Programska aplikacija za prepoznavanje ljudskih aktivnosti

Major, Krešimir

SVEUČILIŠTE U ZAGREBU

FAKULTET STROJARSTVA I BRODOGRADNJE

Završni rad

Krešimir Major

Zagreb, 2020./21.

SVEUČILIŠTE U ZAGREBU

FAKULTET STROJARSTVA I BRODOGRADNJE

Završni rad:

Programska aplikacija za prepoznavanje ljudskih aktivnosti

(Software application for human action recognition)

Profesor: Doc. dr. sc. Tomislav Stipančić

Student: Krešimir Major

0035211042

Zagreb, 2020./21.

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE
Središnje povjerenstvo za završne i diplomske ispite
Povjerenstvo za završne ispite studija strojarstva za smjerove:
proizvodno inženjerstvo, računalno inženjerstvo, industrijsko inženjerstvo i menadžment, inženjerstvo materijala i mehatronika i robotika

Sveučilište u Zagrebu Fakultet strojarstva i brodogradnje	
Datum	Prilog
Klasa: **602 - 04 / 21 - 6 / 1**	
Ur.broj: **15 - 1703 - 21 -**	

ZAVRŠNI ZADATAK

Student: **Krešimir Major** Mat. br.: 0035211042

Naslov rada na hrvatskom jeziku: **Programska aplikacija za prepoznavanje ljudskih aktivnosti**

Naslov rada na engleskom jeziku: **Software application for human action recognition**

Opis zadatka:

Modeli temeljeni na neuronskim mrežama koje su trenirane na određenom skupu podataka mogu naučiti više ili manje uspješno prepoznavati ljudske aktivnosti. Takvi se modeli potom koriste u različite svrhe, npr. kod zaključivanja robota u interakciji s čovjekom, kod predviđanja ljudskih potreba u sklopu semantičkog weba i sl. Razvoj algoritama umjetne inteligencije temeljenih na metodama dubokog učenja kod prepoznavanja ljudskih aktivnosti u današnje vrijeme sve je više u fokusu istraživača diljem svijeta.
U radu je potrebno:

- proučiti metodologiju prepoznavanja ljudskih aktivnosti koja se temelji na metodama dubokog učenja,
- odabrati odgovarajući skup podataka za učenje i testiranje kako bi se odredile domene djelovanja modela,
- odrediti programsku podršku te unaprijed uvježbanu (eng. pre-trained) arhitekturu neuronske mreže za prepoznavanja ljudskih aktivnosti,
- odrediti optimalne parametre rada modela.

Razvijeni model za prepoznavanje ljudskih aktivnosti potrebno je eksperimentalno verificirati na unaprijed snimljenim video materijalima.
U radu je potrebno navesti korištenu literaturu te eventualno dobivenu pomoć.

Zadatak zadan:
30. studenoga 2020.

Zadatak zadao:
Doc. dr. sc. Tomislav Stipančić

Datum predaje rada:
1. rok: 18 veljače 2021.
2. rok (izvanredni): 5. srpnja 2021.
3. rok: 23. rujna 2021.

Predviđeni datumi obrane:
1. rok: 22.2. – 26.2.2021.
2. rok (izvanredni): 9.7.2021.
3. rok: 27.9. – 1.10.2021.

Predsjednik Povjerenstva:
Prof. dr. sc. Branko Bauer

Izjavljujem da sam ovaj rad izradio samostalno koristeći znanja stečena tijekom studija i navedenu literaturu.

Zahvaljujem se Doc. dr. sc. Tomislavu Stipančiću na ukazanom povjerenju pri zadavanju ovog zadatka, pomoći i savjetima tijekom izrade rada.

Također zahvaljujem se svojoj obitelji, uz čiju podršku i razumijevanje sam došao do kraja preddiplomskog studija.

Krešimir Major

Sadržaj

Popis slika i tablica

1. Uvod

Veliki i ubrzani razvoj znanosti na području dubokog učenja, algoritama umjetne inteligencije, u zadnjih 10 do 15 godina omogućio je istraživačima da pomoću računala postižu simulirati sve kompleksnije radove ljudskog mozga. Temelj rada ljudskog mozga su neuroni pa su se tako razvile mnoge vrste umjetnih neuronskih mreža koje pokušavaju simulirati rad mozga. Jedna od zanimljivijih i jako kompleksnih radnji je i ova koja će biti detaljno opisana u ovome radu, prepoznavanje ljudskih aktivnosti u videima. Funkcija prepoznavanja događaja na videima je jako kompleksna pa se zbog toga model programa za prepoznavanje ljudskih aktivnosti temelji na neuronskim mrežama. Takve neuronske mreže su trenirane i učene na velikom skupu podataka nakon kojih mogu relativno uspješno prepoznavati aktivnosti ljudi. Ovakvi modeli programa imaju široku primjenu i koriste se u različite svrhe, kao npr. kod interakcije robota s čovjekom, kod predviđanja ljudskih potreba.

Za početak ovog rada, u prvim poglavljima, teorijski će te biti upoznati s pojmovima dubokog učenja i neuronskih mreža. Na detaljan način se objašnjava struktura i rad pojedinog neurona u umjetnim neuronskim mrežama, a nakon toga se bavimo strukturom i radom cijelih neuronskih mreža.

Kasnije se upoznaje i objašnjava rad konvolucijskih neuronskih mreža, koje su nam bitne za ovaj rad jer se one koriste za raspoznavanje značajki na slikama. Zatim nakon toga se prikazuje neuronska mreža koja se koristi u ovoj programskoj aplikaciji za prepoznavanje ljudskih aktivnosti u videima. Objašnjava se njena struktura i njen princip rada.

Na kraju rada u poglavljima 7 i 8 detaljno se prolazi kroz program koji koristi našu neuronsku mrežu i prikazuju se rezultati rada naše neuronske mreže pri prepoznavanju ljudskih aktivnosti u već snimljenim videima.

2. Duboko učenje

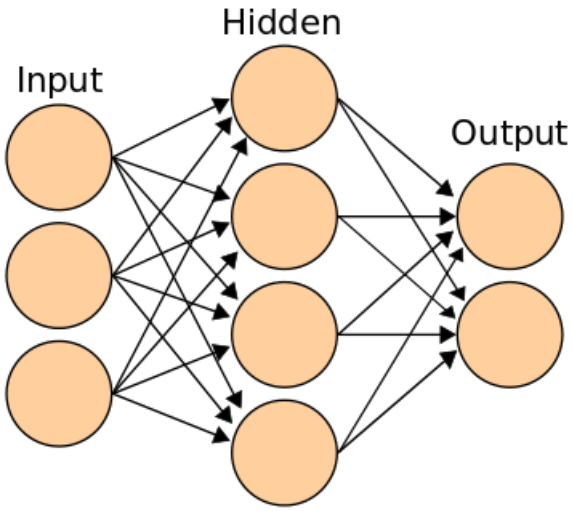
Duboko učenje (eng. Deep learning) je funkcija umjetne inteligencije koja imitira rad ljudskog mozga. Oponaša način procesiranja podataka i kreiranja uzoraka za korištenje u donošenju odluke.

Duboko učenje se ubrzano počelo razvijati zadnjih 10 godina. Dio je većeg područja nazvanog strojno učenje, a strojno učenje je podskup umjetne inteligencije. Umjetna inteligencija je grana računalne znanosti koja se bavi simulacijom inteligentnog ponašanja u računalima. Pojam je 1956. prvi počeo koristiti John McCarthy s američkog MIT-a (Massachusetts Institute of Technology). Umjetna inteligencija je pokušaj oponašanja načina na koji čovjek razmišlja. Pomoću inteligentnih softvera se nastoji automatizirati rad, raspoznati govor ili slike, dijagnosticirati bolesti u medicini te doprinijeti znanstvenim istraživanjima.

Slika 1 – Duboko učenje

Duboko učenje je podvrsta strojnog učenja, u kojemu se u svrhe rješavanja problema koriste umjetne neuronske mreže. Umjetne neuronske mreže se mogu definirati kao niz algoritama namijenjenih za prepoznavanje nekog uzorka. Njihova građa zasniva se na građi ljudskog mozga. Koriste se umjetni neuroni koji su dizajnirani i pokušavaju imitirati rad bioloških neurona.

Umjetne neuronske mreže su izgrađene kao ljudski mozak, s neuronskim čvorovima povezanima međusobno kao mreža, spadaju u najranije i najuspješnije algoritme strojnog učenja. Mreže se sastoje od više slojeva koji služe za progresivno izvlačenje značajki više razine iz neobrađenog ulaza, sposobne za učenje iz nestrukturiranih i neoznačenih podataka.



Slika 2 – Umjetna neuronska mreža

Riječ duboko u dubokom učenju odnosi se na broj slojeva kroz koje se podaci transformiraju. Preciznije, sustavi dubokog učenja imaju veliki lanac transformacije podataka od ulaza do izlaza.

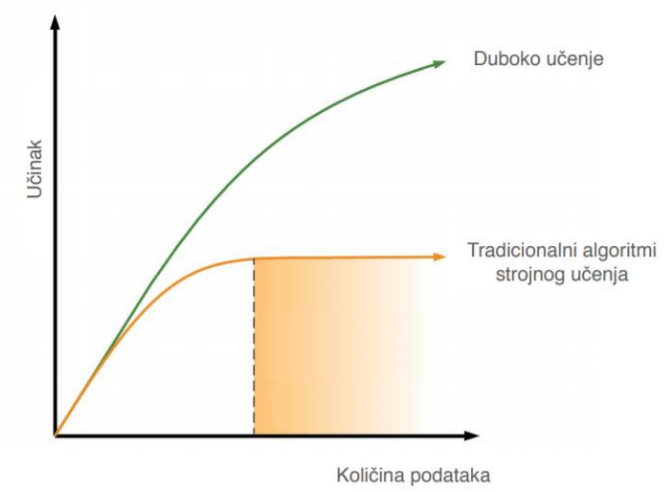
Duboko učenje je koncept koji omogućuje računalu izgradnju kompliciranih modela na temelju jednostavnijih koncepata, kao što su: rubovi, krugovi i razni jednostavni oblici. To je pristup strojnom učenju koji se temelji na poznavanju ljudskog mozga, statistike i primijenjene matematike. U dubokom učenju svaka razina, svaki sloj umjetne neuronske mreže uči se transformaciji svojih ulaznih podataka. Na primjer, u aplikaciji prepoznavanja lica na slici, neobrađeni ulaz može biti matrica piksela; prvi sloj mreže mogao bi transformirati piksele i prepoznati rubove na slici (rub lica, usta, oči itd. ili rub slike), drugi sloj bi mogao transformirati i prepoznati raspored rubova, treći sloj bi mogao prepoznati nos i oči i četvrti sloj bi mogao prepoznati da se na slici nalazi ljudsko lice. Proces dubokog učenja može sam naučiti koje značajke gdje optimalno postaviti, u koju razinu umjetne neuronske mreže.

Dok tradicionalni programi rade analizu podataka linearno, duboko učenje koristi hijerarhijsku razinu umjetne neuronske mreže za provođenje procesa učenja što omogučuje nelinearno procesiranje podataka.

Duboko učenje evoluiralo je uz bok s digitalnom erom što je dovelo do eksplozije podataka u svakom obliku iz svakog dijela svijeta. Velika količina podataka (tzv. Big data) izvlači se većinom iz izvora poput socijalnih mreža, internetskih pretraživača, komercijalnih platformi, itd. Posljednjih godina se dogodio veliki rast u popularnosti ovog pristupa i njegovoj korisnosti, temeljen uglavnom na sve snažnijim računalima, velikim količinama podataka i raznim tehnikama za učenje dubokih neuronskih mreža. Primjena dubokog učenja je profitabilna te se njime koriste najveće svjetske tehnološke kompanije: Google, Microsoft, Facebook, Apple, itd.

Duboko učenje procesira veliku količinu podataka pomoću modela baziranih na procesima u ljudskom mozgu, tražeći sličnosti i korelacije između njih. Umjetna inteligencija dubokog učenja prođe kroz velike količine podataka, raspliće ih i razumije ih u puno kraćem vremenskom periodu u odnosu na čovjeka (par sekundi u odnosu na par godina).

Duboko učenje može učiti i zaključivati iz podataka koji su neoznačeni i neorganizirani. Koristi se za detektiranje objekata, prepoznavanje govora, prevođenje jezika, prepoznavanje aktivnosti, donošenje odluka, itd. Što više podataka tim više znanja i bolja pouzdanost sustava. Strojno učenje stječe znanje izvlačenjem uzoraka iz podataka. Pripada u skup umjetne inteligencije i njime se omogućava učenje sustava temeljeno na podacima, umjesto na programiranju.



Slika 3 – Usporedba Dubokog i strojnog učenja

3. Neuronske mreže

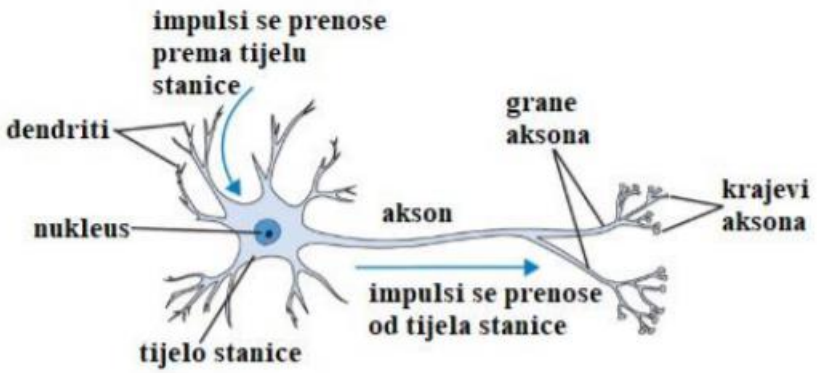
Umjetne neuronske mreže (eng. Artificial neural networks, ANN) simuliraju rad bioloških mreža u mozgu. One su skup umjetnih neurona koji su međusobno povezani i interaktivni kroz operacije obrade signala. Mreža može imati niz ulaza ili jedan ulaz i uvijek jedan izlaz, između kojih se nalazi jedan ili više tzv. skrivenih slojeva (tzv. višeslojne mreže). Pojedinačni neuroni su, kao i slojevi, međusobno spojeni vezama kroz koje idu signali . Veze među njima se aktiviraju ako je zadovoljen uvjet postavljen tzv. aktivacijskom funkcijom.

Biološki neuron

Neuron ili živčana stanica se smatra osnovnom jedinicom živčanog sustava i najsloženija je u ljudskom organizmu, glavni tip stanica koje tvore mozak. Živčani sustav se sastoji od oko 86 milijardi međusobno povezanih neurona. Njihova uloga se može konceptualizirati kao prihvaćanje, obrađivanje i odašiljanje podataka.

Osnovni dijelovi neurona (živčane stanice) su:

- dendriti
- tijelo stanice
- akson



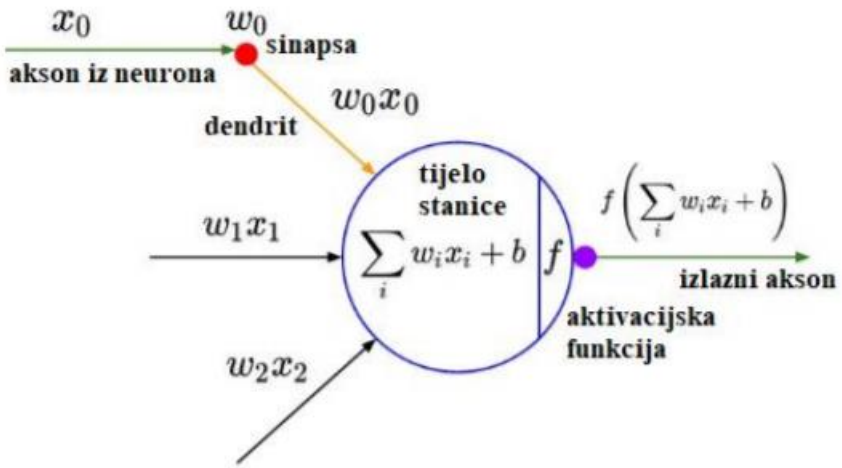
Slika 4 – Biološki neuron

Dendriti su kraći produžeci koji s osjetnih organa ili drugih živčanih stanica dovode živčano uzbuđenje na tijelo stanice. Aksonima je funkcija prenositi živčane impulse s tijela stanice na druge živčane stanice ili izvršne organe – mišićna vlakna ili žlijezde. Akson se grana i povezuje pomoću sinapsi s dendritima drugih neurona.

Svaki podražaj neurona dovodi do promjena na membrani tako što se otvaraju pore za propust iona. Ako je stimulus dovoljno jak da prijeđe prag podražljivosti onda se ionski kanali potpuno otvore i na taj način se propuštaju živčani impulsi.

Umjetni neuron

Kod umjetnih neurona možemo vidjeti njihovu sličnost s biološkim neuronima, odnosno, možemo vidjeti da su koncipirani tj. da im je rad sličan onome kao kod bioloških neurona.



Slika 5 – Umjetni neuron

Kod umjetnih neurona, signal koji putuje aksonom (x0) se množi s w0 na sinapsi. Jačina sinapse (težina w) je parametar koji se može naučiti te kontrolira utjecaj jednog neurona na drugi. Ti signali pomnoženi težinskim faktorima se potom sumiraju sukladno tome kako se u tijelu bioloških neurona sumiraju potencijali, pa se konačni iznos uspoređuje s pragom osjetljivosti (podražljivosti). Ako je vrijednost veća od tog praga umjetni neuron će dati izlazni signal (aktivirat će se izlazni aksion).

4. Umjetne neuronske mreže

Umjetna neuronska mreža (eng. Artificial neural network) je mreža koja napravljena i sadrži skup međusobno povezanih jedinica, neurona, koji su organizirani u slojeve. To je tehnika tj. način dubokog učenja koji oponaša rad ljudskog mozga s ciljem simuliranja postupka učenja kod čovjeka.

Neuronske mreže nam odlično služe za rješavanje problema klasifikacije i predviđanja, odnosno općenito svih problema kod kojih postoji odnos između ulaznih i izlaznih varijabli, bez obzira na visoku složenost te veze (nelinearnost). Danas se neuronske mreže primjenjuju u mnogim segmentima života poput medicine, bankarstva, strojarstva, geologije, fizike itd.

Najčešće se koriste za sljedeće zadatke:

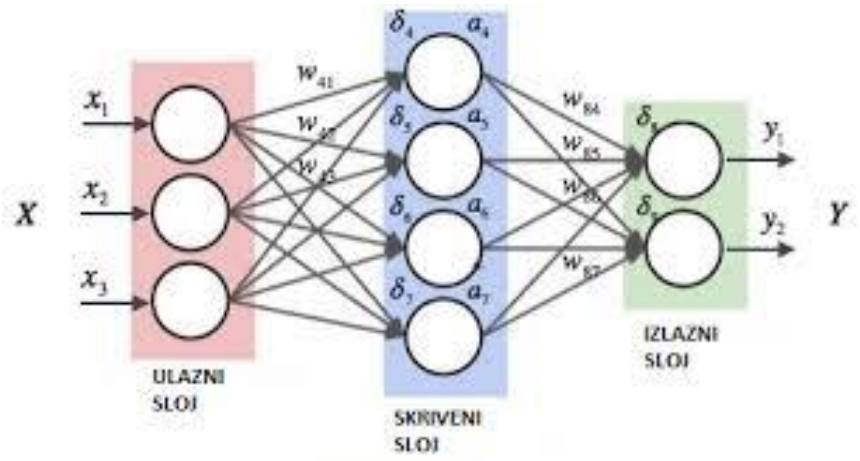
- raspoznavanje uzoraka,
- obrada slike,
- obrada govora,
- problemi optimizacije,
- nelinearno upravljanje,
- obrada nepreciznih i nekompletnih podataka,
- simulacije i sl.

Neke od značajki neuronskih mreža su:

- Vrlo su dobre u procjeni nelinernih odnosa uzoraka.
- Mogu raditi s nejasnim ili manjkavim podacima tipičnim za podatke iz različitih senzora, poput kamera i mikrofona, i u njima raspoznavati uzorke.
- Robusne su na pogreške u podacima
- Stvaraju vlastite odnose između podataka koji nisu zadani na eksplicitan način.
- Mogu raditi s velikim brojem varijabli ili parametara.
- Sposobne su formirati znanje učeći iz iskustva (tj. primjera).

Neke od poznatijih vrsta umjetnih neuronskih mreža su:

- Perceptron
- Konvolucijske neuronske mreže (eng. Convolutional Neural Networks, CNN)
- Povratne neuronske mreže (eng. Recurrent Neural Networks, RNN)
- Unaprijedne ili acikličke neuronske mreže (eng. Feed-forward Neural Networks)
- Ograničeni Boltzmannovi strojevi (eng. Restricted Boltzmann Machines)
- Autoenkoderi (eng. Auto-encoders)
- Radijalne mreže (eng. Radial Basis Function Neural Networks, RBF)



Slika 6 – Primjer neuronske mreže

Rad i struktura neuronske mreže

Neuroni

Neuroni su dijelovi neuronske mreže u kojima se izvršavaju matematičke operacije. U njih ulaze izlazni signali iz prijašnjeg sloja, ili ulazni podaci ako je riječ o početnom (ulaznom) sloju neuronske mreže, pomnoženi s određenim koeficijentom, koji se u neuronskim mrežama naziva težina (w). Težine mogu smanjiti ili povećati iznos svakog ulaza u neuron. Nakon što su u neuron ušli svi izlazi iz prijašnjeg sloja, odnosno ulazni podaci, pomnoženi svojim težinama, oni se sumiraju te se na tu sumu dodaje pomak (engl. bias):

$$z = w^t x + b$$

w^t – transponirani vektor težina

x – vektor ulaza x

b – pomak

Pomak omogućuje pomicanje aktivacijske funkcije lijevo ili desno, ovisno o potrebi. Dodaje se svakom neuronu. Nije povezan s prijašnjim slojevima.

Aktivacijske funkcije

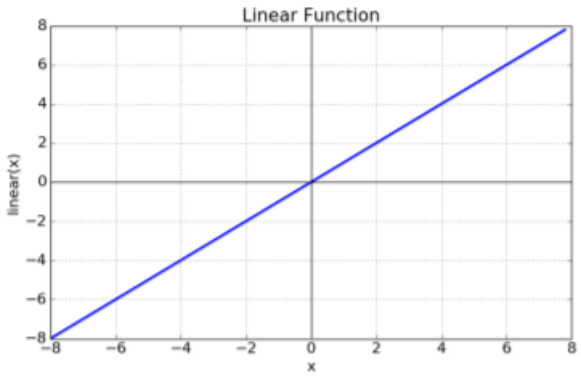
Nakon što se u neuronu obavila funkcija sumiranja ta suma z prolazi kroz aktivacijsku funkciju. Izlaz iz aktivacijske funkcije se označava:

$$y = f(z) = f(w^t x + b)$$

Hoće li se taj neuron i u kolikoj mjeri aktivirati i sudjelovati u idućim slojevima, ovisi o aktivacijskoj funkciji koja se koristi. U svakom se sloju nalazi određeni broj neurona. Postoji veliki izbor aktivacijskih funkcija no u praksi se koriste samo neke koje su se pokazale korisnima. Spomenut ćemo četiri različite aktivacijske funkcije .

– Linearna aktivacijska funkcija koja preslikava svoj ulaz pomnožen s nekom konstantom na izlaz. Ovakav tip aktivacijske funkcije se ne koristi u dubokim neuronskim mrežama zato što onemogućava učenje mreže.

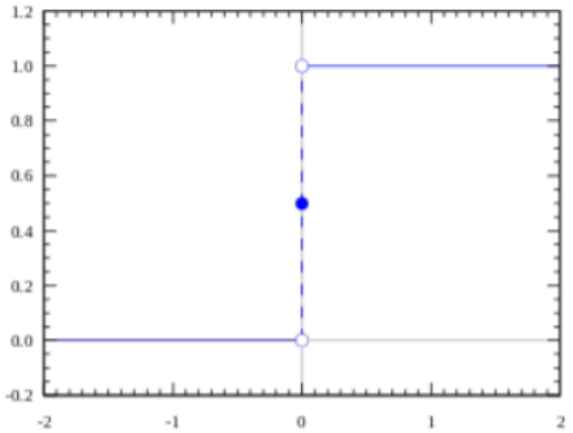
$$A(Y) = cY$$



Slika 7 – Linearna funkcija

– Step funkcija koja u neuronima funkcionira kao prekidač. Izlaz funkcije može poprimiti samo dvije različite vrijednosti ovisno o tome da li je ulaz manji ili veći od nekog praga.

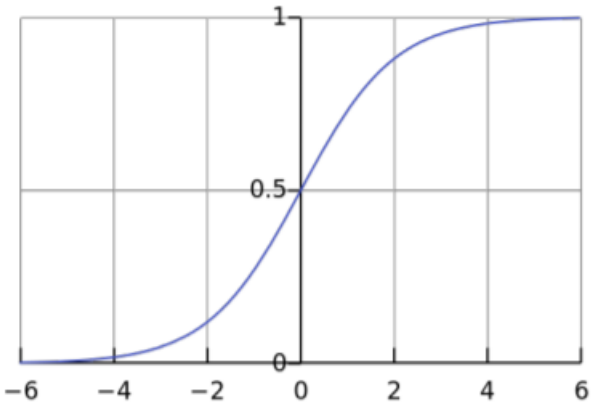
$$A(Y) = \begin{cases} 1, \text{ ako je Y} \geq 0 \\ 0, \text{ ako je Y} < 0 \end{cases}$$



Slika 8 – Step funkcija

– Sigmoidalne aktivacijske funkcije se najčešće koriste u praksi kod dubokih neuronskih mreža. Ovakve funkcije su derivabilne na cijeloj domeni i ograničene su što su dobra svojstva za algoritam unazadne propagacije i učenje mreže. Dvije najčešće korištene sigmoidalne funkcije su logistička funkcija i funkcija hiperbolnog tangensa.

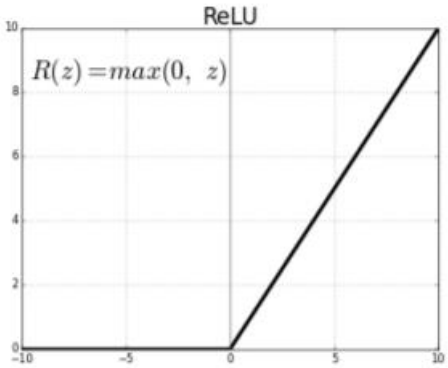
$$A(Y) = \frac{1}{1 + e^{-Y}}$$



Slika 9 – Sigmoidalna funkcija

– Zglobna aktivacijska (eng. ReLu - Rectified linear unit) funkcija se koristi kod gotovo svih konvolucijskih neuronskih mreža. Manje je računalno zahtjevna nego sigmoidalna ili tanh jer uključuje jednostavnije matematičke operacije. Zglobna aktivacijska funkcija pretvara sve negativne brojeve u 0, a pozitivne ostavlja onakvima kakvi su bili.

$$A(Y) = \begin{cases} 0, \text{ ako je Y} < 0 \\ \text{Y, ako je Y} \geq 0 \end{cases}$$

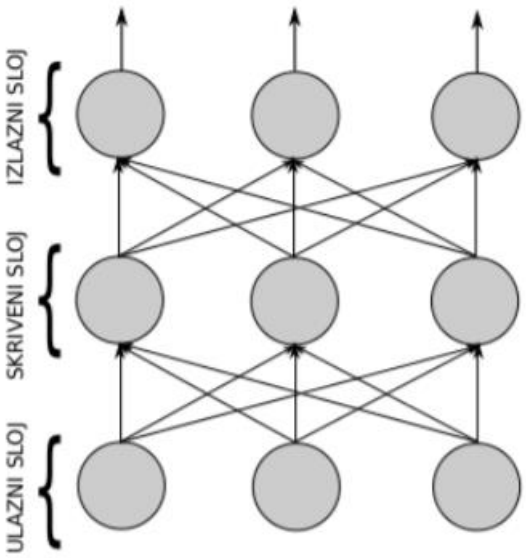


Slika 10 – ReLu funkcija

U umjetnoj neuronskoj mreži, aktivacijska funkcija je funkcija koja mapira ulaze čvorova u odgovarajuće izlaze, tako da se izlazom neurona uvijek smatra vrijednost funkcije z nakon aktivacije. Koriste se u skrivenim slojevima i u izlaznom sloju. Većina aktivacijskih funkcija su nelinearne i odabrane su na ovaj način kako bi se mogli razvrstati podaci koji se ne mogu odvojiti linearnim metodama. Nelinearne aktivacijske funkcije omogućuju neuronskim mrežama izračunavanje složenijih funkcija.

Struktura neuronske mreže

Neuronske mreže bazirane su na skupu velikog broja međusobno povezanih neurona. Neuroni su u neuronskim mrežama organizirani po slojevima. Postoje ulazni, skriveni i izlazni slojevi. Na ulaze neurona u ulaznom sloju se dovode podaci koje je potrebno klasificirati, koliko ulaznih podataka imamo, toliko ulaznih neurona imamo. Izlazi neurona ulaznog sloja su spojeni s ulazima neurona skrivenog sloja. Skrivenih slojeva može biti više pa su zato izlazi neurona skrivenih slojeva povezani s ulazima neurona idućih skrivenih slojeva ili s ulazima neurona izlaznog sloja, za svaki sloj se proizvoljno odabere broj neurona. Izlaz iz neurona izlaznog sloja se interpretira kao klasa koju je mreža klasificirala. Koliko postoji mogućih klasa, toliko treba biti izlaznih neurona.



Slika 11 – Potpuno povezana neuronska mreža

Dubokim neuronskim mrežama se nazivaju mreže koje imaju dva ili više skrivenih slojeva. Pokazalo se da su duboke neuronske mreže pogodnije za kompleksnije probleme klasifikacije i da ostvaruju dobre rezultate. Svaki sloj mreže obrađuje podatke na drugoj razini apstrakcije i na temelju tih podataka donosi neku odluku, odnosno daje neki izlaz. Kretanjem od ulaznog sloja prema izlaznom razina apstrakcije se povećava te se grade kompleksniji i apstraktniji koncepti odlučivanja. Naprimjer kod prepoznavanja slika ulazni sloj obrađuje podatke na razinama piksela dok izlazni sloj radi na najapstraktnijoj razini i daje rezultat klasifikacije. Mogli bismo reći da s većim brojem slojeva možemo preciznije podijeliti problem klasifikacije na niz jednostavnih odluka koje se mogu donijeti na razinama piksela. S većim brojem slojeva ta podijeljenost je finija i preciznija.

U stvarnosti rad dubokih neuronskih mreža je simuliranje nelinearne funkcije s velikim brojem parametara (težina). Naučena mreža simulira funkciju koja je točno ona funkcija koja za dane ulaze daje izlaze koji su točni rezultati klasifikacije. Jasno je da su za probleme klasifikacije često potrebne složene funkcije koje nisu jednostavne. Veći broj slojeva duboke neuronske mreže povećava tu složenost i omogućuje pronalazak takvih funkcija.

Učenje neuronske mreže

Neuronske mreže imaju veliki broj parametara (težina) koje je potrebno odrediti kako bi mreža radila dobru klasifikaciju, zbog toga je potrebno znati kako odrediti te parametre. Kad se na temelju prolaska ulaznih podataka kroz skrivene slojeve dobije određeni izlaz u izlaznom sloju mreže, taj se izlaz uspoređuje s izlazom kojeg je mreža trebala dati te se računa pogreška pomoću funkcije gubitka. Na temelju te pogreške mreža uči podešavajući pritom svaku težinu.

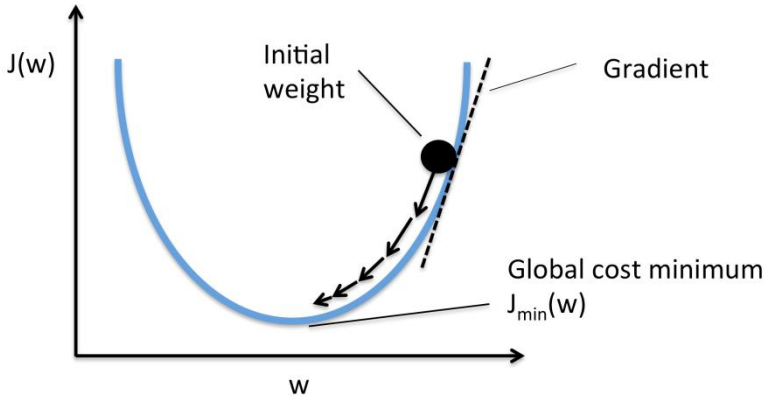
Postoje dva algoritma ključna za rad i učenje neuronske mreže, to su:

- Algoritam unaprijedne faze (eng. feedforward)
- Algoritam povratne faze (eng. backpropagation)

Algoritam unaprijedne faze (eng. feedforward) nam omogućuje sami rad neuronske mreže. Za svaki sloj se računa njegov izlaz krećući od ulaznog. Na ulazu ulaznog sloja su podaci za klasifikaciju dok je njegov izlaz ulaz sljedećeg sloja. Jedino na što treba obratiti pažnju je na povezanost slojeva koja za svaki neuron određuje njegovu povezanost s neuronima prethodnog sloja.

Algoritam povratne faze (eng. backpropagation) je određivanje greške i gradijenata u svakom sloju te podešavanje težina na temelju gradijenata i tako smanjujući grešku neuronske mreže. Prvo se pomoću algoritma feedforward dobije odziv mreže za neki ulaz. Zatim se izračunaju greške izlaznog sloja (greške se računaju na svakom neuronu). Zatim se za prethodni sloj određuje utjecaj neurona na greške u idućem sloju te se izračuna greška prethodnog sloja. Zatim se izračuna gradijent greške po težinama koje povezuju te slojeve te se te težine podešavaju. Ovaj postupak se ponavlja za svaki ulaz i određen broj puta.

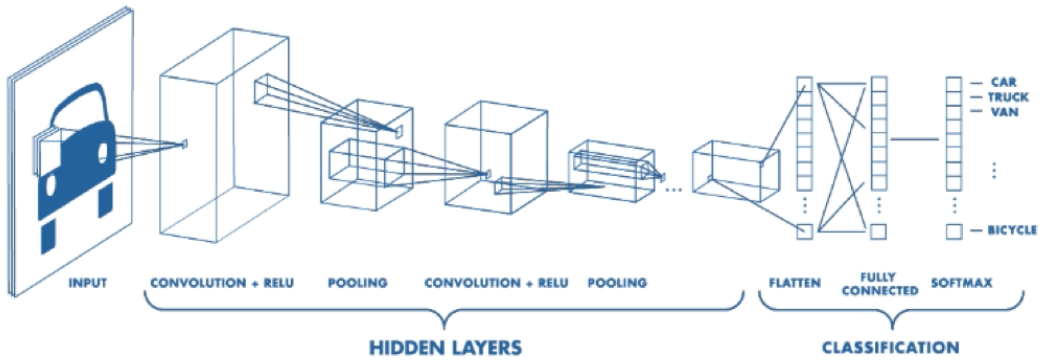
Cilj učenja je doći što bliže globalnom minimumu funkcije greške, to se postiže pomoću računanja gradijenata funkcije za svaku iteraciju nakon čega slijedi podešavanje parametara kako bi se počeli kretati u negativnom smjeru gradijenta. Stopa učenja η (eta) je mali pozitivni broj koji nam govori koliko brzo ćemo se kretati u smjeru negativnog gradijenta, odn. koliko brzo mreža uči. Gradijent pokazuje u smjeru rasta funkcije pa je zato kod podešavanja težina i pragova potrebno dodati negativan predznak jer pokušavamo minimizirati funkciju.



Slika 12 – Prikaz gradijentne metode učenja mreže

5. Konvolucijske neuronske mreže

Konvolucijske neuronske mreže (eng. Convolutional Neural Network - CNN) mogu se opisati kao nadogradnja nad običnim višeslojnim unaprijednim mrežama. Ova mreža je duboka neuronska mreža specijalizirana za raspoznavanje slika. Oponaša način na koji vizualni korteks mozga obrađuje i raspoznaje slike. One su s vremenom postale najpopularniji tip mreža za upotrebu kod bilo kakvih zadataka koji zahtijevaju klasifikaciju i/ili segmentaciju objekata na slici. Ime su dobile prema matematičkoj operaciji koja se upotrebljava u ovim mrežama: konvoluciji. Ove mreže je najjednostavnije definirati kao neuronske mreže koje umjesto standardnog načina množenja matrica (koji se obavlja u unaprijednim neuronski mrežama) koriste konvoluciju. One će svejedno na svom kraju prije samih krajnjih izlaza sadržavati gusto povezan sloj neurona, poput onih u standardnim unaprijednim mrežama



Slika 13 – Konvolucijska neuronska mreža

Konvolucijska, kao i obična, neuronska mreža sastoji se od ulaznog, izlaznog i jednog ili više skrivenih slojeva. Konvolucijske neuronske mreže najčešće kreću s jednim ili više konvolucijskih slojeva, zatim slijedi sloj sažimanja, pa ponovo konvolucijski sloj i tako nekoliko puta. Kod konvolucijskih neuronskih mreža specifični su konvolucijski slojevi i slojevi sažimanja. Osim njih često se koriste i potpuno povezani slojevi. Mreža najčešće završava s jednim ili više potpuno povezanih slojeva koji služe za klasifikaciju. Arhitektura konvolucijskih neuronskih mreža pokazala se izrazito dobra u radu sa slikama i prepoznavanju značajki s istih. One uočavaju primitivnije oblike u ranijim skrivenim slojevima, a složenije oblike u kasnijim.

Od 2012. godine konvolucijske neuronske mreže se počinju ubrzano razvijati te su preuzele vodeću ulogu u vizijskim sustavima. Danas najbolje konvolucijske mreže premašuju ljudsku izvedbu, što se smatralo nemogućim prije samo nekoliko desetljeća.

Struktura konvolucijskih neuronskih mreža

Razlika između konvolucijske neuronske mreže i klasične neuronske mreže je u tome što se kod konvolucijske neuronske mreže nastoji održati prostorna struktura. To je najvidljivije kod primjera slike koja može biti dvodimenzionalna ili trodimenzionalna. Kod slike svaki piksel ima određenu vrijednost te je njegova pozicija u odnosu na ostale piksele bitna. Zapisivanjem slike u višedimenzionalnom obliku umjesto u obliku potpuno povezanog sloja se uvelike smanjuje broj parametara koje mreža mora naučiti jer su neuroni u konvolucijskim mrežama povezani samo sa svojim receptivnim poljem u prijašnjem sloju, umjesto sa svim neuronima.

Struktura konvolucijske neuronske mreže slična je običnoj neuronskoj mreži jer također sadrži težine koje se mogu naučiti te aktivacijske funkcije koje rade pobudu nad neuronima. Konvolucijske neuronske mreže uzimaju u obzir da je ulazni sloj slika ili njoj sličan element te se njena arhitektura organizira na drugačiji način. Tako slojevi CNN-a imaju 3 dimenzije: visinu, širinu i dubinu. Visina i širina ulaznog sloja mreže ovisi o visini i širini slike dok dubina može varirati ali obično se uzima 3 (R, G i B komponenta boje) ili 1 (za crno-bijele ili jednobojne slike). Prednost volumnog rasporeda težina i nepotpune povezanosti u odnosu na običnu unaprijednu neuronsku mrežu jest znatno manji broj parametara. Uz to, uporabom posebnih vrsta slojeva CNN radi operacije koje učenjem i pravom arhitekturom mogu efikasno klasificirati slike

Slojevi koji čine konvolucijsku neuronsku mrežu su:

- Ulazni sloj (eng. input layer)
- Konvolucijski sloj (eng. convolutional layer)
- ReLU sloj
- Sloj sažimanja (eng. pooling layer)
- Potpuno povezani sloj (eng. fully-connected layer)
- Izlazni sloj (eng. output layer)

Ulazni sloj

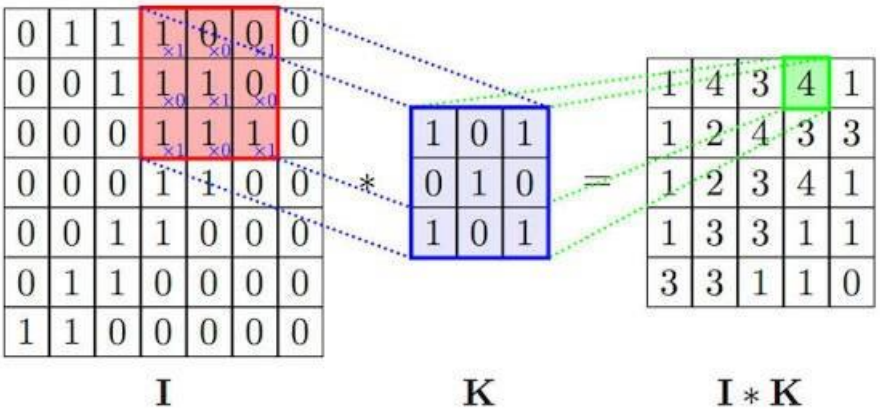
U ulazni sloj se postavlja slika u 3 dimenzije: W×H×D. W predstavlja širinu slike, H visinu, a D njenu dubinu. Ako je slika crno-bijela onda je D jednak 1, ako je u boji onda D iznosi 3 (za svaku od osnovnih boja: crvena, zelena, plava – po jedna slika).

Konvolucijski sloj

Konvolucijski sloj jedan je od glavnih dijelova svake konvolucijske neuronske mreže. Konvolucija je matematička operacija dviju funkcija kako bi se dobila treća funkcija koja opisuje kako oblik jedne funkcije utječe na drugu. Za operaciju konvolucije su zadužene jezgre (eng. filters) dimenzija F×F od kojih svaka prelazi preko ulazne matrice, obavljajući pritom operaciju konvolucije. Konvolucijski sloj funkcionira tako da pomoću jezgri pretvara ulazne slike u izlazne, tzv. mape značajki (eng. feature maps). Broj jezgri određuje broj mapa značajki na izlazu iz konvolucijskog sloja. Mapa značajki je matrica koja nastaje kao rezultat primjenjivanja jezgre na prijašnjem sloju. Prvi konvolucijski slojevi prepoznaju jednostavnije oblike (rubove, boje) dok su slojevi dublje u mreži u stanju prepoznati kompliciranije oblike. Proces se sastoji od toga da se na ulaznu matricu primijeni unaprijed određeni broj jezgri, na svaku jezgru se doda pomak (bias) te na kraju nelinearna aktivacijska funkcija, najčešća koja se koristi za konvoluciju je zglobna aktivacijska (ReLu) funkcija (slika 10.), kako bi se dobila izlazna matrica (mapa značajki). Veličina jezgre se definira visinom i širinom njene matrice. U većini se slučajeva za jezgre koriste manje matrice dimenzija 3x3 ili 5x5. Svaki element unutar jezgre, odnosno njene matrice je jedna težina. Jezgra veličine 5x5 bi tako imala 25 težina. Jezgre su uobičajeno manje od ulaznih matrica, a njihov broj u konvolucijskom sloju odgovara broju mapa značajki koje ćemo dobiti nakon tog sloja. Pri unaprijednoj fazi potrebno je konvoluirati jezgru (eng. filter) s ulazom. Rezultat konvolucije je dvodimenzionalna mapa značajki koja predstavlja odziv jezgre na svakoj prostornoj poziciji. Kako bi bilo moguće definirati točnu veličinu izlaza konvolucijskog sloja potrebno je definirati i korak pomaka filtra (eng. stride). Korak pomaka filtra označava za koliko će se jezgra pomicati po širini odnosno visini tijekom konvolucije.

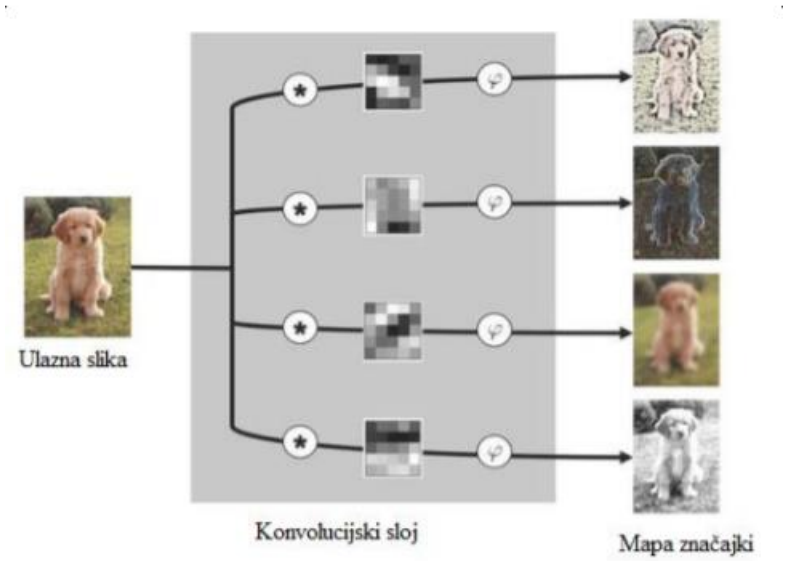
Na primjer u ulazu u konvolucijski sloj se nalazi slika koju predstavlja matrica dimenzija 7x7. U konvolucijskom se sloju nalazi jezgra dimenzije 3x3. Množenjem jezgre u konvolucijskom sloju s ulaznom matricom dobije se mapa značajki. Množenje se odvija tako da jezgra prelazi preko ulazne matrice, u njoj se povezuje s lokalnom regijom koja je jednake dimenzije kao jezgra i obavlja operaciju konvolucije (množenje matrica). Nakon prve operacije konvolucije dobije se vrijednost jednog neurona u mapi značajki. Nakon što je izračunat prvi neuron, jezgra se nastavlja kretati u desno po ulaznoj matrici nekim korakom s (eng. stride) te se spaja sa sljedećom lokalnom regijom kako bi izračunala vrijednost drugog neurona u mapi značajki.

Jezgra se kreće po ulaznoj matrici s lijeva na desno te se tako i rezultati konvolucije zapisuju u mapu značajki. Kad je jezgra, krećući se korakom s u desno došla do desnog ruba matrice, spušta se za korak s prema dolje te opet s lijeva na desno računa vrijednosti neurona u mapi značajki.



Slika 14 – Proces konvolucije jezgre i ulazne matrice

Mreža će naučiti težine unutar jezgre kako bi se jezgra aktivirala na mjestima gdje prepoznaje određena slikovna svojstva, kao na primjer određene vrste rubova ili slično. Svaki sloj najčešće se sastoji od više jezgri i svaka od njih će generirati jednu dvodimenzionalnu mapu značajki. Izlaz sloja bit će sve mape značajki, odnosno izlaz će biti više dvodimenzionalnih matrica gdje svaka predstavlja jednu dubinu izlaza.



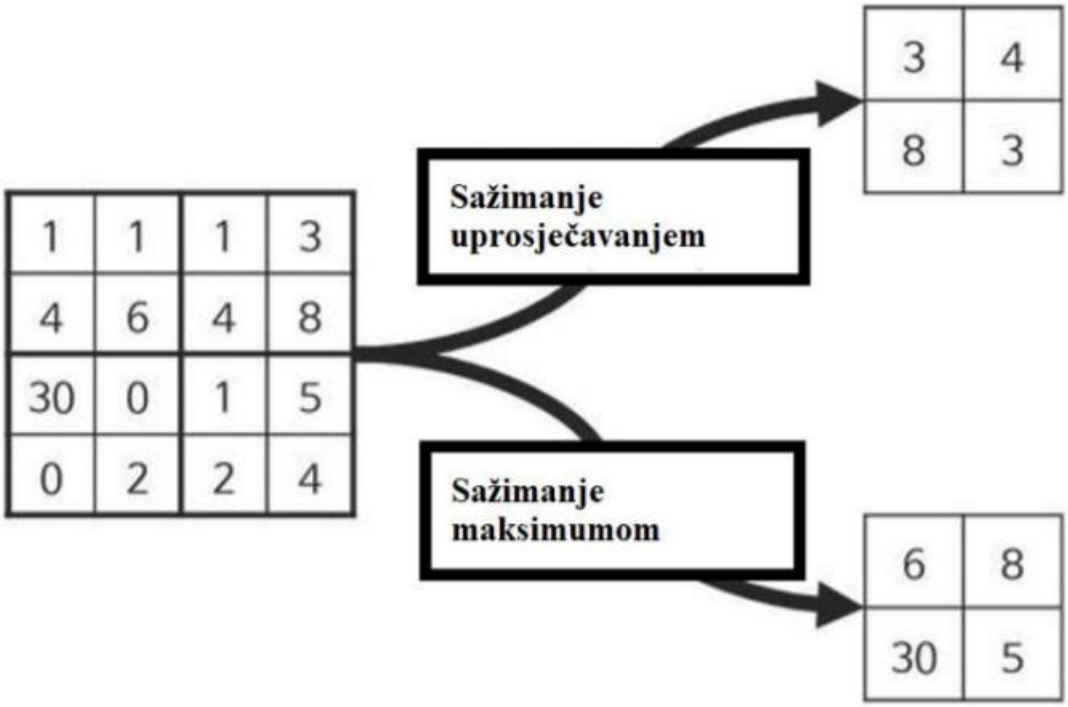
Slika 15 – Pretvorba ulazne slike u četiri mape značajki

Sloj sažimanja

Sažimanje (eng. pooling) se koristi kako bi se smanjila vjerojatnost pojave prenaučenosti, ali i kako bi se smanjenjem dimenzija mapa značajki smanjilo opterećenje za računalo tijekom procesa učenja. Ovaj sloj ne sadrži parametre koje mreža mora naučiti. Sloj sažimanja (eng. Pooling layer) se kreće isto kao i jezgra po ulaznoj matrici nekim korakom s i dimenzijom F x F te se kao izlaz dobiva manja matrica u odnosu na ulaznu. Sloj sažimanja smanjuje prenaučenost mreže, ali i memorijske zahtjeve jer smanjuje broj parametara

Ovaj sloj smanjuje ulaznu matricu tako što od susjednih piksela izuzima one piksele s najvećom vrijednosti ili računa njihovu srednju vrijednost. Broj piksela iz kojih se izuzima vrijednost može varirati ovisno o veličini matrice koja vrši sažimanje (eng. pooling) i njenom koraku. Kao i kod konvolucijskog sloja manji dio uzorka grupira se i obrađuje se, te rezultira jednom vrijednošću.

U ovim slojevima također postoje jezgre (eng. filters) s kojima prolazimo po ulaznoj mapi značajki. Mapa se sažima tako da se okvir predstavi s jednom vrijednošću. Na primjer, okvir veličine 2 × 2 se reprezentira s jednom vrijednošću dobivenom iz 4 vrijednosti unutar okvira čime se mapa značajki smanjuje 4 puta. Okvir se najčešće pomiče tako da se svaka vrijednost iz mape značajki koristi u samo jednom sažimanju. Pomak okvira (slika 16) bi za navedeni primjer bio jednak 2 u horizontalnom i vertikalnom smjeru.



Slika 16 – Sažimanje uprosječivanjem i sažimanje maksimumom

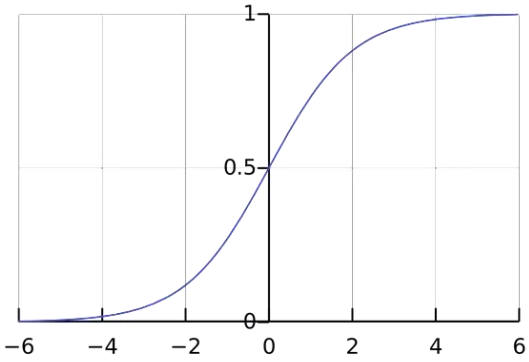
Na slici 16 se mogu vidjeti dva načina sažimanja:

- sažimanje uprosječivanjem (eng. mean pooling)
- sažimanje maksimumom (eng. max pooling).

Kod sažimanja uprosječivanjem grupirani podaci zamjenjuju se aritmetičkom sredinom grupiranih vrijednosti, dok se kod sažimanja maksimalnom vrijednosti grupirane vrijednosti zamjenjuju maksimalnom vrijednošću.

Potpuno povezani sloj

Potpuno povezani sloj (eng. fully-connected layer) se obično koristi u završnim slojevima konvolucijske mreže, no to ne mora biti slučaj. Ovaj sloj je identičan slojevima u potpuno povezanoj unaprijednoj neuronskoj mreži. Koristi se jer se propagacijom slike kroz mrežu nastoje smanjiti njene dimenzije. Tada ima smisla raditi s njima budući da je potpuna povezanost kvadratni broj veza između slojeva. Primjerice, za sliku dimenzija 225x225x3 ulazni sloj imao bi 151 875 ulaznih vrijednosti. Kad bi to potpuno povezali sa skrivenim slojem koji ima 1 000 neurona, imali bismo već $3x10^6$ težina za učenje. Zbog toga se potpuno povezani slojevi koriste u kasnijim fazama. Izlazi iz konvolucijske neuronske mreže za klasifikaciju slika većinom predstavljaju klase kojima slika može pripadati, dok se za aktivaciju tipično koristi Softmax funkcija.



Slika 17 – Softmax funkcija

Softmax aktivacijska funkcija normalizira rezultate dobivene na izlazima mreže transformirajući ih da budu pogodni za klasifikaciju, u rasponu [0, 1]. Na izlazima daje vjerojatnost za svaku klasifikaciju i kada se zbroje te vrijednosti dobije se 1.

Prepoznavanje slika pomoću konvolucijskih neuronskih mreža

Mreže ne percipiraju slike kao slike, već kao niz matrica, zbog toga će svaka slika u konvolucijskoj umjetnoj neuronskoj mreži biti predstavljena s brojkama. Kod slika će nam ulazne vrijednosti biti pikseli. Konvolucija kod slika je karakteristična po tome što se mora uzeti u obzir da slike posjeduju podatke kroz tri područja, jer se sve slike definiraju kao kombinacija određene količine crvene, plave i zelene boje (RGB). U sva tri područja određeni piksel poprima tri vrijednosti koje variraju od 0-255 (po jednu za svako područje). Vrijednost 255 predstavlja bijelu boju, a vrijednost 0 crnu boju. Ako slike nisu RGB, one će imati samo jedno područje unutar kojeg će se svi pikseli moći zamijeniti brojkom između 0 i 255.

Ako se za svaki piksel uzme da predstavlja jednu ulaznu vrijednost, onda možemo reći da ćemo RGB sliku dimenzija *512 x 512* zapravo predstaviti kao matricu dimenzija *512 x 512 x 3*. Budući da za svaki piksel uzimamo po jedan neuron smanjenjem piksela u upotrebi kroz konvoluciju smanjit će se potreban broj neurona za rad mreže. Uspješnost i preciznost mreže neće biti smanjena i mreža će brže učiti.

Slika 18 – Prikaz kako mreža vidi svaku sliku

Na slici 18, lijevo, vidimo izvornu sliku onako kako ju čovjek percipira, dok na desnoj strani vidimo matricu, koju percipira i koristi mreža, u kojoj je svaki piksel prikazan nekom brojčanom vrijednošću od 0 do 255.

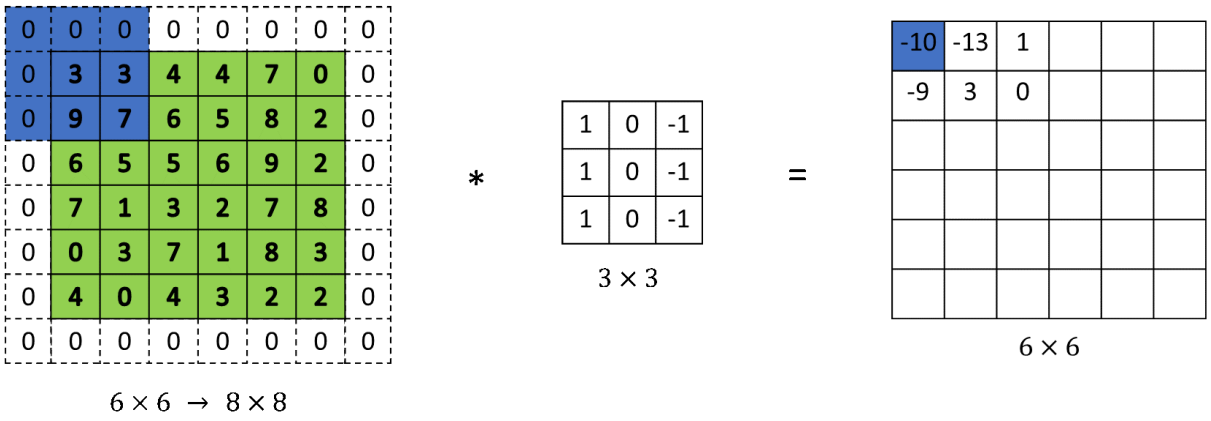
Na tri načina se regulira rezultat konvolucije slika, pomoću dubine, koraka i popunjavanja (eng. padding). Dubina podrazumijeva količinu filtera iskorištenih na nekoj slici. Budući da nas zanimaju razni podaci može se izvesti konvolucija na istoj slici s različitim

filterima, što će utjecati na izlazni rezultat. Filter konvolucije pomiče se po slici određenim korakom. Korak se bira kako bi najbolje odgovarao pojedinoj situaciji, ali postoje preferirane vrijednosti koraka. Što je korak veći, to će se konvolucijom više smanjiti dimenzije neke slike.

Popunjavanje se često koristi kako bi postojala kontrola nad brojem izlaznih neurona. Najčešće se upotrebljava popunjavanje nulama (eng. zero padding). Matrice se popune s nulama kako bi oblikom odgovarale filterima.

Popunjavanjem se postižu tri efekta:

- Sprječava se prebrzo smanjivanje veličine ulaznih podataka
- Omogućuje se korištenje mreža s više slojeva
- Bolje se mogu očuvati podaci koji se nalaze uz rubove



Slika 19 – Primjer popunjavanja nulama (zero padding)

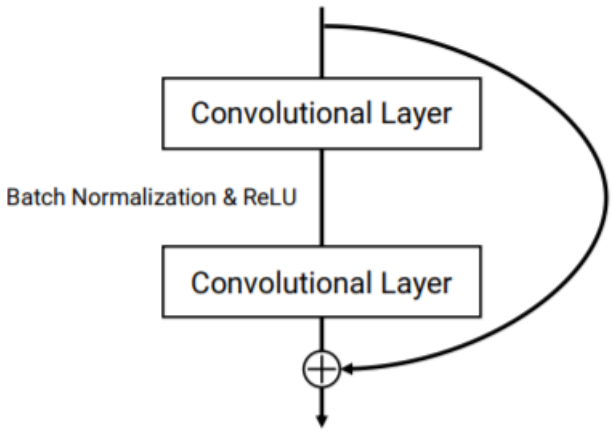
Postoje razne vrste algoritama konvolucijskih neuronskih mreža, a neki od njih su:

- LeNet
- AlexNet
- VGGNet
- GoogLeNet
- ResNet
- ZFNet

6. 3D ResNet – Rezidualna neuronska mreža

Neuronska mreža 3D ResNet je konvolucijska rezidualna neuronska mreža (eng. Residual neural network) koja koristi 3D konvoluciju nad slikama. U ovom radu koristi se za prepoznavanje ljudskih aktivnosti u videima. 3D konvolucijske neuronske mreže imaju jako dobre rezultate kada se uče na velikoj bazi podataka. Koriste se rezidualne neuronske mreže jer nam one, u usporedbi s ostalim neuronskim mrežama, daju najbolje rezultate kod klasificiranja aktivnosti u videima. Kod običnih neuronskih mreža koje ne koriste rezidualne blokove, a imaju veliki broj slojeva, s povećanjem dubine točnost predviđanja dolazi u zasićenje i može doći do pojave nestajućih gradijenata. Propagacijom pogreške u nazad koristi se lančano pravilo, gradijenti tada teže u nulu što sprječava daljnje učenje, kod previše slojeva rezultati su se nakon neke točke počeli pogoršavati. Kako bi se održala dubina neuronske mreže uveli su se rezidualni blokovi koji su spajanjem svakih nekoliko slojeva mreže omogućili veliki broj slojeva u mreži. Njihovim uvođenjem i dodavanjem slojeva pospješuju se rezultati neuronskih mreža.

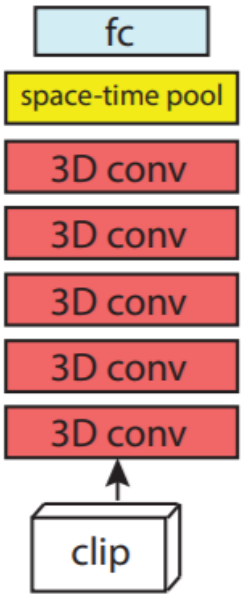
Rezidualne neuronske mreže u svojoj arhitekturi se sastoje od rezidualnih blokova. Struktura tih blokova im je takva da na ulazu blokova se granaju dvije grane. U prvoj grani blokovi imaju 3D konvolucijske slojeve, dok je druga grana prečica. Prečica koristi funkciju identiteta tj. preslikava ulaz i spaja se na kraju bloka s prvom granom. Kako se prolaskom slike kroz konvolucijske slojeve njena dimenzija smanjuje, a na kraju bloka se primjenjuje zbrajanje matrica konvoluirane slike i slike preslikane s ulaza, imamo slučaj nepodudaranja dimenzija za zbrajanje. Zbog toga se na konvoluiranu sliku odn. matricu primjenjuje proširivanje (eng. zero padding) kako bi se dimenzije podudarale i zbrajanje bilo moguće.



Slika 20 – Rezidualni blok 3D ResNet neuronske mreže

Rad i struktura 3D ResNet-18 neuronske mreže

U ovom radu u programu za prepoznavanje ljudskih aktivnosti koristi se već naučena 3D ResNet neuronska mreža koja sadrži 18 slojeva.



Slika 21 –3D ResNet-18 neuronska mreža

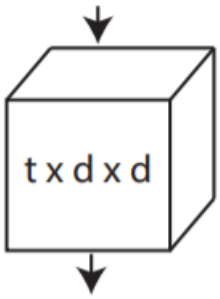
Ova neuronska mreža primjenjuje 3D prostorno-vremensku (eng. spatiotemporal) konvoluciju nad videima kod kojih se traži klasifikacija ljudskih aktivnosti.

Na ulazu u mrežu se dobavlja video (eng. clip) veličine *3 x L x H x W*

3 – 3 kanala boje, RGB (eng. red, green, blue – crvena, zelena, plava)
L – broj sličica (eng. frames)
H – visina slike
W – širina slike

Mreža za svoj ulaz uzima videe koji se sastoje od L RGB sličica (eng. frames) veličine *112 x 112* i njih onda propagira kroz svoje 3D konvolucijske slojeve. Kao i kod konvolucije slika tako i kod 3D konvolucije videa koristimo filtere za pronalaženje određenih značajki. Filteri su povezani u 3D kroz vremensku i prostornu dimenziju. Njihova veličina određena je izrazom t x d x d.

t – vremenski opseg filtra
d – prostorna širina i visina



Slika 22 – Prostorno-vremenski 3D filter

Svaki rezidualni blok se sastoji od dva konvolucijska sloja sa ReLu aktivacijskim funkcijama nakon svakog sloja. Izlaz rezidualnog bloka je definiran izrazom: $z_i = z_{i-1} + F(z_{i-1}; W_i)$. Funkcija F implementira sastav dviju konvolucija pomnoženih sa svojim težinama W i primjenu ReLu aktivacijskih funkcija. Tenzor z_i je u ovom slučaju četverodimenzionalan i ima veličinu N_i *x L x* H_i *x* W_i. N_i je broj filtera iskorištenih u i-tom bloku.

Prolaskom ulaznog videa kroz 3D konvolucijske slojeve dolazi do prostorno-vremenskog smanjivanja dimenzija. Pa se tako u prvoj 3D konvoluciji događa samo prostorno smanjivanje s filterima dimenzija *1 x 2 x 2*, dok u kasnijim 3., 4. i 5. 3D konvoluciji dolazi do prostorno-vremenskog smanjivanja primjenom filtera dimenzija *2 x 2 x 2*. Izlaz tenzora zadnje konvolucije ima veličinu $\frac{L}{8}$ *x 7 x 7*.

layer name	output size	R3D-18	R3D-34
conv1	L×56×56	3×7×7, 64, stride 1 × 2× 2	
conv2_x	L×56×56	$\begin{bmatrix} 3\times3\times3, 64 \\ 3\times3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3\times3, 64 \\ 3\times3\times3, 64 \end{bmatrix}\times3$
conv3_x	$\frac{L}{2}\times28\times28$	$\begin{bmatrix} 3\times3\times3, 128 \\ 3\times3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3\times3, 128 \\ 3\times3\times3, 128 \end{bmatrix}\times4$
conv4_x	$\frac{L}{4}\times14\times14$	$\begin{bmatrix} 3\times3\times3, 256 \\ 3\times3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3\times3, 256 \\ 3\times3\times3, 256 \end{bmatrix}\times6$
conv5_x	$\frac{L}{8}\times7\times7$	$\begin{bmatrix} 3\times3\times3, 512 \\ 3\times3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3\times3, 512 \\ 3\times3\times3, 512 \end{bmatrix}\times3$
	1×1×1	spatiotemporal pooling, fc layer with softmax	

Tablica 1 – Prikaz rezidualnih blokova i veličina filtera i izlaza za arhitekture 3D ResNet neuronskih mreža s 18 i 34 slojeva

Serija 3D konvolucijskih rezidualnih blokova završava sa slojem u kojem se primjenjuje globalno prosječno sažimanje (eng. average pooling) koje nam daje vektor s 512 značajki (eng. features). On se prosljeđuje potpuno povezanom sloju koji nam za izlaz daje vjerojatnosti klasifikacija kroz softmax funkciju, dobijemo konačno klasifikacijsko predviđanje. Izlazna veličina zadnjeg sloja ovisi o broju klasifikacija, što je u ovom slučaju 400 jer radimo s bazom podataka Kinetics 400.

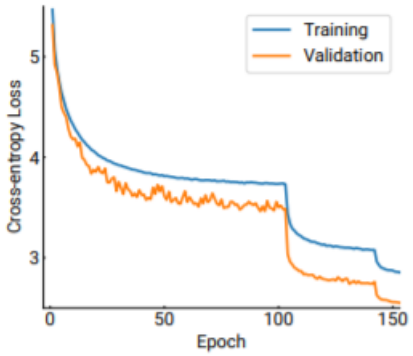
Trening (učenje) mreže s bazom podataka Kinetics 400

Baza podataka Kinetics 400 sadrži 400 klasificiranih ljudskih aktivnosti. Za svaku aktivnost postoji od 400 do 1150 videa, svaki video traje otprilike 10 sekundi. Trenutna verzija ove baze podataka ima 306 245 videa i ti videi su podijeljeni u tri grupe. Prva grupa su videi za treniranje (učenje) mreže kojih ima od 250 do 1000 za svaku klasificiranu aktivnost. Druga grupa su videi koji služe za procjenu preciznosti rada, klasifikacije, mreže, u toj grupi je 50 videa po aktivnosti. U trećoj grupi su videi koji služe za testiranje rada mreže i sadrži 100 videa po aktivnosti. Videi su preuzeti s You tubea i variraju im rezolucija kao i količina sličica (eng. frames) po videu.

Videi su skalirani na veličinu *128 x 171*, a onda svaki isječak posebno bude izrezan s prozorom veličine *112 x 112* koji onda ulazi u mrežu. Za trening se uzme L broj povezanih sličica iz videa. Ova 3D ResNet neuronska mreža je trenirana (naučena) na dvije veličine: $L = 8$ i $L = 16$.

Parametri rada 3D ResNet - 18 mreže

Tijekom rada naše neuronske mreže u prvoj fazi treniranja (učenja), zbog velikog broja parametara koji trebaju biti naučeni, u ovoj mreži njih nekoliko milijuna, postavljena je brzina učenja (eta) $\eta = 0{,}01$.



Slika 23 – Smanjenje greške tijekom učenja i procjene preciznosti rada

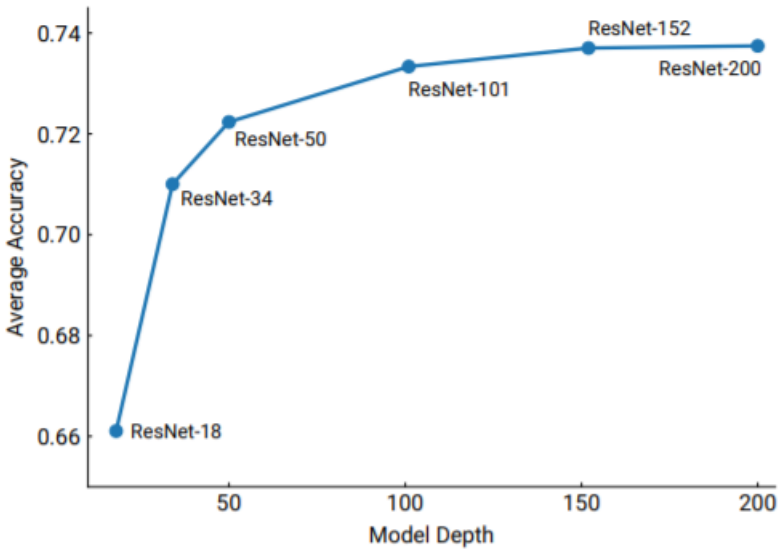
Povećanjem broja epoha kroz koje mreža uči približavamo se globalnom minimumu funkcije greške i samim time se smanjuje postotak greške (vidimo na slici 23), odnosno postiže se sve veća preciznost rada mreže (eng. accuracy).

Rezultate rada mreže na drugoj grupi videa, koji služe za procjenu preciznosti rada mreže, izražavamo preko postotka koji nam pokazuje u koliko slučajeva mreža točno klasificira događaje na videima.

Method	Top-1	Top-5	Average
ResNet-18	54.2	78.1	66.1
ResNet-34	60.1	81.9	71.0
ResNet-50	61.3	83.1	72.2
ResNet-101	62.8	83.9	73.3
ResNet-152	63.0	**84.4**	**73.7**
ResNet-200	**63.1**	**84.4**	**73.7**

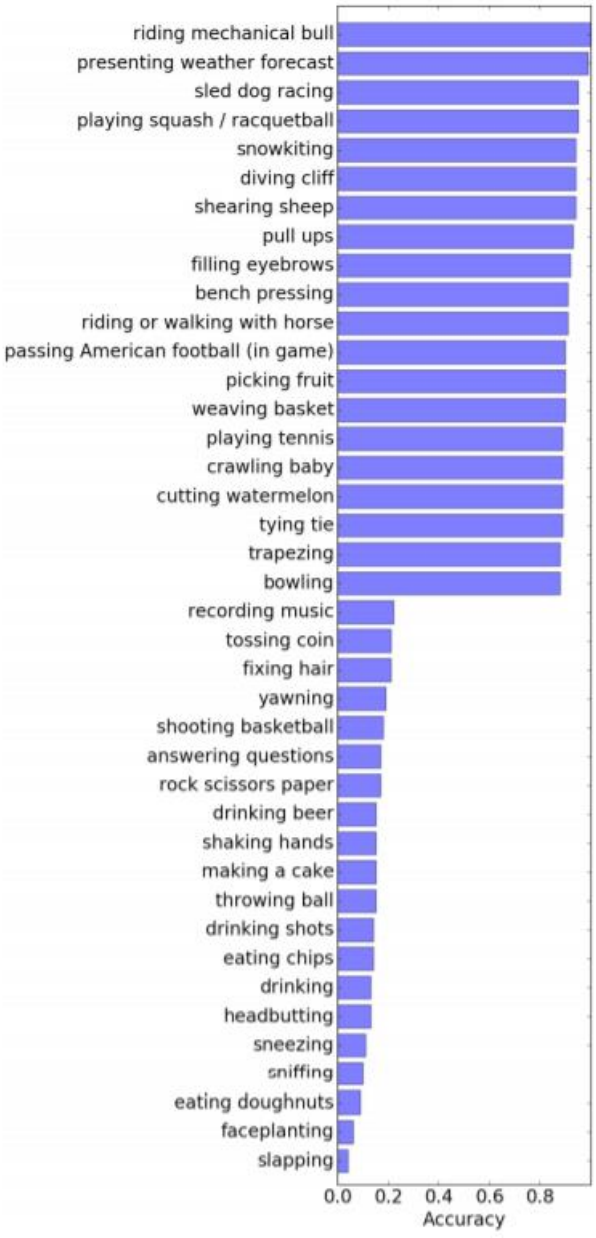
Tablica 2 – Preciznosti rada ResNet mreže s različitim brojem slojeva

U tablici 2 možemo vidjeti kako parametar preciznosti rada (eng. accuracy) naše 3D ResNet-18 mreže, koja se koristi u ovom radu, iznosi *66,1%* . Nadalje možemo vidjeti kako ista takva ResNet neuronska mreža sa sve većim brojem slojeva može postići bolje rezultate preciznosti rada.



Slika 24 – Graf preciznosti rada ResNet mreže s različitim brojem slojeva

Preciznost rada mreže u iznosu od *66,1%* je prosječna preciznost rada naše mreže u prepoznavanju svih klasifikacija ljudskih aktivnosti koje sadrži baza podataka Kinetics 400. Na sljedećoj slici, slika 25, moći ćete vidjeti 20 aktivnosti za koje mreža ima najveći postotak i 20 aktivnosti za koje ima najmanji postotak preciznosti rada.



Slika 25 – 20 najbolje i 20 najlošije prepoznatih aktivnosti u videima

7. Program korištenja naučene 3D ResNet neuronske mreže za prepoznavanje ljudskih aktivnosti

Nakon objašnjene strukture i rada 3D ResNet neuronske mreže dolazi detaljan prikaz i objašnjenje programa, pisanog u programskom jeziku Python, koji koristi već naučenu konvolucijsku neuronsku mrežu 3D ResNet za prepoznavanje ljudskih aktivnosti u već snimljenim videima. Neuronska mreža je naučena i istrenirana na bazi podataka Kinetics 400.

Programi, programski okviri i biblioteke potrebne za izvršavanje programa

Potrebno je imati instalirani programski jezik Python, pošto se u njemu programira, i program u kojem će se vršiti zapisivanje samog koda. U ovom slučaju koristi se program Notepad++ i programira se u verziji Python 3.7 . Samo pokretanje python skripti i ovog programa se izvodi u programu Anaconda. Anaconda je slična programu command prompt, ali ona nam omogućuje postavljanje takozvanih okoliša (eng. environments) u kojima možemo izolirano koristiti module i biblioteke potrebne za ovaj zadatak.

Za pokretanje python skripte u kojoj se koristi 3D ResNet neuronska mreža, zbog kompliciranih kompjutorskih i računalnih zahtjeva koje sama mreža mora ispuniti, potrebno je instalirati pytorch. Pytorch se koristi kod dubokog učenja i umjetne inteligencije, to je biblioteka strojnog učenja koja se koristi za razvoj i osposobljavanje modela dubokog učenja temeljenih na neuronskim mrežama. Pri pokretanju programskog koda potrebno je ostvariti okoliš (eng. environment) pytorch u kojem će se pokrenuti i izvršavati program prepoznavanja ljudskih aktivnosti. Isto tako potrebno je instalirati biblioteku Albumentations. To je biblioteka za transformaciju i procesiranje slika prvenstveno za duboko učenje neuronskih mreža.

Struktura programa

Program se sastoji od dviju python skripti, action_recognition.py i utils.py. Sadrži ulaznu mapu (eng. input folder) u kojoj se nalaze videi koji će se koristiti, u ovom slučaju koristit ćemo nekoliko njih čije rezultate predviđanja će te kasnije vidjeti. Uz to još imamo datoteku labels.txt u kojoj je sadržano svih 400 oznaka koje nam trebaju za klasificiranje događaja u videima. Važno je da se datoteke i skripte nalaze spremljeni u istoj mapi.

Prvo će se napisati kod u skripti utils.py.

Utils.py

Ova skripta služi za transformaciju veličina slika, za ulaz u neuronsku mrežu, i za učitavanje klasifikacija iz datoteke labels.txt.

```
1   import albumentations as A
2   transform = A.Compose([
3       A.Resize(128, 171, always_apply=True),
4       A.CenterCrop(112, 112, always_apply=True),
5       A.Normalize(mean = [0.43216, 0.394666, 0.37645],
6                   std = [0.22803, 0.22145, 0.216989],
7                   always_apply=True)])
8
9   with open('labels.txt', 'r') as f:
10          class_names = f.readlines()
11          f.close()
```

Na početku prvo odredimo kako će se transformacija slike vršiti korištenjem biblioteke albumentations. U trećem redu radi se promjena veličine ulaznih slika na veličinu *128 x 171*, veličina koja se koristi kod učenja. Nakon toga se primjenjuje centralno izrezivanje ulaznih sličica na veličinu *112 x 112* i vrši se normalizacija video sličica, pomoću naredbi mean i std.

Od reda 9 do 11 radi se čitanje datoteke labels.txt i naziva klasifikacija. Time završava prva python skripta utils.py.

Action_recognition.py

U ovoj skripti zapisujemo kod za prepoznavanje ljudskih aktivnosti koristeći model 3D ResNet neuronske mreže. Svaki kod nadalje napisan ulazi u skriptu action_recognition.py.

```
1  import torch
2  import torchvision
3  import cv2
4  import argparse
5  import time
6  import numpy as np
7  import utils
```

U prvom djelu skripte ubacujemo sve module i biblioteke koje su nam potrebne za izvršavanje daljnjeg koda. Isto tako ubacujemo kod iz skripte utils.py.

Nakon toga definiramo analizator argumenata koji nam služi za analiziranje kodova koje dajemo tijekom izvedbe action_recognition.py skripte.

```
10 parser = argparse.ArgumentParser()
11 parser.add_argument('-i', '--input', help='path to input video')
12 parser.add_argument('-c', '--clip-len', dest='clip_len', default=16, type=int,
13                     help='number of frames to consider for each prediction')
14 args = vars(parser.parse_args())
15 print(f"Number of frames to consider for each prediction: {args['clip_len']}")
```

U redu 11 je naredba --input preko koje ćemo pri pokretanju ovog koda napisati put do naših videa spremljenih u mapi input. U redu 12 je naredba --clip-len. Pomoću nje određujemo broj sličica za davanje klasifikacije tijekom unaprijedne propagacije dok se obrađuje video korištenjem modela 3D ResNet neuronske mreže. To znači ako damo vrijednost 16, model će odrediti klasifikaciju gledajući 16 uzastopnih sličica iz videa. Iznos varijable clip-len i podatke varijable input upisujemo tijekom pokretanja ove python skripte u programu anaconda.

Naredba clip-len je bitna jer broj sličica po pretpostavci klasifikacije utječe i na kvalitetu pretpostavke i na brzinu pretpostavke (koliko će biti sličica po sekundi – eng. Frames per second FPS). Korištenjem više sličica nam može donijeti bolje pretpostavke, ali uz manji broj sličica po sekundi, slično tome smanjivanjem broja sličica za svaku unaprijednu propagaciju možemo povećati broj sličica po sekundi, ali možemo dobivati pogrešne pretpostavke klasifikacija.

```
18  class_names = utils.class_names
19  device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
20  model = torchvision.models.video.r3d_18(pretrained=True, progress=True)
21  model = model.eval().to(device)
```

U redu 18 nastavljamo s pisanjem koda dobavljanjem imena klasifikacija direktno iz utils.py skripte . U redu 19 kroz kod provjeravamo sadrži li računalo CUDA (eng. Compute Unified Device Architecture). CUDA omogućuje ubrzanje izvedbe računalno intenzivnih aplikacija iskorištavanjem snage jedinice za obradu grafike. Također je poželjno da CUDA bude instalirana na računalu na kojem se koristi ovaj programski zadatak.

Za dobavljanje modela neuronske mreže 3D ResNet koristit će se modul torchvision, red 20. U redu 21 učitavamo model naše neuronske mreže na računalo.

Nastavljamo s pisanjem koda, u redu 23, u kojem inicijaliziramo OpenCV biblioteku i učitavamo video. OpenCV biblioteka nam služi za strojno učenje i obradu slika.

```
23  cap = cv2.VideoCapture(args['input'])
24  if (cap.isOpened() == False):
25      print('Error while trying to read video. Please check path again')
26  frame_width = int(cap.get(3))
27  frame_height = int(cap.get(4))
28  save_name = f"{args['input'].split('/')[-1].split('.')[0]}"
29  out = cv2.VideoWriter(f"outputs/{save_name}.mp4",
30                        cv2.VideoWriter_fourcc(*'mp4v'), 30,
31                        (frame_width, frame_height))
32  frame_count = 0
33  total_fps = 0
34  clips = []
```

U redovima 26 i 27 dobijemo visinu i širinu sličica (frames) videa. Nakon toga definiramo ime pod kojim će se spremiti video koji prođe kroz neuronsku mrežu.

Frame_count nam služi za brojanje ukupnog broja sličica koje su prošle, a total_fps nam služi da dobijemo broj sličica po sekundi. Nakon toga koristimo listu clips=[] u koju ćemo pridodati svaku sličicu iz videa prije nego ih pošaljemo u 3D ResNet model neuronske mreže.

Slanje videa u 3D ResNet model neuronske mreže se vrši kroz veliku while funkciju.

```
38  while(cap.isOpened()):
39      ret, frame = cap.read()
40      if ret == True:
41          start_time = time.time()
42          image = frame.copy()
43          frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
44          frame = utils.transform(image=frame)['image']
45          clips.append(frame)
46          if len(clips) == args['clip_len']:
47              with torch.no_grad():
48                  input_frames = np.array(clips)
49                  input_frames = np.expand_dims(input_frames, axis=0)
50                  input_frames = np.transpose(input_frames, (0, 4, 1, 2, 3))
51                  input_frames = torch.tensor(input_frames, dtype=torch.float32)
52                  input_frames = input_frames.to(device)
53                  outputs = model(input_frames)
54                  _, preds = torch.max(outputs.data, 1)
55
56                  label = class_names[preds].strip()
57              end_time = time.time()
58              fps = 1 / (end_time - start_time)
59              total_fps += fps
60              frame_count += 1
61              wait_time = max(1, int(fps/4))
62              cv2.putText(image, label, (15, 25),
63                          cv2.FONT_HERSHEY_SIMPLEX, 0.8, (0, 0, 255), 2,
64                          lineType=cv2.LINE_AA)
65              clips.pop(0)
66              cv2.imshow('image', image)
67              out.write(image)
68              if cv2.waitKey(wait_time) & 0xFF == ord('q'):
69                  break
70      else:
71          break
```

Na početku while koda, u redu 39, zapisujemo kod za traženje i čitanje videa. Ako video postoji prvo se zapisuje početno vrijeme početka klasifikacije ljudskih aktivnosti u videu (redovi 40 i 41).

Dva bitna reda koda u prvoj if funkciji su nam redovi 43 i 44. U redu 43 transformiramo slike u RGB boje iz BGR sustava boja koji je standardni u OpenCV biblioteci, dok u 44. redu transformiramo slike koristeći transformacije koje smo definirali u utils.py skripti. U redu 45 dodajemo svaku sliku na listu clips.

U redu 46 dolazimo do druge if funkcije u kojoj provjeravamo ako je broj sličica u listi clips jednak argumentu clip_len, kojeg smo definirali na početku. Ako je tako, nastavljamo dalje s ispisivanjem koda.

U redu 48 konvertiramo listu clips u NumPy niz i spremamo ga kao varijablu input_frames. Dalje u redu 49 dodajemo listi još jednu dimenziju da postane petero-dimenzionalna, što nam je bitno za rad naše neuronske mreže.

U redu 50 zapisujemo kod za transponiranje varijable input_frames da njena veličina bude definirana kao *[batch_size, 3, num_clips, height, width].* U našem slučaju njihova veličina će iznositi *[1, 3, 16, 112, 112].*

Nakon toga u redu 51 konvertiramo varijablu input_frames u torch tenzor. Torch tenzor je višedimenzionalna matrica koja sadrži elemente jednog tipa podataka. U redu 52 učitavamo slike (eng. frames) na naše računalo, a u redu 53 dodajemo i učitavamo video u naš model neuronske mreže 3D ResNet i nakon toga izlaz mreže spremimo u varijablu outputs.

U redu 54 dobijemo indeks klasifikacije mreže iz varijable outputs, a u redu 56 mapiramo taj indeks s nazivom klasifikacije koji smo već prije definirali.

U redu 57 zapisuje se vrijeme završetka klasifikacije ljudskih aktivnosti u videu. Od reda 58 do reda 60 računamo ukupni broj sličica koje su prošle kroz neuronsku mrežu i broj sličica po sekundi (FPS).

U redu 62 stavljamo pretpostavku klasifikacije mreže na sliku koju želimo pokazati, također u redu 65 pokazujemo zadnju sliku iz liste clips. U redovima 66 i 67 pokazujemo sliku s nazivom klasifikacije, koju je mreža pretpostavila, na ekranu našeg računala i to se onda sprema. S time završavamo funkciju while.

U završnom djelu koda zatvaraju se svi prozori na kojima se je video prikazivao i računa se i ispisuje se konačan broj sličica po sekundi (FPS).

```
74  cap.release()
75  cv2.destroyAllWindows()
76  avg_fps = total_fps / frame_count
77  print(f"Average FPS: {avg_fps:.3f}")
```

8. Klasificiranje ljudskih aktivnosti u videima

Nakon što smo napisali kodove u našim python skriptama vrijeme je da ih pokrenemo koristeći naše već snimljene videe. U nastavku komentirat ćemo i prikazati rezultate klasificiranja na 3 različita videa:

1. Streličarstvo (eng. archery)
2. Sviranje klavira (eng. playing piano)
3. Udarac nogom u glavu (eng. high kick)

Za početak otvaramo program anaconda. U njemu stvorimo okoliš pytorch u kojem dolazimo do mape u kojoj nam se nalaze naše python skripte i naša mapa s videima koje koristimo. Pošto prvo prikazujemo rad programa na videu sa streličarstvom, početni kod treba izgledati ovako:

>python action_recognition.py --input input/archery.mp4 --clip-len 16

Broj 16 na kraju koda nam pokazuje koliko uzastopnih sličica će neuronska mreža koristiti za klasificiranje aktivnosti, u ovom slučaju je to 16. Nakon upisivanja ovog koda pokreće se video na kojem se mogu vidjeti klasifikacije koje je odredila naša neuronska mreža.

a)

b)

c)

Slika 26 – Prikaz tri slike: a, b i c, svaka sa svojom klasifikacijom u gornjem lijevom kutu, slike su uzete u različitim trenucima videa

Kao što vidimo na slikama a i c naša neuronska mreža je točno klasificirala događaje u videu, dok na slici b vidimo da je došlo do krive klasifikacije, barbequing (roštiljanje). Razlog tome može biti nedovoljna naučenost mreže na slučajeve kada čovjek samo nosi u rukama luk i strijelu.

Dalje nastavljamo s puštanjem videa u kojem vidimo kako čovjek svira klavir. U Anaconda programu upisujemo sljedeći kod:

>python action_recognition.py --input input/playing_piano.mp4 --clip-len 16

Isto kao i s prethodnim videom, odlučili smo tražiti klasifikacije za 16 uzastopnih sličica.

Slika 27 – Prikaz dviju slika: a i b, svaka sa svojom klasifikacijom u gornjem lijevom kutu, slike su uzete u različitim trenucima videa

Kod ovog videa u kojem čovjek svira klavir vidimo da dolazi do pogrešne klasifikacije naše mreže, koja nam vraća povratnu informaciju kako je riječ o sviranju orgulja, a ne klavira. Do ove pogreške dolazi zbog razloga jer su instrumenti orgulje i klavir relativno slični, dok neuronska mreža nije dovoljno dobro naučena za prepoznavanje kada je riječ o sviranju klavira.

Za treći primjer rada naše 3D ResNet neuronske mreže uzet je video u kojem se radi nožni udarac u visini glave. Isto kao i za prethodne videe tražimo svaku klasifikaciju za 16 uzastopnih sličica. U programu anaconda upisujemo sljedeći kod:

>python action_recognition.py --input input/high_kick.mp4 --clip-len 16

a) b)

Slika 28 – Prikaz dviju slika: a i b, svaka sa svojom klasifikacijom u gornjem lijevom kutu, slike su uzete u različitim trenucima videa

U ovom videu se prikazuje udarac nogom u glavu (eng. high kick), dok naša neuronska mreža zbog nedovoljno dobre naučenosti, ali i sličnosti između udaraca, ljudsku aktivnost u ovom videu klasificira kao udarac sa strane (eng. side kick).

U sva tri primjera vidimo kako mreža relativno dobro i precizno prepoznaje aktivnosti na videima. Do pogrešaka se dolazi kada mreža nije dovoljno dobro naučena ili kada dolazi do poklapanja relativno sličnih, a opet različitih, podataka. Primjer toga je klasificiranje mreže da se radi o sviranju orgulja dok se svira klavir i o prepoznavanju jedne vrste udarca (eng. side kick) umjesto drugog (eng. high kick). Jedan od načina da povećamo šanse za ispravnu klasifikaciju je povećanje broja sličica (frames), u ovim primjerima 16, koje kao grupa ulaze u neuronsku mrežu, za promjenu stavimo 32 sličice, i za koje na kraju mreže kao izlaz dobijemo klasifikaciju. S većim brojem sličica omogućava se mreži da s više podataka koji dođu do zadnjeg sloja mreže lakše odredi ispravnu klasifikaciju, ali zbog kompliciranosti funkcija u mreži i u duljeg vremena prolaska ulaznih podataka kroz mrežu kao posljedicu imamo smanjen broj sličica po sekundi (FPS), što znači dulje vrijeme rada mreže za davanje klasifikacija. Suprotno tome ako bismo smanjili broj sličica u grupi koja ulazi u mrežu, odlučimo se za vrijednost 8, postigli bismo veći broj sličica po sekundi (FPS), ali zbog manjeg broja podataka koji ulaze u mrežu i dolaze do zadnjeg sloja mreža je sklonija pogrešnoj klasifikaciji aktivnosti. Takve razlike u klasificiranjima zbog promjene broja sličica koje ulaze u mrežu kao grupa možemo vidjeti i na ova tri prethodno predstavljena primjera.

9. Zaključak

Kroz ovaj rad cilj je bio upoznati vas s teorijskim dijelom umjetne inteligencije i neuronskih mreža i pokušati vam objasniti sve bitne pojmove pomoću kojih ćete bolje razumjeti rad neuronske mreže koja se koristi za prepoznavanje ljudskih aktivnosti u već snimljenim videima. Sa detaljnim objašnjavanjem strukture i rada pojedinih neurona i cijelih neuronskih mreža došli smo do konvolucijske neuronske mreže, do 3D ResNet mreže s 18 slojeva, koja se koristi u ovome radu u programskoj aplikaciji za prepoznavanje ljudskih aktivnosti. Treniranje i učenje neuronske mreže kako bi ona imala veću preciznost i bolje klasificirala i prepoznavala podatke koji ulaze u nju je dugačak i kompliciran proces. Kada je riječ o radu neuronske mreže na prepoznavanju događaja u videima potrebna je velika baza podataka na kojoj mreža ući, pa se tako u ovome radu koristi mreža koja je učila 400 različitih aktivnosti na preko 300 tisuća videa koji su bili prikupljeni. U poglavlju klasificiranje ljudskih aktivnosti u videima može se vidjeti kvaliteta i preciznost rada naše neuronske mreže. Vidimo kako je mreža u stanju dosta precizno prepoznati događaje u videima, ali isto tako se vidi i sklonost pogrešnom klasificiranju. Kako bi se smanjile greške u radu tj greške mreže kod klasificiranja događaja, naša neuronska mreža bi se trebala onda učiti i trenirati na još većoj bazi podataka s više videa koji se isto tako bave ljudskim aktivnostima. Takvo rješenje već postoji, pa tako uz našu bazu podataka Kinetics 400, koja je korištena u ovom primjeru, postoje i baze podataka Kinetics 600 i Kinetics 700 koje sadrže 600 odnosno 700 različitih ljudskih aktivnosti na više od 500 tisuća različitih videa koji se koriste za trening i učenje neuronske mreže u prepoznavanju ljudskih aktivnosti u videima. S dovoljno velikom bazom podataka na kojoj učimo i treniramo mrežu i s pravilnim podešavanjem parametara po kojima mreža radi, možemo postići veliku preciznost i točnost u radu naše neuronske mreže.

Za kraj kada shvatimo koliko je veliki potencijal i kako su velike mogućnosti rada umjetne inteligencije i neuronskih mreža, možemo samo s nestrpljenjem očekivati buduće uspjehe na poljima umjetne inteligencije i neuronskih mreža.

10. Literatura

[1] Nielsen Michael - Neural Networks and Deep Learning (god. 2019.)

[2] Židov Ivan – Uvod u neuronske mreže (god. 2018.)

[3] Kežman Domagoj - Inteligentna dijagnostika kvarova rotacijske opreme male brzine vrtnje (god. 2019.)

[4] Kopljar Damir – Konvolucijske neuronske mreže (god. 2016.)

[5] Delovski Boris - Primjena neuronskih mreža za segmentaciju tkiva u biomedicinskim snimkama pacijenata (god. 2019.)

[6] Mrđen Josip – Prepoznavanje rukopisa strojnim učenjem i primjena u identifikaciji popunjenih polja na uplatnici (god. 2018.)

[7] Tran D., Wang H., Torresani L., LeCun Y., Ray J. - A Closer Look at Spatiotemporal Convolutions for Action Recognition

[8] Kay Will - The Kinetics Human Action Video Dataset