazni sloj, ponavljajući skriveni sloj i izlazni sloj. Ulazni sloj prima ulazne podatke u svakom vremenskom koraku, Izlazni sloj koji izvršava finalnu predikciju i ponavljajući skriveni sloj koji sadržava skriveno stanje. Skriveno stanje funkcija je trenutnog ulaza i skrivenog stanja u vremenskom koraku t-1. To stanje računa se na idući način.

$$h_t = \mathrm{f}(W_{xh}x_t + \mathrm{W}_{hh}\mathrm{h}_t - 1 + \mathrm{b}_h)$$

U prikazanoj formuli W_{xh} predstavlja težinsku matricu za ulazne podatke skrivenog stanja x_t, dok je W_{hh} matrica težina skrivenog stanja koje je trenutno zapamćeno. b_h predstavlja pristranost za trenutni sloj dok je funkcija f neka aktivacijska funkcija kao što je tangens hiperbolni ili ReLU.



Slika 4 Usporedba RNN i FNN arhitektura

Na Slici 4 prikazana je jedna jednostavna inačica Recurrent Neural Network sa koje se jasno vidi kako izgledaju te poveznice i kako se razlikuje u odnosu na Feedforward mrežu. Svaki neuron skrivenog sloja ima poveznicu sa samim sobom kako bi zapamtio skriveno stanje za sljedeći korak. Nakon što podaci prođu kroz zadnji skriven sloj oni se prosljeđuju izlaznom, sloju. Izlazni sloj generira izlaz u svakom vremenskom koraku, to se računa na idući način.

$$y_t = g(W_{hy}h_t + b_y)$$

U prikazanoj formuli W_{hy} predstavlja matricu težina koje se primjenjuju kada podaci prolaza sa skrivenog sloja na izlazni. Funkcija g predstavlja aktivacijsku funkciju izlaznog sloja. U slučaju klasifikacije kao aktivacijsku funkciju najčešće se koristi softmax dok se za regresijske zadatke koristi funkcija identiteta.

Postoji više različitih tipova RNN-ova, a neki od najučestalijih su Long short-term memory mreže i dvosmjerne RNN mreže. Long short-term memory mreže vrsta su RNN-a koju

karakterizira sposobnost pamćenja dugoročnih ovisnosti. Dizajnirane su s namjerom da izbjegnu problem dugoročnih ovisnosti koji je inače čest u klasičnim RNN-ovima. Long short-term memory mreže ne predviđaju idući korak samo s obzirom na ulaznu sekvencu već selektivno pamte ili zaboravljaju dijelove informacija kroz duži period vremena. To ih čini ekstremno efektivnim u odrađivanju zadataka vezanih uz vremenske podatke, sekvence i modeliranje jezika.

Long short-term memory mreže uvode memorijsku ćeliju C_t koja se sastoji od troja vrata. Ta vrata kontroliraju protok informacije kroz ćeliju i su dizajnirana kako bi regulirala koje informacije će se zapamtiti, a koje zaboraviti tijekom treniranja i predikcije.

Prva vrata koja je potrebno spomenuti su vrata za zaboravljanje. Vrata za zaboravljanje odlučuju hoće li se informacija unutar ćelije odbaciti ili sačuvati. Kao parametre vrata za zaboravljanje uzimaju prethodno stanje h_{t-1} te trenutni ulazni podatak x_t. Nad njima se zatim primjenjuje sigmoidna funkcija. Tako se osigurava se na izlaz dobivaju vrijednosti između nula i jedan za svaki broj u ćeliji C_{t-1} . Formula za to glasi.

$$f_t = \sigma(\mathrm{W}_f \cdot [\mathrm{h}_{t-1}, \mathrm{xt}] + \mathrm{b}_f)$$

U prikazanoj formuli W_f je matrica težina za ta vrata a b_f pristranost tih vrata. Iduća vrata koja je bitno analizirati su vrata za ulazne podatke. Ova vrata kontroliraju nove informacije koje će se pohraniti u stanje ćelije. Sastoje se od dva dijela, prvi dio su sami ulazi dok su drugi dio vrijednosti koje su kandidati za čuvanje, odnosno to su one vrijednosti koje imaju potencijal kreirati nove informacije. Matematički to izgleda ovako.

$$\mathrm{i}_t = \sigma(\mathrm{W}_i \cdot [\mathrm{h}_{t-1}, \mathrm{x}_t] + \mathrm{b}_i)$$

$$\widetilde{\mathrm{C}_\mathrm{t}} = \tanh(\mathrm{W}_\mathrm{C} \cdot [\mathrm{h}_{\mathrm{t}-1}, \mathrm{x}_\mathrm{t}] + \mathrm{b}_\mathrm{C})]$$

Nakon što podatak prođe kroz prethodno navedena vrata potrebno je ažurirati vrijednosti ćelija. To se radi tako što se kombinira izlaz vrata za zaboravljanje i vrata za ulaz i to se matematički zapisuje na sljedeći način.

$$\mathrm{C}_\mathrm{t} = \mathrm{f}_\mathrm{t} \odot \mathrm{C}_{\mathrm{t}-1} + \mathrm{i}_\mathrm{t} \odot \widetilde{\mathrm{C}_\mathrm{t}}$$

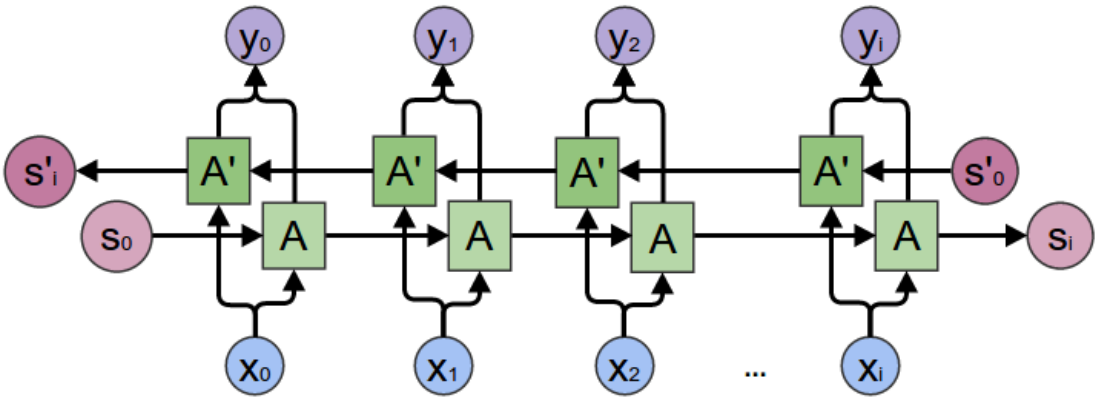
Kada se ćelije ažuriraju ostaje još samo prolaz kroz vrata za izlaz. Vrata za izlaz odlučuju koje će biti iduće skriveno stanje h_t to stanje će ujedno služiti i kao izlaz. Na ovim vratima filtrira se stanje ćelije te se spaja s ulazom prijašnjeg skrivenog stanja kako bi se generiralo novo skriveno stanje. Računski gledano to izgleda ovako.

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$

$$h_t = o_t \odot \tanh(C_t)$$

Sad kada je jasno kako funkcionira ovakva mreža vidljivo je da je pogodna za zadatke kao što su obrada prirodnog jezika (predviđanje iduće riječi, prijevod, generiranje teksta), vremenska analiza (predviđanje dionica, vremenska prognoza), prepoznavanje govora, detekcija anomalija, itd.

Drugi često korišten tip RNN-ova jest Dvosmjerni RNN. Dvosmjerni RNN nadilazi probleme klasičnog RNN-a tako što sadrži dva sloja skrivenih stanja. Prvi sloj procesira sekvencu od ulaza prema izlazu dok je drugi sloj procesira suprotnim smjerom, odnosno od izlaza prema ulazu. To omogućava mreži da kod predikcije uzima u obzir informacije iz prošlosti i iz budućnosti prije pravljenja predikcije. Arhitekturno to znači da se mreža sastoji od tri sloja. Ulazni sloj procesira sekvencu u svom prirodnom poretku, od prvog do posljednjeg elementa. Povratni sloj procesira sekvencu u obrnutom redoslijedu, počevši od zadnjeg elementa prema prvom. Izlazni sloj spaja izlaze iz oba sloja kako bi napravio finalnu predikciju. Svako skriveno stanje u izlaznom sloju funkcija je skrivenih stanja ulaznog i izlaznog sloja. Grafički to bi se moglo prikazati na sljedeći način.



Slika 5Prikaz arhitekture dvosmjernog RNN-a

Neka se za primjer koristi sekvenca $X = \{x_1, x_2, x_3, \dots, x_T\}$ kao ulazni podatci u mrežu onda se prolaz unaprijed i prolaz u nazad mogu se izraziti na sljedeći način.

Prolaz unaprijed

$$h_t^{(fwd)} = f\left(W^{(fwd)} \cdot x_t + U^{(fwd)} \cdot h_{t-1}^{(fwd)}\right)$$

Prolazu unazad

$$h_t^{(bwd)} = f\left(W^{(bwd)} \cdot x_t + U^{(bwd)} \cdot h_{t+1}^{(bwd)}\right)$$

U navedenim formulama $W^{(fwd)}$ i $W^{(bwd)}$ predstavljaju matrice težina za konekcije ulaza i skrivenog sloja, a $U^{(fwd)}$ i $U^{(bwd)}$ predstavljaju matrice težina za konekcije skrivenog sloja na skriveni sloj.

Nakon što su podaci propagirani kroz mrežu dolaze do izlaznog sloja. U izlaznom sloju kombiniraju se izlazi ulaznog i povratnog sloja kako bi se generirala predikcija. Način na koji će se informacije spajati ovise o tipu zadatka no najčešći načini su:

Ulančavanje

$$y_t = g\left(h_t^{(fwd)} \oplus h_t^{(bwd)}\right)$$

Zbroj

$$y_t = g\left(h_t^{(fwd)} + h_t^{(bwd)}\right)$$

Prosjek

$$y_t = g\left(2h_t^{(fwd)} + h_t^{(bwd)}\right)$$

Kao što je pokazano ovakve mreže dosta su moćan alat kada ih koristimo za zadatke kao što su prepoznavanje imenovanih entiteta, analiza raspoloženja iz teksta, prepoznavanje govora te prepoznavanje govornika. BiRNN briljira u ovakvim zadacima radi svoje pojačane sposobnosti shvaćanja konteksta kao i svoje prilagodljivosti na različite sekvencijalne podatke. No ovakve mreže generalno imaju veliku računsku kompleksnost što ih čini dosta skupima. Uz to s obzirom na količinu podataka koje pamte memorijski su zahtjevne te može doći do problema kod pamčenja dugačkih sekvenci. Također ovakve mreže nisu pogodne za rad u stvarnom vremenu.

1.3.3. Generative Adversarial Networks (GAN)

Generative Adversarial Networks (GAN) su vrsta neuronskih mreža koja je dizajnirana kako generirali realistični uzorci podataka. Osmislio ih je Ian Goodfellow sa svojim timom 2014 godine. GAN se zapravo sastoji od dvije neuronske mreže, jedna od tih mreža je generator dok je druga diskriminator. Ove dvije mreže uključene su u proces teorije igara kako bi jedna drugu unaprijedile kroz neki period vremena.

Mreža koja služi kao generator zadužena je za kreiranje uzoraka podataka kao što su slike, audio ili tekst koji su slični ulaznim podacima. Ona uzima nasumične vektore šuma, često je to iz Gaussove distribucije, kao ulaz te ih transformira u uzorke podataka. Cilj generatora je generirati uzorak koji je nemoguće razaznati od stvarnog uzorka.

Diskriminator ima u cilju prepoznati razliku između pravih uzoraka i onih generiranih od strane generatora. Izlaz diskriminator jest vjerojatnost koja određuje je li uzorak stvaran, odnosno došao iz skupa za treniranje, ili je uzorak lažan odnosno generiran pomoću generatora.

Te dvije mreže treniraju se suparnički gdje se generator i diskriminator natječu jedan protiv drugoga. Generator pokušava nasamariti diskriminatora tako što producira sve realističnije

uzorke, dok diskriminator pokušava biti sve bolji u prepoznavanju lažnih uzoraka. Takvo natjecanje se nastavlja dok generator ne počne generirati uzorke koji se ne mogu razaznati od pravih, odnosno kada diskriminator-ova stopa uspjeha nije bolja od nasumičnog pogađanja (otprilike 50% šanse da točno odabere).

Matematički gledano taj proces može se zapisati na sljedeći način.

$$min_G max_D V(D,G) = E_{x\sim p_{\mathrm{dt}}(x)}[\log D\,(x)] + E_{z\sim p_z(z)}\left[\log\left(1 - D\big(G(z)\big)\right)\right]$$

U navedenoj formuli x predstavlja uzorke, a z predstavlja nasumičan šum koji se predaje kao ulaz generatoru. D(x) je diskriminator-ova predikcija vjerojatnosti da je x stvaran uzorak dok je G(z) generirani uzorak nastao pomoću šuma z.

Treniranje ovakvog modela odvija se tako da se obje mreže prvo inicijaliziraju nasumičnim težinama. Zatim se diskriminator trenira serijom pravih i serijom generiranih uzoraka pomoću čega ažurira svoje težine. Funkcija gubitka diskriminatora najčešće mjeri koliko dobro on klasificira podatke. Generator se trenira da bi mogao generirati podatke za diskriminator. Njegova funkcija gubitka temelji se na povratnoj informaciji koju dobije od diskriminatora što se najčešće odvija preko unatražnog prolaza korištenjem gradijenta funkcije gubitka diskriminatora. Ovaj proces ponavlja se iterativno te se mreže ažuriraju alterirajući između generatora i diskriminatora.

Ovakve mreže imaju široku primjenu. Neki od zadataka koji su pogodni za ovakve mreže su sljedeći. Generiranje slika, odnosno kreacija realističnih slika objekata ljudi i scena. Povećanje podataka nekog skupa podataka koji se koristi za strojno učenje kako bi se dodali dodatni uzorci koji se mogu koristiti za treniranje. Super rezolucija koja je proces u kojemu se povećava rezolucija slike, dosta često se koristi u medicinske svrhe ili za povećanje kvalitete satelitskih slika. Također ove mreže mogu se primijeniti za prebacivanje slika iz jedne domene u drugu kao što je recimo prebacivanje skeča u foto realističnu sliku, te naravno mogućnost generiranja teksta i sintetiziranje govora.

GNN mogu se susresti s nekoliko problema. Prvi problem koji se javlja jest stabilnost treniranja. Mreže ovakve arhitekture poznate su po tome da ih je teško trenirati jer je teško pronaći balans između diskriminatora i generatora. Također dosta je teško mjeriti kvalitetu i

raznolikost generiranih podataka. Na kraju tu je još i etička upitnost generiranja lažnog sadržaja koji se čini autentičan.

Sad kada je jasno kako funkcioniraju neke od najosnovnijih arhitektura bitno je naglasiti kako je za zadadni zadatak najpogodnije koristiti konvolucijske neuronske mreže. Zašto su baš konvolucijske mreže pogodne za ovaj zadatak te kako one funkcioniraju objasnit će se detaljno u zasebnom poglavlju no prije toga bitno je shvatiti pojam dubokog učenja i gdje se ono može primijeniti.

2. Duboko učenje

Duboko učenje podskup je metoda strojnog učenja temeljenih na umjetnim neuronskim mrežama s reprezentacijskim učenjem. Pridjev "duboki" odnosi se na korištenje višestrukih slojeva u mreži. U dubokom učenju, svaka razina uči transformirati svoje ulazne podatke u malo apstraktniji i složeniji prikaz. Ako se kao primjer pogleda aplikacija za prepoznavanje lica sa slike, neobrađeni unos može biti matrica piksela; prvi reprezentacijski sloj može apstrahirati piksele i kodirati rubove; drugi sloj može sastavljati i kodirati raspored rubova; treći sloj može kodirati nos i oči; a četvrti sloj može prepoznati da slika sadrži lice. Važno je naglasiti da proces dubokog učenja može samostalno naučiti koje značajke optimalno postaviti na koju razinu. Ovo ne eliminira potrebu za ručnim podešavanjem; na primjer, različiti brojevi slojeva i veličine slojeva mogu pružiti različite stupnjeve apstrakcije.

Potreba za dubokim učenjem javila se iz toga što su tradicionalne metode strojnog učenja zahtijevale veliku količinu ljudskog truda kako bi model bio adekvatno treniran. Ako se sagleda primjer jednog modela za prepoznavanje životinjskih slika, korištenjem tradicionalnih metoda strojnog učenja to bi zahtijevalo ručno označiti stotine tisuća slika životinja, napraviti algoritme koji obrađuju te slike, testirati algoritme na nepoznatim slikama, prepoznati zašto su pojedini rezultati pogrešni te unaprijediti bazu podataka tako da označimo nove slike kako bi povećali točnost. Taj proces zove se nadzirano učenje te se njegovu točnost moguće poboljšati samo ako je dostupan dovoljno širok i raznolik skup ulaznih podataka.

Modeli koji koriste duboko učenje, s druge strane, nemaju takve probleme te jako lagano shvaćaju nestrukturirane podatke te primjećuju generalne stvari bez ručnog izvlačenja značajki. Aplikacije koje koriste duboko učenje mogu analizirati velike količine podataka dublje te tako pronaći neke nove poveznice za koje možda nisu bile trenirane.

Duboki modeli referiraju se na to da se u njima, po uzoru na ljudski mozak, koristi stotine pa čak i tisuće slojeva kako bi se obavilo treniranje. Na ovakvim modelima poželjno je koristiti nenadzirano učenje pošto se tako modelu omogućava da samostalno izvuče karakteristike, značajke i poveznice kako bi mogle davati adekvatne izlaze za nestrukturirane podatke.

2.1. Primjene

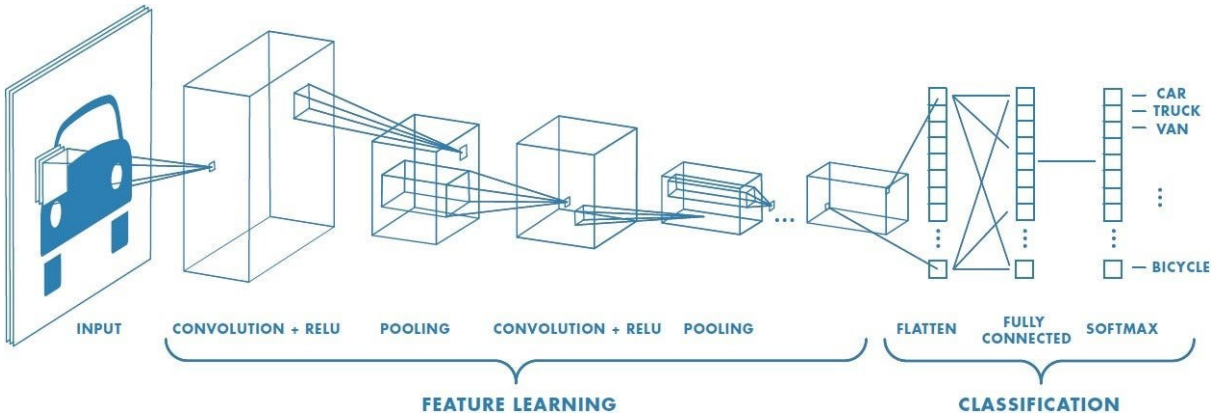
Duboke neuronske mreže mogu se primijeniti na bezbroj načina u današnje doba. Neke od čestih primjena su u predikcija preporuka reklama, prepoznavanje lica, predikcija prognoze, prepoznavanje objekata sa slike itd. Najbitnije primjene vide se u području računalnog vida pošto se upravo ovakve mreže koriste u autonomnim autima ili robusnim sustavima za prepoznavanje lica. U zdravstvu ovakve mreže primjenjuju se kako bi se olakšalo dijagnosticiranje medicinskih slika te kreiranje personalnih tretmana ovisno o pacijentovim podacima. I ovo su samo neke od značajnih primjena uz adekvatan skup podataka i s razvojem tehnologije jasno je kako se ovakvi modeli mogu primijeniti na gotovo svaki aspekt ako za njega postoji dovoljno podataka što ih čini predvodnikom za rješenja vođena umjetnom inteligencijom.

3. Konvoluvijske neuronske mreže

Konvolucijske neuronske mreže (CNN ili ConvNets) su klasa neuronskih mreža posebno pogodnih za zadatke koji uključuju ulazne podatke u obliku slike i videozapisi. Takve neuronske mreže (CNN) postale su oslonac za područje računalnog vida i proširile svoj utjecaj na ostala područja koja se bave obradom i generiranjem slike. Nastale su iz temeljne potrebe za analizom složenih vizualnih podataka, te su postali pokretačka snaga izvanrednih napretka u prepoznavanju slika, detekciji objekata i razumijevanju scena.

Razvoj CNN-ova započeo je iz želje za oponašanjem sposobnosti ljudskog mozga u obradi vizualnih podatka. Kako su znanstvenici nastojali razviti modele koji mogu razlučiti hijerarhijske obrasce i prikaze u slikama, tako je arhitektura CNN-ova je evoluirala kako bi obuhvatila lokalna receptivna polja, zajedničke težine i učenje hijerarhijskih značajki. Inspirirani ljudskim vizualnim sustavom CNN-ovi se ističu u različitim procesima kategorizacije, sređivanja i selekcije mjesta nastalih prostornih vizuala, omogućujući im da automatski uče zamršene značajke iz neobrađenih slikovnih podataka bez da koriste izvedene značajke.

Konvolucijske neuronske mreže u srži slične su kao i većina specijaliziranih neuronskih mreža no razlikuju se u par ključnih stvari. Slika 6 prikazuje shematski prikaz jedne tipične konvolucijske mreže na osnovu kojega će se pobliže objasniti ključne dijelove te mreže, koji su ono što je čine različitom od većine neuronskih mreža.



Slika 6 Jednostavan prikaz arhitekture konvolucijske neuronske mreže

3.1. Ulazni podaci

Kao što se vidi na Slici 6 Ulazni podaci u Konvolucijsku neuronsku mrežu su pretežito slike. Kao ulazni tenzor uzima se trodimenzionalno polje u kojemu se mreži daje do znanja koja je visina i širina slike kao i koliko kanala postoji. Ako je slika sivih nijansi koristiti će se jedan kanal, a ako se obrađuje slika koja je u boji koriste se 3 kanala kako bi se moglo predstaviti svaku od boja(crvena, zelena, plava). Nekada je poželjno imati još jednu dimenziju ulaznog senzora koja govori koliki će biti batch size odnosno koliko slika će se uzeti u svakoj iteraciji treniranja. Ono što je bitno naglasiti da podaci, uglavnom prije ulaska u mrežu, prolaze kroz pred procesiranje. Pred procesiranje je postupak kojemu je cilj poboljšati kvalitetu podataka. Taj postupak obuhvaća nekoliko koraka koji se mogu i ne moraju primijeniti, a će se primijeniti samo oni koraci koji su potrebni kako bih se dobila maksimalna kvaliteta podataka. Mogući koraci su sljedeći:

Normalizacija

Cilj normalizacije jest skalirati vrijednosti piksela na određeni interval kako bi model lakše mogao učiti. To se obavlja tako da se vrijednosti piksela prebace u float vrijednosti te se skaliraju u interval od [0,1] ili [-1,1] ovisno o zahtjevu zadatka. Neki od uobičajenih načina normalizacije su sljedeći:

1. Min-Max normalizacija

Pronalaze se minimalna i maksimalna vrijednost unutar skupa podataka te se na svaki podatak primjenjuje iduća transformacija:

$$X_{norm} = X - \frac{X_{min}}{\mathrm{X}_{max} - \mathrm{X}_{min}}$$

2. Robusno skaliranje

Podaci se skaliraju ovisno o medijanu i interkvartalnom intervalu što ih čini manje osjetljivim na ekstreme

$$X_{robust} = X - \frac{median}{IRQ}$$

3. L2 Normalizacija

Normaliziranje vektora jediničnom normom

$$X = \frac{X}{||X||^2}$$

4. Promjena veličine

Promjenu veličine obavljamo kako bi osigurali da su svi podaci jednake veličine koja je predodređena.

5. Rezanje

Ovom metodom otklanjaju se nepotrebni dijelovi slike kako bi se stavio fokus na dijelove koji su bitni.

6. Izmjena podataka

Ovim postupkom dobivamo na raznolikosti podatka što je potrebno kada su dobiveni podaci previše slični. Podatke se može transformirati rotacijom, zumiranjem, okretanjem, itd. Povećanjem raznolikosti podataka postiže se mogućnost generalizacije modela.

7. Standardizacija

Vrijednosti piksela se centriraju oko nule te se onda standardiziraju. Taj proces se radi tako da se vrijednosti piksela oduzme srednja vrijednost piksela te se ta razlika podjeli sa standardnom devijacijom.0

3.2. Konvolucijski sloj

Konvolucijski sloj ključna je komponenta u konvolucijskim neuronskim mrežama (CNN). Ovaj sloj uvelike unaprjeđuje obradu podataka čija struktura nalikuje mreži, a najbolji primjer takve strukture podataka jest slika. Sastavljen od filtera koje je moguće naučiti, ovaj sloj izvodi operaciju konvolucije, te tako izvlači hijerarhijske značajke iz ulaza. Iskorištavanjem zajedničkih težina, konvolucijski sloj smanjuje broj parametara, što povećava učinkovitost računanja i pospješuje generalizaciju. Lokalna receptivna polja hvataju prostorne odnose, omogućujući mreži da razazna raznolike obrasce. Korištenje aktivacijskih funkcija, od kojih se najčešće koristi Rectified Linear Units (ReLU), uvodi se nelinearnost u model što pospješuje sposobnost modela da generalizira. Kombiniranjem više filtera i slojeva, konvolucijskom sloju se omogućuje da automatski nauči hijerarhijske prikaze, što ga čini kamenom temeljcem za prepoznavanje slika i zadatke računalnog vida.

3.2.1. Konvolucijski filteri (kernel)

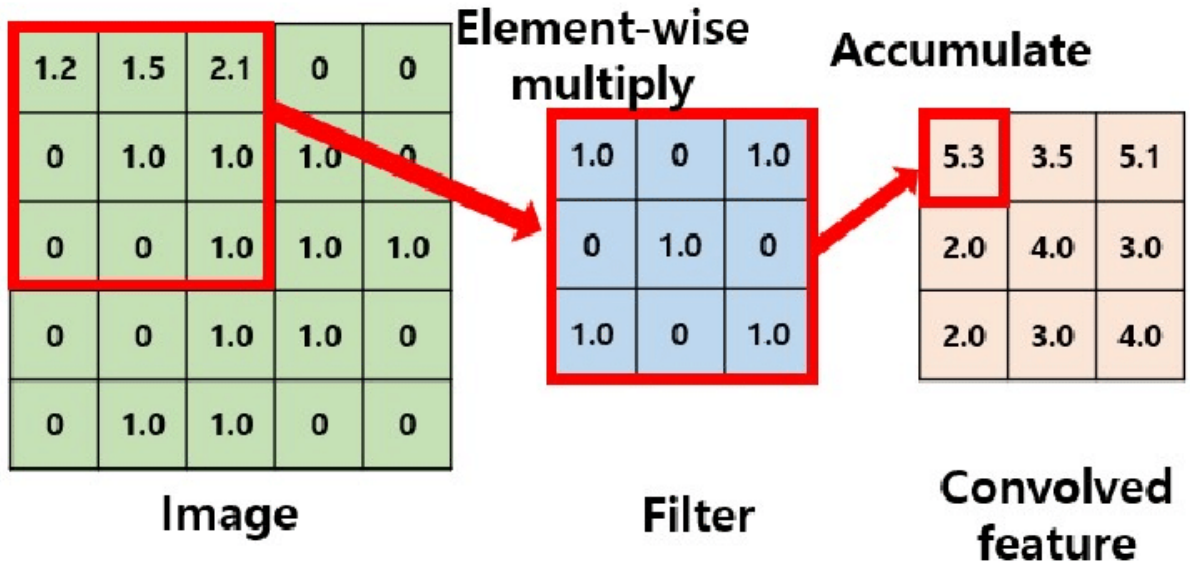
U srcu konvolucijskog sloja nalazi se konvolucijski filter, to je mala matrica težina koje je moguće naučiti koja je srž ekstrakcije značajki iz podatka. Ovi filteri, koji su reprezentacija vizualnih receptora, kreću se po ulaznim podacima s jednom zadaćom, a ta zadaća jest pronaći karakteristične obrasce. Svaki filter, kojeg karakterizira njegovo receptivno polje, ponaša se kao specijalizirani detektor za značajke koji je opredijeljen za određeni aspekt kao na primjer rubovi, teksture ili neke kompleksnije strukture. Filteri, prolazeći po podacima, obavljaju elementarne operacije te kao rezultat dobivamo aktivacijske mape koje pokazuju jezgru detektirane značajke. Parametri, koje je moguće naučiti, prolaze kroz konstantnu optimizaciju korištenjem unatražnog prolaza kako bi se prilagodili zadanom zadatku. Korištenjem dijeljenih težina znatno se smanjuje računalno opterećenje te se pospješuje generalizacija, time se neuronskoj mreži omogućava da prepoznaje obrasce nevezano za njihovu poziciju u prostoru. Jedna od najbitnijih stvari kada se sagleda inicijalizacija filtera jest pronalaženje adekvatne veličine filtera pošto taj odabir uvelike utječe na sposobnost mreže da prepozna detalje ili širi kontekst značajki koje vadimo. Kako bi se odabrala adekvatna veličina filtera potrebno je uzeti u obzir nekoliko stvari, kao prvo treba provjeriti veličinu značajki, ako tražimo neke manje značajke bolji su filteri manjih veličina, ako pak tražimo veće značajke možda je poželjno uzeti veći filter. Nakon toga treba uzeti u obzir rezoluciju slike (manji filteri bolje funkcioniraju na slikama veće rezolucije), i zadnje treba uzeti u obzir zadatak koji obavlja mreža. Najčešće kako bi napravili ispravan odabir potrebno je eksperimentirati s veličinama dok ne pronađemo adekvatnu. Kako bi dobili maksimalnu iskoristivost neophodno je slagati različite filtere s različitim veličinama receptivnih polja. Tako građeni slojevi rade tako da se niži konvolucijski slojevi bave detaljima dok se više slojevi bave shvaćanjem šire slike te se time omogućava mreži da shvaća kompleksne strukture.

3.2.2. Operacija konvolucije

Sama operacija konvolucije neophodan je matematički proces za izvlačenje značajki iz danih ulaznih podataka. To se obavlja tako da Konvolucijski filter, čija će oznaka biti K, primjeni na ulazni tenzor kojega će se označiti sa S. Ako se sada pogleda mapa značajki, element koji se nalazi na poziciji (i, j) može se izračunati na pomoću iduće funkcije:

$$S(i,j) = (K * I)(i,j) = \sum_m \sum_n I(m,n) \cdot K(i-m, j-n)$$

U navedenoj formuli parametri m i n predstavljaju dimenzije filtera, te se zbrajanje odvija na lokalnom području ulaznog tenzora. Elementarni produkt težina filtera i odgovarajućih ulaznih veličina zbrajaju se te se tako dobiva izlazna mapa značajki. Operacija konvolucije koristi dijeljene parametre gdje se iste težine filtera koriste za različite prostorne pozicije. Na taj način dosta se smanjuje broj parametara koje je potrebno naučiti unutar mreže.



Slika 7 Prikaz djelovanja konvolucije

Sa slike 7 jasno se vidi vizualni prikaz gore navedene formule. Ovdje je prikazan matrični izgled slike na koju djeluje filter veličina 3x3. Iz slike jasno se vidi da se adekvatne vrijednosti sa slike pomnožene s filterom te zbrojene daju element u mapi značajki kao što to iziskuje formula.

3.2.3. Dijeljenje parametara

Dijeljenje parametara specifična je karakteristika konvolucijskih slojeva. Ono što je bitno za shvatiti jest da se pod dijeljenje parametara smatra korištenje istog seta težina preko više pozicija te se time smanjuje broj potrebnih parametara modela. Ova strategija omogućava ostvarivanje efikasnijeg treniranja modela kojima je potrebna manja računalna snaga.

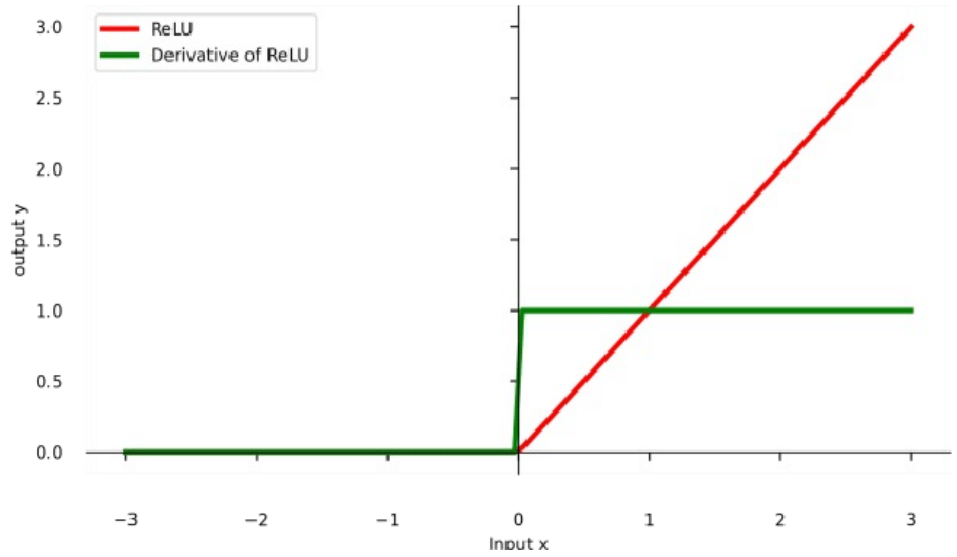
Podijeljene težine služe kao lokalni detektori značajki koji prepoznaju obrasce neovisno o njihovoj lokaciji. To se poravnava s načelom da su određene značajke, kao što su rubovi i teksture, neovisne o prostornom položaju. Posljedično dijeljenje parametara omogućava dozu neovisnosti o translaciji što pospješuje sposobnost mreže da generalizira naučene značajke na različitim položajima. Efikasnost koja se dobiva na ovaj način vrlo se lagano prepozna kada uzmemo u obzir alternativu. Ako zamislimo da za svaku poziciju filter ima jedinstvene težine, to bi dovelo do ogromnog rasta količine parametara što bi, ne samo ekstremno povećalo potrebnu računalnu snagu, već bi stavilo mrežu u opasnost od prenaučenosti. Na ovaj način izbjegavamo ove probleme već i omogućujemo mreži da se fokusira na apstraktne reprezentacije dok podatak prolazi kroz slojeve tako što efikasno hvata lokalne obrasce.

3.2.4. RELU aktivacijska funkcija

U ovom poglavlju pokazat će se zašto je RELU funkcija adekvatan odabir za zadatke koje se bave prepoznavanjem slike kao što je naš zadatak. Objasnit će se kako funkcionira RELU aktivacijska funkcija te njene prednosti nad sigmoidom i tangensom hiperbolni. Matematički zapis RELU funkcije prikazan je formulom.

$$\mathrm{ReLU}(x) = \max(0, x)$$

Iz formule vidi se da je x ulazna varijabla te da funkcija radi tako da vraća maksimum između nule i dobivenog podatka što zapravo znači da funkcija vraća nulu u slučaju kada je primila negativnu vrijednost. Ako je funkciji dana pozitivna vrijednost ona će ju jednostavno vratiti. To se također jasno vidi i sa grafa funkcije na Slici 8.



Slika 8 Graf RELU funkcije i njene derivacije

Na Slici 8 prikazan je graf RELU funkcija iz prikazanog grafa se jasno vidi zašto je ona tako pogodna za zadatke koji uključuju prepoznavanje objekata na slici. Kao prvo očito je da je funkcija ne linearna. Uvođenjem ne linearnosti u model omogućujemo mu da uči kompleksne ne linearne veze između podataka. To je ključno kada je potrebno prepoznavati obrasce i značajke. Također pošto funkcija efektivno miče sve negativne vrijednosti ona razrjeđuje model tako što smanjuje broj aktiviranih neurona. Uz to funkcija je relativno jednostavna te ne zahtijeva veliku računalnu snagu kako bih obavila svoj proračun. Kada analiziramo ove značajke RELU funkcije jasno je da ona dosta rasterećuje količinu računanja koja je potrebna tokom prolaska podatka kroz model. Jedan od čestih problema koji se pojavljuje kod treniranja dubokih neuronskih mreža jest problem nestajućeg gradijenta. Problem nestajućeg gradijenta posebno je izražen kod mreža s mnogo slojeva. Do tog problema dolazi kada je gradijent funkcije gubitka neuronske mreže u odnosu na parametre neuronske mreže postane ekstremno malen. To smanjenje gradijenta događa se za vrijeme unatražnog prolaza tijekom treniranja mreže. Rezultat tog umanjivanja gradijenta jest da se težine u mreži smanje na ekstremno male vrijednosti te se mreža počinje mučit sa sposobnosti da nauči bilo kakvu smislenu reprezentaciju značajki. RELU se dosta efektivno bori s tim problemom tako što dopušta isključivo pozitivne gradijente za pozitivne vrijednosti. Neke od mogućih varijacija RELU funkcije koje mogu biti korisna alternativa, ovisno o situaciji, su sljedeće. Prva varijacija koja će se analizirati jest Leaky RELU. Leaky

RELU osmišljen je kako bih se riješio problem umirućeg RELU-a (problem gdje se neki od neurona postaju neaktivni za vrijeme treniranja). Formula ove varijante RELU operacije glasi.

$$\mathrm{LReLU}(x) = \max(\alpha x, x)$$

Koeficijent α predstavlja neku jako malu pozitivnu konstantu sa kojom će se množiti ulazni podatak kako bi ga se dovelo u okolinu nule. Iduća na red dolazi Parametric RELU(PReLU). Ova inačica RELU funkcije radi slično kao i leaky RELU no ona sadrži parametar, u formuli najčešće označen kao a, kojeg je moguće naučiti. Pomoću tog parametra odrađuje se nagib negativnog djela funkcije te omogućava neuronskoj mreži da adaptivno nauči najbolji nagib za vrijeme treniranja. Formula za ovu varijantu RELU jest sljedeća

$$\mathrm{PRReLU}(x) = \max(0, x) + a * \min(0, x)$$

Zadnja varijanta koju je bitno obijasniti jest Exponential linear unit(ELU). Ova varijanta glatko rješava problem umirućeg RELU-a te njena formula glasi

$$\mathrm{ELU}(x) = x \;\; if \; x > 0, else \; a * (\exp(x) - 1)$$

Prikazani su mnogi pozitivni aspekti korištenja RELU funkcije kao aktivacijske funkcije, no bitno je naglasiti da to ne znači da je to apsolutno rješenje. Odabir aktivacijske funkcije ovisit će prvenstveno o zadatku kojeg trebamo ispuniti. Bez obzira na sva svoja pozitivna svojstva može se dogoditi da našem zadatku više paše koristiti sigmoidu ili pak tangens hiperbolni.

3.3. Sloj sažimanja (pooling layer)

Sloj sažimanja ključna je komponenta svake konvolucijske mreže te se najčešće nalazi direktno iza konvolucijskog sloja. Njegova svrha jest sažimanje uzorkovanja mape značajki. Prilikom sažimanja bitno je da se očuvaju osnovne prostorne informacije dok se dimenzije

podataka smanjuje. Sažimanje se odvija tako da se ulazna mapa značajki podjeli na dijelove koji se međusobno ne preklapaju te se nad svakom regijom podataka izvršava sažimanje. Postoje dvije vrste sažimanja koje se koriste. Max-Pooling sažimanje može se opisati idućom formulom

$$MaxPool(x) = max(region(x))$$

dok se AvgPooling opisuje formulom

$$AvgPool(x) = \frac{\sum region(x)}{RegionSize}$$

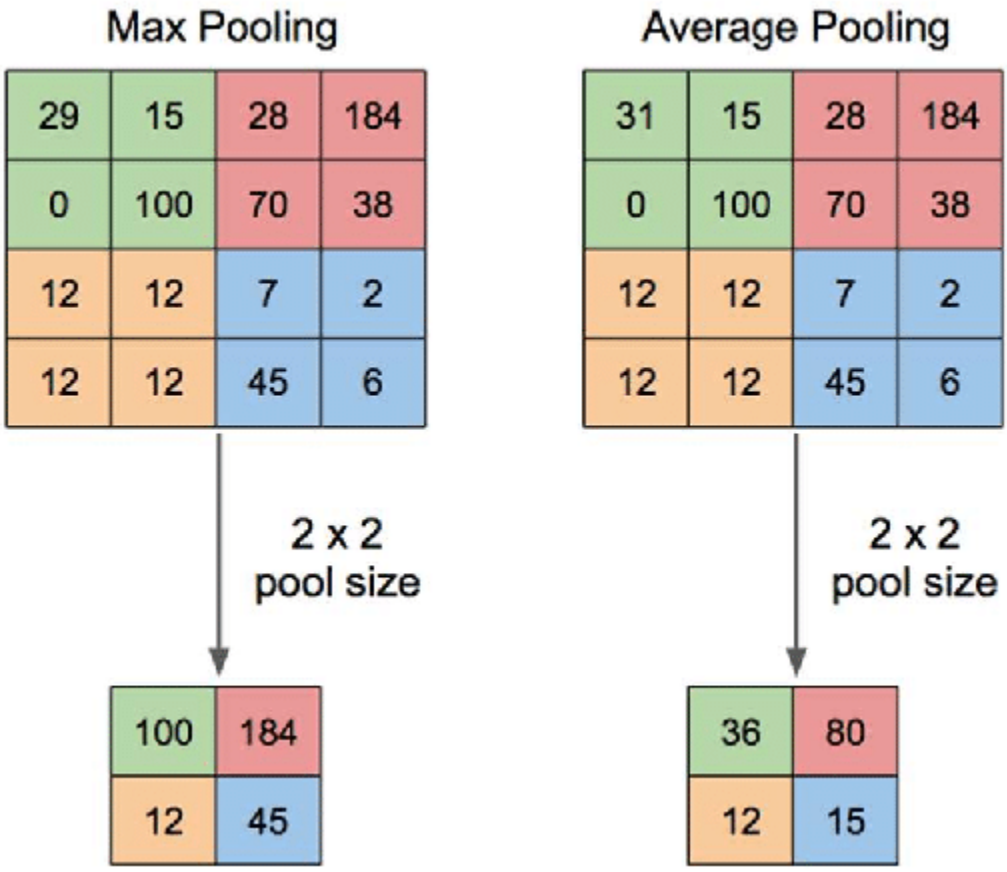
U prikazanim formulama Region(x) predstavlja odabrani dio mape značajki dok RegionSize predstavlja broj elemenata unutar odabrane regije. Ono što se jasno vidi iz formula jest da Max pooling odrađuje koji element će biti odabran tako da jednostavno odabere najveći element unutar dobivene regije, dok AvgPooling računa prosječnu vrijednost elementa u regiji te nju prosljeđuje dalje i tako sažima dimenzije podataka.

Kada govorimo o sloju sažimanja bitno je naglasiti dva karakteristična parametra a to su korak i zadebljanje. . Korak određuje za koliko mjesta će se okvir sažimanja pomaknuti prilikom uzimanja svake iduće regije, ovaj parametar utječe na jačinu sažimanja. Zadebljanje se koristi kako bih okvir sažimanja zahvatio cijelo područje ulaznih podataka, a to se ostvaruje dodavanjem piksela zanemarivih vrijednosti koji neće utjecati na sam proračun sažimanja. Veličinu izlaznih podataka nakon sažimanja može se izračunati prema sljedećoj formuli:

$$\text{Output Size} = \frac{\text{Input Size} - \text{Pool Size} + 2 \times \text{Padding}}{\text{Stride}} + 1$$

Input size označava veličinu ulaznog podatka. Filter size predstavlja veličinu okvira koji će se micati po ulaznim podacima kako bih se obavilo sažimanje. Dodaje se dvostruki padding pošto se on postavlja s obje strane slike. Stride predstavlja korak pomicanja okvira. Na kraju se na rezultat dodaje jedan. Jedinica se dodaje kako bi osigurali da postoji barem jedna pozicija u kojoj filter prekriva cijeli ulazni podatak. Bez te jedinice može se dogoditi da se izračuna manja veličina od očekivane te da se izgube informacije o rubovima ulaznog podatka.

Ono što dobivamo korištenjem ovog sloja jest smanjenje broja parametara u daljnjim slojevima to utječe prvenstveno na efikasnost programa. Uz to poboljšava sposobnost mreže da se prilagođava malim promjenama u obrascima tako što grupira lokalne značajke. Korištenjem ovog sloja omogućava se modelu da na nižim slojevima hvata lokalne obrasce te pomoću njih kreira kompleksniju reprezentaciju u višim slojevima.



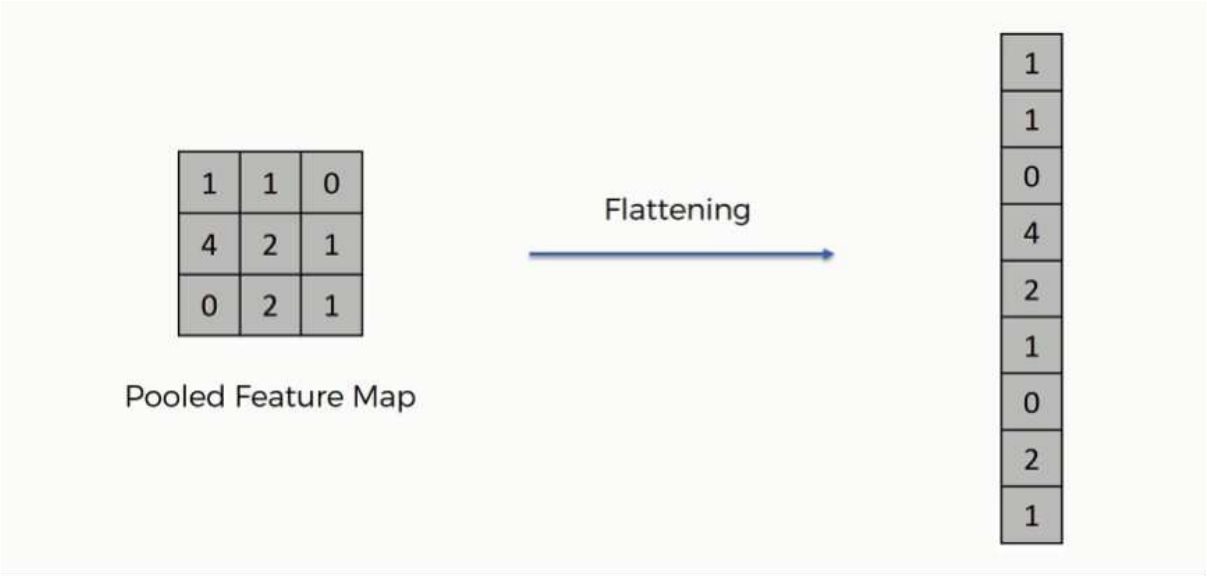
Slika 9 Prikaz MAX i AVG pool fiunkcija

3.4. Klasifikacijski sloj

Klasifikacija je ključan aspekt konvolucijskih neuronskih mreža. Svrha klasifikacijskog sloja jest da pretoči podatke dobivene iz konvolucijskog sloja u reprezentaciju pojedinih predikcija klasa. Unutar ovog sloja podacima se pridjeljuju probabilističke vrijednosti koje označuju vjerojatnost da dani primjer pripada nekoj od mogućih klasa. Sve ovo najčešće se odvija tako što se podaci provlače kroz tri različita sloja a to su flatten, Potpuno povezani sloj i softmax sloj. Kombinacija ova tri sloja omogućava mreži da ispravno klasificira podatke te da im pridjeljuje vjerojatnosti pripadnosti nekoj klasi na ispravan način. Da ovaj sloj ne postoji podaci bi bili teško razumljivi i nečitki za ljude.

3.4.1. Flatten sloj

Flatten sloj ima ključnu ulogu u transformaciji podataka kako bih se podaci iz prostorne hijerarhijske mape značajki mogli bezbrižno proslijediti potpuno povezanom sloju na daljnju obradu. Matematički gledano u ovom sloju prostorni trodimenzionalni tenzor dimenzija HxWxC, koji se dobiva iz konvolucijskog sloja, pretvara se u linearni jednodimenzionalni niz. Ovom pretvorbom lokalno izvlačenje značajki postaje globalno što je potrebno potpuno povezanom sloju. Flatten sloj sprječava problem eksplozije parametara. Eksplozija parametra problem je koji se javlja u potpuno povezanom sloju. Do problema dolazi kada na ulaz mreže dovodimo visoko dimenzijske ulazne podatke što dovodi do eksponencijalnog povećanja u broju težina i pristranosti u mreži. Radi tog povećanja dolazi do veće potražnje memorije kao i korištenje više resursa za provedbu računanja. Uz to s velikim brojem parametara povećava se vrijeme treniranja kao i vjerojatnost da model postane prenaučen. To se dešava jer će model uz stvari koje je poželjno da nauči također učiti i na šumu što će u konačnici rezultirati smanjenjem sposobnosti generalizacije na novim podacima. Kao što se vidi iz naravi problema Flatten sloj savršeno paše kako bi se vjerojatnost tog problema znatno umanjila. Sa Slike 10 može se vidjeti kako funkcionira flatten sloj na nekom primjeru.



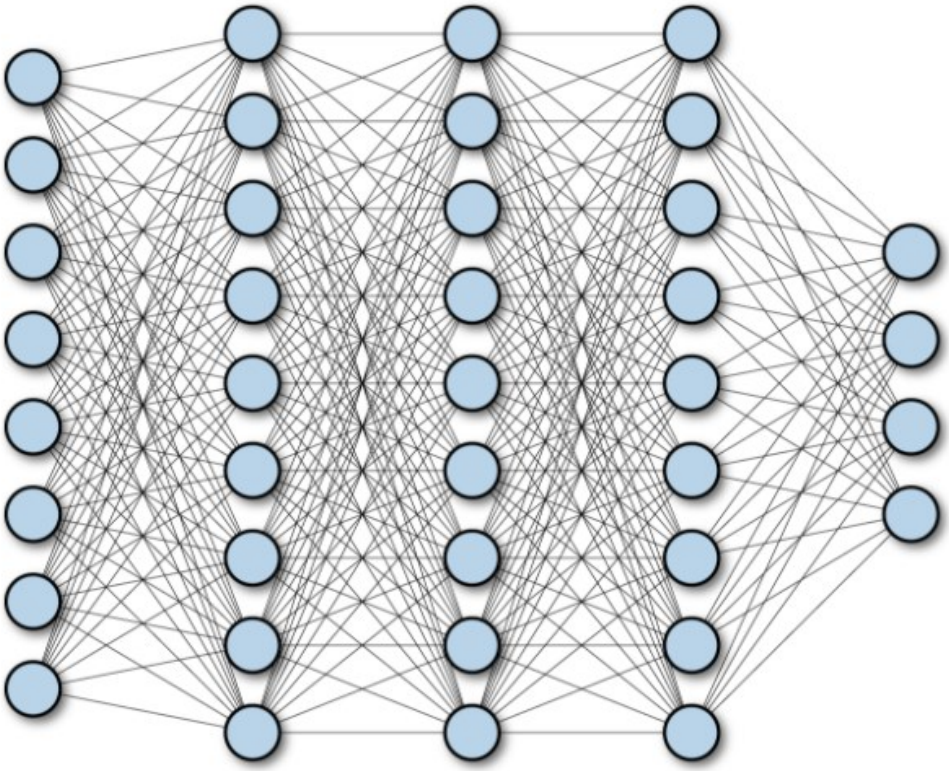
Slika 10 Rad Flatten sloja

3.4.2. Potpuno povezani sloj

Potpuno povezani sloj koristi se tokom klasifikacije radi njegove sposobnosti da hvata kompleksne značajke visokih razina te ulazim podacima pridodaje globalni kontekst. To znači da je ovaj sloj savršen kandidat koji bi sažeo sve lokalne značajke u jednu odluku te kategorizirao zadani ulazni podatak u neku od klasa. Bitna karakteristika potpuno povezanog sloja jest ta da su svi neuroni u sloju povezani sa svim neuronima u prethodnom sloju čime se ostvaruje gusta međusobno povezana struktura. Matematički gledano ovaj sloj obavlja iduću operaciju.

$$Y = \sigma(WX + b)$$

Ova formula predstavlja jednostavno matrično množenje. X predstavlja ulazne podatke u sloj koji su u ovom slučaju jednodimenzionalni vektor koji je dobiven od flatten sloja, W predstavlja matricu težina potpuno povezanog sloja, a b predstavlja pristranost. Nakon odrađivanja matričnog množenja W sa X potrebno je dodati pristranost b, zatim prije vraćanja izlaza potrebno je još primijeniti neku nelinearnu aktivacijsku funkciju(RELU, sigmoida, tanh) kako bi izlaz bio zadovoljavajući.



Slika 11 Primjer potpuno povezanih slojeva

Prednost potpuno povezanih slojeva je ta da omogućuju neuronskoj mreži da prepoznaje obrasce koji se potencijalno protežu preko cijelog ulaznog podatka. To ga čini ekstremno pogodnim za zadatke u kojima je potrebno holistički shvatiti ulazne podatke kao što je to kod prepoznavanje sa slike. Korištenjem ovakvih slojeva lagano se dolazi do integracije podataka s različitih dijelova ulaznih podataka što pospješuje ekstrakciju globalnih značajki.

Sagledaju li se sve navedene prednosti potpuno povezani sloj čini se kao svršeno rješenje, no bitno je naglasiti da se treba paziti kod korištenja potpuno povezanih slojeva. Naime kako potpuno povezani sloj ima mnogo parametara te zahtjeva visoku računalnu snagu za izvršavanje računanja postoji veliki rizik od prenaučenosti. Taj rizik uvelike dolazi do izražaja kada je skup podataka za učenje ograničen. Također uz visoki broj parametara dolazi i velika memorijska zahtjevnost što može dovesti do smanjene skalabilnosti neuronske mreže

3.4.3. Softmax

Softmax je matematička funkcija koja pretvara vektor realnih brojeva u probabilističku distribuciju. Kada govorimo o korištenju softmax funkcije u kontekstu konvolucijskih neuronskih mreža, ona se najčešće koristi kao završni sloj koji na izlaz daje vjerojatnosti pripadnosti podatka nekoj od zadanih klasa. Ako se na primjer funkciji kao ulaz predaje vektor x koji ima dužinu K, gdje K predstavlja broj klasa prema kojima svrstavamo podatke, onda se matematički softmax izražava na idući način.

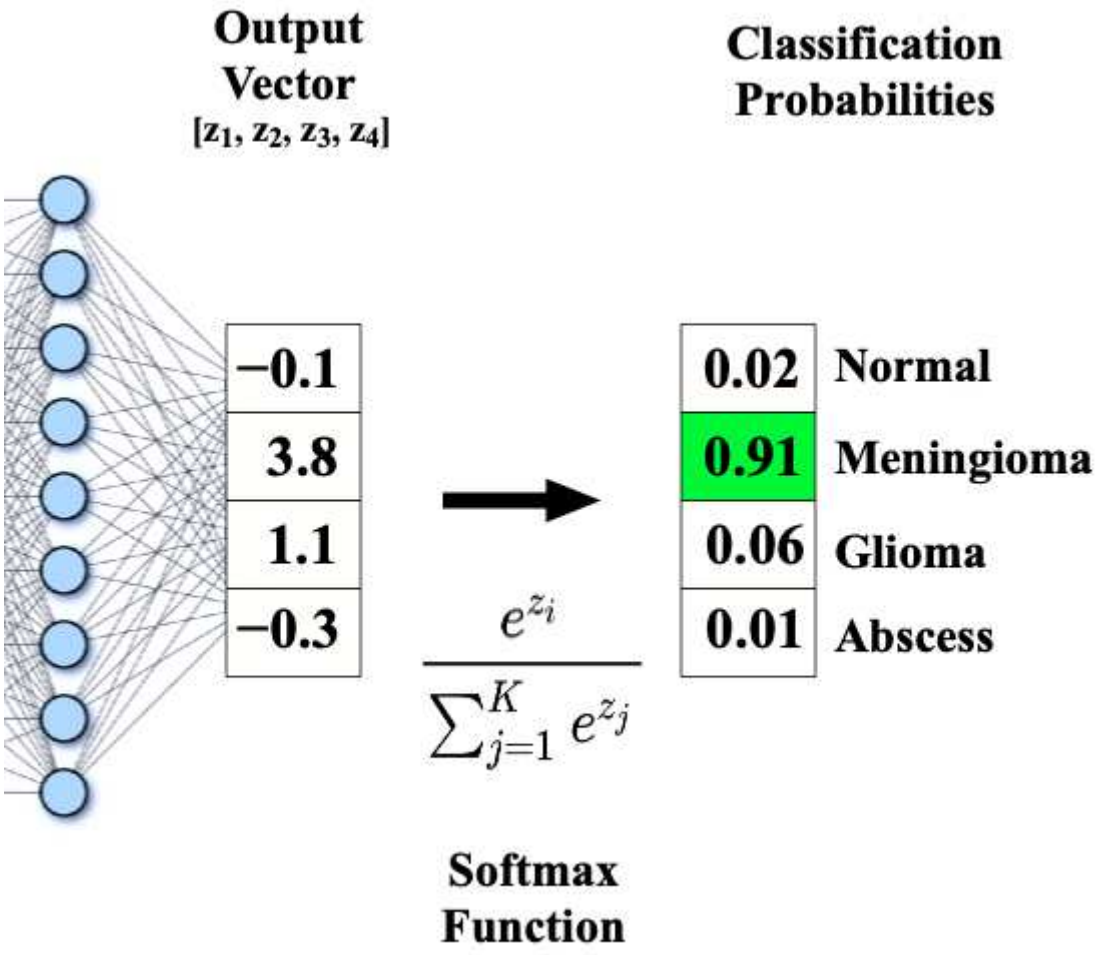
$$\text{Softmax}(x)_i = \frac{e^{x_i}}{\sum_{j=1}^{N} e^{x_j}}$$

U gore navedenoj formuli softmax(x) i predstavlja vrijednost i-tog elementa dobivenog nakon provođenja softmax operacije. Prilikom računanja eksponenciju se koristi kako bi se naglasilo i normaliziralo magnitude ulaznih vektora. Time je ograničeno da se izlazne vrijednosti nalaze u intervalu [0,1] te da zbroj tih vrijednosti bude jedan što osigurava da izlaz ima probabilističku distribuciju.

Unutar same mreže softmax se postavlja nakon potpuno povezanog sloja. To znači da softmax kao ulaz dobiva izlazne podatke potpuno povezanog sloja. Konačno odabir klase odvija se po formuli

$$predicted\_class = argmax_i\, y_i$$

Ova formula govori da se odabire ona klasa čija je vjerojatnost najveća. Softmax je ključna operacija u konvolucijskim neuronskim mrežama jer dozvoljava jasno shvaćanje rezultata kao i uvid u sigurnost modela za svaku klasu. Uz to, za vrijeme treniranja, softmax se koristi u kombinaciji s unakrsnim gubitkom entropije kako bi navodio model prema pravilnom odabiru tako što ga kažnjava ovisno o razlici između predviđene klase i stvarne klase podatka.



Slika 12 Primjer softmax funkcije

4. Preneseno Učenje

U rastućem području strojnog učenja uvijek se traže efikasniji i efektivnije modeli. Ova činjenica dovela je do toga da se istražuju inovativni pristupi. Jedan od tih pristupa, koji se pokazao vrijednim pažnje, jest preneseno učenje. Preneseno učenje koristi se znanjem stečenim za jedan zadatak kako bi se unaprijedila funkcionalnost nekog drugog modela koji je zadužen za neki drugi, ali tematski povezani, zadatak. Prelazak na korištenje prenesenog učenja pokazao se kao stepenica za velike napretke u računalnom vidu i obradi prirodnog jezika kao i u područjima zdravstva i robotike. Sposobnost prijenosa znanja kroz različite domene uvelike ubrzava treniranje i pospješuje prilagodljivost i generalizaciju modela. U ovom poglavlju objasniti će se neki od osnovnih koncepata prenesenog učenja te pogledati koje su sve prednosti korištenja prenesenog učenja i na koje sve probleme se može naići u ovoj domeni.

4.1. Osnovni koncepti

Kada govorimo o osnovnim konceptima na kojima je zasnovano prijenosno učenje prvoa stvar na koju je potrebno obratiti pažnju jest prijenos znanja. Prijenos znanja u kontekstu stvarnog svijeta obično se svodi na usmenu predaju, zapisivanje znanja za buduće generacije itd. No kada se to sagleda u domeni strojnog učenja taj postupak zapravo će značiti iduće. Želimo prenijeti sve naučene reprezentacije, parametre i značajke iz izvornog zadatka na novi zadatak. Radi toga model na koji se znanje prenijelo konvergira brže te pokazuje bolje performanse na skupovima podataka s ograničenim brojem označenih podataka

Iduća bitna stavka koju moramo pogledati jest prilagođavanje domeni. Ovaj koncept javlja se kao posljedica toga da u stvarnom svijetu obično dolazimo do scenarija u kojem zadaci imaju svoju jedinstvenu distribuciju podataka. Prilagođavanje domene odnosi se konkretno na to da model podesimo tako da može funkcionirati pravilno na ciljanoj domeni bez obzira na to što se ta domena razlikuje od inicijalne domene za koju je model napravljen. Ovim postupkom modelu su omogućava da nadvlada probleme koju su specifični za ciljanu domenu, te tako poboljšavamo sposobnost generalizacije modela. Postoji nekoliko vrsta metoda koje se mogu koristiti kako bi prilagodili domenu.

Postoji više vrsta Metoda temeljene na značajkama, a neke od njih su analiza glavnih komponenata koja za cilj ima smanjenje dimenzije značajki kako bi se značajke

korespondirale s dimenzijama ciljane domene distribucije. Također, jedna od metoda temeljena na značajkama je učenje značajki koje su invarijante na domenu, što podrazumijeva ekstrakciju onih značajki koje su nepomične na cijeloj domeni i to se radi kako bi se osiguralo da se model fokusira na bitne aspekte neovisno o ciljanoj domeni. Metode temeljene na instancama uključuju metode dodjeljivanja težine instancama inicijalne domene kako bi se smanjio utjecaj razlike u domenama, te suparničko treniranje gdje se dodaje druga mreža koja usklađuje distribuciju značajki između inicijalne i ciljane domene za vrijeme treniranja modela.

Metode bazirane na modelima u koje ubrajamo precizno podešavanje modela kao i metode nad cjelinama koje kombiniraju predikcije više različitih modela treniranih na inicijalnim i ciljnim podacima.

Metode na nivou podataka koje se odnose na manipulaciju ulaznih podataka kako bi se postiglo prilagođavanje domeni. U ove metode spadaju funkcije kao što su izmjenjivanje podataka, gdje se generiraju podaci za treniranje tako što se postojeći podaci transformiraju, te samo-uvježbavanje to jest treniranje modela da bude invarijantan na transformacije specifične za domenu.

Pred-trenirani modeli ključna su komponenta kod prenesenog učenja. Ovi modeli inicijalno se treniraju na ekstremno velikim skupovima podataka te su namijenjeni da obavljaju generičke zadatke. Takvi modeli mogu se precizno podesiti kako bi se prilagodili specifičnim zadacima s relativno malim skupovima podataka. Ovaj pristup se pokazao posebno djelotvoran kada radimo sa zadacima koji imaju ograničene resurse i količinu podataka na kojima se može učiti.

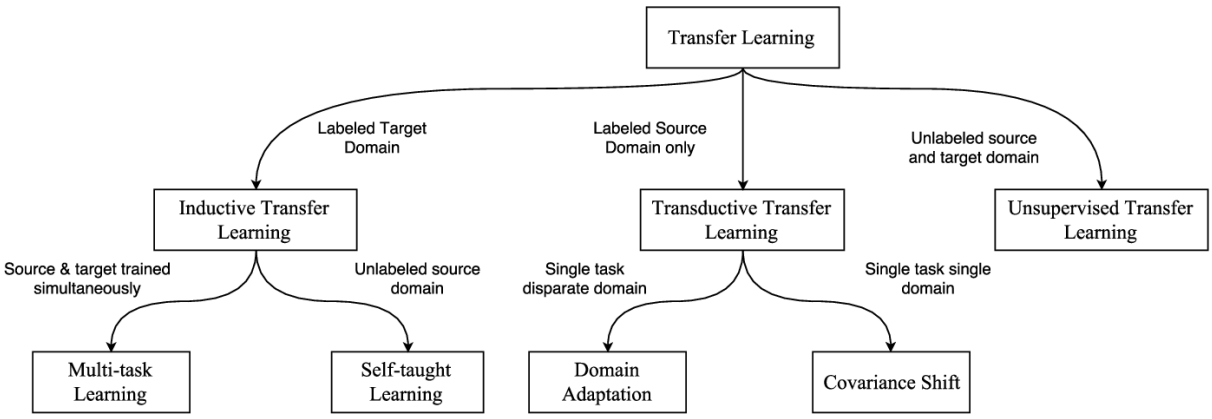
Bitna stvar za uzeti u obzir kod pred-treniranih modela jest precizno podešavanje. Precizno podešavanje predstavlja proces namještanja parametara pred-treniranih modela kako bi on moga funkcionirati na novom zadatku. Ovim procesom model se specijalizira za novi zadatak dok zadržava sve bitne informacije koje je stekao za vrijeme originalnog treniranja. Modelu se poboljšavaju performanse bez odbacivanja stečenog znanja šireg konteksta.

Zadnja stvar koju moramo spomenuti kada govorimo o prenesenom učenju jest važnost sličnosti zadataka. Efikasnost prenesenog učenja usko je povezana sa sličnosti između inicijalnog zadatka za koji je model bio namijenjen te ciljanog zadatka za koji želimo iskoristiti model. Kao što je i logično zadaci koji imaju više sličnih značajki i karakteristika vidjet će veću prednost u korištenju prenesenog učenja. To je zato što je jedna domena lakše

prihvatljiva i primjenjiva na drugu domenu. Ovaj dio procesa podrazumijeva stratešku selekciju izvornog modela kako bi se maksimizirala efikasnost prijenosa znanja.

4.2. Vrste prenesenog učenja

Unutar ovog poglavlja analizirati će tri glavne vrste prenesenog učenja. Za svaku od vrsta objasniti će se kako funkcionira te koje su njezine prednosti i nedostaci. Također objasnit će se zašto se određena vrsta koristi za određeni tip zadatka.



Slika 13 Prikaz grananja tipova prenesenog učenja

4.2.1. Induktivno preneseno učenje

Induktivno preneseno učenje je vrsta prenesenog učenja koja se fokusira na to da se iskoristi znanje inicijalne domene kako bi se poboljšale performanse modela na ciljanoj domeni koja je povezana s inicijalnom ali joj nije jednaka. To znači da su označeni podaci korišteni za treniranje uglavnom jednaki kao i podaci koji su bili korišteni za pred-treniranje, no zadatak koji će mreža obavljati razlikuje se od zadatka inicijalne mreže. Postoje dvije podvrste induktivnog prenesenog učenja a to su Paralelno učenje više zadataka i Samouko učenje. Paralelno učenje više zadataka odnosi se na realizaciju modela koji je sposoban izvoditi više različitih zadatak koji su slični. Ovakav način rada podrazumijeva da se tijekom treniranja model učio na više zadataka istovremeno koristeći se sličnostima između zadataka kako bih poboljšao performanse modela. Dijeljenjem reprezentacije između dva zadatka Paralelno učenje više zadataka omogućava bolju generalizaciju na izvornom zadatku. Korištenje ovakvog pristupa pruža brojne prednosti kao što su smanjenje potrebnih ulaznih podataka, manja vjerojatnost prenapučenosti zbog dijeljenja reprezentacija te povećana brzina učenja.

No bitno je naglasiti da s takvim načinom učenja dolaze i svojevrsni problemi. Glavni problem koji se javlja kod ovog pristupa jest kako kombinirati signale više različitih zadataka u jedan model. Taj problem može se reducirati korištenjem metoda grupiranja zadataka ili preklapanja zadataka. Druga podvrsta Induktivnog prenesenog učenja jest samouko učenje. Ovaj pristup podrazumijeva scenarij u kojem model uči na zadatku za koji su ulazni podaci označeni te se to znanje primjenjuje na ciljani zadatak čija je domena nema označene podatke. Model se trenira na velikoj količini označenih podataka kako bi naučio dobre značajke koje se mogu iskoristiti za specifičan zadatak. Ovo rješenje postaje jako pogodno za zadatke u kojima raspolažemo s malim brojem ulaznih podataka ili ako je prikupljanje podataka skupo.

4.2.2. Transduktivno preneseno učenje

Transduktivno preneseno učenje uključuje prijenos znanja iz određene inicijalne domene na drugu, ali sličnu ciljanu domenu, s glavnim fokusom na slučaj gdje inicijalna domena ima veliku količinu označenih podataka, dok ciljana domena sadrži samo ograničenu količinu označenih podataka. U transduktivnom učenju, model je već bio izložen skupu za treniranje i skupu za testiranje, te predviđa oznake testnog skupa na osnovu onoga što je naučio iz poznatog skupa podataka. Ovaj pristup posebno je koristan kada je skup označenih podataka za ciljano područje malen, jer koristi veliku količinu označenih podataka iz izvorne domene kako bi poboljšao performanse modela na ciljanom području koje ima ograničenu količinu označenih podataka. Primjer transduktivnog prijenosa učenja je recimo model koji se bavi prepoznavanjem različitih emocija na osobi uz ograničenu količinu ulaznih podataka iz ciljane domene. Neki od čestih problema koji se javljaju kod ove vrste prenesenog učenja su sljedeći. Razlika u domeni odnosno postojanje značajnih razlika između izvorne i ciljane domene. To može predstavljati veliki izazov u transduktivnom prijenosu znanja. Kako bi mogli prilagoditi izvornu domenu našoj ciljanoj domeni, pogotovo u slučaju kada se one značajno razlikuju, zahtjeva korištenje robusnih tehnika prilagođavanja domene. Također postoji rizik od negativnog prijenosa. Negativan prijenos događa se kada znanje preneseno na ciljanu domenu pogorša performanse modela umjesto da ih poboljša. To može nastati ako su izvorna i ciljana domena previše različite ili ako znanje nije znanje koje prenosimo relevantno za ciljani zadatak. Ovaj problem rješava se na način da se uvedu jednostavne metode koje detektiraju negativan prijenos kako bi smanjile kolektivni šum klasa i samim time produljile vrijeme do negativnog prijenosa.

Vrlo je bitno izabrati koji slojevi će se prenositi, a na kojima će se odrađivati precizno podešavanje. Ovaj postupak je ključan u transduktivnom prenesenom učenju. Mogućnost prijenosa značajki smanjuje se kako razlika između ciljane i izvorne domene povećava. Određivanje optimalnih slojeva za prijenos predstavlja netrivijalan zadatak. Niži slojevi obično su zaslužni za hvatanje generičkih značajki te ih je kao takve puno lakše prenijeti nego više slojeve koji su zaduženi za hvatanje specifičnih značajki. To se može ostvariti tako da se prouče aktivacijski obrasci sloja te se nauči koje značajke oni predstavljaju. Neke od ostalih strategija koje se koriste kako bi odlučili koje su korištenje pred-treniranih modela, precizno podešavanje najbitnijih slojeva za dani zadata k te eksperimentiranje s kombiniranjem različitih prenesenih i precizno podešenih slojeva kako bi se pronašla optimalna kombinacija. Pravilnim izborom slojeva moguće je razriješiti izazove vezane za transduktivno prenošenje znanja. Ako se sagledaju ovi izazovi vidi se da se transduktivno preneseno učenje suočava s kompleksnim izazovima te je ključno implementirati tehnike koje se bave tim izazovima kako bi omogućili maksimalne performanse modela.

4.2.3. Nenadzirano preneseno učenje

Nenadzirano preneseno učenje vrsta je prenesenog učenja u kojoj se znanje stečeno za vrijeme učenja na označenim podacima izvorne domene primjenjuje kako bi se poboljšale performanse modela na specifičnoj ciljanoj domeni za koju postoji jako malo ili uopće ne postoje označeni podaci. Ključ ovog pristupa jest sposobnost modela da generalizira podatke iz izvorne domene te to efikasno primjeni na povezanu ali različitu ciljanu domenu. Ovaj pristup omogućava rješavanje problema manjka podatka unutar ciljane domene što ga čini ekstremno pogodnim za realne slučajeve u kojima je dohvaćanje označenih podataka jako nepraktično ili skupo. Ovaj pristup koristi se nekim od tehnika korištenim u nenadziranom učenju. Te tehnike omogućavaju modelu da izvlači značajne informacije iz podataka bez navođenja i označenih primjera. Neke od mogućih tehnika su navedene u nastavku. Grupiranje gdje se podaci skupljaju u grupe oko centroida ovisno o sličnosti podatka u odnosu na taj centroid ili hijerarhijsko grupiranje gdje se podaci slažu u strukturu nalik stablu. Zatim se mogu koristiti generativne metode kao recimo auto enkoder kako bi mreža naučila efikasne reprezentacije kodiranjem i dekodiranjem podataka. Također jedna od metoda koja se primjenjuje jest korištenje apriornih algoritama kako bi se pronašli česti skupovi u ulaznom skupu te se otkrile asocijacije između varijabli.

Matematički gledano nenadzirano preneseno učenje podrazumijeva optimizaciju parametara modela(ϑ) minimizacijom funkcije gubitka karakteristične za zadani zadatak. Pred treniranje na izvornoj domeni pokušava minimizirati $min_{\theta} L_{T_S}(f_{\theta}(x), y)$ gdje $f_{\theta}(x)$ predstavlja modelov izlaz za dani ulazni podatak x s parametrima ϑ , dok je y oznaka za taj podatak odnosno označava klasu kojoj podatak treba pripadati. Nakon optimizacije te funkcije prelazi se na precizno podešavanje funkcije $min_{\theta} L_{T_T}(f_{\theta}(x), yT)$ gdje je yT oznaka klase u ciljanoj domeni.

5. Tehnologije korištene za realizaciju zadatka

U sklopu ovog poglavlja objasniti će se tehnologije koje su korištene za realizaciju programa koji klasificira tipove vozila. Pojasniti će se zašto su pojedine tehnologije odabrane te kako su primijenjene na zadani zadatak. Također analizirati će se neki od algoritama koji su korišteni za realizaciju zadatka te objasniti kako oni funkcioniraju.

5.1. Tensorflow 2

Tensorflow je open-source biblioteka za strojno učenje koju je razvio Google Brain tim. Osmišljena je kako bih se olakšao razvoj i treniranje modela strojnog učenja pogotovo u okviru zadataka kao što su duboko učenje i aplikacije zasnovane na neuronskim mrežama. Tensorflow pruža široki izbor alata, biblioteka i zajedničkih resursa kako bi programerima omogućio lakšu izgradnju i implementaciju modela strojnog učenja na više platformi. Tensorflow pruža mnoge pogodnosti kao što su fleksibilnost modela na širok broj aplikacija, skalabilnost tako što omogućava treniranje na različitim hardverskim platformama uključujući CPU, GPU, TPU, korištenje važnih API-ja kao keras, dobar ekosustav aspektu postojanja Tensorflow lite za mobilne aplikacije i tensorflow.js za web aplikacije te ima jako veliku zajednicu što znači da postoji mnogo informacija na internetu.

Tensorflow 2 označio je značajan napredak u odnosu na Tensoflow radi toga što je njegov inicijalni način rada koristi trenutno izvršavanja. Trenutno izvršavanje znači da se operacije mogu izvoditi u trenutku u kojem su pozvane, za razliku od Tensorflowa gdje je inicijalni način rada bio izvršavanje na temelju grafa. To znači da se prije izvršavanja naredbi gradio prikaz koji je bio reprezentacija tijeka izvođenja pojedinih matematičkih i računalnih operacija u obliku direkcijskog grafa. Također Tensorflow 2 integrirao je keras API kao službeni API Tensorflow platforme. Keras API kreiran je kako bi korisnici jednostavno i intuitivno mogli izrađivati vlastite slojeve, funkcije gubitka i ostale stvari potrebne u izradi neuronskih mreža.

Nakon što su analizirane osnovne rada Tensorflow biblioteke postaje jasno zašto se baš ona javlja kao savršen kandidat kada govorimo o građenju duboke neuronske mreže. Njenim

korištenjem ostvaruje se pristup gotovim konvolucijskim slojevima kao i keras API-ju što olakšava pisanje vlastitih implementacija slojeva i funkcija.

5.2. OpenCV

OpenCV (Open Source Computer Vision) je opsežan javni alat dizajniran za korištenje kod zadataka u području računalnog vida i obrade slika. Inicijalno alat je razvio Intel no sada je taj projekt vodi zajednica. Ovaj alat nudi mnoštvo funkcionalnosti neke od kojih su obrada slike i videa, prepoznavanje i detekcija objekata, kalibracija kamere i geometrijske transformacije. Alat je kompatibilan s raznim programskim jezicima, od kojih se Python ističe kao najpopularniji, što omogućava da se koristi u raznim aplikacijama.

Konkretno gledano za zadani problem ovaj alat pokazuje se kao poželjan iz više razloga. Kao prvo sadrži mnoštvo implementiranih algoritama kao Haar cascades i HOG te raznovrsne algoritme iz područja dubokog učenja što uvelike olakšava implementaciju sustava za prepoznavanje objekata. Osim toga velika količina funkcionalnosti koju alat pruža daje određenu dozu fleksibilnosti s obzirom na specifičan zadatak.

5.3. OIDv4-Toolkit

OIDv4-Toolkit je alat je koji je vezan za Open Images Dataaset(OID). Open Images dataset je skup podataka koji sadrži označene slike te je dizajniran za evaluaciju modela računalnog vida s fokusom na prepoznavanje objekata. Skup podataka sadrži nekoliko milijuna slika označenih okvirima, te sadrži velik broj različitih klasa što ga čini bogatim resursom za zadatke povezane uz prepoznavanje objekata. Cilj OID skupa podataka je podržavanje napredaka u području računalnog vida tako što nudi širok, dobro označen skup podataka koji obuhvaća širok spektar situacija iz stvarnog svijeta.

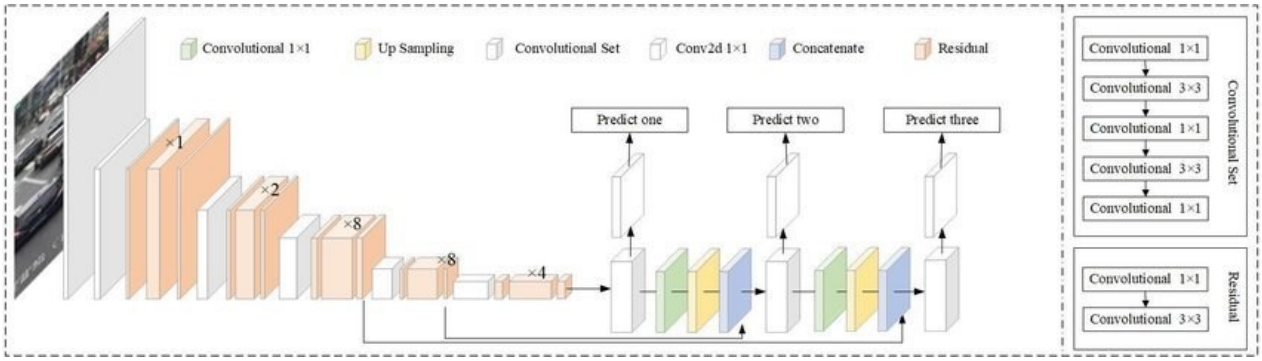
Sami OIDv4-toolkit koristi se kako bi se mogli koristiti i upravljati podacima OID skupa podataka. Unutar toolkita nalaze se mnogi alati i skripte koje omogućavaju dohvaćanje, obradu i rad na podacima iz skupa što ga čini pristupačnim za osobe koje rade na projektima vezanim uz detekciju objekata.

Ovaj alat konkretno će se koristiti kako bi dohvatili naš skup podataka sa željenim klasama, označili taj skup i pripremili ga za obradu tijekom provedbe procesa treniranja.

5.4. YOLOv3

Yolo odnosno you only look once algoritam je za prepoznavanje objekata u stvarnom vremenu koji je predstavljen je 2015 godine te je kombinirao kreaciju okvira i klasifikacijsku regresiju u jedan korak. Funkcionirao je na način da se mapa značajki podjeli na 7x7 ćelija te se za svaku ćeliju odvija predikcija. Na ovaj način značajno se smanjila kompleksnost računanja. 2018 izlazi verzija YOLOv3 koja je kompetitivna s dosadašnjim algoritmima za prepoznavanje objekata.

YOLOv3 funkcionira na idući način. Ulazni podatak dijeli se na SxS mrežu ćelija. Zatim podatak šaljemo u dva različita smjera. Prvi smjer gradi okvire oko potencijalnih objekata te računa sigurnost u odluku za svaki dok drugi smjer računa vjerojatnost pripadnosti nekoj od klasa za svaku od ćelija. Nakon toga, uzevši u obzir oba smjera, gradi se odluka te se odlučuje koji će se okviri ostaviti te kojim klasama oni pripadaju. Većinu vremena dogodit će se da se objekt nalazi unutar više okvira. Taj problem rješava se korištenjem IOU odnosno intersection over union principa. Sami IOU je granica koja se postavlja na vrijednost iz intervala 0 do 1. Zatim se računa koeficijent za svaki okvir po formuli površina presjeka/površina unije te se odbacuju svi okviri s manjim koeficijentom od zadanog. Slika 14 daje jasni prikaz arhitekture YOLOv3 mreže.



Slika 14 Ilustracija arhitekture YOLOv3 algoritma

Sa Slike 14 bitno je uočiti sljedeće. U srži YOLOv3 mreže nalazi se Darknet-53 arhitektura duboke neuronske mreže te ona služi kako bi se izvršilo izvlačenje značajki. Ovdje u igru dolazi preneseno učenje. Darknet-53 sastoji se od pedeset i tri sloja među kojima su konvolucijski i rezidualni slojevi. Ovi slojevi su obično pred-trenirani na nekom širokom

skupu podataka kao što je ImageNet kako bi se naučile neke generičke značajke koje se onda mogu iskoristiti u našoj YOLOv3 mreži.

Druga bitna stvar za uočiti jest da se predikcija izvršava tri puta. Svaka predikcija izvršava se na većoj SxS mreži. Time omogućavamo modelu da pronalazi manje objekte na slikama pošto veličina određenog objekata na slici nije uvijek jednaka. Prva predikcija izvršava se konvolucijom koja će vratiti izlaz dimenzija NxNx(Bx(5+C)). NxN predstavlja dimenzije mape značajki dobivene iz Darknet djela mreže. B označava broj okvira koje je moguće pronaći u svakoj od ćelija, odnosno YOLOv3 pokušava predvidjeti mogući broj okvira za svaku ćeliju. Brojka 5 označava broj atributa koji opisuju svaki okvir, ti atributi su x i y koordinata centra okvira, dužina i visina okvira te vjerojatnost da je promatrana ćelija centar okvira. C parametar na m označava broj prepoznatih klasa.

Ostale dvije predikcije odvijaju se na isti način no mreža ćelija je gušća. Zadnja stvar koju je bitno naglasit jest korištenje ne-maksimalne supresije odnosno kod finalne predikcije maknuti će se svi okviri koji nisu maksimalni.

5.5. Flask

Za kreiranje web aplikacije preko koje će se učitavati slike bilo je potrebno odabrati odgovarajući okvir. Tu se kao savršeni kandidat javlja Flask okvir. Flask je jednostavan i fleksibilan mikro web okvir za Python. Dizajniran je kako bi podržao razvoj web aplikacija i API-ja, pritom srezujući troškove na minimum. Flask se predvodi modularnim dizajnom te tako omogućuje programerima da biraju i integriraju samo one komponente koje su im neophodno potrebne. Sagrađen je korištenjem WSGI alata te Werkzug i Jinja2 pogonima za izradu predložaka.

Flask pruža osnovne alate za usmjeravanje, rukovanje zahtjevima i renderiranje predložaka pritom zadržavajući načela jednostavnosti i lakoće korištenja. Flask je pogodan za korištenje na malim projektima jednako kao i na velikim skalabilnim aplikacijama. To je zato što postoji širok raspon dodataka kao što su Flask-SQLAlchemy za rukovanje bazama ili Flask-WTF za upravljanje formama.

Značajke koje ga čine pogodnim za realizaciju zadatka su sljedeće: minimalistička jezgra koja omogućuje jednostavno prilagođavanje i integraciju željenih komponenti, izravan

dizajn i lako shvatljivi API, širok broj ekstenzija koje mogu pokriti sve što je potrebno za realizaciju zadatka, modularan je u slučaju da je aplikaciju potrebno proširiti te postoji velika i aktivna zajednica koja ga konstantno razvija. Uza sve te dobre značajke Flask se možda čini kao savršeno rješenje za bilo koji tip aplikacije, no i kod Flaska postoje nedostaci.

Neki od nedostataka su mali broj ugrađenih opcija za razliku od okvira kao Django, Flask zahtjeva ručno podešavanje učestalih operacija kao npr. Autentifikacija, zatim iako je moguće koristiti Flask za kreiranje velikih aplikacija nekada to može zahtijevati više posla i temeljito znanje konfiguracije web servera kao i poznavanje njegovog postavljanja.Zadnji od glavnih problema Flaska jest taj da pošto je okvir jako fleksibilan i većina mogućnosti se mora vlastoručno realizirati može doći do nekonzistentnosti koda prilikom rada većeg broja programera na istom kodu. No nedostaci Flaska za realizaciju zadanog zadataka su nepostojeći pošto je aplikacija mala i jednostavna te služi isključivo za učitavanje slike u model.

6. Struktura programa

Kada se govori o strukturi koda prvo je bitno spomenuti način na koji se gradi YOLO mreža. Prvo je bitno napraviti kada gradimo mrežu jest raščlaniti konfiguracijsku datoteku. To radimo tako da u konfiguracijskoj datoteci prolazimo liniju po liniju, kada linija započinje znakom [radit će se o početku konfiguracije određenog sloja te će se nakon toga čitati parametri za dani sloj i spremati u mapu tog sloja. Nakon što se to odradi definira se model ulaza te se ti ulazi normaliziraju. Kada su ulazi normalizirani iterira se po slojevima i inicijaliziramo ih njihovim parametrima.

Postoji pet vrsta slojeva koji se mogu pojaviti. Prvo tu je konvolucijski sloj. Prilikom nailaska na konvolucijski sloj provjerava se trebali dodati proširenje zatim se izvršava konvolucija te se provodi normalizaciju serije ako je bila postavljena za zadani sloj tijekom konfiguracije. Idući sloj koji za koji se pojavljuje jest sloj povećanja uzorkovanja to se izvršava preko Keras metode UpSampling2D. Ta metoda izvršava proširenje ulaza za dani korak koji se predaje kao parametar preko konfiguracije. Ovaj sloj je bitan jer se povećanjem uzorkovanja pospješuje detekcija manjih objekata kao i detekcija objekata na različitim skalama.

Nadalje jedan od mogućih slojeva je sloj usmjeravanja. Sloj usmjeravanja bitan je kod YOLO algoritma jer funkcionira na način koji omogućava modelu da ponovno koristi značajke prošlih slojeva čime se dobiva bolja točnost na manjim objektima kao i veća fleksibilnost modela radi čega se povećava prilagodljivost modela na razne skupove podataka. To konkretno funkcionira na način da se kroz parametar zadaje sloj s kojim želimo spojiti izlaz trenutnog sloja.

To zadajemo na dva moguća načina. Prvi način je singularnim negativnim brojem koji označava koliko slojeva unazad je sloj čiju mapu značajki želimo postaviti na izlaz ovog sloja. Ako je kao parametar primljeno dva broja prvi označava koliko koraka unazad je prvi sloj čija će se mapa značajki spojiti s mapom značajki koja se nalazi na izlazu sloja koji se nalazi na mjestu koje označava drugi broj. Spajanjem te dvije mape značajki dobit će se izlaz ovog sloja.

Četvrti sloj po redu koji se može pojaviti jest sloj prečaca. Sloj prečaca funkcionira na način da kao parametar postavljamo broj koraka kojih moramo otići unazad da dođemo do onog sloja s čijim izlazom će se spojiti izlaz sloja ispred ovog. Kombinaciju ta dva izlaza

predstavlja izlaz ovog sloja koji se onda šalje dalje u mrežu. Ovaj sloj je dosta bitan radi toga što stvara unatražne konekcije slične onima u ResNet arhitekturi te time omogućava glađi protok gradijenta tijekom unatražnog prolaza čime se umanjuje problem nestajućeg gradijenta. Posljedično tome treniranje postaje stabilnije te se dublje mreže mogu trenirati efikasnije. Zadnji sloj koji će se koristiti jest yolo sloj. Unutar ovog sloja prvo se deklariraju maske i sidra. Maske služe kako bi odabrali koja sidra želimo koristiti u ovom sloju dok sidra označavaju unaprijed definirane okvire raznih veličina koje koristimo kao referentni predlošci za kreiranje pravih okvira.

Nakon toga izvlače se centri i dimenzije okvira te pouzdanost rezultata i vjerojatnosti pojedinih klasa. Na izvučene vrijednosti se zatim primjenjuje sigmoida kako bi se normalizirale. Tada slažemo sidra da odgovaraju prostornim dimenzijama izlaza te skaliramo dimenzije okvira eksponencijalnom aktivacijskom funkcijom. Još preostaje generirat i postaviti mrežu koordinata kako bi se centri okvira mogli prilagoditi na mapu značajki. Kada se to odradi radi se finalna predikcija koja sadrži sve komponente koje su potrebne kako bih se iscrtali okviri na slici.

6.1. Postavljanje težina

Kada govorimo o postavljanu težina moramo prvo znati kako su originalne težine zapisane. Težine se u start nalaze zapisane u binarnoj datoteci spremljene kao float podaci. Bitno je znati da su težine spremljene na jedan od dva načina. Način na koje su spremljene ovisi o tome na koji se sloj odnose. Naime težine se odnose samo na konvolucijske slojeve te s obzirom na to je bitno, je li težina koju unosimo vezana za konvolucijski sloj s normalizacijom serije ili bez. Iteracijom se prolazi kroz slojeve dok ne naiđe konvolucijski. Kada je konvolucijski sloj identificiran provjeravamo dali postoji normalizacija serije te ovisno o tome ili čitamo sve potrebne parametre ili ako ne postoji čitamo pristranost te postavljamo težine sloja na pročitane vrijednosti. Nakon postavljanja svih vrijednosti spremamo ih u Tensorflow formatu kako bi ih koristili tijekom predikcije.

6.2. Provođenje predikcije

Prije nego što se objasni točno kako se provodi finalna predikcija bitno je spomenuti neke dodatne funkcije koje su potrebne kako bih sve funkcioniralo. Prva bitna funkcija jest funkcija za ne maksimalnu supresiju. Ova funkcija kao ulaz prima podatke dobivene na

izlazu yolo mreže. Ti podaci se zatim razdijele te se okviri normaliziraju i izračunava se vrijednost svakog okvira. Nakon toga izvršava se ne maksimalna supresija pozivom funkcije iz Tensorflow biblioteke. Na ovaj način filtrirani su svi oni okviri čija vrijednost ne zadovoljava postavljenu granicu.

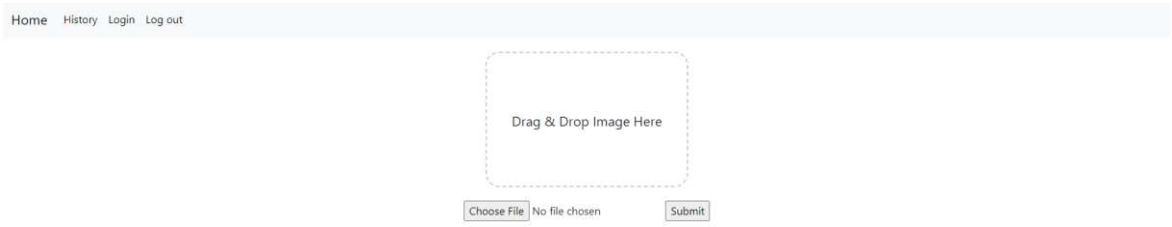
Iduća bitna funkcija jest funkcija koja pretvara koordinate dobivene iz yolo algoritma u format gdje parovi predstavljaju koordinate gornjeg lijevog kuta i donjeg desnog kuta okvira. Pošto yolo algoritam vraća koordinate centra i veličinu okvira lako je izračunati te dvije točke. Nakon računanja točki tako spremljeni okviri šalju se u ne maksimalnu supresiju. Zadnja bitna pomoćna funkcija jest funkcija za crtanje okvira na slici. Ta funkcija prolazi kroz dobivene okvire i iscrtava ih na dobivenoj slici korištenjem cv2 biblioteke. Nakon što nacrta okvir nadodaje mu zaglavlje s imenom predviđene klase kao i vjerojatnosti pripadnosti toj klasi.

Sad kada su objašnjene bitne funkcije može se graditi glavna petlja programa. U glavnoj petlji za početak pripremimo sve granice i maksimalne veličine izlaza kao i veličinu modela te postavljamo putanje do potrebnih datoteka. Zatim kreiramo YOLO model i učitavamo težine. Nakon što se slika učita prilagođava se veličini modela te se predaje modelu kako bi izvršio predikciju. Kada se odradi predikcija ona se prosljeđuje funkciji koja odrađuje ne maksimalnu supresiju te miče sve nepotrebne okvire. Nakon toga okviri i slika se predaju funkciji koja iscrtava okvire. Kada su okviri nacrtani za sliku se generira nasumično ime te se ona pohranjuje.

6.3. Grafičko sučelje

Kao što je navedeno, kod odabira tehnologija za implementaciju web aplikacije korišten je Flask. Struktura aplikacije te prikaz njenog korištenja detaljnije su objašnjeni u ostatku ovog poglavlja.

Kreirano je više html predloška koji služe kako bi se generirali različiti prikazi za korisnika. Unutar glavne aplikacije smještene su rute koje definiraju funkcionalnosti web aplikacije. Ruta početne stranice generira funkcionalnost i html početne stranice koja će biti prikazana korisniku. Izgled početne stranice prikazan je na Slici 16.



Slika 15 Početna stranica

Kao što je vidljivo na Slici 16 ideja je vrlo jednostavna. Korisniku se omogućava odabir slike na jedan od dva načina. Korisnik može sliku učitati na dva načina, odabirom iz izbornika pritiskom na gumb „Choose File“ ili može jednostavno povući sliku mišem u okvir, „Drag & Drop“ funkcionalnost. Kada korisnik učita sliku ispisuje mu se ime slike radi dodatne provjere je li željena slika učitana.

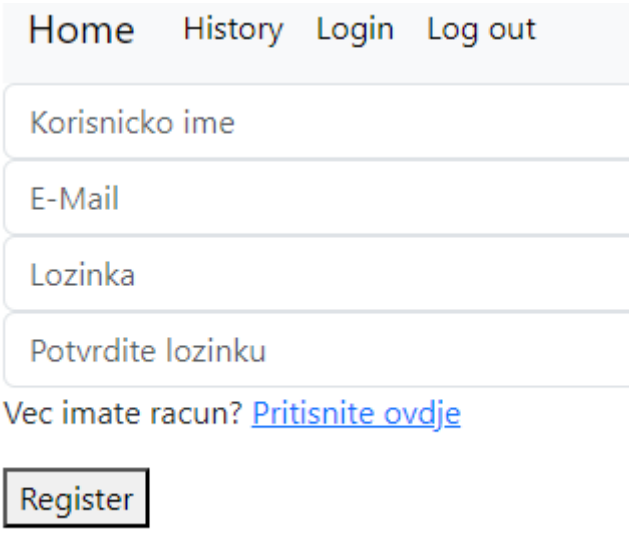
Slika 16 Prikaz sučelja za odabir fotografije

Kada željena slika odabrana, pritiskom na gumb „Submit“, šalje se POST zahtjev prema serveru koji sliku prosljeđuje YOLO mreži gdje se ona obrađuje na prethodno objašnjen način. Nakon što je mreža obradila sliku i generirala rezultat, obrađena slika s rezultatom, prosljeđuje se na server koji prikazuje tu sliku u obliku html predloška. Prikaz generirane slike pokazan je na slici 18.

Slika 17 Prikaz rezultata

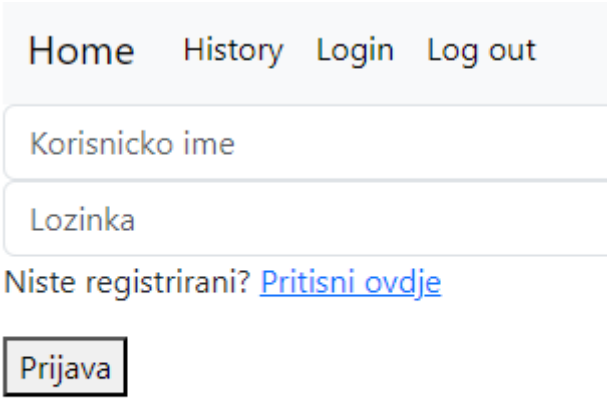
Kao što se vidi iz prikaza na Slici 18 slika je obrađena te se na njoj jasno vide okviri i imena klasa kao i njihove vjerojatnosti. Ispod slike nalazi se opcija povratka na početnu stranicu gdje je moguće učitati novu sliku.

Također, korisniku aplikacije omogućuje se kreacija korisničkog računa koji korisniku pruža uvid u prethodno obrađene slike. Sam izgled registracije prikazana je na Slici 19.



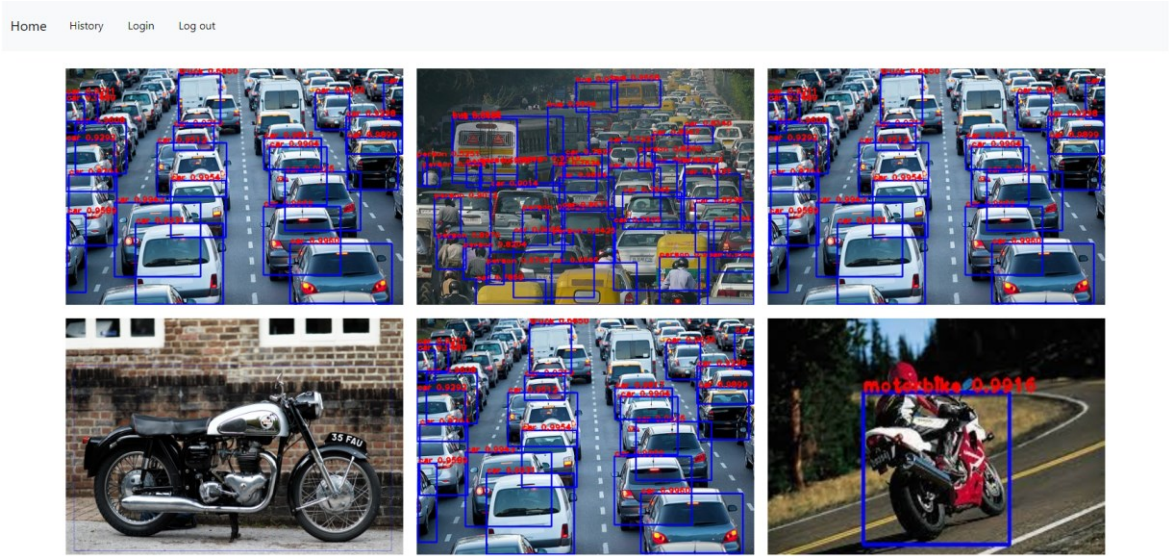
Slika 18 Registracija

Prilikom registracije, korisnikovi podaci pohranjuju se u jednostavnu bazu podataka koja se nalazi lokalno na serveru. Nakon registracije, korisniku se prijava u aplikaciju omogućuje preko prikaza na Slici 20. Prijavom u sustav, na server se šalje POST zahtjev s podacima za prijavu koji se zatim uspoređuju sa zapisima u bazi podataka kako bi se provjerilo odgovaraju li podaci za prijavu.



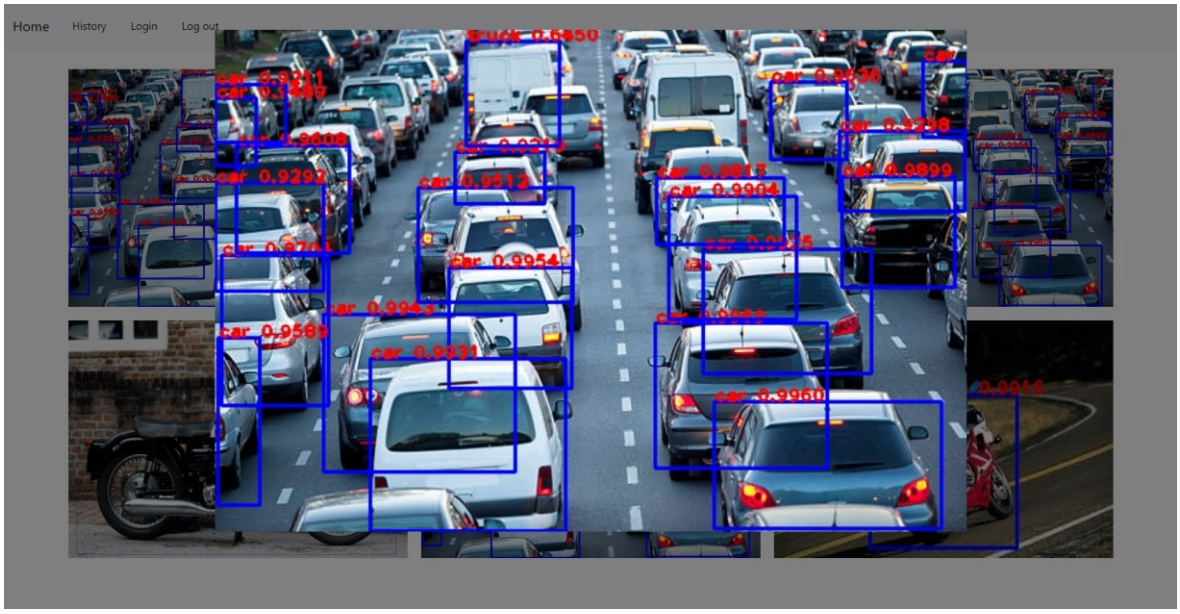
Slika 19 Prijava

Prijavljeni korisnik ima mogućnost prikaza povijesti korištenja aplikacije. Korisniku, koji je prijavljen, pritiskom na opciju „History“ iz navigacijske trake, prikazuju se sve prethodno generirane slike. Izgled tog prikaza vidljiv je sa Slike 21.



Slika 20 Prikaz povijesti

Odabirom neke od slika iz opcije „History“, korisniku se otvara modalni prikaz odabrane slike i omogućuje uvećavanje te slike.



Slika 21 Uvećani prikaz slike

Konačno, pritiskom na opciju „Log out“, korisnik se odjavljuje i onemogućuje mu se pregled povijesti korištenja aplikacije.

Zaključak

U današnje doba, svijet postaje sve užurbaniji, pa tako i promet postaje sve gušći te se raznolikost motornih vozila u prometu znatno povećava. S povećanjem količine prometa te naprednom tehnologija jasno je da se otvara prostor za primjenu računalnog vida kao i tehnika umjetne inteligencije na sve aspekte prometa. U sklopu ovog rada prikazana je upravo jedna takva primjena. Korištenjem YOLOv3 algoritma i tehnikama dubokog učenja kreirana je konvolucijska neuronska mreža iz koje se dobiva model koji je sposoban razlikovati različite tipove vozila, na slici ih prikazati ograđene okvirima s oznakama njihove klase i numeričku vrijednost koja predstavlja pouzdanost predikcije.

Kroz rad su prikazane osnove neuronskih mreža i dubokog učenja, te je detaljno razjašnjen princip rada konvolucijskih neuronskih mreža. Također, objašnjeno je što je to preneseno učenje i zašto je njegovo korištenje ekstremno pogodno za zadani zadatak. Odabirom prikladne tehnologije za realizaciju traženog modela, provedena je analiza kako bi se dobio odgovor zašto su baš te tehnologije dobar izbor za izvedbu prepoznavanja tipova vozila. Konačno, obrađena je realizacija odabranih algoritama u kodu programa, te je prikazan rad modela na određenom skupu podataka kroz razvijenu web aplikaciju.

Kada se sagleda kompletna realizacija zadataka, može se zaključiti da su rezultati zadovoljavajući te da model uspješno prepoznaje većinu vozila koja se nalaze na slici, no to ne znači da ne postoji prostora za napredak. Za početak skup podataka na kojem se odvijalo treniranje može biti veći i adekvatniji za pojedinu primjenu. Zatim ako razmišljamo o konkretnoj primjeni jednog ovakvog modela jasno je da se veća primjenjivost ostvaruje primjenom modela na video umjesto na sliku. Za budući rad i unaprjeđenje modela za kompleksnije zadatke preporučuje se korištenje novijih verzija YOLO algoritma, ali je potrebno naglasiti da za dani zadata, korištena verzija daje jako dobre rezultate.

Literatura

[1] https://www.ibm.com/topics/deep-learning

[2] https://www.fer.unizg.hr/predmet/dubuce1_a

[3] https://en.wikipedia.org/wiki/Convolutional_neural_network

[4] https://www.analyticsvidhya.com/blog/2021/05/convolutional-neural-networks-cnn/

[5] https://www.geeksforgeeks.org/convolutional-neural-network-cnn-in-machine-le

[6] https://www.datacamp.com/tutorial/introduction-to-convolutional-neural-networks-cnns

[7] https://pjreddie.com/darknet/yolo/

[8] https://viso.ai/deep-learning/yolov3-overview/

[9] https://www.ibm.com/topics/transfer-learning

[10] https://machinelearningspace.com/yolov3-tensorflow-2-part-1/

[11] https://github.com/pythonlessons/TensorFlow-2.x-YOLOv3

[12] https://pylessons.com/YOLOv3-TF2-introduction

Sažetak

Klasifikacija tipova vozila u prometu korištenjem dubokih neuronskih mreža

U današnje vrijeme, napretkom u području prepoznavanja objekata i računalnom vidu, jasno se vidi mogućnost primjene tih tehnologija u prometu i transportu. Te tehnologije mogu se primjenjivati na razne aspekte prometa kao što su regulacija prometa, sigurnost u prometu te razvoj autonomnih vozila. U sklopu ovog rada treba ostvariti model koji je će na slici prepoznavati vozila te ih klasificirati u kategorije kao što su automobil, motocikl, autobus, kamion ili kombi. Za realizaciju tog zadatka koristiti biblioteku Tensorflow 2 programskog jezika Python s ciljem izgradnje dubokog modela koji je baziran na konvolucijskim neuronskim mrežama te treniran korištenjem metoda prenesenog učenja.

Ključne riječi: Duboke neuronske mreže, konvolucija, Tensorflow, Prepoznavanje objekata, Python

Summary

Classification of vehicle types in traffic using deep neural networks

Nowadays, with progress in the field of object recognition and computer vision, the possibility of applying these technologies in traffic and transport is clearly visible. These technologies can be applied to various aspects of traffic such as traffic regulation, traffic safety and the development of autonomous vehicles. As part of this work, a model should be created that will recognize vehicles in the image and classify them into categories such as car, motorcycle, bus, truck or van. To realize this task, we use the Tensorflow 2 library of the Python programming language with the aim of building a deep model based on convolutional neural networks and trained using transfer learning methods.

Keywords: Deep neural networks, convolution, Tensorflow, Object recognition, Python