Učinkovitost različitih vrsta neuronskih mreža u klasificiranju grafičkih prikaza

Kovačević, David

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Sveučilišni studij

UČINKOVITOST RAZLIČITIH VRSTA NEURONSKIH MREŽA U KLASIFICIRANJU GRAFIČKIH PRIKAZA

Završni rad

David Kovačević

Osijek, 2022.

1. UVOD

Posljednjih godina raste interes za vizualizaciju podataka zbog njezine sposobnosti da prezentira smislene uvide u podatke sve veće veličine. Stoga je vizualizacija važna za znanstvenike jer im pomaže u istraživanju, analizi i objavljivanju njihovih rezultata. U 21. stoljeću, u vremenu kada se sve više pojavljuje zanimanje za ljudski mozak stvaraju se nove ideje kako napraviti matematički model koji se može usporediti s radom ljudskog mozga. Kako je mozak najsloženiji organ kojeg je jako teško istražiti i imitirati njegov rad pred znanstvenicima u današnje vrijeme težak je zadatak napraviti model koji se može usporediti s ljudskim mozgom. Neuronske mreže su algoritmi koji oponašaju rad mozga i njihov zadatak je određivanje točnih detalja o objektu.

Određivanje točnih detalja o objektu, poput razlike između psa i mačke, često je potrebno jer nekvalificirani ljudi ne znaju dovoljno o objektu da bi točno odredili kojoj vrsti taj objekt pripada. Umjetna neuronska mreža trenirana podatkovnim skupom koji sadrži na tisuće različitih slika vrsta objekata mogla bi riješiti ovaj problem. Kod treniranja neuronske mreže kao ulaze koriste se slike na kojima je prikazana poznata vrsta objekta te na taj način mreža dobiva mogućnost prepoznavanja značajki promatrane klase. Trenirana neuronska mreža će kao izlaz dati numeričku vrijednost koja predstavlja vjerojatnosti pripadnosti nekoj klasi. Takav princip prepoznavanja može se koristiti i na drugim objektima, objektima poput modela grafičkih kartica, vrsta biljaka (npr. pšenica i kukuruz), vrste mikroupravljača te ostalih u kojima se mogu primjetiti neke oku vidljive razlike. Naravno s vremenom problemi prepoznavanja i analize postali su sve složeniji pa iz tog razloga se počinju upotrebljavati kompleksnije mreže s većim brojem slojeva i samih neurona u tim slojevima kako bi točnije i jasnije pratili kompleksnost problema.

Razvojem grafičkih procesora (*engl. Graphical Processing Unit – GPU)* duboke neuronske mreže postale su prikladan alat za rješavanje brojnih vrsta problema. Kombinacijom naprednih algoritama, povećanje računalne moći i podataka dovelo nas je do spoznaje o neuronskim mrežama te sve većem korištenju istih npr. analiza dizajna kemijskih proizvoda, detektori i simulacije kvarova komponenti zrakoplova i sl.

U drugom poglavlju se pronalazi rješenje treniranja mreža uz teorijsku podlogu što su umjetne neuronske mreže, plitke neuronske mreže (*engl. Shallow Neural Networks*), konvolucijske neuronske mreže (*engl. Convolutional Neural Networks*), povratne neuronske mreže (*engl. Recurrent Neural Networks*).

U trećem poglavlju u eksperimentalnom dijelu objašnjena je implementacija mreža, koje mreže su korištene te su uspoređeni njihovi rezultati i donijeti je zaključak na temelju izlaznih rezultata.

1.1. Zadatak završnog rada

Cilj ovog završnog rada je istražiti i opisati povratne neuronske mreže (*engl. Recurrent Neural Networks*), konvolucijske neuronske mreže (*engl. Convolutional Neural Networks*), plitke neuronske mreže (*engl. Shallow Neural Network*) te napraviti analizu i usporedbu rezultata između mreža RNN, CNN i RNN-CNN mreža uz podatkovni skup ICDAR 2019 data set of chart images.

2. PREGLED PODRUČJA RADA

Uz sve veću količinu podataka dostupnih iz različitih izvora u različitim formatima, potrebni su novi načini analize velikih skupova podataka. U tom kontekstu, tehnike vizualizacije informacija mogu se koristiti za predstavljanje velikih količina podataka, koristeći vizualne elemente koji mapiraju apstraktne podatke i interpretiraju ih na intuitivniji i objektivniji način [1]. Razvijene su različite tehnike vizualizacije informacija [1], a izbor jedne tehnike za analizu skupa podataka ima velik utjecaj na rezultate koji se iz njih izvlače [2], budući da je moguće generirati prilično različite grafove za isti skup podataka.

Huang, Zong i Tan [3] uvode zadatak klasificiranja slika grafova. Za obuku i prepoznavanje, detektori oblika koriste se za predstavljanje slike kao skupa značajki iz pronađenih oblika. Za rješavanje ovog problema učenja koristi se modificirani algoritam Diverse Density. Faktor korelacije (*engl. Correlation factor - CF*) svakog oblika izračunava se za svaku vrstu slike, a naučeni CF-ovi se koriste za klasificiranje nove slike, iz identifikacije oblika s visokim CF-om. Njihov rad koristio je 4 klase grafova i postigao prosječnu točnost od 76,762%.

Prasad et al. [4] imali su za cilj klasificirati nekoliko vrsta slika grafova koristeći značajke temeljene na obliku i prostornim odnosima. Za obuku, značajke temeljene na segmentaciji, istaknutosti krivulje, histogram usmjerenih gradijenata (*engl. Histogram of Oriented Gradients - HOG*) [5] i SIFT (*engl. Scale-Invariant Feature Transform*) [6] izdvajaju se iz svake slike, tako da se sličnost između slika izračunava pomoću Pyramid Match-a algoritam. Support Vector Machine (*SVM*) se koristi za klasifikaciju na temelju ovih rezultata sličnosti. Koristili su 5 vrsta grafova i imali su prosječnu točnost od 83,8%.

Tang et al. [7] predložio je DeepChart, okvir za klasifikaciju slika grafova koristeći kombinaciju konvolucijskih neuronskih mreža *(engl. Convolutional Neural Networks)* i Deep Belief mreža.

U ovom pristupu, CNN se koristi kao ekstraktor značajki, kroz težine dobivene iz potpuno povezanog sloja, koji pohranjuje karakteristike slike. Iz duboko skrivenih značajki dobivenih od CNN-a, Deep Belief Network koristi se za izradu predviđanja vrsta grafova. Korišteno je 5 klasa i predložena metoda je postigla prosječnu točnost od 75,4%.

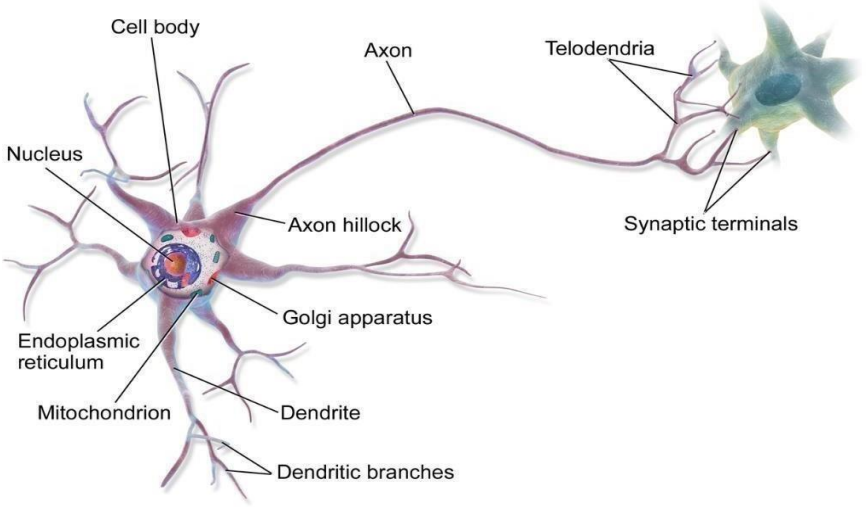
Jung et al. [8] predstavljaju interaktivni sustav za klasifikaciju i izdvajanje podataka iz grafičkih slika, nazvan ChartSense. Njihov je rad imao za cilj korištenje CNN-a za klasifikaciju, a za provjeru valjanosti ovog koraka napravljena je usporedba između tri CNN arhitekture (LeNet1 [9], AlexNet [10] i GoogLeNet [11]). Koristili su skup podataka Revision chart, ali su prikupili više slika kako bi stvorili veći korpus. Koristeći svoj novi skup podataka, postigli su točnost od 91,3% s GoogLeNetom.

Primjetno je da su neki radovi već koristili CNN-ove za klasificiranje slika grafova; međutim, nisu koristili robusnije arhitekture. Canziani, Paszke i Culurciello [12] usporedili su različite tipove CNN arhitektura dostupnih u literaturi koristeći ImageNet skup podataka. Uspoređujući ove rezultate s arhitekturama CNN-a koje su do sada korištene za klasifikaciju slika grafova, vidljivo je da postoje složenije arhitekture s boljim performansama koje se mogu testirati. Stoga ovaj rad predlaže procjenu ovih robusnijih arhitektura i njihovu usporedbu s konvencionalnim klasifikatorima.

U ovom radu koriste se raznovrsni načini treniranja mreža koji su opisani u prethodno navedenim istraživanjima. Konkretno za ovaj rad korištena je konvolucijska neuronska mreža koja je dizajnirana za ImageNet challenge iz 2014. godine pod nazivom GoogLeNet koja je sadržavala 22 sloja.

2.1. Što su umjetne neuronske mreže i što je neuron?

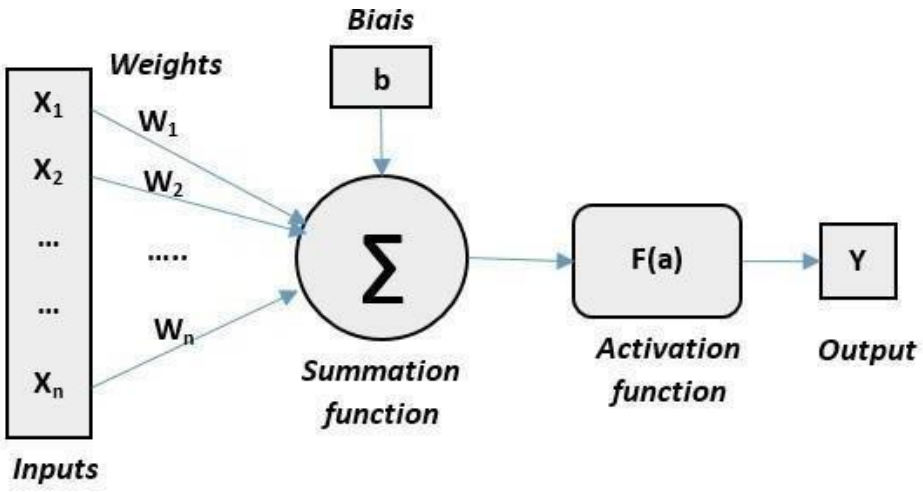
Umjetne neuronske mreže vrsta su algoritama koji se baziraju na živim bićima koje posjeduju živčani sustav tj. njihovim neuronskim mrežama. Neuron kao njihova osnovna gradivna jedinica vrlo je bliska biološkom neuronu prikazanom na slici 2.1.



Slika 2.1. *Prikaz biološkog neurona*

Na slici 2.1. vidi se da postoje i umjetni neuroni koji se temelje na principu rada biološkog neurona. Osnovna stvar koja je bila potrebna da se izgradi umjetni neuron je ta da postoji broj točaka, odnosno veza koje kontinuirano djeluju na određenu promatranu točku, također da postoji intenzitet uzbuda (težina veza) te da postoji odsutnost rivalne točke koja se ne nalazi u funkcionalnoj vezi s nekom promatranom točkom, a u koju bi se "pražnjenje" ostalih točaka moglo skrenuti. Ove tri tvrdnje su poslužile kako bi znanstvenici mogli izgraditi osnovnu strukturu neurona. Umjetni neuron kao i biološki ima ulaz, prijenos te izlaz. (x_1, x_2) predstavljaju ulaze na dendritima koji primaju neku informaciju (najčešće signal) iz prethodnih neurona koji se nalaze u mreži. Prijenos kod neurona govori da svaki neuron sadrži nekakvu prijenosnu funkciju u sebi kojom ona djeluje na ulaze i na izlaze (y_1, y_2) te na izlazu dalje propagira signal. Kakav signal će biti na izlazu ovisi o ulazu jer se oslanja na kvalitetu i pouzdanost prijenosne funkcije te se kroz trening težine funkcija mijenjaju ovisno o podražaju te na posljetku prijenosna funkcija odlazi u aktivacijsku. Može se reći da svaki

od ulaza sadrži svoju težinu te ujedno ta težina stvara pojačanje ili oslabljenje signala danog ulaza za danu funckiju.

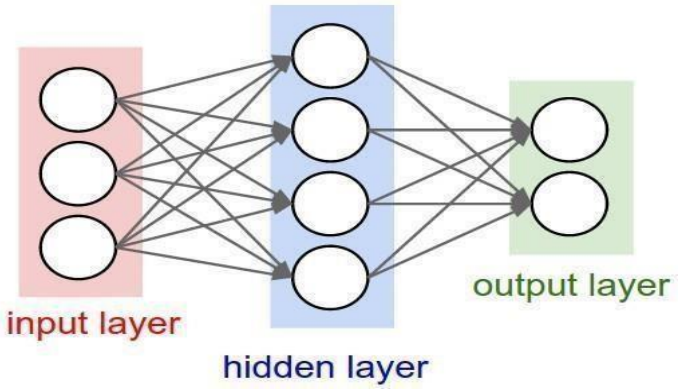


Slika 2.2. *Matematički model neurona*

Model neurona s matematičkog stajališta gleda se na način da svaki ulaz ima svoju težinsku vrijednost, dok se iste postavljaju tokom treninga ovisno o kompleksnosti ulaza.

Ulazni signali kojih ima "n" označavaju se s x_1, x_2, ... x_n, dok težine ulaznih signala označavaju se s ω_1, ω_2, ... ω_n. Izlaz svim slijedećim spojenim neuronima predstavlja funkcija nad sumom svih ulaza množenih svojom težinom. Na slici 2.2. prikazano je da se tijelo neurona zamjenjuje sumatorom, dok funkciju dendrita izvršavaju ulazi u sumator, a za izlaz sumatora kaže se da je to akson umjetnog neurona. Uloga praga je da se osjetljivost bioloških neurona preslikava u aktivacijsku funkciju. Veza umjetnog neurona i njegove okoline postiže se na način da se funkcijske sinaptičke veze biološkog neurona preslikavaju na težinske faktore s njegovom okolinom. Za težinske faktore može se pretpostaviti da su pozitvini ili negativni, a danas kod suvremenih neuronskih mreža može biti i neka funkcija (varijabilan težinski faktor).

Težinski faktori poistovjećuju se sa sinapsama biološkog neurona, iz razloga što se povezuju izlazi iz okoline neurona tj. izlaze drugih neurona s ulazima sumatora. Ulaz aktivacijske funkcije koja može biti linearna ili nelinearna povezujemo na izlaz sumatora .



Slika 2.3. *Neuronska mreža s jednim skrivenim slojem*

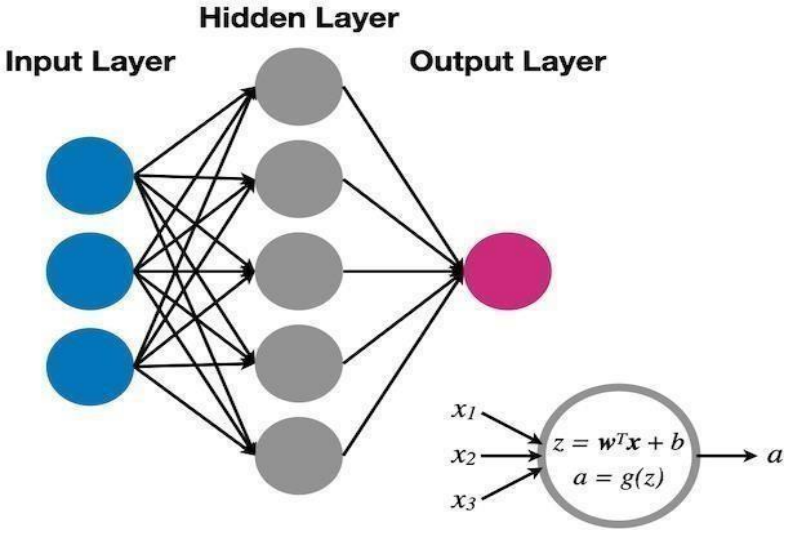
Na slici 2.3. zbog jednostavnosti kreiranja modela neuroni su u većini slučajeva poslagani u slojevima, ali to nužno ne znači da signal koji prolazi kroz model odmah završava na izlazu već se unutar procesa odvijaju razne funkcionalnosti između slojeva koje povećavaju točnost i daju bolje rezultate. Na slici 2.3. vidi se kako postoji jedan skriveni sloj što govori da mreža posjeduje linearnu povezanost dok za neke složenije procese mreža može imati više skrivenih slojeva između ulaza i izlaza.

2.1.1. Vrste umjetnih neuronskih mreža

Kategorizacija umjetnih neuronskih mreža ovisi o postavljenom kriteriju za koje se dobivaju različite neuronske mreže. Važno je za znati da paralelan skup neurona čini jedan sloj neuronske mreže. Umjetne neuronske mreže prema količini slojeva mogu se gledati kao jednoslojne i višeslojne. Za mreže sastavljene od više slojeva uobičajeno je za reć da su one izgrađene od ulaznog i izlaznog sloja, a između ta dva sloja da se nalaze skriveni slojevi. Ako se govori da signal propagira samo u jednom smjeru, od ulaza prema izlazu, govori se o tzv. (*engl. Feedforward Nerual Networks*) odnosno unaprijednim neuronskim mrežama, no ako postoji barem jedna povratna petlja u kojoj signal propagira u suprotnom smjeru, govori se o tzv. (*engl. Recurrent Neural Networks* ili *feedback*) koje se još nazivaju povratnim neuronskim mrežama. U ovom radu će se obratiti pažnja na RNN mrežama te mrežama koje imaju više skrivenih slojeva te njihovu učinkovitost u klasificiranju grafičkih prikaza.

2.1.2. Plitke neuronske mreže (*engl. Shallow Neural Networks*) i njihova struktura

Kada se čuje za pojam neuronska mreža odmah se pretpostavlja da se radi o mreži koja ima puno skrivenih slojeva, ali postoje mreže koje sadrže samo nekoliko skrivenih slojeva i to su upravo plitke neuronske mreže (*engl. Shallow Neural Networks*). Najčešće plitke neuronske mreže sastoje se od jednog do dva skrivena sloja koji čine mrežu. Razumijevanje plitke neuronske mreže daje uvid o tome što se zapravo događa unutar duboke neuronske mreže. Na slici 2.4. vidi se plitka neuronska mreža (*engl. Shallow Neural Network*) s parametrima koji su objašnjeni u daljnjem razmatranju.



Slika 2.4. *Plitka neuronska mreža s jednim slojem*

2.1.3. Rad plitke neuronske mreže i aktivacijske funkcije

Zapravo, cilj je shvatiti kako cijela neuronska mreža izračunava izlaz Y za dani ulaz X, upravo to može se prikazati slijedećim jednadžbama:

$$\boldsymbol{Z}_{[1]} = \boldsymbol{W}_{[1]T}\boldsymbol{X} + \boldsymbol{b}_{[1]} \quad \textbf{(2-1)}$$

Jednadžba (2-1) izračunava srednji izlaz Z[1] prvog skrivenog sloja

$$\boldsymbol{A}_{[1]} = \boldsymbol{\sigma}(\boldsymbol{Z}_{[1]}) \quad \textbf{(2-2)}$$

Jednadžba (2-2) izračunava konačni izlaz A[1] prvog skrivenog sloja

$$\boldsymbol{Z}_{[2]} = \boldsymbol{W}_{[2]T}\boldsymbol{A}_{[1]} + \boldsymbol{b}_{[2]} \quad \textbf{(2-3)}$$

Jednadžba (2-3) izračunava srednji izlaz Z[2] izlaznog sloja

$$y\text{^} = \boldsymbol{A}_{[2]} = \boldsymbol{\sigma}(\boldsymbol{Z}_{[2]}) \quad \textbf{(2-4)}$$

Jednadžba (2-4) izračunava konačni izlaz A[2] izlaznog sloja koji je ujedno i konačni izlaz cijele neuronske mreže

Nakon uvida kako plitka mreža izgleda i od čega se sastoji mora se uzeti još jedan vrlo važan parametar za opisivanje mreže, a to je aktivacijska funkcija. Poznato je da je neuronska mreža u osnovi skup matematičkih jednadžbi i težina. Kako bi se neuronska mreža učinila izdržljivijom, tako da radi dobro u različitim scenarijima, koriste se aktivacijske funkcije.
Ove funkcije uvode nelinearna svojstva u neuronske mreže. U slijedećim koracima objašnjeno je zašto su aktivacijske funkcije ključne za svaku neuronsku mrežu, a kao primjer će nam poslužiti plitka neuronska mreža.

Bez aktivacijskih funkcija neuronska mreža ima slijedeći oblik **(2-5)** :

$$\boldsymbol{Z}^{[1]} = \boldsymbol{W}^{[1]T}\boldsymbol{X} + \boldsymbol{b}^{[1]} \quad \textbf{(2-5)}$$

$$y\text{^} = \boldsymbol{Z}^{[2]} = \boldsymbol{W}^{[2]T}\boldsymbol{Z}^{[1]} + \boldsymbol{b}^{[2]} \quad \textbf{(2-6)}$$

Nakon uvođenja supstitucije za Z[1] iz jednadžbe (2-5) u jednadžbu (2-6) dobivaju se slijedeće jednadžbe:

$$\boldsymbol{Z^{[1]} = W^{[1]T}X + b^{[1]}} \quad \textbf{(2-7)}$$

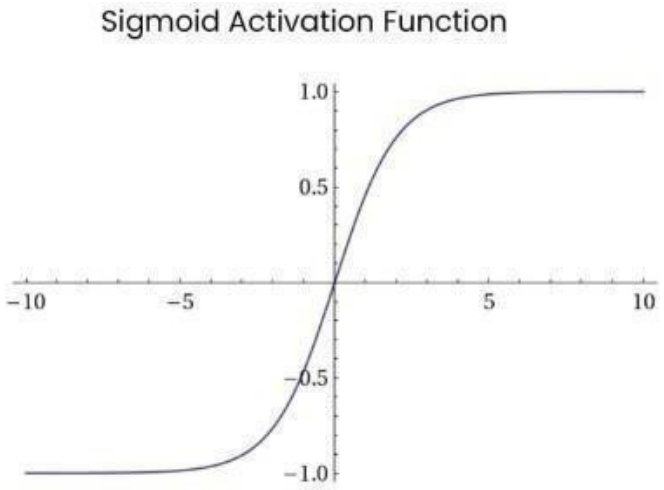
$$\boldsymbol{y^{\wedge} = Z_{[2]} = W_{[2]T}W_{[1]T}X + W_{[2]T}b_{[1]} + b_{[2]}} \quad \textbf{(2-8)}$$

$$\boldsymbol{y^{\wedge} = Z^{[2]} = w_{new}X + b_{new}} \quad \textbf{(2-9)}$$

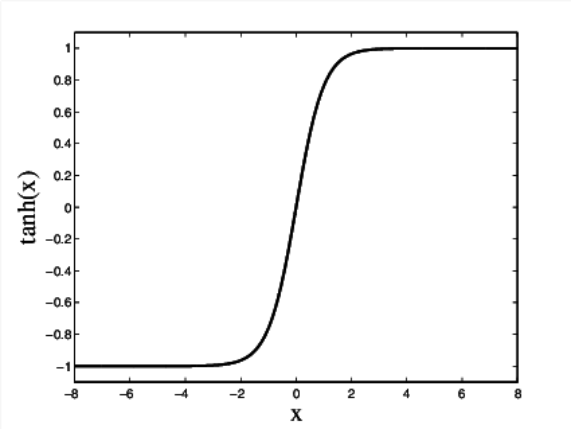
Kao što se vidi u jednadžbi **(2-7)** izlaz će biti linearna kombinacija nove težinske matrice W, ulaza X i novog biasa b (koji se interpretira kao konstantna linearna funkcija).

Iz jednadžbe **(2-7)** vidi se da nema značaja prisutnih neurona i težina u skrivenom sloju, stoga za uvođenje nelinearnosti u mrežu koriste se aktivacijske funkcije. Postoji mnoštvo aktivacijskih funkcija koje se mogu koristiti, a neke od njih su Sigmoid (2.5.), Tanh (2.6.), ReLU (2.7.), Leaky ReLu (2.8.) i druge.

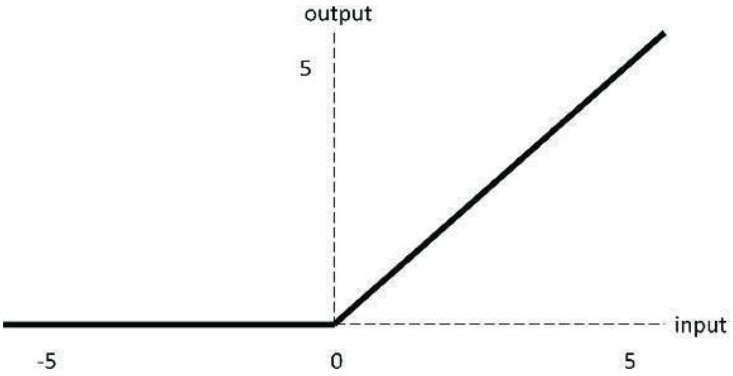
Pozitivna stvar aktivacijskih funkcija je da za određeni sloj se može izabrati različita aktivacijska funkcija, što znači da se ne mora za svaki sloj koristiti ista aktivacijska funkcija već se ona može mijenjati za svaki sloj.



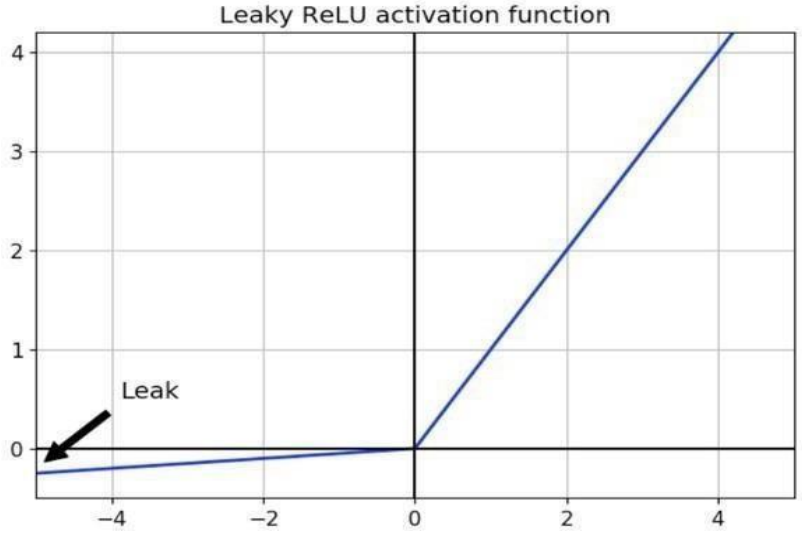
Slika 2.5. *Sigmoid aktivacijska funkcija*



Slika 2.6. *Tanh aktivacijska funkcija*



Slika 2.7. *ReLu aktivacijska funkcija*



Slika 2.8. *Leaky ReLu aktivacijska funkcija*

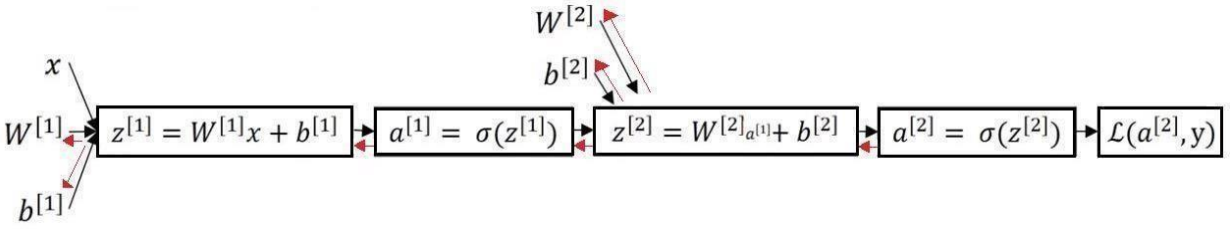
Poznato je da je matrica težine W neuronske mreže nasumično inicijalizirana. Neka se W1, matrica težine sloja 1 i W2, matrica težine sloja 2 inicijaliziraju s 0 ili bilo kojom drugom vrijednošću. Ako bi matrice W1 i W2 bile iste težine, aktivacije neurona u skrivenom sloju bile bi iste odnosno derivacije aktivacije bile iste, stoga bi neuroni u tom skrivenom sloju modificirali težine na sličan način te samim time ne bi bilo značaja imati više od 1 neurona u određenom skrivenom sloju. Naravno, to nije pogodno te umjesto toga želi se da svaki neuron u skrivenom sloju bude sam po sebi jedinstven, da ima različitu težinu i da radi kao jedinstvena funkcija pa zato iz ovog razloga težine se inicijaliziraju nasumično. Najbolja metoda za inicijalizaciju je Xavierova inicijalizacija.

Matematički definirana Xavierova inicijalizacija **(2-10)** :

$$W^{[l]} \sim N(\mu = 0, \mu^2 = \frac{1}{n^{[l-1]}} \tag{2-10}$$

$$b^{[l]} = 0 \tag{2-11}$$

Jednadžba **(2-10)** govori da težinska matrica W određenog sloja l je nasumično izabran broj iz normalne distribucije iz uvjeta da je $\mu = 0$ i varijance μ^2 = multiplikativni inverz od broja neurona u sloju l-1. Bias b iz jednadžbe **(2-11)** je inicijaliziran s nula u svim slojevima. Iz prethodne spoznaje da se težine neuronske mreže inicijaliziraju nasumično zaključuje se da kako bi se koristila neuronska mreža čija bi predviđanja bila ispravna, moraju se ažurirati dobivene težine. Metoda pomoću koje se ažuriraju te težine poznata je kao gradijentni spust (*engl. Gradient Descent*). U slijedećem primjeru prikazano je kako ta metoda funkcionira.



Slika 2.9. *Gradient Descent*

Na slici (2.9.) grafički se vidi metoda korištenja gradijentnog spusta (*engl. Gradient Descent*). Širenje prema naprijed (*engl. forward propagation*) označeno je s crnim strelicama i koristi se kako bi se izračunao izlaz za dani ulaz X. Širenje prema unatrag (*engl. backward propagation*) označeno je s crvenim strelicama i koristi se za ažuriranje težinske matrice W[1], W[2] i biasa b[1] i b[2].

Sam izračun se svodi da se izračunavaju derivacije ulaza u svakom koraku Gradient Descent grafa.

Koristeći se spoznajom da je gubitak L matematički definiran na način prikazan u jednadžbi (2-12) :

$$L(\hat{y}, y) = -[y log\hat{y} + (1-y)\log(1-\hat{y})] \quad \textbf{(2-12)}$$

Koristeći jednadžbu (2-12) za gubitak L te koristeći sigmoidalnu funkciju kao aktivacijsku za skriveni i izlazni sloj, uz pomoć lančanog pravila derivacija dobivaju se slijedeće jednadžbe (2-13) – (2-19):

$$dA^{[2]} = \frac{\delta L(A^{[2]},Y)}{\delta A^{[2]}} = \frac{-Y}{A^{[2]}} + \frac{1-Y}{1-A^{[2]}} \quad \textbf{(2-13)}$$

$$dZ^{[2]} = \frac{\delta L(A^{[2]},y)}{\delta Z^{[2]}} = \frac{\delta L(A^{[2]},y)}{\delta A^{[2]}} * \frac{\delta A^{[2]}}{\delta Z^{[2]}} = A^{[2]} - Y \quad \textbf{(2-14)}$$

$$dW^{[2]} = \frac{\delta L(A^{[2]},y)}{\delta W^{[2]}} = \frac{\delta L(A^{[2]},y)}{\delta Z^{[2]}} * \frac{\delta Z^{[2]}}{\delta W^{[2]}} = dZ^{[2]}A^{[1]T} \quad \textbf{(2-15)}$$

$$db^{[2]} = \frac{\delta L(A^{[2]},y)}{\delta b^{[2]}} = \frac{\delta L(A^{[2]},y)}{\delta Z^{[2]}} * \frac{\delta Z^{[2]}}{\delta b^{[2]}} = dZ^{[2]} \quad \textbf{(2-16)}$$

$$dA^{[1]} = \frac{\delta L(A^{[2]},Y)}{\delta A^{[1]}} = \frac{\delta L(A^{[2]},Y)}{\delta Z^{[2]}} * \frac{\delta Z^{[2]}}{\delta A^{[1]}} = dZ^{[2]}W^{[2]} \quad \textbf{(2-17)}$$

$$dW^{[1]} = \frac{\delta L(A^{[2]},y)}{\delta W^{[1]}} = \frac{\delta L(A^{[2]},Y)}{\delta Z^{[1]}} * \frac{\delta Z^{[1]}}{\delta W^{[1]}} = dZ^{[1]}X^{T} \quad \textbf{(2-18)}$$

$$db^{[1]} = \frac{\delta L(A^{[2]},y)}{\delta b^{[1]}} = \frac{\delta L(A^{[2]},y)}{\delta Z^{[1]}} * \frac{\delta Z^{[1]}}{\delta b^{[1]}} = dZ^{[1]} \quad \textbf{(2-19)}$$

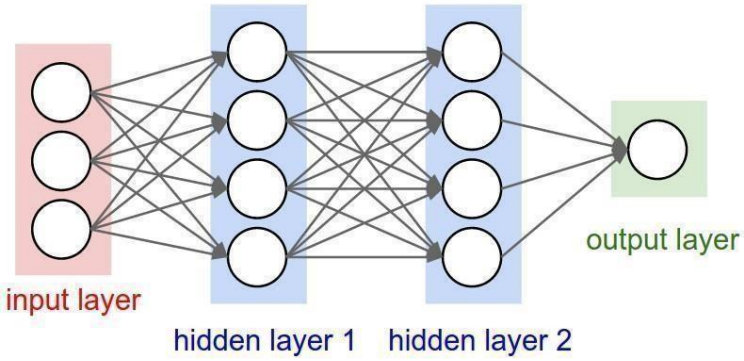
U jednadžbi (2-19), operacija "*" predstavlja množenje po produktu, a "σ'" predstavlja derivaciju sigme (sigmoidalna aktivacijska funkcija).

2.2 Konvolucijske neuronske mreže (*engl. Convolutional Neural Networks*)

2.2.1. Konvolucijske neuronske mreže i njihova struktura

Umjetne neuronske mreže (*engl. Artificial Neural Networks*) su računski procesni sustavi koji su kako je prethodno rečeno uvelike inspirirani načinom na koji biološki živčani sustavi (kao što je ljudski mozak) rade. ANN uglavnom sastoje se od velikog broja međusobno povezanih računalnih čvorova koji se nazivaju neuroni, od kojih se oni isprepliću u slojevima kako bi kolektivno učili iz zadanog ulaza kako bi dobili optimizirani konačni izlaz. Način na koji umjetne neuronske mreže rade je da se učitava ulaz u obliku višedimenzionalnog vektora na čiji ulazni sloj će se neuroni podijeliti na ostale skrivene slojeve, ovisno koliko ih ima. Skriveni slojevi će donositi odluke iz prethodnog sloja i određivati je li proces unutar skrivenog sloja šteti ili poboljšava konačni rezultat, a to se naziva dubokim učenjem.

Konvolucijske neuronske mreže (CNN) analogne su tradicionalnim oblicima ANN-a i samim time smatraju se podvrstom ANN-a. Analogne su po tome što se obje sastoje od neurona koji se samooptimiziraju učenjem. Svaki neuron će i dalje primati ulaz i izvoditi operaciju (kao što je skalarni produkt praćen nelinearnom funkijom) – temelj brojnih umjetnih neuronskih mreža. Od ulaznih vektora slike do konačnog izlaza iz koje se dobiva ocjena mreže, cijela mreža će i dalje izražavati jednu funkciju koja se razlikuje za svaki neuron unutar mreže, a to je težina. Posljednji sloj će sadržavati gubitke neuronske mreže povezane klasama te sve stvari koje se mogu primjeniti na ANN-u mogu se primjeniti i na CNN. Jedina značajna razlika između CNN-a i tradicionalnih ANN-a je u tome što CNN-i se uvelike i prvenstveno koriste u području prepoznavanja uzoraka unutar slika, a to pomaže da se prepoznaju detalji koji čine sliku specifičnom. Sama ideja iza neuronske mreže navodi da za neki zadani skup podataka se napravi predikcijski model za prepoznavanje uzoraka ili ranije spomenutih slika. Na slici 2.10. prikazana je troslojna mreža te način na koji su ti slojevi i sami neuroni povezani.



Slika 2.10. *Prikaz troslojne mreže*

Govoreći o samim konvolucijskim mrežama, svrstavaju se u oblik umjetnih neuronskih mreža koje su u proteklih desetak godina najzaslužnije što se tiče napretka u područjima računalne tehnike. U stvarnosti, objekti snimljeni kamerom su vrlo mali te je njihova detekcija vrlo kompleksna. Detekcija objekta je metoda koja se koristi da se odredi lokacija (pozicija) objekta na slici. U današnje vrijeme sve više aspekata života se pokazuju statistički, prikazom grafova. Podaci koji ulaze u statističku analizu uzimaju se tokom dužeg vremenskog razdoblja te će i same metode morati biti puno zahtjevnije i naprednije kako bi se moglo rukovati tolikom količinom podataka.

2.3. Povratne neuronske mreže (*engl. Recurrent Neural Networks*)

RNN (*engl. Recurrent Neural Network*) je vrsta umjetne neuronske mreže koja koristi sekvencijalne podatke ili podatke vremenske serije. Sekvencijalni podataci su podaci kod kojih je poznato da neka točka unutar skupa podataka (*engl. dataset*) ovisi o drugim točkama unutar skupa podataka dok podaci vremenske serije bilježe podatke u dosljednim vremenskim intervalima, odnosno, podaci o poprečnom presjeku sastoje se od nekoliko varijabli snimljenih u isto vrijeme.

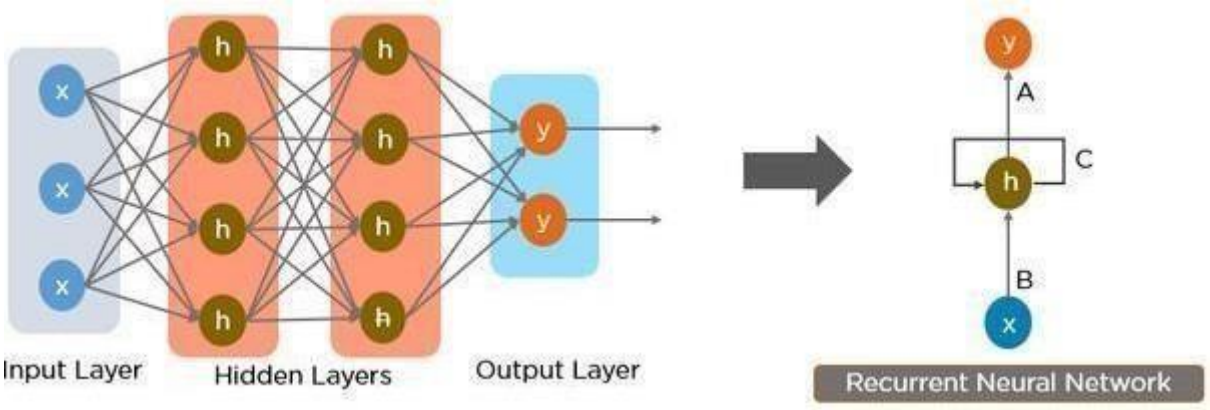
2.3.1. Općenito o RNN mrežama

Algoritmi koji opisuju RNN su algoritmi dubokog učenja koji se obično koriste za redovne ili vremenske probleme, kao što su prijevod jezika, prepoznavanje govora, obrada prirodnog

jezika itd. Široka primjena RNN-a je upravo u aplikacijama koje se koriste na dnevnoj bazi poput Google Translate-a, Siri, Google glasovno pretraživanje i druge. Također, nedavno je Gmail implementirao novu značajku da kada korisnik platforme piše neku tekstualnu poruku da mu sustav predviđa koja riječ bi mogla biti slijedeća iz konteksta poruke, a to je ono za što je zaslužna RNN mreža. Način na koji radi RNN je da sprema izlaz određenog sloja i vraća ga natrag na ulaz kako bi se predvidio izlaz sloja.

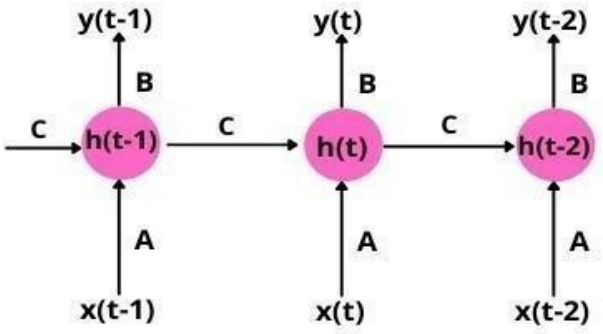
2.3.2. Način dobivanja RNN iz feedforward neuronske mreže te njihova parametrizacija

Na slici (2.11.) prikazano je kako od klasične feedforward neuronske mreže se dobiva RNN.



Slika 2.11. *Prikaz jednostavne RNN mreže*

Na slici (2.11.) vidi se da čvorovi u različitim slojevima neuronske mreže su komprimirani (potisnuti) da tvore jedan sloj RNN-a. Iz slike se prepoznaje da je "x" ulazni sloj, ″h″ je skriveni sloj dok ″y″ predstavlja izlazni sloj. Parametri A, B i C se nazivaju mrežnim parametrima te se koriste za poboljšanje rezultata modela. Za bilo koji trenutak t, trenutni ulaz je kombinacija x(t) i x(t-1) dok se izlaz u bilo kojem trenutku vraća u mrežu kako bi joj poboljšao krajnji rezultat (*engl. output*).



Slika 2.12. *Potpuno povezana RNN mreža*

Parametri koji opisuju sliku ***(2.12.)***:

$h(t)$ – novonastalo stanje

f_c - funkcija s parametrom c

$h(t-1)$ – prethodno stanje

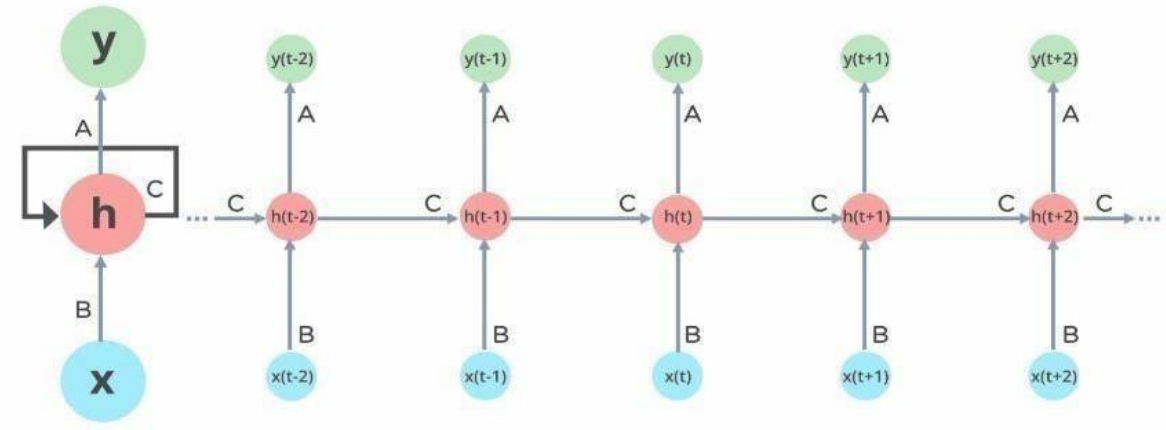
$x(t)$– ulazni vektor za trenutak t

Ako se znaju poznati parametri može se odrediti novonastalo stanje h(t) iz jednadžbe (4-1):

$$h(t) = f_c(h(t-1), x(t)) \tag{4-1}$$

2.3.3. Način rada RNN mreža

U RNN mrežama informacije kruže kroz petlju do srednjeg skrivenog sloja.



Slika 2.13. *RNN mreža*

Iz slike (2.13.) zaključuje se da ulazni sloj “x“ uzima ulaz u neuronsku mrežu i obrađuje ga i prosljeđuje u srednji sloj. Kao svaka neuronska mreža i ova se može sastojati od više slojeva tako da srednji sloj “h“ se sastoji od više skrivenih slojeva za koji svaki od njih ima svoju vlastitu aktivacijsku funkciju, težinu i bias. Treba pripaziti jer se ne može u svim slučajevima koristi RNN. Ako postoji neuronska mreža u kojoj prethodni sloj ne utječe na različite parametre različitih slojeva u prijevodu neuronska mreža nema memoriju, tada se može koristiti rekurentna neuronska mreža.

3. EKSPERIMENTALNI DIO

U eksperimentalnom djelu ovoga rada opisan je postupak implementacije već prethodno naučenih modela neuronske mreže te kako se evaluira skup podataka ICDAR 2019 dataset za postojeću mrežu. Programski kod je napisan u Python-u te zbog velike zahtjevnosti samih mreža koristila se platforma Google Colab koja pruža da s manje računalne snage preko njihovih servera se treniraju mreže. Programski kod je izvršen pomoću grafičke procesorske jedinice (*engl. Graphics processing unit – GPU*), što je uvelike povećalo brzinu učenja modela za razliku od središnje procesorske jedinice (*engl. Central processing unit - CPU*), koja je puno sporija i samim time otežava učenje modela. Prilikom pisanja programskog koda za model koji se trenira koristio se Keras. Osim Kerasa, koristio se i TensorFlow koji je poslužio za rješvanje zahtjevnih problema s numeričkog stajališta.

ICDAR 2019 podatkovni skup je skup slika koji se sastoji od 7 klasa (Vertical bar,Vertical box, Horizontal bar, Horizontal box, Pie, Scatter, Line) te je zadatak bio sortirati trening i testing data set s pripadajućim klasama. Za trening set uzeto je 300 slika za svaku od prethodno navedenih klasa što ukupno iznosi 2100 slika za cijeli trening dataset dok se za testing set uzelo 32 slike za svaku od prethodno navedenih klasa što ukupno iznosi 224 slike za cijeli testing dataset.

3.1. Eksperimentalni dio za podatkovni skup ICDAR 2019

Prilikom pripreme podataka kod rada s ICDAR 2019 podatkovnim skupom, bilo je potrebno sve slike prilagoditi ulaznom sloju neuronske mreže. Minimalna prilagodba koju je potrebno napraviti je prilagodba rezolucije i količine boja.

Zadane slike mogu se povećati, odnosno može im se povećati rezolucija, no to ne bi donijelo više informacija o slici osim toga treniranje mreže u kojoj postoji velik podatkovni skup s velikom rezolucijom slika u istom bi oduzelo puno vremena te samim time potrebno je koristiti puno jača računala da bi se mogla istrenirati mreža. Dobro rješenje za ovaj problem je zadržati veličinu slike i izgraditi mrežu s manje parametara, ali i dalje da ista ta mreža ima sposobnost prepoznavanja s velikom točnošću. Na temelju ovih zahtjeva su znanstvenici preporučili korištenje CNN mreža. Za neuronsku mrežu koja se trenira postavljeno je da je veličina slike 64x64 px iz razloga da mreža koja se trenira bude brža i točnija tj. da postoji

manja mogućnost da se dobiju veća odstupanja od pretpostavljenog, a kod za prilagodbu prikazan je na slici 3.1.

```
[ ]  IMAGE_SIZE = [64, 64]


     train_path = '/content/drive/MyDrive/Traning'
     test_path = '/content/drive/MyDrive/Testing'
```

```
from keras.preprocessing.image import ImageDataGenerator

train_datagen = ImageDataGenerator(rescale = 1./255)

test_datagen = ImageDataGenerator(rescale = 1./255)

trainig_set = train_datagen.flow_from_directory(train_path,
                                                target_size=(64,64),
                                                batch_size=5,
                                                class_mode='categorical')

test_set = test_datagen.flow_from_directory(test_path,
                                            target_size=(64,64),
                                            batch_size=5,
                                            class_mode='categorical')

classes = ('Verticalbox', 'Verticalbar', 'Scatter', 'Pie',
           'Line', 'Horizontalbox', 'Horizontalbar')
```

Slika 3.1. *Prilagodba slika za treniranje mreže*

Nakon što su slike prilagođene veličini od 64x64 px, slijedeći korak koji mreža radi je da uzima nasumično slike iz danog ICDAR 2019 podatkovnog skupa koje su joj potrebne za trening, odnosno da bi na primjerima slika iz podatkovnog skupa kasnije naučila razliku između pojedinih klasa, a kako se to radi prikazano je na slici 3.2.

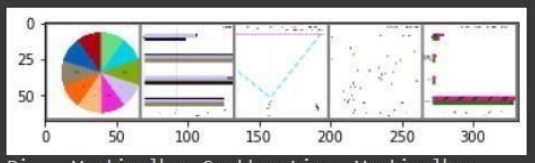
```python
import matplotlib.pyplot as plt
import numpy as np

# funkcija za prikaz slike


def imshow(img):
    img = img / 2 + 0.5     # !(normalizacija)
    npimg = img.numpy()
    plt.imshow(np.transpose(npimg, (1, 2, 0)))
    plt.show()


# uzimamo random slike za trening
dataiter = iter(trainloader)
images, labels = dataiter.next()

# funkcija za prikaz slika
imshow(torchvision.utils.make_grid(images))
# funkcija za prikaz teksta (labelsa)
print(' '.join(f'{classes[labels[j]]:5s}' for j in range(batch_size)))
```




```
Pie   Verticalbar Scatter Line  Verticalbar
```

***Slika 3.2**. Funkcija za prikaz i uzimanje nasumičnih slika iz podatkovnog skupa*

3.2. Učinkovitost CNN za ICDAR 2019

Za ovu testnu CNN mrežu koristio se prethodno navedeni ICDAR 2019 podatkovni skup. Kako bi učenje modela bilo valjano slike su se morale obraditi i prilagoditi ulaznom sloju neuronske mreže. Za ulaznu rezoluciju slika postavljeno je da nam je ona 64x64 px (piksela). Nakon što se izvršila prilagodba rezolucije slike, drugi minimalni uvjet prilagodbe slike je prilagodba boja. Kako se orginalne slike sastoje od RGB koeficjenata koji se nalaze u rasponu od 0-255 to bi uvelike stvaralo problem za treniranje modela. Orginalne slike potrebno je skalirati na vrijednosti da se nalaze u intervalu vrijednosti od 0 do 1 što se radi na način tako da se podijeli vrijednost svakog piksela s gornjom granicom RGB koeficjenata koja iznosi 255 (1/255.). Kada su ispunjena ova dva minimalna uvjeta za obradu slika, sve je spremno za učenje modela.

3.2.1. Implementacija i evaluacija CNN

Na slici 3.3. prikazana je implementacija korištene CNN mreže. U prvom sloju mreže *conv.1* vidi se da postoji 3 ulazna kanala za odabranu sliku te 6 izlaznih kanala, odnosno kanala koji su stvoreni konvolucijom. Broj 5 predstavlja veličinu konvolucijskog kernela. Nadalje, funkcijom *MaxPool2d* smanjuju se takozvane prostorne dimenzije volumena izlazne slike. U drugom sloju *conv.2* postoji 6 ulaznih kanala te 16 izlaznih kanala, samim time mreža postaje sve dublja i njena analiza je sve točnija. Primjećuje se da je razlika ulaznih kanala za prvi i drugi sloj različita što govori da veličina izlaznog kanala prvog sloja postaje ulazni kanal drugog sloja. Funckijom *fc1* određeno je da mreža koristi linearni sloj koji se sastoji od 16 kanala što opet govori da je to broj izlaznih kanala drugog sloja, a širina i visina kernela je 13. Funkcija *softmax* se postavlja nakon obrade prvog i drugog sloja iz razloga kako bi se osiguralo da dobiveni rezultat bude u intervalu od [0,1] te njegova suma da bude 1. U drugom djelu koda definira se funkcija "naprijed" (*engl. forward),* ona služi tako da definira izlaz neuronske mreže, konkretno, poziva se kada se neuronska mreža primjenjuje na ulaznu varijablu. Vidljivo je da je ulazna varijabla za ovu mrežu definirana kao "x".

```
[ ]
    class Net(nn.Module):
        def __init__(self):
            super().__init__()
            self.conv1 = nn.Conv2d(3, 6, 5)
            self.pool = nn.MaxPool2d(2, 2)
            self.conv2 = nn.Conv2d(6, 16, 5)
            self.fc1 = nn.Linear(16 * 13 * 13, 120)
            self.fc2 = nn.Linear(120, 84)
            self.fc3 = nn.Linear(84, 7)
            self.softmax = nn.Softmax(dim=1)

        def forward(self, x):
            x = self.pool(F.relu(self.conv1(x)))
            x = self.pool(F.relu(self.conv2(x)))
            x = torch.flatten(x, 1) # spljostivanje (flattening) svega osima batcha
            x = F.relu(self.fc1(x))
            x = F.relu(self.fc2(x))
            x = self.fc3(x)
            x = self.softmax(x)
            return x


    net = Net()
    model_parameters = filter(lambda p: p.requires_grad, net.parameters())
    params = sum([np.prod(p.size()) for p in model_parameters])
    print(params)

    print('Network Structure : torch.nn.Linear(2,1) :\n',net)
```

Slika 3.3*. Implementacija mreže*

Nakon što mreža prođe kroz oba sloja, konkretno za ovaj primjer dobivaju se rezultati u slijedećem obliku koji je prikazan na slici 3.4.

```
338231
Network Structure : torch.nn.Linear(2,1) :
 Net(
  (conv1): Conv2d(3, 6, kernel_size=(5, 5), stride=(1, 1))
  (pool): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (conv2): Conv2d(6, 16, kernel_size=(5, 5), stride=(1, 1))
  (fc1): Linear(in_features=2704, out_features=120, bias=True)
  (fc2): Linear(in_features=120, out_features=84, bias=True)
  (fc3): Linear(in_features=84, out_features=7, bias=True)
  (softmax): Softmax(dim=1)
)
```

Slika 3.4. *Rezultati nakon prilagođavanja slojeva neuronske mreže*

Gore definirani parametri (*engl. out features*) neuronske mreže koji se mogu naučiti su 7.

Rezultat mreže trenirane na ICDAR 2019 podatkovnom skupu prikazan u tablici 3.1.

Tablica 3.1. Rezultat mreže trenirane na ICDAR 2019 datasetu

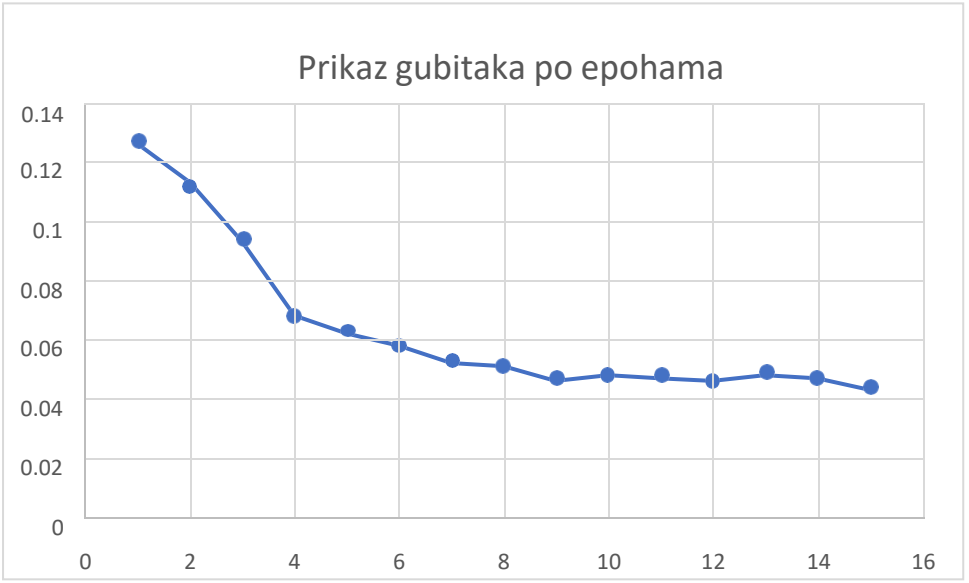
Epoha 1, min loss	Epoha 2, min loss	Epoha 3, min loss	Epoha 4, min loss	Accuracy (Trening)
0,093	0,058	0,046	0,043	56%

Iz tablice se primjećuje da kako se prolazi kroz epohe treniranja da se gubitak (*engl. loss*) smanjuje što znači da mreža smanjuje gubitke prilikom treniranja i da postaje sve stabilnija. Mreža nam daje točnost od 56% što nije idealno no ne može se očekivati puno veća točnost jer se ne radi s vrlo dubokim i zahtjevnim neuronskim mrežama već se koriste najjednostavniji primjer za treniranje modela.

Slika 3.5. prikazuje kako se gubitak smanjuje kroz svaku epohu.

```
[13] Epochs: 100% 5/5 [12:09<00:00, 92.32s/it]
      [1,   131] loss: 0.126
      [1,   262] loss: 0.113
      [1,   393] loss: 0.093
      Epoch: 01 | Epoch Time: 8m 42s
      [2,   131] loss: 0.068
      [2,   262] loss: 0.062
      [2,   393] loss: 0.058
      Epoch: 02 | Epoch Time: 0m 51s
      [3,   131] loss: 0.052
      [3,   262] loss: 0.051
      [3,   393] loss: 0.046
      Epoch: 03 | Epoch Time: 0m 51s
      [4,   131] loss: 0.048
      [4,   262] loss: 0.047
      [4,   393] loss: 0.046
      Epoch: 04 | Epoch Time: 0m 51s
      [5,   131] loss: 0.048
      [5,   262] loss: 0.047
      [5,   393] loss: 0.043
      Epoch: 05 | Epoch Time: 0m 51s
      Finished Training
```

Slika 3.5. *Smanjivanje gubitaka kroz svaku epohu*



Slika 3.6. *Grafički prikaz smanjivanja gubitaka kroz epohe*

Dok slika 3.7. prikazuje kolika je točnost mreže za naš podatkovni skup

```
[19] correct = 0
     total = 0
     # gradient postavljen na 0
     with torch.no_grad():
         for data in testloader:
             images, labels = data
             # funkcija koja izracunava output
             outputs = net(images)
             # klasa s najveci postotkom uzimamo kao "tocku gledista"
             _, predicted = torch.max(outputs.data, 1)
             total += labels.size(0)
             correct += (predicted == labels).sum().item()

     print(f'Accuracy of the network on the test images: {100 * correct // total} %')

     Accuracy of the network on the test images: 56 %
```

Slika 3.7. *Točnost mreže za odabrani podatkovni skup*

U tablici 3.2. vidi se kolika je točnost za svaku pojedinu klasu nakon završetka treniranja modela:

Tablica 3.2. Točnost modela za svaku klasu

Vertical box	Vertical bar	Scatter	Pie	Line	Horizontal box	Horizontal bar
3,1%	93,8%	93,8%	100%	37,5%	40,6%	40,6%

Prikazani rezultati za svaku klasu nisu isti te kada bi se opet trenirala ista mreža ne bi se dobili isti rezultati, razlog tomu je što težine koje se inicijaliziraju nisu uvijek iste već se pri treniranju mreže nasumično uzimaju te zato rezultati variraju.

Na slici 3.8. i 3.9. prikazano je kako se za istu mrežu dobivaju različiti rezultati točnosti za pojedinu klasu

```
correct_pred = {classname: 0 for classname in classes}
total_pred = {classname: 0 for classname in classes}


with torch.no_grad():
    for data in testloader:
        images, labels = data
        outputs = net(images)
        _, predictions = torch.max(outputs, 1)
        # prikupljanje tocnih predikcija klasa
        for label, prediction in zip(labels, predictions):
            if label == prediction:
                correct_pred[classes[label]] += 1
            total_pred[classes[label]] += 1


# funkcija koja printa tocnost mreze za odredene klase slika
for classname, correct_count in correct_pred.items():
    accuracy = 100 * float(correct_count) / total_pred[classname]
    print(f'Accuracy for class: {classname:5s} is {accuracy:.1f} %')
```

```
Accuracy for class: Verticalbox is 3.1 %
Accuracy for class: Verticalbar is 93.8 %
Accuracy for class: Scatter is 93.8 %
Accuracy for class: Pie   is 100.0 %
Accuracy for class: Line  is 37.5 %
Accuracy for class: Horizontalbox is 40.6 %
Accuracy for class: Horizontalbar is 40.6 %
```

Slika 3.8. *Rezultat točnosti prilikom prvog treniranja*

```
correct_pred = {classname: 0 for classname in classes}
total_pred = {classname: 0 for classname in classes}


with torch.no_grad():
    for data in testloader:
        images, labels = data
        outputs = net(images)
        _, predictions = torch.max(outputs, 1)
        # prikupljanje tocnih predikcija klasa
        for label, prediction in zip(labels, predictions):
            if label == prediction:
                correct_pred[classes[label]] += 1
            total_pred[classes[label]] += 1


# funkcija koja printa tocnost mreze za odredene klase slika
for classname, correct_count in correct_pred.items():
    accuracy = 100 * float(correct_count) / total_pred[classname]
    print(f'Accuracy for class: {classname:5s} is {accuracy:.1f} %')
```

```
Accuracy for class: Verticalbox is 84.4 %
Accuracy for class: Verticalbar is 12.5 %
Accuracy for class: Scatter is 9.4 %
Accuracy for class: Pie   is 100.0 %
Accuracy for class: Line  is 100.0 %
Accuracy for class: Horizontalbox is 75.0 %
Accuracy for class: Horizontalbar is 28.1 %
```

Slika 3.9. *Rezultat točnosti klase prilikom drugog treniranja*

3.3. Učinkovitost VGG11 mreže za ICDAR 2019

VGG11 je oblik konvolucijske neuronske mreže koji se koristi za klasifikaciju slika koje su većih dimenzija. Samim brojem u imenu mreže može se zaključiti da se mreža sastoji od 11 skrivenih slojeva. Model mreže koja se trenira postiže preciznost od 99,11% na skupu podataka ICDAR 2019 za koji je uzeto da je 2100 slika trening dataset dok je 224 slike testing dataset uz to da su slike podijeljene na 7 klasa.

3.3.1. Implementacija i evalucija VGG11

Baza svih VGG mreža je poprilično ista, jedina razlika koja se pojavljuje između mreža koje imaju različit broj slojeva je funkcija *self.features = features* kojoj se proslijeđuje argument nakon što se konstruira VGG mreža. Kao što je prikazano na slici 3.10. u VGG mrežama koriste se *AdaptiveAvgPool2d* slojevi za razliku od CNN-a gdje se koriste *MaxPool* slojevi. U adaptivnim slojevima (*engl. Adaptive Layers)* navodi se željena izlazna veličina sloja udruživanja umjesto veličine filtra koju koristi sloj. U funkciji koja je predstavljena s *nn.Linear(512 * 7 * 7, 4096)* vidljivo je da završni konvolucijski sloj VGG mreže ima 512 filtara što vrijedi za sve VGG mreže što uvelike pomaže jer klasifikator uvijek ostaje isti iz razloga što je veličina za svaku konfiguraciju 7 x 7. Prednost korištenja adaptivnih slojeva je to što omogućuje primjenu modela na slike različitih veličina no postoji i uvjet, a on je da minimalna slika za VGG iznosi 32 x 32.

```python
class VGG(nn.Module):
    def __init__(self, features, output_dim):
        super().__init__()

        self.features = features

        self.avgpool = nn.AdaptiveAvgPool2d(7)

        self.classifier = nn.Sequential(
            nn.Linear(512 * 7 * 7, 4096),
            nn.ReLU(inplace=True),
            nn.Dropout(0.5),
            nn.Linear(4096, 4096),
            nn.ReLU(inplace=True),
            nn.Dropout(0.5),
            nn.Linear(4096, output_dim),
        )

    def forward(self, x):
        x = self.features(x)
        x = self.avgpool(x)
        h = x.view(x.shape[0], -1)
        x = self.classifier(h)
        return x, h
```

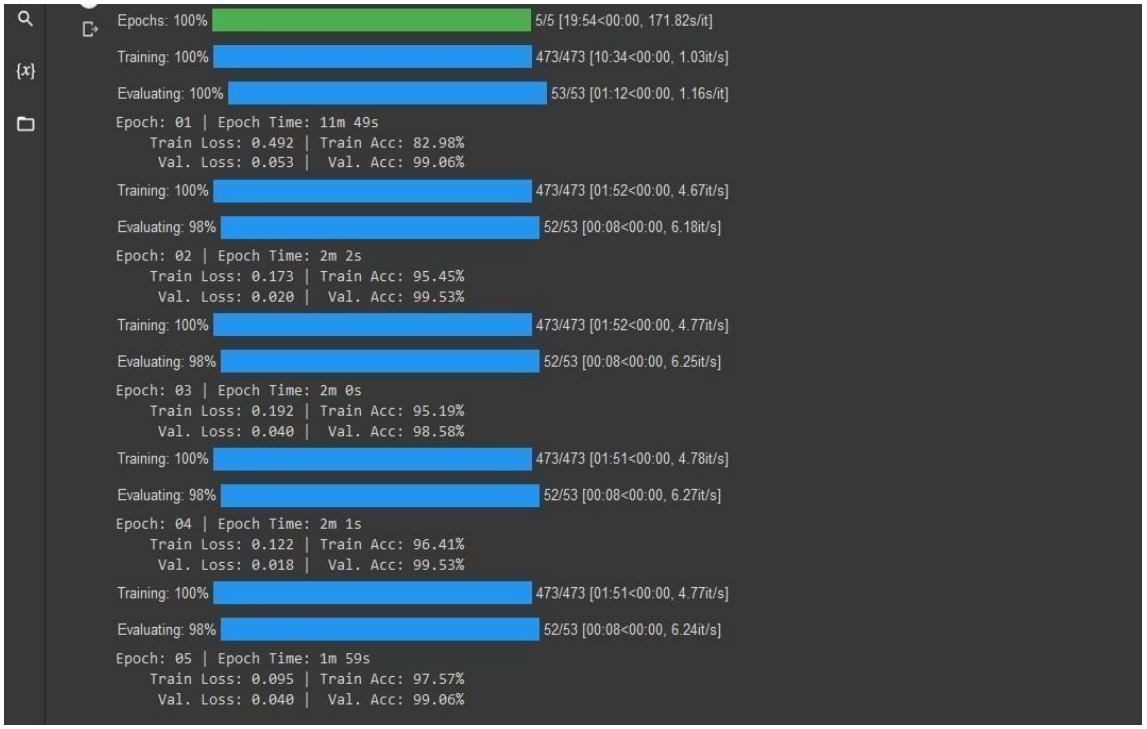
Slika 3.10. *Konfiguracija VGG mreže*

Na slici 3.11. prikazan je rezultat mreže u kojem se vidi kako se broj filtara povećava kao višekratnik izlaznog kanala prvog sloja mreže koji iznosi 64 i raste sve do 512 što je ujedno i karakteristični broj filtara za VGG mrežu.

```
Sequential(
  (0): Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (2): ReLU(inplace=True)
  (3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (4): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (5): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (6): ReLU(inplace=True)
  (7): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (8): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (9): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (10): ReLU(inplace=True)
  (11): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (12): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (13): ReLU(inplace=True)
  (14): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (15): Conv2d(256, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (16): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (17): ReLU(inplace=True)
  (18): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (19): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (20): ReLU(inplace=True)
  (21): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (22): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (23): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (24): ReLU(inplace=True)
  (25): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (26): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (27): ReLU(inplace=True)
  (28): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
)
```

Slika 3.11. *Output VGG11 mreže za postavljeni broj filtara*

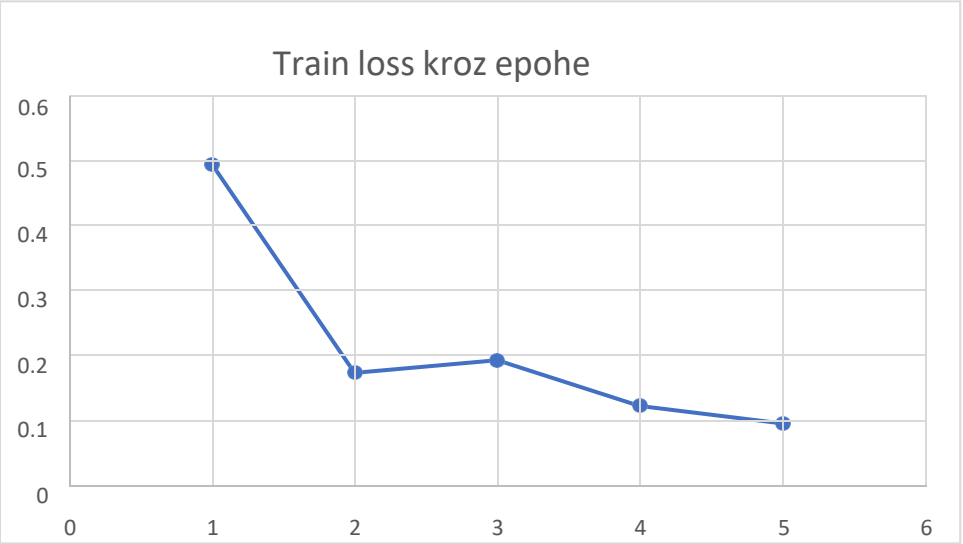
Prilikom treniranja ovakvog modela, računa se preciznost *(engl. Accuracy)*, funkcija gubitaka *(engl. loss)* korištenjem kategoričke unakrsne entropije (*engl. Categorical crossentropy*). Koristi se optimizacijski algoritam Adam, koji je zapravo poboljšana inačica stohastičkog gradijentnog spusta *(engl. Stohastic Gradient Descent – SGD).* Model treniran na način da ga se provlači kroz 5 epoha prilikom kojih se za svaku iteraciju propagira cijeli podatkovni skup u obliku male serije *(engl. mini batch).* Parametrom veličina serije (*engl. batch size)* određuje se veličina serije koja se propagira kroz mrežu te sama serija ovisi o veličini podatkovnog skupa koji se koristi za učenje.



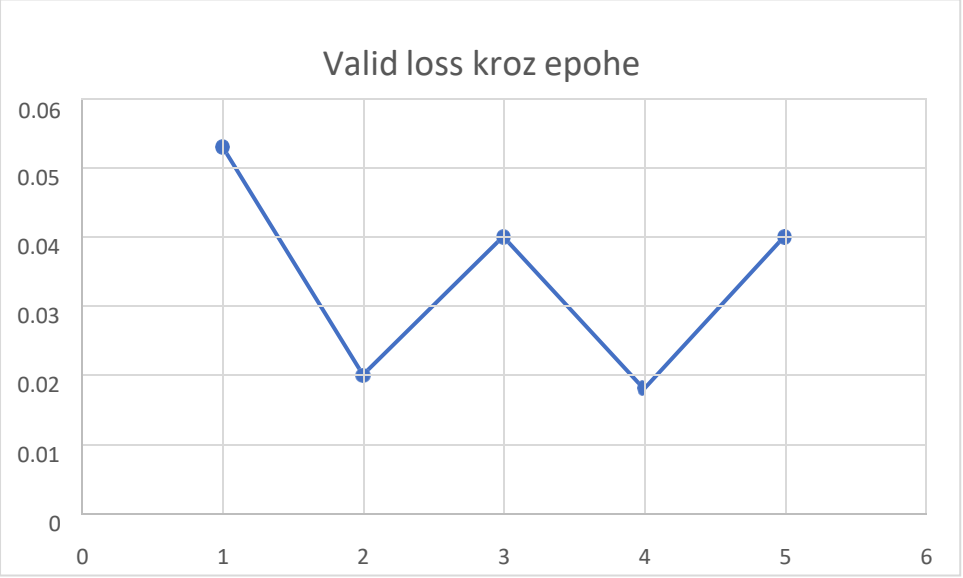
Slika 3.12. *Detaljan prikaz gubitaka kroz epohe*

Na slici 3.12. prikazani su gubitci unutar svake epohe treniranja mreže.

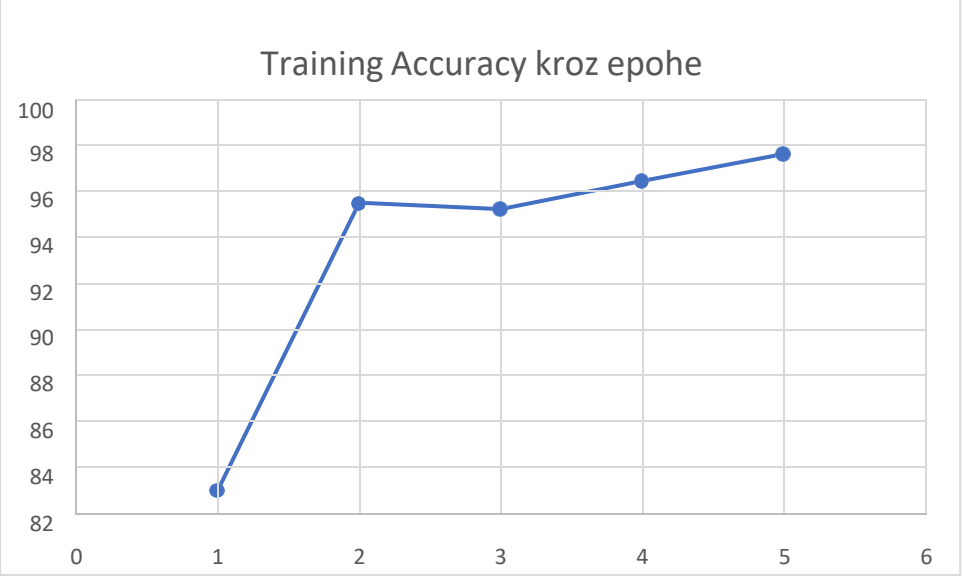
U daljnjem razmatranju prikazani su grafički prikazi za svaki segment treniranja mreže, Train Loss, Valid Loss, Train Accuracy, Valid Accuracy.



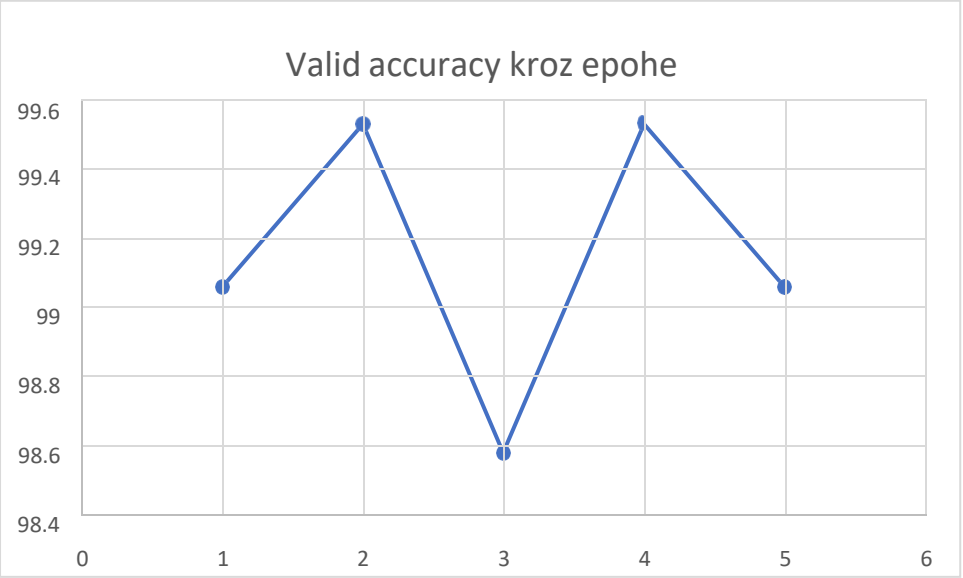
Slika 3.13. *Grafički prikaz Train lossa kroz epohe*



Slika 3.14. *Grafički prikaz Valid lossa kroz epohe*



Slika 3.15. *Grafički prikaz Training Accuracy kroz epohe*



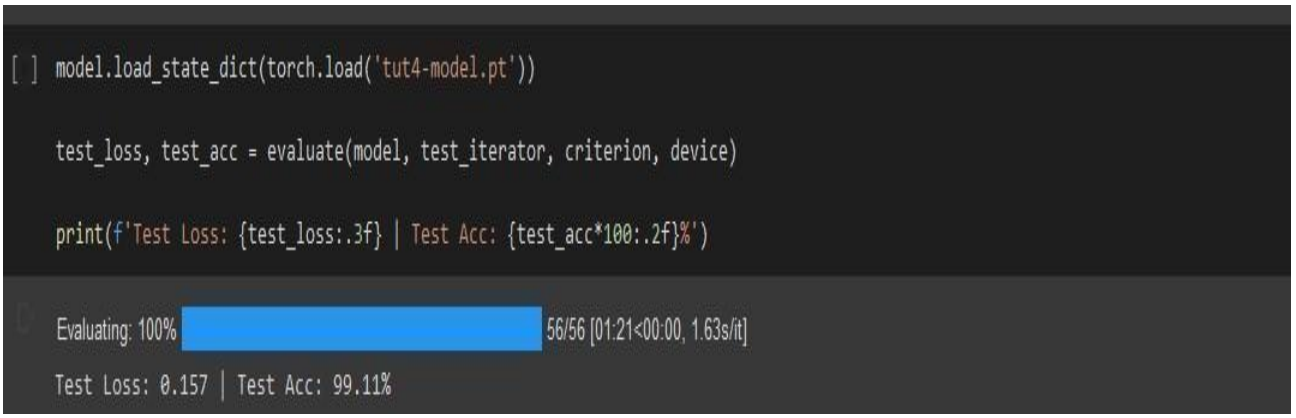
Slika 3.16. *Grafički prikaz Valid Accuracy kroz epohe*

3.3 Tablica modela VGG11

Train loss	Valid loss	Train Accuracy	Valid Accuracy
0,095	0,040	97,57%	99,06%

U tablici 3.3. prikazan je rezultat učenja VGG11 mreže na skupu podataka od 2100 slika. Valjana točnost (*engl. Valid Accuracy*) je veća nego točnost treninga (*engl. Train Accuracy*)

što može biti razlog da je testni skup (*engl. testing data*) vrlo jednostavan za identificiranje od strane mreže pa je samim time i točnost veća.



3.17. *Prikaz testnih rezultata VGG11 mreže*

3.4. Tablica modela VGG11 za testni skup

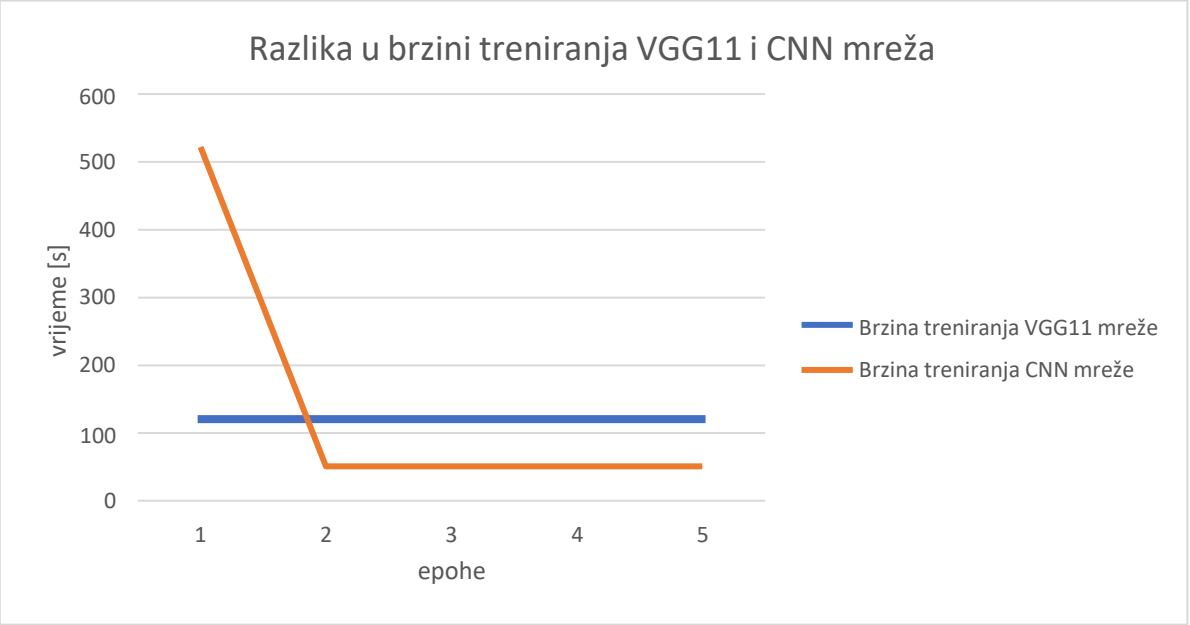
Test loss	Test Accuracy
0,157	99,11%

U tablici 3.4. prikazan je rezultat učenja VGG11 mreže na testnom skupu podataka od 224 slike.

Točnost testa (*engl. Test Accuracy*) iznosi 99,11% što znači da mreža ima vrlo visoku točnost na zadanom podatkovnom skupu.

3.4. Usporedba CNN i VGG11 neuronske mreže

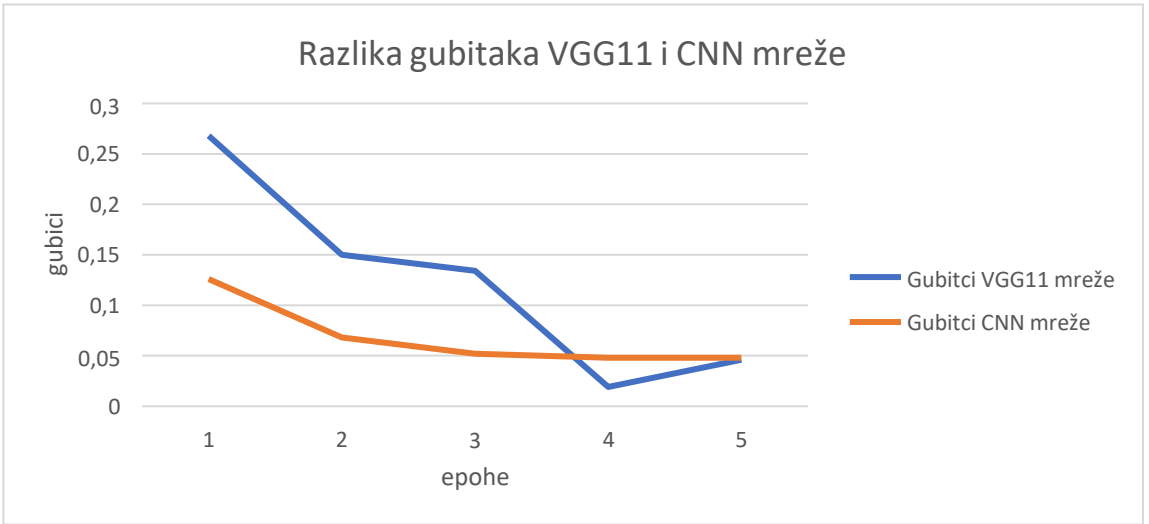
Po rezultatima je vidljivo da VGG11 neuronska mreža daje bolju pouzdanost u treniranju ICDAR 2019 podatkovnog skupa što je i očekivano. VGG11 neuronska mreža sastoji se od 11 slojeva te je samim time puno preciznija u treniranju što se može vidjeti po izlaznim rezultatima. Kako se po rezultatima može vidjeti razlika što se tiče točnosti i gubitaka tako se može vidjeti i po vremenu treniranja kroz epohe. Na slici 3.18. prikazana je razlika duljine treniranja CNN i VGG11 mreže izražena u sekundama.



3.18. *Razlika u brzini treniranja VGG11 i CNN mreža*

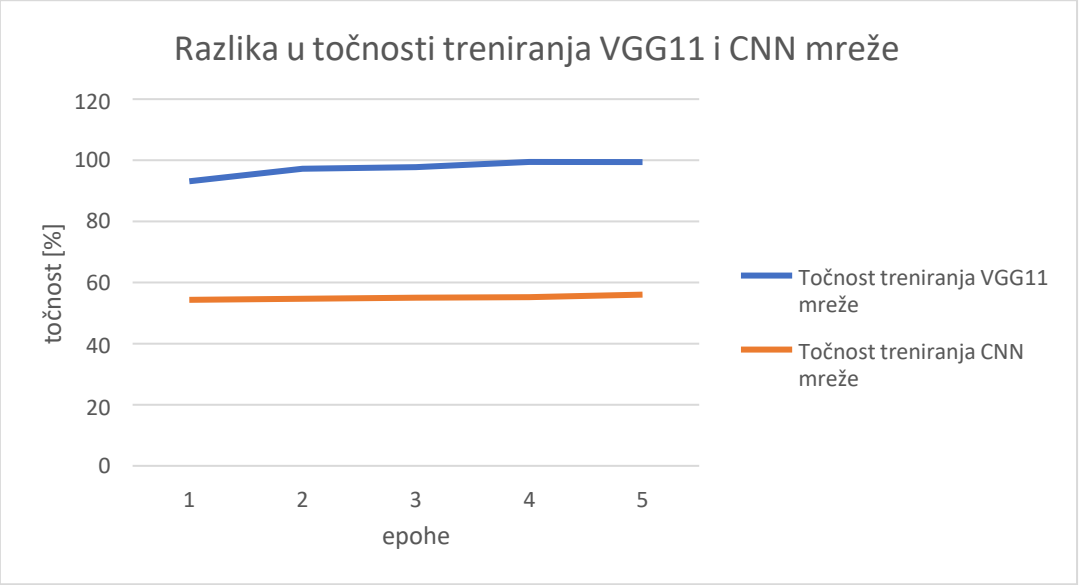
Sa slike 3.18. vidljivo je da jedino u prvoj epohi je CNN mreža sporija u odnosu na VGG11 mrežu. Kroz preostale 4 epohe primjećuje se da VGG11 mreža ima veće vrijeme, odnosno brzinu treniranja. Razlog tomu, kao što je prije navedeno, je taj što je VGG11 vrlo duboka neuronska mreža za razliku od CNN koja se sastoji od svega par slojeva.

Također, još jedan vrlo važan parametar kod uspoređivanja mreža je parametar gubitaka (*engl. loss*). Na slici 3.19. grafički je prikazano kakve gubitke ima pojedina mreža koja se uspoređuje.



3.19. *Razlika gubitaka VGG11 i CNN mreže*

Na slici 3.19. vidljivo je da su tokom prve tri epohe gubitci veći kod VGG11 mreže nego kod CNN mreže što nam govori da nam model VGG11 mreže sadrži više grešaka (*engl. error*) što znači da se tokom prve tri iteracije model mreže loše ponaša. U četvrtoj i petoj epohi treniranja gubici u VGG11 mreži su manji što znači da se mreža bolje ponaša nakon svake iteracije optimizacije dok model CNN mreže se kroz svaku iteraciju slično ponaša.



3.20. *Razlika u točnosti treniranja VGG11 i CNN mreže*

Na slici 3.20. prikazana je razlika u točnosti treniranja VGG11 i CNN mreže u odnosu na izlazne rezultate. VGG11 mreža je puno točnija te nakon pete epohe njena točnost iznosi 99,37% dok točnost treniranja CNN nakon pete epohe iznosi 56%.

4. ZAKLJUČAK

Iz dobivenih rezultata zaključuje se da korištenje dubokih umjetnih neuronskih mreža je učinkovitije i pouzdanije od korištenja umjetnih neuronskih mreža s manje slojeva, odnosno plićih neuronskih mreža. Zadatak ovog završnog rada bila je učinkovitost različitih vrsta neuronskih mreža u klasificiranju grafičkih prikaza no neuronske mreže ne moraju nužno klasificirati grafove, mogu se koristiti u klasificiranju drugih različitih objekata.

Iako su rezultati kod CNN relativno dobri, preporuča se korištenje VGG11 mreže kao trenutno rješenje jer nam mreža daje bolju učinkovitost i relativno brzo treniranje pomoću Google Colab platforme. Korištena je Google Colab platforma preko koje je i sa slabijim računalima omogućeno treniranje vrlo zahtjevnih mreža na način da platforma nudi direktno spajanje na njihove servere koji omugućavaju olakšano treniranje mreže. Naravno, moguće je bilo korištenje i još dubljih mreža, primjerice VGG16/VGG19, ali Google Colab platforma ne omogućava besplatno korištenje za tako složene mreže.

Dobrim korištenjem računala i snalaženjem u literaturama vezanim uz ovakav problem omogućuje se svakome da napravi kod za bilo koju vrstu umjetnih neuronskih mreža te da radi klasifikaciju objekata po željenim klasama, bilo da objekt bude slika, video, tekst i slično.

LITERATURA

1) Muzammil Khan and Sarwar Shah Khan, "Data and information visualization methods and interactive mechanisms: A survey", In International Journal of Computer Applications, 2011. , preuzeto 7. rujan 2022.
2) 2. Geral Franciscani et al., "An annotation process for data visualization techniques", In Proceedings of International Conference on Data Analytics, 2014. , preuzeto 7. rujan 2022.
3) Weihua Huang, Siqi Zong and Chew Lim Tan, "Chart image classification using multipleinstance learning", In Applications of Computer Vision. WACV'07 IEEE Workshop on. IEEE, str. 27-27, 2007. , preuzeto 7. rujan 2022.
4) V. Shiv Naga Prasad, Behjat Siddiquie, Jennifer Golbeck and Larry S. Davis, "Classifying computer generated charts", In International Workshop on Content-Based Multimedia Indexing. CBMI'07. IEEE 8592, 2007. , preuzeto 7. rujan 2022.
5) Navneet Dalal and Bill Triggs, "Histograms of Oriented Gradients for Human Detection", In IEEE Computer Society Conference on Computer Vision and Pattern Recognition 2005. CVPR. IEEE, str. 886-893, lipanj 2005. , preuzeto 7. rujan 2022.
6) David G. Lowe, "Object Recognition from Local Scale-Invariant Features", The proceedings of the seventh IEEE international conference on Computer vision, vol. 2, str. 1150-1157, 1999. , preuzeto 7. rujan 2022.
7) Binbin Tang et al., "Deepchart: Combining deep convolutional networks and deep belief networks in chart classification", Signal Processing 124. ACM, str. 156-161, srpanj 2016.
8) Daekyoung Jung et al., "Chartsense: Interactive data extraction from chart images", Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems. ACM, str. 6706-6717, svibanj 2017. , preuzeto 7. rujan 2022.
9) Yann LeCun, Léon Bottou, Yoshua Bengio and Patrick Haffner, "Gradient-based learning applied to document recognition", Proceedings of the IEEE 86 No. 11. IEE, str. 2278-2324, 1999. , preuzeto 7. rujan 2022.
10) Alex Krizhevsky, Ilya Sutskever and Geoffrey E. Hinton, "Imagenet classification with deep convolutional neural networks", In Advances in Neural Information Processing Systems, str. 1097-1105, 2012. , preuzeto 7. rujan 2022.
11) Christian Szegedy et al., "Going deeper with convolutions", In Proceedings of the IEEE

Conference on Computer Vision and Pattern Recognition (CVPR '15), str. 1-9, lipanj 2015.

12) Alfredo Canziani, Adam Paszke and Eugenio Culurciello, "An Analysis of Deep Neural Network Models for Practical Applications", svibanj 2016. , preuzeto 7. rujan 2022.
13) Keiron O'Shea, Ryan Nash, "An Introduction to Convolutional Neural Networks", 2015. , preuzeto 10. srpanj 2022.
14) Alex Sherstinsky, "Fundamentals of Recurrent Neural Network (RNN) and Long Short – Term Memory (LSTM) Network", 2020. , preuzeto 1. rujan 2022.
15) Jianxin Wu, "Introduction to Convolutional Neural Networks", 2017. , preuzeto 10. srpanj 2022.
16) B.Novaković, D.Majetić,M.Široki: Umjetne neuronske mreže, Zagreb,1998., preuzeto 28. srpanj 2022.
17) Joško Markučič, "Generalizacijsko svojstvo unaprijedne neurosnke mreže", 2016. , preuzeto 28. srpanj 2022. dostupno na:
18) CS231n Convolutional Neural Networks for Visual Recognition, preuzeto 12. rujan 2022. dostupno na: https://cs231n.github.io/neural-networks-1/
19) Imad Dabbura, "Gradient Descent Algorithm and Its Variants", 2017., preuzeto 12. rujan 2022. dostupno na: https://towardsdatascience.com/gradient-descent-algorithm-and-its-variants-10f652806a3
20) "A Brief History Of Neural Network Architectures", TOPBOTS, 2017., preuzeto 12. rujan 2022. dostupno na: https://www.topbots.com/a-brief-history-of-neural-network-architectures/
21) Qiwei Lin, Ruixun Zhang, Xiuli Shao, "CNN and RNN mixed models for image classification", 2019. , preuzeto 1. srpanj 2022.
22) Jiang Wang, Yi Yang, Junhua Mao, Zhiheng Huang, Chang Huang, Wei Xu, "CNN-RNN: A Unified Framework for Multi-Label Image Classification", 2016. , preuzeto 7. srpanj 2022.
23) Monica Bianchini, Franco Scarselli, "On the Complexity of Neural Network Classifiers: A comparsion Between Shallow and Deep Architectures", 2014. , preuzeto 7. rujan 2022. , dostupno na:
24) Simon Haykin, "Neural Networks and Learning Machines Third Edition", 2009. , McMaster University, Canada , preuzeto 20. kolovoz 2022.

25) Fangyuan Lei, Xun Liu, Qingyun Dai, Bingo Wing-Kuen Ling, “Shallow convolutional neural network for image classification“, 2019. , preuzeto 10. srpanj 2022. , dostupno na: https://link.springer.com/article/10.1007/s42452-019-1903-4

26) Suzana Dumančić, “Neuronske mreže“, 2014. , preuzeto 28. lipanj 2022.

27) Bojana Dalbelo Bašić, Marko Čupić, Jan Šnajder, “Umjetne neuronske mreže“, 2008. preuzeto 8. lipanj 2022

SAŽETAK

Duboke umjetne neuronske mreže danas su najpouzdaniji, najbrži, te najefektivniji način za rješavanje problema vezanog uz prepoznavanje slika. Korištenjem dubokih umjetnih neuronskih mreža omogućava se učenje mreže na velikoj količini podataka čije rezultate kasnije možemo upotrijebiti u analizi istih (npr. prepoznavanje lica osoba na aerodromima).

U ovom radu opisane su osnovne stvari potrebne za razumijevanje umjetnih neuronskih mreža te je opisano koje su mreže pouzdanije za jednak dataset. Na kraju je pokazano da je VGG11 mreža postigla najbolje rezultate i da je u usporedbi s ostalima njeno korištenje najbolje pri klasificiranju grafičkih prikaza.

Ključne riječi: analiza podataka, prepoznavanje slika, umjetne neuronske mreže, VGG11

ABSTRACT

Neural Networks Efficiency in Chart Image Classification

Deep artificial neural networks are today the most reliable, fastest and most effective way to solve the problem of image recognition. By using deep artificial neural network on large amount of data, the results of which can later be used in the analysis of the same (e.g. recognizing a person's face at airports).

This paper describes the basic things needed to understand artificial neural networks and describes which networks are more reliable for the same dataset. In the end, it was shown that the VGG11 network achieved the best results and that, compared to the others, its use is the best for graphic classifying representation.

Keywords: artificial neural networks, data analysis, image recognition, VGG11

ŽIVOTOPIS

David Kovačević rođen je u Zagrebu 10.05.2000. Živi u mjestu Popovača koje se nalazi u Sisačko – moslavačkoj županiji. Pohađao je osnovnu školu „Osnovna škola Popovača“ u Popovači pa opću gimnaziju „Srednja škola Tina Ujevića“ u Kutini. 2019. godine upisuje se na „Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek“ (FERIT). Nakon prve godine opredijelio se na smjer elektroenergetika.