Primjena neuronskih mreža za otkrivanje napada

Ćosić, Antonio

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Antonio Ćosić

Primjena neuronskih mreža za otkrivanje napada

DIPLOMSKI RAD

Varaždin, 2020.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Antonio Ćosić

JMBAG: 0036481844

Studij: Informacijsko i programsko inženjerstvo

Primjena neuronskih mreža za otkrivanje napada

DIPLOMSKI RAD

Mentor:

Doc. dr. sc. Ivković Nikola

Varaždin, studeni 2020.

Antonio Ćosić

Izjava o izvornosti

Izjavljujem da je moj završni/diplomski rad izvorni rezultat mojeg rada te da se u izradi istoga nisam koristio drugim izvorima osim onima koji su u njemu navedeni. Za izradu rada su korištene etički prikladne i prihvatljive metode i tehnike rada.

Autor potvrdio prihvaćanjem odredbi u sustavu FOI-radovi

Sažetak

Sigurnost informacijskog sustava je njegovo najvažnije svojstvo kojim se krajnjem korisniku jamči sigurno i kvalitetno korištenje njegovih mogućnost. Kako bi razina kvalitete sigurnosti informacijskog sustava bila na zavidnoj razini potrebno je pratiti i konstantno poboljšavati njegove komponente. Komponente bitne za sigurnost informacijskog sustava su okruženje u kojem se sustav razvija i instalira, složena autorizacija kroz veliki broj uloga s ograničenjima, te infrastruktura koja podržava jako brzo pokretanje sustava iz *backup* datoteka ili prebacivanje uloge na rezervnu okolinu (*eng. fallback*). Prilikom razvoja informacijskog sustava, vodeći se standardnim smjernicama i implementacijama, sigurnost možemo dovesti na zadovoljavajuću razinu. Unatoč kvalitetnoj implementaciji sustava i dalje postoji mogućnost da će potencijalni napadač naći novu ranjivost i napraviti štetu sustavu. Trenutak u kojem će se dogoditi napad, informacijski sustav ne može predvidjeti, ali aktivnosti napadača kroz mrežne slojeve može označiti kao potencijalne napade i poduzeti određene mjere. Kako bi informacijski sustav to bio u mogućnosti napraviti potreban mu je podsustav koji će se brinuti isključivo za otkrivanje takvih aktivnosti (*eng. Intrusion Detection System, IDS*). Razvoj takvog sustava tehnikama učenja neuronskih mreža (*eng. Deep Learning*) u središtu je promatranja ovog rada u kojem su implementirani modeli duboke, konvolucijske i povratne neuronske mreže. Provedena testiranja nad testnim setom podataka pokazala su da je najbolje rezultate za obje klasifikacije, binarnu i višerazrednu, postigla duboka neuronska mreža. U arhitekturi modela duboke neuronske mreže za binarnu klasifikaciju su dva skrivena sloja, a u arhitekturi modela za višerazrednu klasifikaciju su tri.

Ključne riječi: sigurnost, neuronske mreže, duboko uče

Sadržaj

1. Uvod

Izgraditi kvalitetan informacijski sustav je izazov za sve sudionike procesa razvoja i isporuke sustava klijentu. Postavlja se pitanje koja su to svojstva koja oblikuju razinu kvalitete nekog informacijskog sustava. Odgovor na to pitanje nije jednostavan jer se razvojem tehnologije povećava i broj tih svojstava. No međutim neka svojstva ipak treba izdvojiti kao najvažnija i u toku razvoja potrebno je obratiti pažnju na njih. Koliko je neko svojstvo bitno ili ne najbolje se vidi pri korisnikovom korištenju sustava. Korisniku je bitno intuitivno snalaženje u aplikaciji i vrlo brzo korištenje željenih mogućnosti pa se može reći da je oblikovanje korisničkog iskustva jedno bitno svojstvo. Osim toga korisnik bi htio da prilikom korištenja mogućnosti što manje čeka na odziv i izvršavanje određenih akcija što također stavlja naglasak na optimizaciju sloja obrade podataka. Moglo bi se tu još bitnih svojstava nabrojiti, ali jedno svojstvo se ističe od svih ostalih, a to je sigurnost aplikacije i sustava kojeg koristi korisnik. Sigurnost kao svojstvo se najviše ističe u sustavima koji čuvaju povjerljive podatke. Razviti kvalitetan sigurnosni sustav je izazovan i dugotrajan proces koji zahtjeva dubinsku analizu i obradu podataka te programsku interpretaciju tih podataka kroz algoritme koji na izlazu imaju varijable korisne za daljnje akcije. Završetkom procesa bi se dobio sustav za otkrivanje napada (*eng. Intrusion Detection System*). Takav sustav je u najviše slučajeva izgrađen na način da predviđanje i klasificiranje napada temelji na uzorcima i iz njih uči tehnikama strojnog učenja ili neuronskim mrežama.

U sklopu ovog rada su implementirane neuronske mreže koje uče iz određenog seta podataka i rade dvije vrste klasifikacija, binarnu i višerazrednu. Binarna klasifikacija (*eng. binary classification*) na izlazu određeni paket klasificira kao opasan (1) ili neopasan (0), a klasifikacija po kategorijama (*eng. multiclass classification*) mrežni promet svrstava u jednu od pet kategorija *Dos*, *Probing*, *R2L*, *U2R* i *Normal*.

U svrhu analize i usporedbe izgrađeni i uspoređeni su rezultati klasifikacije tri različite neuronske mreže: duboke (*eng. Deep Neural Network*), konvolucijske (*eng. Convolutional Neural Network*) i povratne (*eng. Recurrent Neural Network).* Skup podataka za učenje i treniranje je *NSL-KDD* [1]. Radi se o skupu podataka koji ima više verzija i namijenjen je učenju i treniranju neuronskih mreža za otkrivanje napada. Njegova su svojstva (*eng. features*) detaljno opisana i objašnjena.

Za realizaciju i izgradnju neuronskih mreža korišten je razvojni alat *Google Colab* [2]*,* programski jezik *Python* [3] te biblioteke *TensorFlow (Keras API)* [4]*, scikit-learn* [5]. Modeli neuronskih mreža su izgrađeni kroz nekoliko slojeva (ulaz, skriveni slojevi, izlaz) i trenirani sa određenim skupom podataka. Nakon izgradnje i treniranja modela određene arhitekture,

model su testirani sa testnim skupom podataka. Ovisno o modelu testovi na izlazu imaju binarnu ili višerazrednu vrijednost. Rezultati su prikazani i uspoređeni u obliku matrice konfuzije (*eng. confusion matrix*).

Prije same implementacije analizirana je arhitektura i svojstva pojedine neuronske mreže kao i uzorci kojima neuronska mreža uči. Kako bi se razumjele ulazne i izlazne varijable u početnim poglavljima ovog rada objašnjeni su pojmovi informacijske sigurnosti i vrsta napada koje su neuronske mreže klasificirale na izlazu.

Zaključak na kraju rada je izveden iz analize rezultata izlaznih varijabli neuronskih mreža u obliku matrice konfuzije i točnosti na osnovu koje se vidi koja neuronska mreža daje najbolje rezultate.

2. Mrežni napadi

Za analizu rezultata neuronskih mreža bitno je razumjeti pojmove informacijske sigurnosti i mrežnog napada jer su zapravo ta dva pojma motivacija za izgradnju mehanizma za otkrivanje napada koji podiže razinu sigurnosti informacijskog sustava. To se postiže otkrivanjem i klasifikacijom mrežnog prometa te pravovremenim alarmiranjem administratora ako se radi o napadu. Otkrivanje mrežnog napada i pravovremeno alarmiranje svojstva su sustava za otkrivanje napada.

2.1. Informacijska sigurnost

Osigurati tajnost, cjelovitost i dostupnost informacije najveći je izazov informacijskog sustava. Ta tri svojstva su velikom broju definicija zajednička i stavlja se fokus na njih. U slučaju da se neko od tih svojstava kompromitira potrebno je poduzeti određene akcije koje moraju biti predefinirane i spremne od strane informacijskog sustava.
Jedna od definicija je:

„Informacijska sigurnost je stanje povjerljivosti, cjelovitosti i raspoloživosti podatka, koje se postiže primjenom propisanih mjera i standarda informacijske sigurnosti te organizacijskom podrškom za poslove planiranja, provedbe, provjere i dorade mjera i standarda.“ [7]

Može se reći da se navedena definicija sastoji od dva dijela. U središtu prvog dijela definicije je stanje tajnovitosti, cjelovitosti i raspoloživosti podatka, dok drugi dio definicije opisuje način na koji se to stanje postiže. Bitno je naglasiti drugi dio koji poopćuje način implementiranja podsustava i primjenu standarda tokom razvoja informacijskog sustava. To znači da je prva razina sigurnosti programski kod napisan prema sigurnosnim standardima koje propisuju organizacije kroz projekte kojima je u cilju poboljšati sigurnost informacijskog sustava. Jedan takav projekt je „Otvoreni projekt sigurnosti web aplikacija“ (*eng. The Open Web Application Security Project, OWASP).* Međutim programski kod napisan po sigurnosnim standardima još uvijek ne jamči razinu sigurnosti koja je dovoljna da bi se sustav pustio na korištenje korisnicima. Da bi informacijski sustav sigurnost podigao za još jednu razinu potrebno je implementirati podsustav koji će pratiti mrežni promet i detektirati pakete koji bi mogli biti dio napada. Takav sustav se naziva sustav za otkrivanje napada (*eng. Intrusion Detection System, IDS*). On se može izgraditi i implementirati kao zaštita za informacijski sustav na različite načine. Kako bi se ispravno implementirao potrebno je poznavati moguće prijetnje sigurnosti sustava i njihova obilježja.

2.2. Napadi

Za izgradnju kvalitetnog sustava za otkrivanje napada potrebno je poznavati napade i njihova svojstva. Unatoč listi poznatih napada i poznavanju svojstava pojedinog napada nije isključena situacija kada napadač može iskoristiti neku dosada nepoznatu ranjivost i na neki novi način napraviti štetu sustavu. U tom slučaju bi sustav za otkrivanje napada takve maliciozne pakete klasificirao kao nepoznate i blokirao ih. Nakon toga bi informacijski sustav bio obaviješten da se radi o nepoznatoj vrsti paketa za koju je potrebna detaljna analiza kako bi se ubuduće takvim paketima znalo rukovati odnosno treba li ih klasificirati kao napad ili ne. Napadi u ovom radu su klasificirani u četiri glavne kategorije: nedostupnost usluge (*eng. Denial-of-Service, DoS),* udaljeni pristup pravima korisnika sustava *(eng. Remote to Local, R2L),* dobivanje prava super korisnika (*eng. User to root, U2R)* i sondiranje ili ispitivanje *(eng. Probing)*. Za nazive napada unutar pojedine kategorije korištena su kodna imena koja nije potrebno prevoditi.

Nedostupnost usluge (*eng. Denial-of-Service, DoS*)

Napad uskraćivanjem usluge vrsta je napada u kojem napadač preplavljivanjem (*eng. flooding*) zahtjeva prema ciljanom serveru nastoji onemogućiti obradu zahtjeva koji dolaze od korisnika. U tu vrstu napada možemo svrstati napade: *land*, *back*, *pod*, *neptune*, *smurf*, *teardrop*, *udpstorm* itd.

Udaljeni pristup pravima korisnika sustava (*eng. Remote to local, R2L*)

Udaljeni napad korisnika je napad u kojem napadač šalje pakete putem mreže na server kojem ne može pristupiti kako bi izložio ranjivosti servera i iskoristio prava koje lokalni korisnik ima na tom serveru. To su napadi *xlock*, *gost*, *xnsnoop*, *phf*, *sendmail* itd.

Dobivanje prava super korisnika (*eng. User to root, U2R*)

U ovu kategoriju spadaju napadi u kojima napadač počinje ulazi u sustav preko normalnog korisničkog računa kojeg posjeduje. Nakon toga pokušava zloupotrijebiti ranjivosti u sustavu kako bi stekao privilegije super korisnika (*eng. root*). To su napadi *perl, xterm, rootkit* itd.

Sondiranje ili ispitivanje (*eng. Probing*)

Sondiranje je napad u kojem napadač prikuplja informacije o stroju ili mrežnom uređaju u cilju pronalaska slabosti ili ranjivosti koje se kasnije mogu iskoristiti za preuzimanje i kompromitiranje sustava. U ovu kategoriju mogu se svrstati napadi *saint, portsweep, mscan, nmap* itd.

2.3. Sustavi za otkrivanje napada

Visoka razina sigurnosti informacijskog sustava podrazumijeva postojanje sustava za otkrivanje napada. Informacijski sustav bi tu tom slučaju u pozadini imao stalnu provjeru mrežnog prometa i pravovremeno alarmiranje u slučaju da se radi o sumnjivom paketu ili nekoj drugoj anomaliji u mrežnom prometu.

Sustav za otkrivanje upada (IDS) je uređaj ili softverska aplikacija koja nadzire mrežu radi zlonamjernih aktivnosti ili kršenja pravila. [9] Neki su IDS-ovi sposobni odgovoriti na otkriveni napad nakon otkrića. Tada IDS možemo klasificirati kao sustav za sprječavanje napada (*eng. Intrusion Prevention Systems, IPS*).

IDS se najčešće dijeli prema mjestu otkrivanja ili prema metodi koja se koristi za otkrivanje. [10, str. 303] Prema mjestu otkrivanja IDS može biti mrežni (*eng. Network Intrusion Detection System, NIDS*) i sustav koji se pokreće na određenom domaćinu (*eng. Host Intrusion Detection System, HIDS*). Druga podjela dijeli IDS na sustav temeljen na potpisu odnosno određenom uzorku (*eng. Signature-based*) i sustav dizajniran za otkrivanje nepoznatih napada (*eng. Anomaly-based*) kojim se najčešće definira model validacije napada algoritmima strojnog učenja.

Sustavi za otkrivanje napada na mreži (NIDS)

Postavljaju se na stratešku točku ili točku unutar mreže za nadzor prometa prema svim uređajima u mreži i sa svih uređaja u mreži. Izvodi analizu prolaznog prometa na cijeloj podmreži (*eng. subnet*) i klasificira napad prema biblioteci poznatih napada. Kada se napad prepozna upozorenje se može poslati administratoru. Dobra primjena NIDS-a bila bi njegovo instaliranje na podmrežu gdje se nalazi vatrozid (*eng. firewall*) kako bi se vidjelo pokušava li netko provaliti u vatrozid. Idealno bi bilo skenirati sav dolazni i odlazni promet, no to bi moglo stvoriti usko grlo koje bi umanjilo ukupnu brzinu mreže. NIDS može povećati svoju točnost implementacijom neuronske mreže.

Sustav temeljen na domaćinu (HIDS)

Nadzire ulazne i izlazne pakete uređaja i upozorit će korisnika ili administratora ako se otkrije sumnjiva aktivnost. Kako bi se uočile nepravilnosti ili anomalije u radu potrebna je snimka stanja (*eng. snapshot*) postojećih sistemskih datoteka koja se uspoređuje sa prethodnom snimkom. Ako su kritične sistemske datoteke promijenjene ili izbrisane šalje se upozorenje administratoru kako bi istražio i analizirao o čemu se radi.

Sustav baziran na potpisu ili uzorku (*Signature-based*)

Odnosi se na otkrivanje napada traženjem specifičnih uzoraka, poput sljedova bajtova u mrežnom prometu ili poznatih malicioznih instrukcija koje koristi maliciozni programski kod.

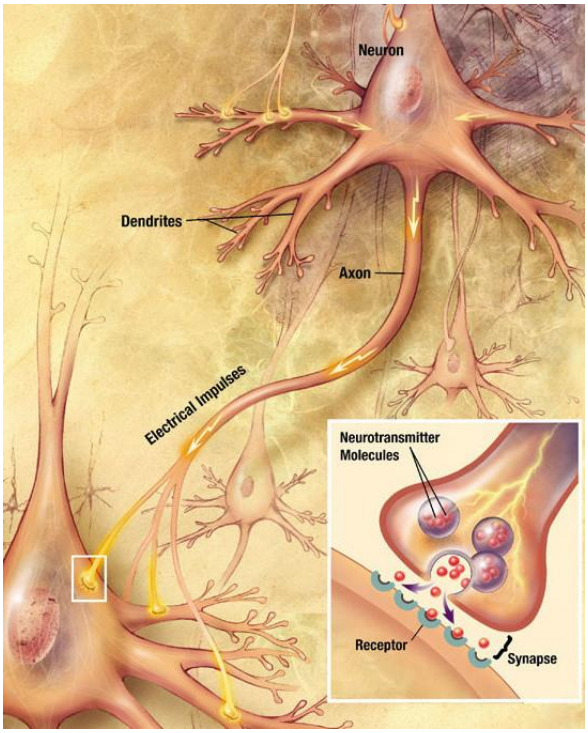
Sustav dizajniran za otkrivanje nepoznatih napada (*Anomaly-based*)

Ova vrsta sustava za otkrivanje napada prvenstveno je uvedena za otkrivanje nepoznatih napada zbog brzog razvoja zlonamjernog programskog rješenja. Osnovni pristup je korištenje strojnog učenja za stvaranje modela koji će raditi predikcije i klasifikaciju. Budući da se modeli mogu trenirati imaju općenitija svojstva u usporedbi s tradicionalnim IDS-ovima temeljenim na potpisima.

Iz prethodne podjele bitno je istaknuti NIDS kao tip jer je implementacija jednog takvog sustava korištenjem različitih neuronskih mreža prikazana u ovom radu. Implementacija NIDS-a neuronskim mrežama poboljšava predviđanje i klasificiranje napada jer se mreže treniraju na velikom skupu podataka.

3. Neuronske mreže

Ljudski mozak je moćno računalo tijela i fascinira način na koji obavlja veliki broj radnji i zadataka istovremeno. Načini i obrasci kojima ljudski mozak funkcionira i naposljetku donosi odluke, motivacija su za istraživanje i razmatranje drugim znanostima. Jedna od znanosti je i računalna znanost kojoj je ljudski mozak zanimljiv zbog brzine učenja i količine podataka koje može spremati i obrađivati. Osnovna jedinica kojom mozak prihvaća, obrađuje i odašilje podatke je neuron. Komunikacija neurona odnosno prenošenje signala sa jednog neurona na drugi odvija se preko sinapse.



Slika 1. Širenje signala između neurona [13]

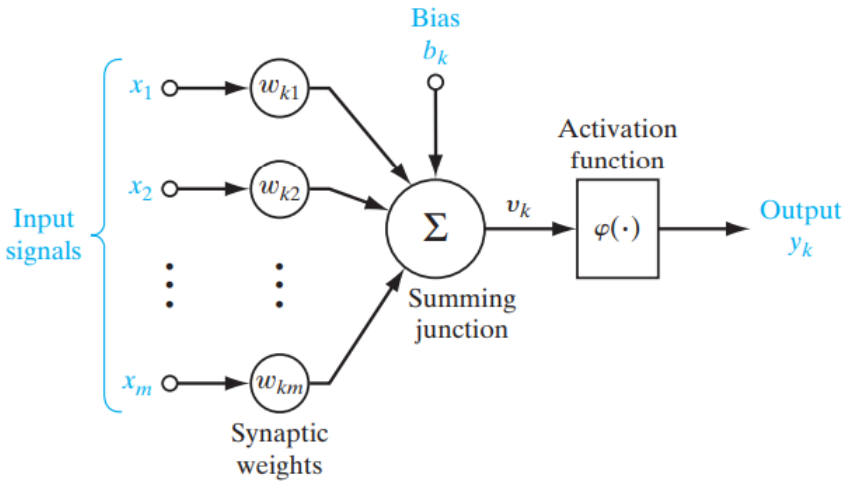
Slika 1 prikazuje način na koji dva neurona međusobno komuniciraju i izmjenjuju signale. U svakodnevnim zadacima i funkcijama mozga sudjeluje različit broj neurona kojih sveukupno prema novijim saznanjima ima oko sto milijardi. Upravo takvo ponašanje, komuniciranje odnosno prijenos signala je fokusu računalne znanosti kojoj je u cilju izgraditi neuronsku mrežu računala koja bi što sličnije oponašala neuronsku mrežu mozga. Takva neuronska mreža je umjetna, ali kada se spominje u kontekstu računalne znanosti dovoljno je reći samo neuronska mreža. Stvarajući umjetne neuronske mreže koje će „učiti“ računalo određenom ponašanju stvaramo umjetnu inteligenciju. Programiranje računala na način da ono poboljšava svoju učinkovitost i performanse na temelju empirijskih podataka nazivamo

strojnim učenjem (*eng. machine learning*), a specijalna grana strojnog učenja koja se bavi tehnikama učenja neuronskih mreža naziva se duboko učenje (*eng. deep learning*). [19]

Prve korake prema razvoju umjetne inteligencije i njenog ozbiljnog shvaćanja učinio je Alan Turing sa svojom teorijom o računanju (*eng. Theory of computation*) u kojoj navodi da računalo bilo koji postupak matematičke dedukcije može simulirati simbolima „0“ ili „1“. To saznanje da računalo može simulirati bilo koji proces formalnog zaključivanja poznatije je kao *Churh-Turing* teza. [11] Kasnije su te teze postale temelj za daljnja istraživanja na području umjetne inteligencije.

Umjetna neuronska mreža je pojednostavljeni model prirodne neuronske mreže. Sastoji se od međusobno povezanih jedinica za obradu koje nazivamo neuronima. S obzirom da neuron u prirodnoj neuronskoj mreži prima ulazne podatke, obrađuje ih i na izlazu vraća podatke slično ponašanje ima i neuron u umjetnoj neuronskoj mreži, ali je potrebno detaljnije objasniti na koji način to funkcionira. To ponašanje se najbolje opisuje kroz strukturu modela neurona koji je prikazan na slici 2.

Model neurona



Slika 2. Model neurona (Prema: Simon Haykin, 2009)

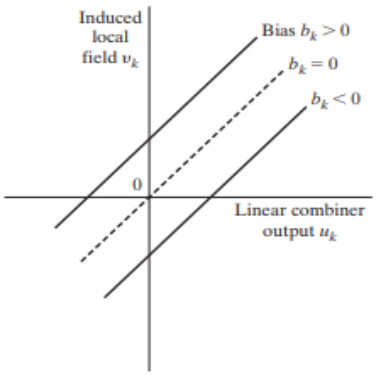
Prema S. Haykin (2009.) neuron ili čvor (*eng. node*) na ulazu prima određeni broj nezavisnih varijabli ili svojstava zajedno sa sinaptičkim težinama (*eng. synaptic weights*) ili kratko težinama. Težine koje zapravo označavaju važnost određene varijable su ključne za razumijevanje načina na koji neuronske mreže uče. Neuronske mreže podešavaju težine kako bi postigle bolji odnosno točniji izlazni rezultat. Inicijalno težine mogu biti dodijeljene nasumično ili svim težinama može biti dodijeljena 0. Kasnije te težine neuronska mreža učenjem kroz

iteracije podešava. Dakle neuron uzima vrijednosti varijabli i dodijeljene određene težine te njihov umnožak zbraja prema sljedećoj formuli:

$$u_k = \sum_{j=0}^{m} w_{kj} x_j$$

Nakon toga se rezultat sume proslijedi u funkciju koju nazivamo funkcijom aktivacije čiji rezultat neuron vraća kao izlaznu vrijednost. Formula aktivacije je na slici 2. označena sa φ. U funkciju aktivacije je moguće kao parametar dati zbroj ukupne sume i pomaka b_k (*eng. bias*) prikazan na slici 3. Pomak je zapravo slobodni koeficijent c koji se dodaje ukupnoj sumi kako bi se podesio izlaz y_k:

$$y_k = \varphi(u_k + b_k)$$

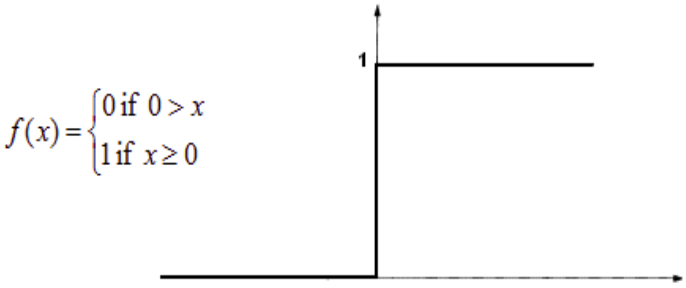


Slika 3. Bias, slobodni koeficijent (Prema S. Haykin)

Izlaz može biti u obliku kontinuirane vrijednosti (npr. cijena), binarne vrijednosti (npr. yes/no, 1/0) ili na izlazu možemo očekivati klasifikaciju u više kategorija pa shodno tome izlaz ima i n različitih vrijednosti. Oznaka k reprezentira broj promatranja što znači da se k-to promatranje sastoji od m ulaznih i n izlaznih varijabli.

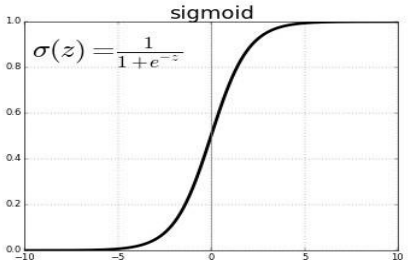
Funkcije aktivacije

Funkcija aktivacije je funkcija kojom neuron računa izlaznu vrijednost i prosljeđuje ju kao konačni rezultat ili kao ulaznu varijablu drugom neuronu. Najviše korištene funkcije aktivacije su funkcija *praga* (*eng. threshold), sigmoidna (eng. sigmoid), ispravljačka (eng. Rectified Linear Unit, ReLU) i hiperbolički tangens (eng. hyperbolic tangent).*



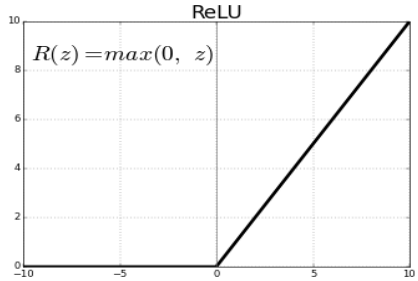
Slika 4. Prag funkcija [14]

Funkcija prag je vrlo jednostavna jer kao rezultat vraća 0 ili 1 po čemu je i dobila naziv. Primjenjiva je za modele neuronskih mreža koji na izlazu imaju binarni vrijednost.



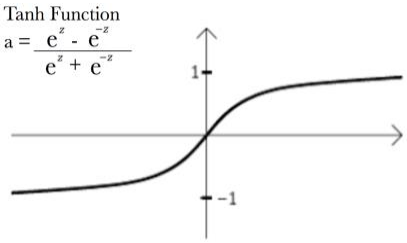
Slika 5. Sigmoidna funkcija [15]

Sigmoidna funkcija poprima vrijednosti između 0 i 1. Prednosti ove funkcije je što daje jasno predviđanje između 0 i 1 i time pomaže neuronu normalizirati izlaz.



Slika 6. Ispravljačka funkcija [15]

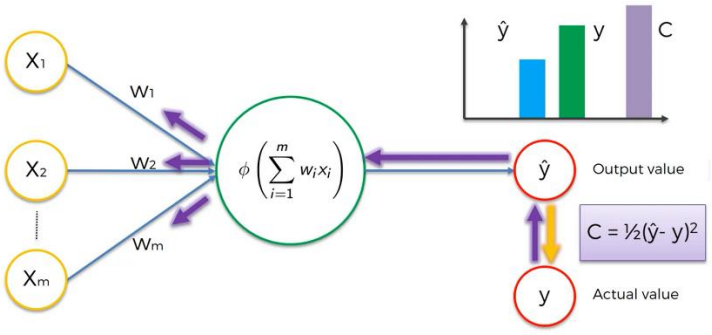
Ispravljačka funkcija je najkorištenija funkcija kod izgradnje modela neuronske mreže jer rješava problem nestajućeg gradijenta (*eng. vanishing gradient problem*).



Slika 7. Hiberbolički tangens [15]

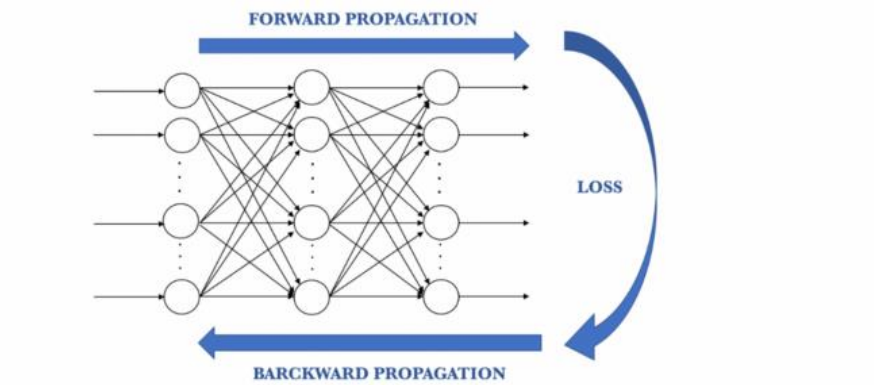
Hiperbolički tangens *tanh* poprima vrijednosti između 1- i 1 i primjenjiv za modele kojima su potrebne vrijednosti ispod 0.

Neuroni funkcijom aktivacije dolaze do izlaznih vrijednosti. Međutim postavlja se pitanje kako neuron zapravo uči odnosno kako neuronska mreža uči i poboljšava svoje rezultate. Odgovor leži u podešavanju težina. Slika 8 prikazuje propagaciju unazad (*eng. backpropagation*) koja podešava težine s obzirom na izlaznu vrijednost funkcije gubitka. Funkcija gubitka se može računati na različite načine. Na slici 8 je prikazana srednja kvadratna pogreška (*eng. mean squared error*). Nakon što se izvrši propagacija unaprijed izlazna vrijednost i prava vrijednost se prosljeđuju u funkciju gubitka čija se vrijednost koristi za podešavanje težina što rezultira boljim rezultatom u sljedećoj iteraciji. Manja vrijednost funkcije gubitka znači i bolje izlazne rezultate. Minimiziranje vrijednosti funkcije gubitka se postiže metodom gradijentnog spusta (*eng. gradient descent*) za jednoslojne i metodom stohastičkog gradijentnog spusta (*eng. stohastic gradient descent*) za višeslojne neuronske mreže. Za jednoslojne je metoda gradijentnog spusta dovoljna za minimiziranje jer znamo iznos pogreške izlaznih neurona. Za višeslojnu neuronsku mrežu gradijentni spust nije dovoljan jer postoji skriveni sloj i za njegove neurone ne znamo koliki je iznos funkcije gubitka i u tom slučaju potrebno je primijeniti metodu stohastičkog gradijentnog spusta.



Slika 8. Propagacija unazad [16]

Može se zaključiti da neuronska mreža uči propagacijom unazad, a u tom procesu pomoću gradijentnog spusta za jednoslojne i stohastičkog gradijentnog spusta za višeslojne minimizira iznos funkcije gubitka. Na slici 8 je prikazana propagacija unazad na najjednostavnijem modelu neuronske mreže koji se naziva *perceptron.* Kao što se vidi na slici 8 *perceptron* na ulazu prima n ulaznih varijabli i na izlazu daje jednu vrijednost. Prema M. Nielsen osoba zaslužena za razvoj *perceptrona* je Frank Rosenblatt. Propagacija unazad se jednako primjenjuje i na složenije modele neuronske mreže što se može vidjeti na slici 9 koja prikazuje višeslojnu ili duboku unaprijednu (*eng. feedforward*) neuronsku mrežu.

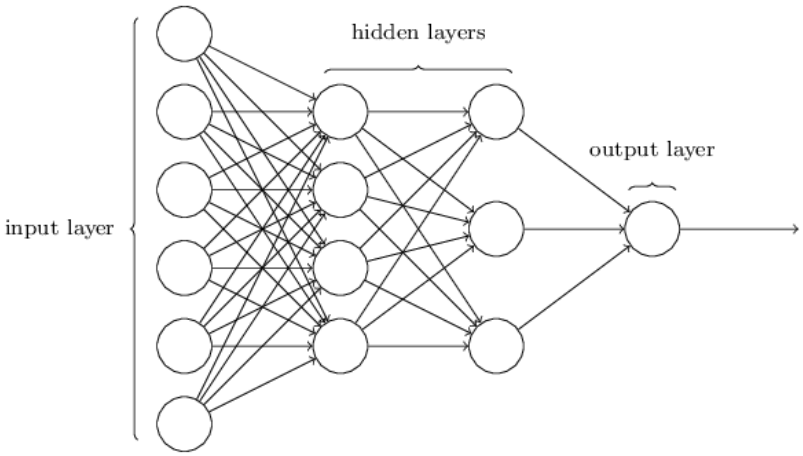


Slika 9. Propagacija unazad za višeslojnu neuronsku mrežu [18]

Definirajući i analizirajući način na koji neuronska mreža uči spomenuti su pojmovi *perceptrona* i višeslojne ili duboke unaprijedne neuronske mreže. Zapravo se radi o podjeli neuronskih mreža prema arhitekturi. S. Haykin smatra arhitekturalno razlikujemo tri osnovne kategorije neuronskih mreža: jednoslojne unaprijedne (*eng. single-layer feedforward networks*), višeslojne unaprijedne (*eng. multilayer feedforward networks*) i povratne (*eng. recurrent networks*) mreže. Povratna mreža se od unaprijedne razlikuje po tome što ima barem jednu povratnu petlju.

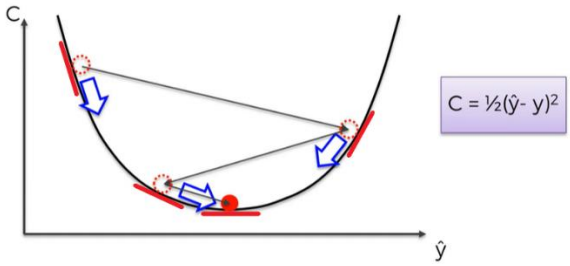
3.1. Duboka neuronska mreža

Duboka neuronska mreža zapravo je višeslojna unaprijedna neuronska mreža. U višeslojnoj neuronskoj osim ulaznih podataka i izlaznih neurona postoje i skriveni slojevi neurona (*eng. hidden layers*). Svaki sloj obrađuje podatke određenom funkcijom aktivacije i sljedećem sloju prosljeđuje izlazne vrijednosti. Na kraju se vrijednost izlazne varijable i prava vrijednost prosljeđuju u funkciju gubitka zbog podešavanja težina propagacijom unazad.



Slika 10. Duboka neuronska mreža [17]

Gradijentni spust



Slika 11. Gradijentni spust [16]

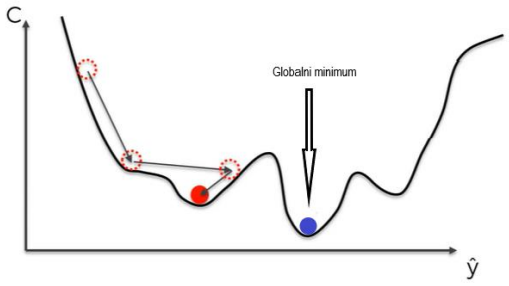
Slika 11 prikazuje kako korake kojima se došlo do minimuma funkcije gubitka i time se postigao najbolji rezultat nakon što se ta vrijednost propagirala unazad. Tih koraka ovisno o početnoj točki ispitivanja može biti i više. Algoritam funkcionira na način da ako gledajući s lijeva na desno linija pada nagib je negativan i potrebno je spustiti se niz krivulju udesno, a ako je nagib pozitivan spuštanje niz krivulju ide ulijevo. Koraci se ponavljaju sve dok vrijednost nagiba ne postane 0. Time je pronađen globalni minimum funkcije gubitka i minimizirana je pogreška. Nagib se računa prema formuli:

$$m = \frac{y_2 - y_1}{x_2 - x_1} = \tan\theta$$

Metodu gradijentnog spusta je moguće primijeniti samo na konveksne funkcije. To znači da nije primjenjiva za višeslojne neuronske mreže kojima funkcije gubitka nisu konveksne. Rješenje za to je stohastički gradijentni spust.

Stohastički gradijentni spust

Ova metoda rješava problem višestrukih lokalnih minimuma funkcije gubitka za višeslojne neuronske mreže. Slika 12 prikazuje funkciju gubitka koja nije konveksna što može rezultirati pronalaskom lokalnog umjesto globalnog minimuma.

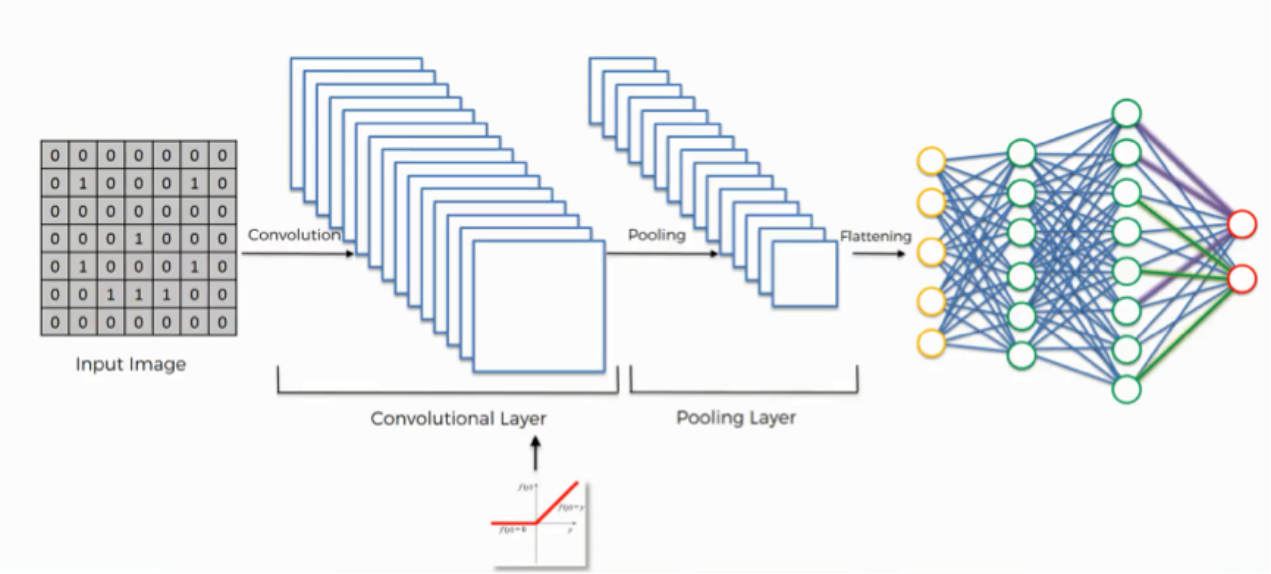


Slika 12. Nekonveksna funkcija gubitka [16]

Pogreška neurona na određenom sloju se procjenjuje na temelju pogrešaka neurona na sljedećem sloju sa kojima je neuron spojen. To se postiže tako da se za svako k-to promatranje zasebno provede podešavanje težina za razliku od grupnog gradijentnog spusta (*eng. batch gradient descent*) koji podešavanje provodi nakon što je učitan cijeli skup podataka. Osim što rješava problem višestrukih lokalnih minimuma, računalo ga brže i izvršava jer podešavanje težina provodi nakon svakog k-tog promatranja.

3.2. Konvolucijska neuronska mreža

Konvolucijske neuronske mreže su po svojoj arhitekturi unaprijedne višeslojne mreže kojoj nisu svi slojevi potpuno povezani (*eng. fully connected*). Najčešća primjena konvolucijskih neuronskih mreža je za klasifikaciju slika, medicinsku analizu slika, prepoznavanje lica i prometnih znakova itd. Primjenu u ovom radu nalazi kroz model za klasifikaciju mrežnog prometa.



Slika 13. Osnovna arhitektura konvolucijske neruonske mreže [16]

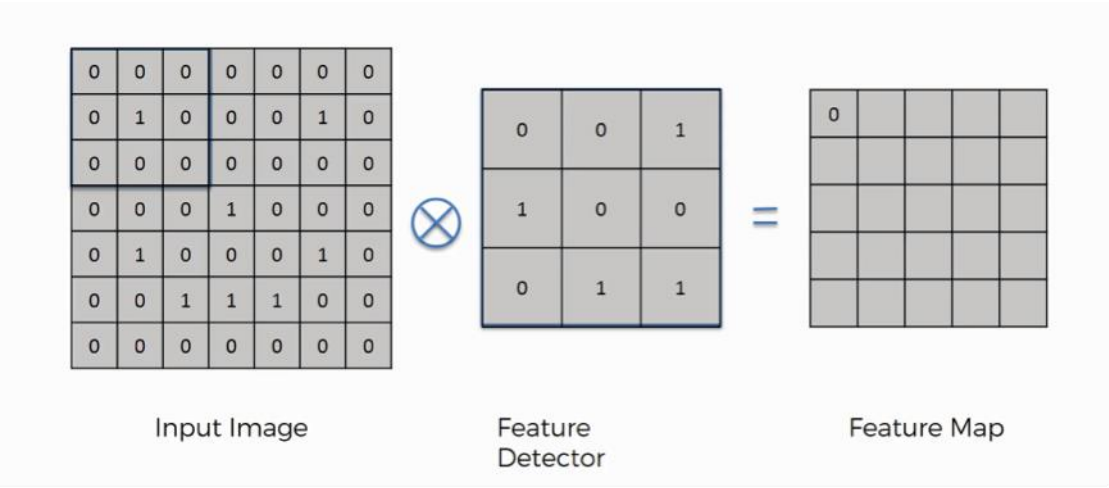
Slika 13 prikazuje sve bitne elemente konvolucijske neuronske mreže, a to su: konvolucijski sloj (*eng. convolutional layer*), sažimanje (*eng. pooling*) i ravnanje (*eng. falttening*) nakon kojeg se kreiraju potpuno povezani slojevi što znači da je svaki neuron prethodnog sloja povezan sa svakim neuronom sljedećeg sloja što ne mora biti slučaj i za djelomično povezane neuronske mreže.

Konvolucijski sloj

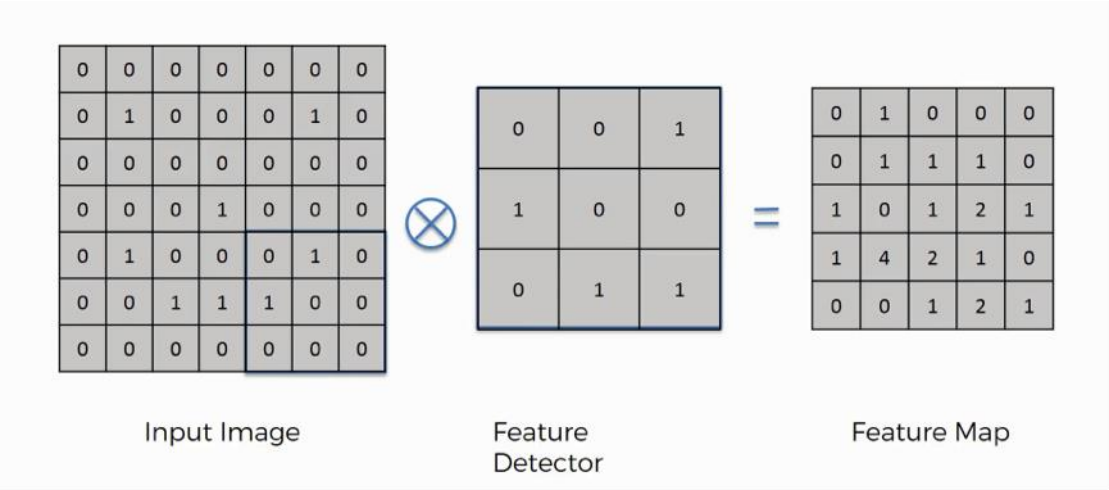
Za razumijevanje konvolucijskog sloja i operacija koje izvršava bitna su tri elementa: ulazna matrica, detektor značajki (*eng, feature detector*) i mapa značajki (*eng. feature map*). Detektor značajki još se naziva i jezgra (*eng. kernel*) ili filter (*eng. filter*). Element detektora značajki može biti samo 0 ili 1, dok element mape značajki može biti bilo koji prirodni broj uključujući i 0. Slika 14 prikazuje početak, a slika 15 kraj konvolucijske operacije kojoj je cilj smanjiti veličinu ulazne matrice kako bi se izdvojila bitna svojstva za promatranje. Operacija se odvija na način da se na početku odredi matrica detektora značajki koja je na navedenim slikama veličine *3x3.* Veličina detektora značajki ovisi naravno o aplikacijskoj domeni i veličini ulaza na koji se primjenjuje. Detektor značajki pomičemo po ulaznoj matrici po određenoj veličini koraka (*eng. strides*). Na slikama 14 i 15 veličina koraka je 1 što znači da se detektor značajki pomiče jedan po jedan element. Veći korak generira manju dimenziju mape značajki.

Filter se pomiče po ulaznoj matrici i primjenjuje operaciju konvolucije nad dijelom matrice koji je istih dimenzija kao i filter. Dio ulazne matrice se izdvaja u zasebnu matricu u svakom koraku i označuje sa A_1, A_2, A_3 … A_n gdje je n broj koraka. Neka je matrica filtera označena sa M. Operacija $\otimes$ predstavlja *Hadamardov produkt* i broj jedinica matrice $B_n =$

$A_n \otimes M$ se zapisuje u mapu značajki koja bilježi broj podudarenih značajki nakon konvolucije.

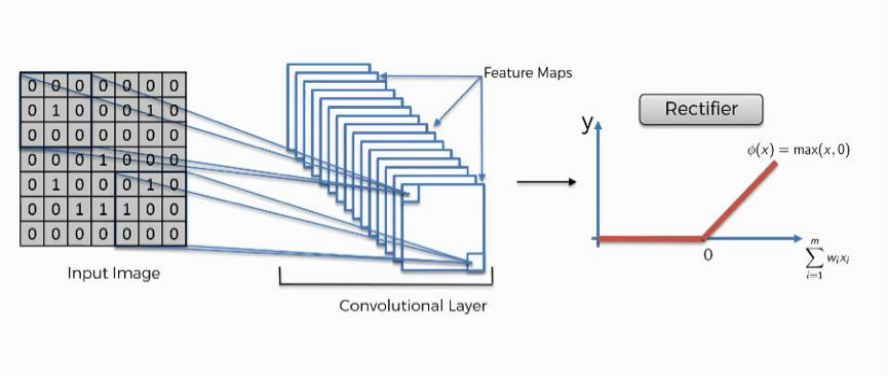


Slika 14. Početak konvolucijske operacije [20]



Slika 15. Završetak konvolucijske operacije [20]

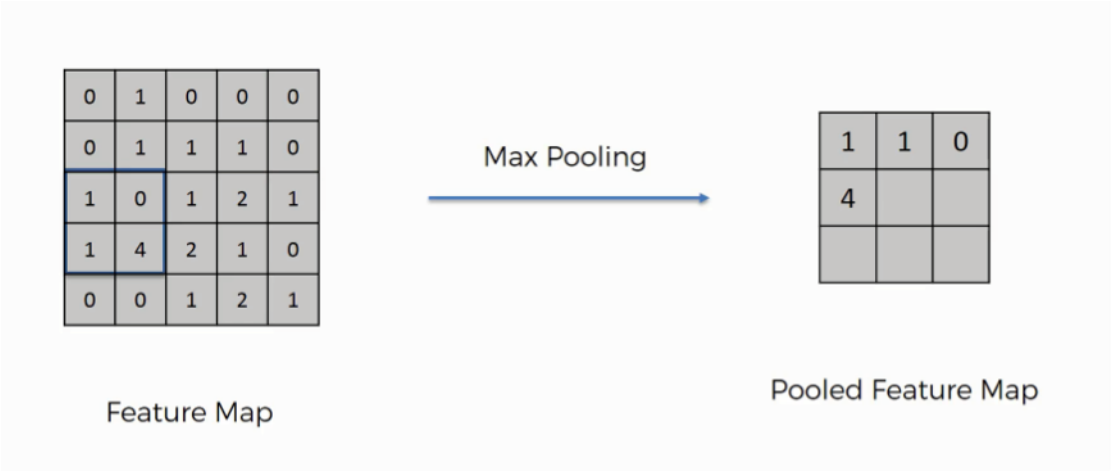
Gore navedeni primjer vrlo je pojednostavljen. U stvarnim složenim zadacima konvolucijske neuronske mreže kroz trening razvijaju više detektora značajki i koriste ih za razvoj nekoliko mapa značajki koje stvaraju konvolucijski sloj što se može vidjeti na slici 16. U konvolucijskom sloju se ovisno o zadanom problemu nakon dobivenih rezultata mapa značajki primjenjuje ispravljačka funkcija koja se pokazala kao najučinkovitija.



Slika 16. Konvolucijski sloj [20]

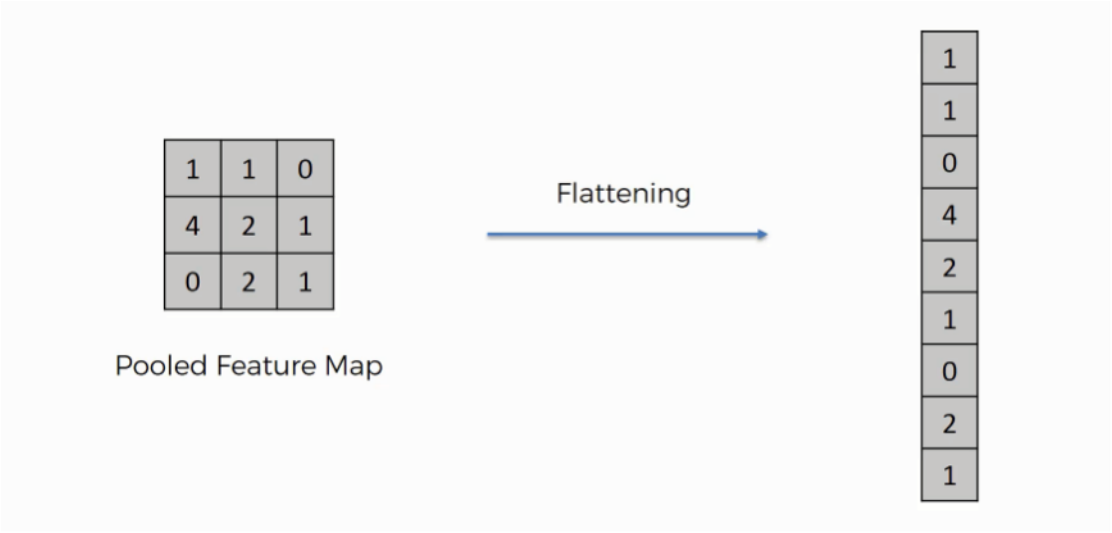
Sloj sažimanja

Konvolucijski sloj sloju za sažimanje prosljeđuje mape značajki. Sloj za sažimanje omogućava konvolucijskoj neuronskoj mreži da detektira svojstvo prikazano na bilo koji način ili u bilo kojem okruženju. Sažimanje može biti prosječno (*eng. mean pooling*), maksimalno (*eng. max pooling*) i ukupno (*eng. sum pooling*). Sažimanje korišteno u programskoj izvedbi konvolucijske mreže za binarnu i višerazrednu klasifikaciju u ovom radu je maksimalno sažimanje.

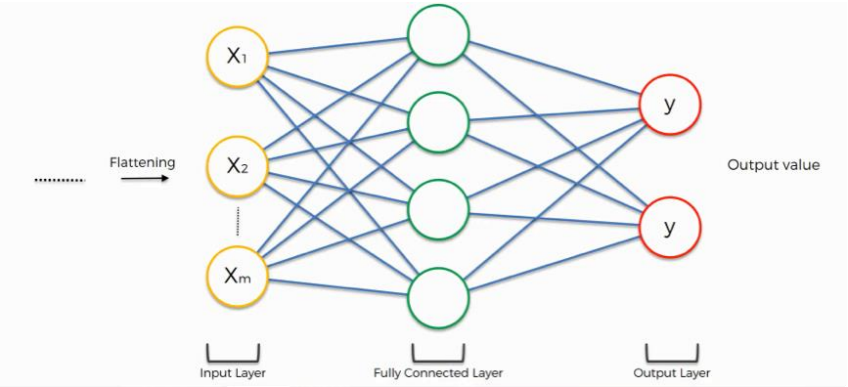


Slika 17. Maksimalno sažimanje [20]

Postupak popunjavanja sažete mape značajki razlikuje se od postupka popunjavanja obične mape značajki. U ovom procesu se odabire dimenzija kojom se ispituju dijelovi mape značajki. Na slici 17 je prikazana matrica *2x2* kojom se djeluje nad mapom značajki određenim korakom. Maksimalnim sažimanjem se uzima u određenom koraku dio matrice dimenzije *2x2* iz kojeg se uzima maksimalna vrijednost i sprema u sažetu mapu značajki. Rezultati sloja sažimanja odnosno sažete mape značajki se ravnaju i prosljeđuju ulaznom sloju potpuno povezane neuronske mreže. Postupak ravnanja je jednostavan i prikazan je slikom 18, a potpuno povezana neuronska mreža slikom 19.



Slika 18. Ravnanje [20]



Slika 19. Potpuno povezana neuronska mreža [20]

Na izlaznom sloju konvolucijske neuronske mreže očekuje se klasifikacija odnosno vjerojatnosna razdioba varijable s K mogućih vrijednosti. Tu mogućnost nudi softmax aktivacijska funkcija, za funkciju gubitka kojom se smanjuje pogreška najbolje rezultate pokazuje unakrsna entropija (*eng. cross-entropy*) softmax klasifikatora. Softmax funkcija $f: \mathbb{R}^K \rightarrow \mathbb{R}^K$ definirana je formulom

$$f(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

Za $i = 1, \ldots, K$ i $z = (z_1, \ldots, z_K) \in \mathbb{R}^K$ gdje z predstavlja ulazni vektor od K realnih brojeva. [21] Nakon djelovanja softmax funkcije svaka komponenta K je normalizirana na vrijednost u intervalu [0, 1], a ukupna suma svih komponenti je 1 kako bi one mogle biti tumačene kao vjerojatnosti.

Unakrsna entropija se definira formulom

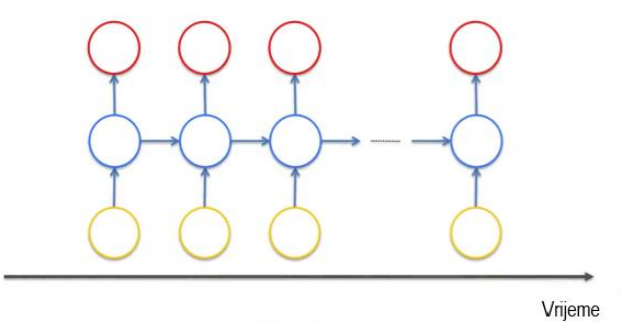
$$H(p,q) = -\sum_{x} p(x) \log q(x)$$

gdje je p istinita ili stvarna razdioba, a q procijenjena razdioba.

3.3. Povratna neuronska mreža

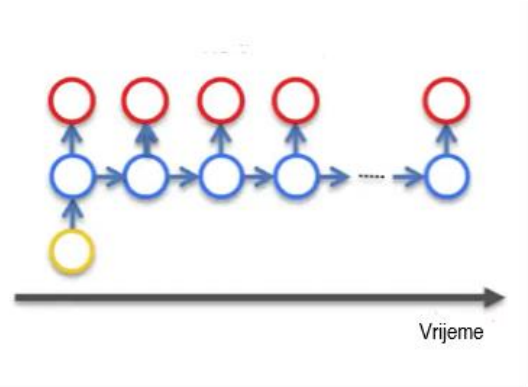
Režnjevi ljudskog mozga se mogu povezati sa određenom strukturom neuronske mreže. Duboke unaprijedne neuronske mreže oponašaju dugoročno pamćenje što znači da se težine koje su dobivene treniranjem mogu iskoristi i nakon treniranja jer su trajno spremljene u memoriju. Stoga ih je moguće povezati sa temporalnim moždanim režnjem. Zadaća konvolucijskih neuronskih mreža je prepoznavati slike, objekte i trenirati i učiti računalni vid pa se zbog toga vežu uz okcipitalni režanj. Za oponašanje kratkoročne memorije i funkcija frontalnog režnja računalna znanost istražuje i konstantno poboljšava povratne neuronske mreže koje se najčešće primjenjuju za previđanje i analizu vremenskih nizova, prepoznavanje govora i sintakse govora, učenje ritma itd. Sami naziv upućuje na postojanje povratne veze u arhitekturi koja ju izdvaja od unaprijedne neuronske mreže.

Osim što neuroni skrivenog sloja povratne neuronske mreže daju izlazne vrijednosti, oni se također vežu i na same sebe kroz određeni vremenski period. Bitno je naglasiti vremenski period jer ideja koja stoji iza povratnih neuronskih mreža je da neuroni imaju neku vrstu kratkoročnog pamćenja, pružajući im mogućnost da se prisjete stanja kojeg su imali prije. Dakle, neuroni mogu sami sebi prenositi podatke u budućnost i analizirati ih. Slika 20 prikazuje neurone skrivenog sloja i njihovo stanje kroz vrijeme. Plavi krug ne predstavlja samo jedan neuron nego sve neurone skrivenog sloja, a strelica udesno označava da svaki neuron ima povratnu vezu.

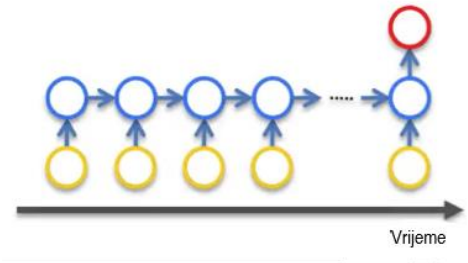


Slika 20. Povratna veza neurona kroz vrijeme

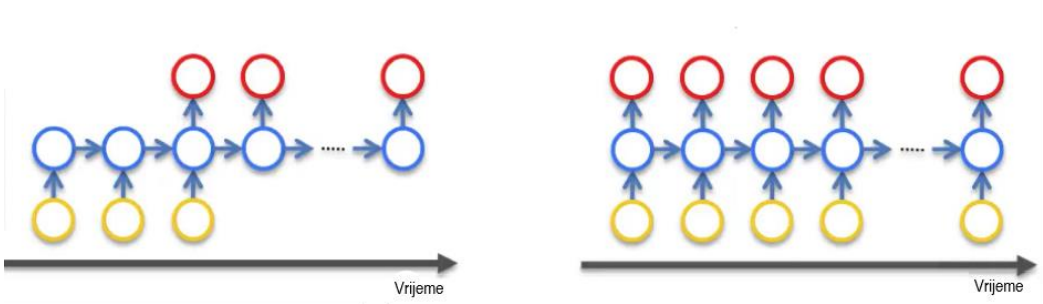
S obzirom na veličinu ulaza i izlaza povratne neuronske mreže se mogu izgraditi na više načina. Tako se razlikuje struktura jedan ulaz prema više izlaza (*eng. one to many*), više ulaza prema jednom izlazu (*eng. many to one*) i više ulaza prema više izlaza (*eng. many to many*).



Slika 21. Jedan ulaz prema više izlaza



Slika 22. Više ulaza prema jednom izlazu

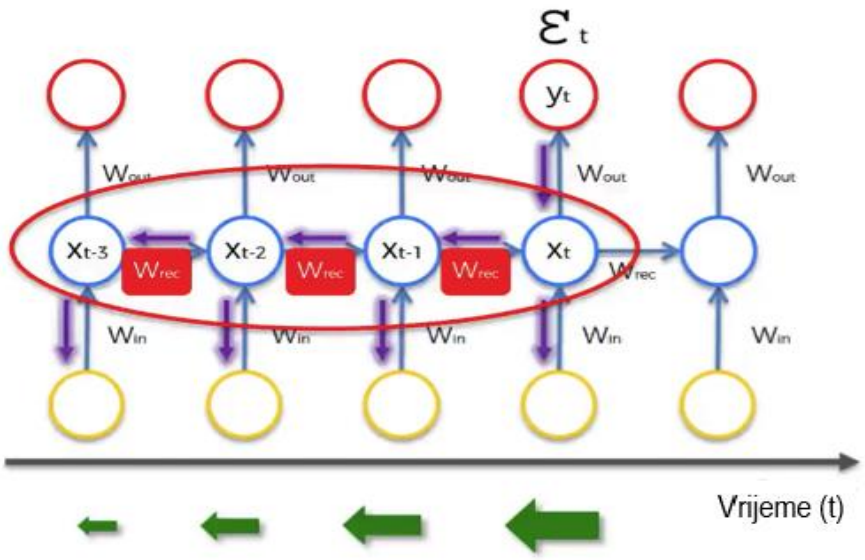


Slika 23. Više ulaza prema više izlaza

Princip jednog ulaza prema više izlaza bi se mogao primijeniti u slučaju da je na ulazu slika, a na izlazu opis slike riječima. Više ulaza prema jednom izlazu najčešće svoju primjenu nalazi kod analize teksta gdje je potrebno analizirati je li neki tekst optimističan ili pesimističan. I na kraju princip više ulaza prema više izlaza se primjenjuje pri prijevodu teksta sa jednog jezika na drugi.

Povratne neuronske mreže u procesu učenja nailaze na problem kod računanja pogreške i njene propagacije unazad u svrhu podešavanja težina. Pogreška se mora propagirati ne samo unazad kroz sve slojeve već i kroz sva stanja neurona u određenim vremenskim trenucima, a težina se odmotavanjem vremenske petlje ponavlja (*eng. weight recurring*) što je problematično. Ponavljajuća težina je označena sa W_{rec}.

Da bi se došlo iz stanja skrivenog sloja X_{t-3} do stanja X_{t-2} potrebno je X_{t-3} pomnožiti sa težinom W_{rec}. Zatim je isto potrebno ponoviti i za stanje X_{t-1} što znači da se X_{t-2} množi sa težinom W_{rec}. Tako se nastavlja dalje sve do stanja X_t. I tu onda nastaje problem jer se uvijek množi istom vrijednošću, a u slučaju da je ta vrijednost mala umnožak se smanjuje drastično. Nakon inicijalizacije težina slučajnim vrijednostima one su jako blizu nuli pa množenjem takvih vrijednosti W_{rec} sa stanjima X_t u određenom vremenskom trenutku t gradijent postaje sve manji i mreži je sve teže podesiti težine. Zbog toga se taj problem naziva nestajanjem gradijenta. Nestajanje gradijenta prikazano je na slici 24 zelenim strelicama. U slučaju da su početne težine veće od jedan dolazi do obrnute situacije gdje gradijent postaje prevelik i u jednom trenutku eksplodira. Ta pojava se naziva eksplodirajućim gradijentom (*eng. exploding gradient*).

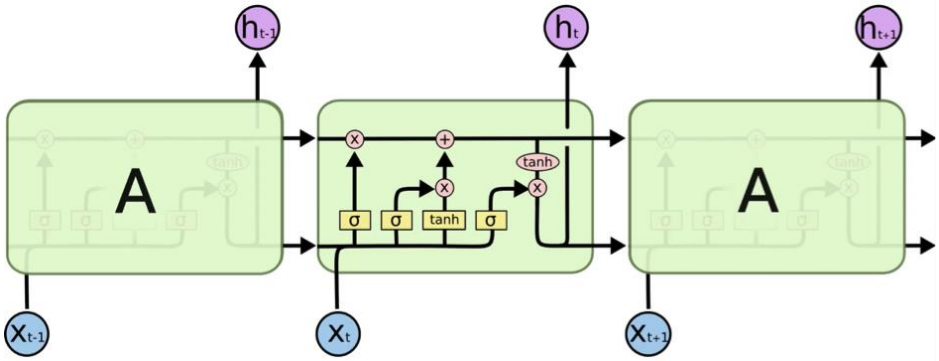


Slika 24. Problem nestajanja gradijenta [22]

Kao rješenje na nestajući i eksplodirajući gradijent nameće se ćelija sa dugoročnom memorijom (*eng. Long Short Term Memory, LSTM*). Postoje i druga rješenja, ali LSTM daje najbolje rezultate pa je u ovom radu korišten kod implementacije modela povratne neuronske mreže. Stoga je potrebno detaljnije objasniti način kojim rješava nestajući i eksplodirajući gradijent.

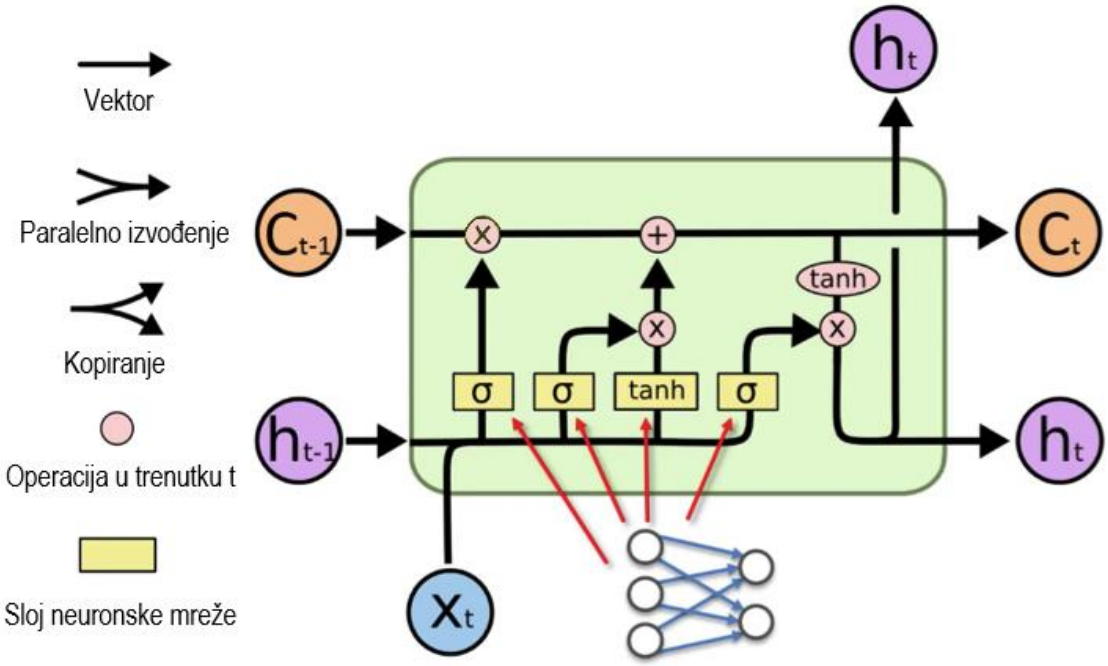
LSTM

Može se zaključiti da u slučaju kada je ponavljajuća težina manja od jedan, $W_{rec} < 1$, dolazi do pojave nestajućeg gradijenta, a ako je veća od jedan, $W_{rec} > 1$, onda dolazi do pojave eksplodirajućeg gradijenta. Rješenje bi bilo osigurati da je ponavljajuća težina jednaka jedan, $W_{rec} = 1$. To svojom arhitekturom omogućava LSTM.



Slika 25. Povratna neuronska mreža sa LSTM-om [22]

Slika 25 prikazuje arhitekturu povratne neuronske mreže sa LSTM-om, a za detaljan opis načina rada LSTM-a prikazana je slika 26 sa legendom znakova.



Slika 26. Detaljan prikaz LSTM-a

Ponavljajuća težina LSTM-a jednaka je jedan, $W_{rec} = 1$. To svojstvo na vrhu arhitekture opisuje ravna linija koja označava memorijski protok koji slobodno teče kroz vrijeme i može se zaustaviti, ili mu se može dodatno priključiti neka druga memorija. Propagacija kroz takav LSTM eliminira problem nestajućeg gradijenta.

Prije nego se krene u detaljniju razradu potrebno je objasniti označavanje na slici 26. C_{t-1} označava ulaz iz neke druge memorijske ćelije, X_t je ulaz u trenutku t, a h_t je izlaz u trenutku t koji ide i na izlazni i na skriveni sloj sljedeće točke u vremenu. Na slici se vidi da svaki blok ima tri ulaza X_t, h_{t-1} i C_{t-1} i dva izlaza h_t i C_t. Svi ulazi i izlazi su vektori. Na arhitekturi se raspoznaju operacije u trenutku t. U jednom bloku ih ima pet i dijele se na tri vrste: „x*“, „+“* i *tanh.* Operacija „x*“* briše, sprema ili prosljeđuje memoriju na izlaz na temelju odluke funkcije aktivacije. „+“ operacija dodaje memoriju u protok, ali samo ako operacija „x*“* propusti memoriju dalje prema njoj. Hiperbolički tangens pretvara ulazne vrijednosti u vrijednosti intervala od -1 do 1.

Kako bi vidjeli slijed akcija dok neuronska mreža uči potrebno je krenuti od ulaza gdje dolazi vrijednost X_t i vrijednost iz prethodnog čvora h_{t-1}. Te dvije vrijednosti zajedno ulaze u funkciju aktivacije koja će svojim izlazom odlučiti treba li izvršiti operacija „x“ izvršiti brisanje. Iste te vrijednosti odnosno vektori vrijednosti se paralelno prosljeđuju na funkciju aktivacije *tanh* i funkciju aktivacije koja odlučuje hoće li vrijednost izračunata funkcijom aktivacije *tanh* biti proslijeđena i dodana memorijskom protoku na vrhu kroz operaciju „+“. Nakon toga se promatra memorija koja ide kroz navedeni memorijski protok na vrhu. Ako operacija „x“ za zatvaranje pušta memoriju u protok, a operacija za spremanje memorije je zatvorena, memorija se neće promijeniti. U suprotnom ako je operacija za puštanje toka zatvorila tok, a operacija za spremanje memorije pustila da se memorija doda, onda se i memorija u protoku na vrhu mijenja. Na kraju na temelju ulaza X_t i h_{t-1} čvor odlučuje koji dio memorije će pustiti kao izlaz.

Osim LSTM-a postoje i njegove varijacije. Najpoznatija je propusna povratna ćelija (*eng. gated recurrent unit, GRU*) koja za razliku od LSTM-a ima dva ulaza.

Različite arhitekture neuronskih mreža se primjenjuju u određenim aplikacijskim domenama što ne znači da ih se striktno može koristiti samo za određeni set problema. U ovom radu su duboka, konvolucijska i povratna neuronska mreža korištene za izgradnju modela za otkrivanje napada.

4. Programska izvedba

Sustavi za otkrivanje napada, IDS, povećavaju razinu sigurnosti i zaštite osjetljivih podataka. U velikoj većini temeljeni su na modelima strojnog ili dubokog učenja. Da bi navedeni modeli mogli učiti potreban im je određeni skup podataka sa specifičnim svojstvima za aplikacijsku domenu. Za programsku izvedbu modela neuronskih mreža u ovom radu korišten je skup podataka NSL-KDD. Za svaku vrstu neuronske mreže (duboka, konvolucijska i povratna) izgrađeni su modeli za binarnu („napad“ ili „nije napad“) i višerazrednu (*DoS, Probe, R2L, U2R* i *Normal*) klasifikaciju. Kategorija *normal* zapravo označava da se ne radi o napadu nego o normalnoj aktivnosti u mrežnom prometu. Rezultati predviđanja svakog modela uspoređeni su matricom konfuzije.

4.1. Skup podataka NSL-KDD

NSL-KDD je skup podataka namijenjen istraživanju na području informacijske sigurnosti. Radi se o ažuriranoj verziji skupa podataka KDD'99. Predviđena je za usporedbu različitih vrsta metoda za otkrivanje napada. Prednosti koje NSL-KDD ima nad KDD'99 skupom podataka su [1]:

- Nema suvišnih zapisa u skupu podataka za treniranje, tako da klasifikacija neće biti pristrana prema češćim zapisima.
- U testnim setovima nema duplih zapisa pa model dok uči nije pristran u slučaju da koristi metode koje imaju bolji postotak otkrivanja na čestim zapisima.
- Broj odabranih zapisa iz svake težinske skupine obrnuto je proporcionalan postotku zapisa u izvornom skupu podataka KDD-a. Kao rezultat toga, stope klasifikacije različitih metoda strojnog učenja razlikuju se u širem rasponu, što čini učinkovitijim precizno vrednovanje različitih tehnika učenja.
- Broj zapisa u trening i testnim skupovima podataka je dovoljan za provođenje eksperimenata nad kompletnim skupom bez potrebe za nasumičnim odabirom manjeg skupa. Zbog toga su rezultati evaluacija različitih istraživačkih radova dosljedni i usporedivi.

Osim NSL-KDD skupa podataka postoje i drugi skupovi podataka koji se koriste u ovom polju. Najpoznatiji su ADFA-ID i ISCX-UNB.

NSL-KDD skup podataka za treniranje i testiranje se sastoji od 42 svojstva koje može biti nominalno (kategorijsko), binarno ili numeričko.

Tablica 1. Popis svojstava NSL-KDD skupa podataka [24]

Br.	Naziv	Opis	Tip varijable	Primjer vrijednosti
1	duration	Trajanje veze	numerička	0
2	protocol_type	Korišteni protokol tokom trajanja veze	nominalna	tcp
3	service	Korišteni servis odredišne mreže	nominalna	ssh
4	flag	Status veze	nominalna	RSTR
5	src_bytes	Broj bajtova prenesenih od izvora do odredišta	numerička	494
6	dst_bytes	Broj bajtova prenesenih od odredišta prema izvoru	numerička	0
7	land	Ako su izvorišna IP adresa i port jednaki odredišnoj IP adresi i portu poprima vrijednost 1 inače 0	binarna	1
8	wrong_fragment	Ukupan broj pogrešnih fragmenata u trenutnoj vezi	numerička	0
9	urgent	Broj hitnih paketa u trenutnoj vezi. Hitni paket je paket kojemu je bit hitnosti (*eng. urgent bit*) aktivan	numerička	0
10	hot	Broj „vrućih“ indikatora u sadržaju. To spadaju pristupi sistemskim datotekama, kreiranje ili izvođenje programskog koda.	numerička	0
11	num_failed_logins	Broj neuspješnih pokušaja prijave u sustav	numerička	10
12	logged_id	Je li korisnik uspješno prijavljen u sustavu. Ako je onda je vrijednost 1 inače 0	binarna	1
13	num_compromised	Broj kompromitiranih uvjeta	numerička	2
14	root_shell	1 ako je dobiven pristup korijenskoj ljusci; inače 0	binarna	0
15	su_attempted	1 ako je pokušano izvršavanje naredbe „su root“; inače 0	binarna	1
16	num_root	Broj pristupa korijenskom korisniku ili broj operacija izvršenih u modu korijenskog korisnika	numerička	2
17	num_file_creations	Broj kreiranja datoteka tijekom veze	numerička	0
18	num_shells	Broj otvaranja ljuske (*eng. number of shell prompts*)	numerička	0
19	num_access_files	Broj operacija nad datotekama za kontrolu pristupa	numerička	0
20	num_outbound_cmds	Broj izlaznih naredbi u ftp sesiji	numerička	2

21	is_hot_login	1 ako se radi o prijavi sa „vruće“ liste, npr. prijava korijenskog korisnika: inače 0	binarna	1
22	is_guest_login	1 ako se radi o prijavi gosta (*eng. guest*); inače 0	binarna	0
23	count	Broj veza na istog odredišnog domaćina koje su iste kao i trenutna veza u protekla dvije sekunde.	numerička	0
24	srv_count	Broj veza prema istom servisu (broj porta) kao i trenutna veza u protekle dvije sekunde	numerička	0
25	serror_rate	Postotak veza koje su aktivirale jednu od vrijednosti „S0“, „S1“, „S2“ ili „S3“ svojstva „flag“ (4) među vezama koje su pribrojene svojstvu „count“ (23)	numerička	0
26	srv_serror_rate	Postotak veza koje su aktivirale jednu od vrijednosti „S0“, „S1“, „S2“ ili „S3“ svojstva „flag“ (4) među vezama koje su pribrojene svojstvu „srv_count“ (24)	numerička	0
27	rerror_rate	Postotak veza koje su aktivirale vrijednost „REJ“ svojstva „flag“ (4) među vezama koje su pribrojene svojstvu „count“ (23)	numerička	0
28	srv_rerror_rate	Postotak veza koje su aktivirale vrijednost „REJ“ svojstva „flag“ (4) među vezama koje su pribrajaju svojstvu „srv_count“ (24)	numerička	0
29	same_srv_rate	Postotak veza koje su trebale isti servis (svojstvo „service“) među vezama koje su pribrojene svojstvu „count“ (23)	numerička	1
30	diff_srv_rate	Postotak veza koje su trebale različite servise (svojstvo „service“) među vezama koje su pribrojene svojstvu „count“ (23)	numerička	1
31	srv_diff_host_rate	Postotak veza koje su trebale drugačije odredište među vezama pribrojenim svojstvu „srv_count“ (24)	numerička	0
32	dst_host_count	Broj veza koje imaju iste odredišne IP adrese domaćina	numerička	150
33	dst_host_srv_count	Broj veza koje imaju isti broj porta	numerička	25
34	dst_host_same_srv_rate	Postotak veza koje su trebale isti servis (svojstvo „service“) među vezama koje	numerička	0.17

		su pribrojene svojstvu „dst_host_count“ (32)		
35	dst_host_diff_srv_rate	Postotak veza koje su trebale različite servise (svojstvo „service“) među vezama koje su pribrojene svojstvu „dst_host_count“ (32)	numerička	0.03
36	dst_host_same_src_port_rate	Postotak veza koje su trebale isti izvorišni port među vezama koje su pribrojene svojstvu „dst_host_srv_count“ (33)	numerička	0.17
37	dst_host_srv_diff_host_rate	Postotak veza koje su trebale različita odredišta među vezama koje su pribrojene svojstvu „dst_host_srv_count“ (33)	numerička	0
38	dst_host_serror_rate	Postotak veza koje su aktivirale jednu od vrijednosti „S0“, „S1“, „S2“ ili „S3“ svojstva „flag“ (4) među vezama koje su pribrojene svojstvu „dst_host_count“ (32)	numerička	0
39	dst_host_srv_serror_rate	Postotak veza koje su aktivirale jednu od vrijednosti „S0“, „S1“, „S2“ ili „S3“ svojstva „flag“ (4) među vezama koje su pribrojene svojstvu „dst_host_srv_count“ (33)	numerička	0
40	dst_host_rerror_rate	Postotak veza koje su aktivirale vrijednost „REJ“ svojstva „flag“ (4) među vezama koje su pribrojene svojstvu „dst_host_count“ (24)	numerička	0.05
41	dst_host_srv_rerror_rate	Postotak veza koje su aktivirale vrijednost „REJ“ svojstva „flag“ (4) među vezama koje su pribrojene svojstvu „dst_host_srv_c ount“ (24)	numerička	0

Tablica 1 opisuje svojstva NSL-KDD skupa podataka. U tablici je najviše numeričkih vrijednosti koje broje neku pojavu ili prikazuju njen postotak u odnosu na povezane pojave. Neka svojstva su povezana sa drugim svojstvima, npr. svojstvo „serror_rate“ koje prikazuje postotak veza koje su aktivirale jednu od vrijednosti svojstva „flag“ među vezama koje su svojim ponašanjem pribrojene svojstvu „count“. Jedno svojstvo nije navedeno, a to je zadnje svojstvo koje zapravo označava kategoriju ili klasu kojoj jedan zapis svojim svojstvima pripada. Kategorije mogu biti određene na dva načina. Prvi način klasificiranja određuje je li zapis potencijalna prijetnja ili ne. Oznake su *attack* i *normal*. Drugi način klasificiranja svrstava zapis u nekoliko kategorija. To su *normal, land, back, pod, neptune, smurf, teardrop, apache2, udpstorm, processtable, worm, mailbomb, satan, ipsweep, nmap, portsweep, mscan, saint,*

guess_passwd, ftp_write, imap, phf, multihop, warezmaster, warezclient, spy, xlock, xsnoop, snmpguess, snmpgetattack, sendmail, httptunnel, named, rootkit, loadmodule, sqlattack, xterm, buffer_overflow, perl, ps. Sve te oznake osim *normal* se mogu svrstati u četiri kategorije *DoS, Probe, R2L, U2R.* Tablica 2 prikazuje grupiranje klasa u četiri navedene kategorije i oznake koje je pojedina kategorija dobila tijekom obrade podataka. Oznaka „5“ je dodijeljena kategoriji *normal* koja označava da se ne radi o prijetnji.

Tablica 2. Grupiranje vrsta napada

Kategorija	Vrsta napada	Oznaka
DoS	land, back, pod, neptune, smurf, teardrop, apache2, udpstorm, processtable, worm, mailbomb	1
Probe	satan, ipsweep, nmap, portsweep, mscan, saint	2
R2L	guess_passwd, ftp_write, imap, phf, multihop, warezmaster, warezclient, spy, xlock, xsnoop, snmpguess, snmpgetattack, sendmail	3
U2R	rootkit, loadmodule, sqlattack, xterm, buffer_overflow, perl, ps	4

Koristeći se tim skupom podataka za treniranje i testiranje modela neuronskih mreža za svaku vrstu neuronske mreže pojedinačno je izgrađena i programski izvedena arhitektura u programskom jeziku *python*. Za programsku izvedbu korištene su biblioteke *Tensorflow (Keras API), scikit learn.* Korišteni alat za crtanje arhitekture je *draw.io* [25], a razvojni alat za programski kod je *Google Colab.*

Zajednički dio programske izvedbe

Za svaku vrstu neuronske mreže programski je zasebno izveden model za binarnu klasifikaciju i model za višerazrednu klasifikaciju. Određeni dijelovi programske izvedbe zajednički su za sve implementacije, a neki samo za modele binarne klasifikacije i modele višerazredne klasifikacije. Stoga je zajednički dio programske izvedbe prikazan prije pojedinačne programske izvedbe modela. Uključivanje biblioteka, čitanje skupa podataka za treniranje i testiranje te skaliranje varijabli zajedničko je svim implementacijama. Označavanje nominalnih vrijednosti numeričkim i prikaz rezultata različiti su za modele binarne klasifikacije i višerazredne klasifikacije. Rezultati oba modela prikazani su matricom konfuzije čije se računanje za binarnu klasifikaciju razlikuje od računanja za višerazrednu klasifikaciju. U procesu obrade podataka, treniranja i testiranja neuronske mreže ili algoritma strojnog učenja, svojstva se promatraju kao varijable pa se umjesto pojma svojstva koristiti pojam varijable.

Uvoz biblioteka i učitavanje skupa podataka

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

```
import pandas as pd

dataset_train = pd.read_csv('kdd_train.csv', header = None)
dataset_test = pd.read_csv('kdd_test.csv', header = None)

X_train = dataset_train.iloc[:, :-1].values
y_train = dataset_train.iloc[:, -1].values

X_test = dataset_test.iloc[:, :-1].values
y_test = dataset_test.iloc[:, -1].values
```

Za učitavanje podataka korištena je *python* biblioteka *pandas.* Podaci su spremljeni u *.csv* tip datoteke.

Označavanje nominalnih svojstava ulaznih podataka

Nominalna svojstva su drugi („protocol_type“), treći („service“) i četvrti („flag“) stupac u NSL-KDD skupu podataka.

```
from sklearn.preprocessing import OrdinalEncoder
oe = OrdinalEncoder()
oe.fit(X_train[:, 1:4])
X_train[:, 1:4] = oe.transform(X_train[:, 1:4])
X_test[:, 1:4] = oe.transform(X_test[:, 1:4])
```

Označavanje (*eng. labelling*) se mora provesti jer modeli neuronskih mreža na ulazu ne mogu primiti nominalne vrijednosti. U tu svrhu korištena je metoda iz biblioteke *scikit-learn, OrdinalEncoder.* Metoda za svaku nominalnu vrijednost dodijeli numeričku vrijednost, npr. za varijablu „protocol_type“, *tcp -> 1.0, udp -> 2.0, icmp -> 3.0*. Za nominalne varijable koje nemaju prirodnu povezanost kao što ima varijabla “poredak” (1, 2, 3, …, n) potrebno je jedno vruće kodiranje (*eng. one-hot encoding*) jer su numeričke vrijednosti uređen skup i neuronske mreže mogu biti u stanju razumjeti i iskoristiti taj odnos. Takvim označavanjem numeričke vrijednosti dobivaju binarni zapis i skup podataka se proširuje za broj različitih vrijednosti varijable koja se označava minus jedan ($n-1$). Tako vrućim kodiranjem svojstva „protocol_type“ koje može poprimiti $n=3$ različite vrijednosti proširuje se ulazni skup podataka za još dvije varijable ($n-1=2$). Nakon toga je broj varijabli ukupno proširen na 43. Osim varijable „protocol_type“, jedno vruće kodiranje potrebno je i za varijable „service“ i „flag“. Varijabla „flag“ može poprimiti $n=11$ različitih vrijednosti, što proširuje ulazni skup podataka na 53 varijable, a broj različitih vrijednosti varijable „service“ je $n=70$ pa jednim vrućim

kodiranjem na kraju dobijemo 122 varijable. Za jedno vruće kodiranje korištene su metode *OneHotEncoder* i *ColumnTransformer* iz *scikit-learn* biblioteke:

```
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(), [1, 2, 3])], remainder='passthrough')
X_train = np.array(ct.fit_transform(X_train))
X_test = np.array(ct.transform(X_test))
```

Osim kodiranja ulaznih nominalnih varijabli trening i test skupa podataka *X_train, X_test* potrebno je napraviti kodiranje trening i testne izlazne varijable koja sadrži vrijednosti kategorija koje također nisu prirodno povezane, *y_train, y_test*. Programski kod za kodiranje izlaznih varijabli modela binarne klasifikacije se razlikuje od programskog koda za kodiranje modela višerazredne klasifikacije. Za binarnu klasifikaciju kodiranje se implementira na način da se vrijednost izlazne varijable „normal“ kodira sa „0“, a svaka druga vrijednost sa „1“. Binarna vrijednost „1“ označava da se radi o anomaliji odnosno potencijalnom napadu, a vrijednost „0“ da se ne radi o napadu i da je sve uredu. Kodiranje za višerazrednu klasifikaciju temelji se na tablici 2.

Kodiranje izlaznih varijabli za binarnu klasifikaciju

```
normal_type = 'normal'

for index, item in enumerate(y_train):
  if item == normal_type:
    y_train[index] = 0
  else:
    y_train[index] = 1

for index, item in enumerate(y_test):
  if item == normal_type:
    y_test[index] = 0
  else:
    y_test[index] = 1
```

Kodiranje izlaznih varijabli za višerazrednu klasifikaciju

```
dos_label = 1
dos_types = {'land', 'back', 'pod', 'neptune', 'smurf', 'teardrop', 'apache2', 'udpstorm','processtable', 'worm', 'mailbomb'}


probe_label = 2
probe_types = {'satan', 'ipsweep', 'nmap', 'portsweep', 'mscan', 'saint'}
```

```
R2L_label = 3
R2L_types = {'guess_passwd', 'ftp_write', 'imap', 'phf', 'multihop', 'warezmaster', 'warezclient', 'spy', 'xlock', 'xsnoop', 'snmpguess', 'snmpgetattack', 'sendmail', 'httptunnel', 'named'}


U2R_label = 4
U2R_types = {'rootkit', 'loadmodule', 'sqlattack', 'xterm', 'buffer_overflow', 'perl', 'ps'}


normal_label = 5
normal_type = 'normal'


unknown_label = 6


for index, item in enumerate(y_train):
  if item in dos_types:
    y_train[index] = dos_label
  elif item in probe_types:
    y_train[index] = probe_label
  elif item in R2L_types:
    y_train[index] = R2L_label
  elif item in U2R_types:
    y_train[index] = U2R_label
  elif item == normal_type:
    y_train[index] = normal_label
  else:
    y_train[index] = unknown_label


for index, item in enumerate(y_test):
  if item in dos_types:
    y_test[index] = dos_label
  elif item in probe_types:
    y_test[index] = probe_label
  elif item in R2L_types:
    y_test[index] = R2L_label
  elif item in U2R_types:
    y_test[index] = U2R_label
  elif item == normal_type:
    y_test[index] = normal_label
  else:
```

```
    y_test[index] = unknown_label

from keras.utils.np_utils import to_categorical
y_train = to_categorical(y_train)
y_test = to_categorical(y_test)
```

Nakon kodiranja izlaznih varijabli potrebno ih je na kraju dekodirati kako bi se kao rezultat klasifikacije vidjele stvarne klase umjesto binarnih vrijednosti.

Skaliranje varijabli

Osim označavanja varijabli potrebno je provesti i skaliranje numeričkih varijabli kako određene varijable ne bi svojim vrijednostima bile dominantnije od drugih. Svako vruće kodiranje varijablu stavlja na početak dvodimenzionalnog polja što znači da su prva 84 stupca („protocol_type“ -> 3, „service“ -> 70, „flag“ -> 11) binarne vrijednosti i njih nije potrebno skalirati. Standardizacija i normalizacija su najčešće korištene metode za skaliranje. Standardizacija se računa po formuli:

$$x_{stand} = \frac{x - mean(x)}{standard\ deviation\ (x)},$$

a normalizacija:

$$x_{norm} = \frac{x - min\ (x)}{max(x) - min\ (x)}.$$

Za skaliranje ulaznih varijabli koristi se metoda standardizacije iz biblioteke *scikit-learn.*

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train[:, 84:] = sc.fit_transform(X_train[:, 84:])
X_test[:, 84:] = sc.transform(X_test[:, 84:])
X_train = X_train.astype(np.float32)
X_test = X_test.astype(np.float32)
```

Matrica konfuzije binarne klasifikacije

```
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
accuracy_score(y_test, y_pred)
```

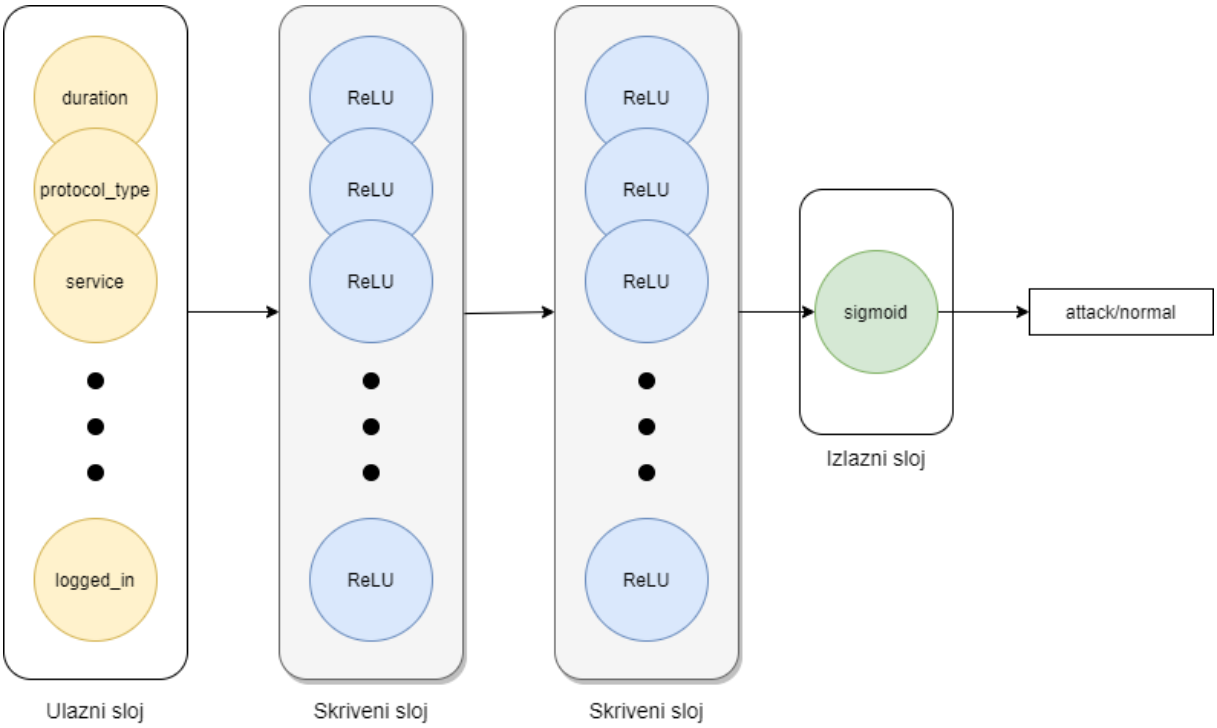
Matrice konfuzije višerazredne klasifikacije

```
from sklearn.metrics import accuracy_score, multilabel_confusion_matrix
mcm = multilabel_confusion_matrix(y_test, y_pred)
cm = confusion_matrix(y_test, y_pred)
accuracy_score(y_test, y_pred)
```

U opisu programske izvedbe pojedinog modela zajednički dio programskog koda je zamijenjen nazivom, npr. ako je u programskoj izvedbi potreban programski kod za matricu konfuzije umjesto cijelog programskog koda koristi se izraz „ `> Matrica konfuzije binarne klasifikacije`“.

4.2. Duboka neuronska mreža

Arhitektura duboke neuronske mreže sastoji se od ulaznog sloja, dva skrivena sloja i izlaznog sloja. U prvom skrivenom sloju nalazi se 16, a u drugom 8 neurona. Neuroni oba skrivena sloja koriste ispravljačku funkciju kao funkciju aktivacije jer se njome postiže najbolji rezultat. Nakon svakog skrivenog sloja primijenjeno je odbacivanje 10% neurona da bi se spriječilo prekomjerno treniranje modela (*eng. overfitting*). Bitno je naglasiti da se neuroni odbacuju slučajnim odabirom. U izlaznom sloju korištena je sigmoidna funkcija jer je potrebno odrediti radi li se o napadu ili ne. Funkcija korištena za računanje pogreške je binarna unakrsna entropija (*eng. binary crossentropy*). Za metriku se traži točnost (*eng. accuracy*). Korištena funkcija optimizacije je *„adam“* jer implementira stohastički gradijenti spust koji rješava problem lokalnih minimuma. Ta funkcija je korištena u svim modelima pa nije posebno objašnjavana prilikom opisivanja drugih modela. Model podešava težine svakih 256 zapisa što je implementirano parametrom *batch_size.* Broj epoha određen je parametrom *epochs* i iznosi 100 jer većim ili manjim brojem epoha nije postignut bolji rezultat. Jedna epoha znači prolaz kroz cijeli skup podataka. Nakon treniranja modela napravljena je klasifikacija, a rezultati i točnost izračunati su matricom konfuzije.



Slika 27. Arhitektura duboke neuronske mreže za binarnu klasifikaciju

Programska izvedba binarne klasifikacije

> Uvoz biblioteka i učitavanje skupa podataka

> Označavanje nominalnih svojstava ulaznih podataka

> Kodiranje izlaznih varijabli za binarnu klasifikaciju

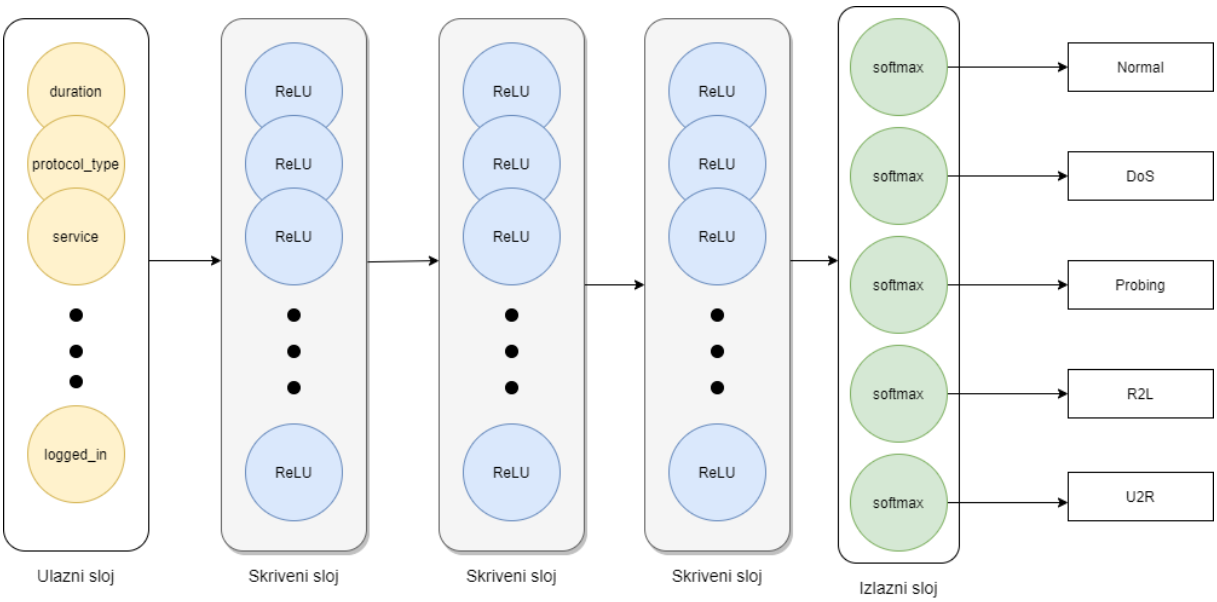
> Skaliranje varijabli

```
from keras.preprocessing import sequence
from keras.models import Sequential
from keras.layers import Dense, Dropout

model = Sequential()
model.add(Dense(units = 16, activation = 'relu'))
model.add(Dropout(0.1))
model.add(Dense(units = 8, activation = 'relu'))
model.add(Dropout(0.1))
model.add(Dense(1, activation = 'sigmoid'))

model.compile(loss = 'binary_crossentropy', optimizer = 'adam', metrics =
['accuracy'])
model.fit(X_train, y_train, validation_data = (X_test, y_test), batch_size =
256, epochs = 100)
```

```
y_pred = model.predict(X_test)
y_pred = (y_pred > 0.5)
```

> Matrica konfuzije za binarnu klasifikaciju

Višerazredna klasifikacija u svojoj arhitekturi za razliku od binarne ima još jedan skriveni sloj zbog toga jer na izlazu treba napraviti višerazrednu klasifikaciju. Prvi skriveni sloj sadrži 32, drugi 16, a treći 8 neurona. Nakon svakog skrivenog sloja je primijenjeno odbacivanje 10% neurona. Funkcija korištena na izlazu je softmax jer na izlazu mora biti pet kategorija. Funkcija za računanje pogreške je kategorička unakrsna entropija (*eng. categorical crossentropy*) proslijeđena u parametar *loss* kao vrijednost *categorical_crossentropy.* Na kraju se računa matrica konfuzije za višerazrednu klasifikaciju.



Slika 28. Arhitektura duboke neuronske mreže za višerazrednu klasifikaciju

Programska izvedba višerazredne klasifikacije

> Uvoz biblioteka i učitavanje skupa podataka
> Označavanje nominalnih svojstava ulaznih podataka
> Kodiranje izlaznih varijabli za višerazrednu klasifikaciju
> Skaliranje varijabli

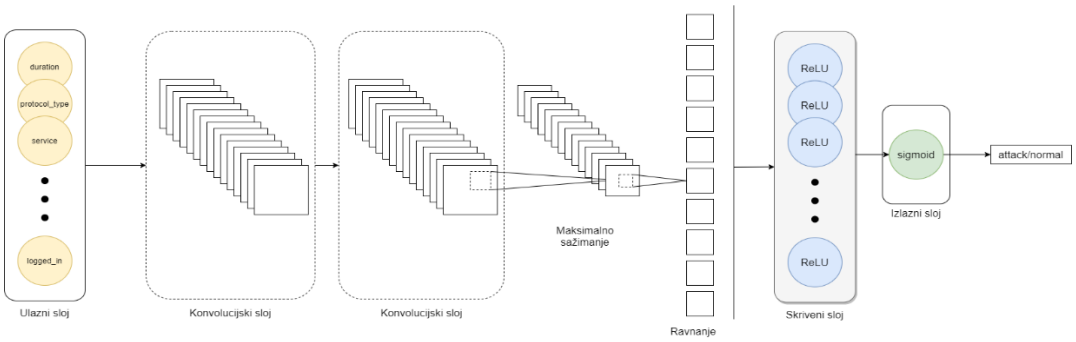
```
from keras.preprocessing import sequence
from keras.models import Sequential
from keras.layers import Dense, Dropout
```

```
model = Sequential()
model.add(Dense(32,activation='relu'))
model.add(Dropout(0.01))
model.add(Dense(16,activation='relu'))
model.add(Dropout(0.01))
model.add(Dense(8,activation='relu'))
model.add(Dropout(0.01))
model.add(Dense(5, activation = 'softmax'))

model.compile(loss = 'categorical_crossentropy', optimizer = 'adam', metrics
= ['accuracy'])
model.fit(X_train, y_train, validation_data = (X_test, y_test), batch_size =
256, epochs = 100)
y_pred = (model.predict(X_test) > 0.5).astype("float32")

y_test = np.argmax(y_test, axis=-1)
y_pred = np.argmax(y_pred, axis=-1)
```

> Matrica konfuzije za višerazrednu klasifikaciju

4.3. Konvolucijska neuronska mreža

Konvolucijska mreža za binarnu klasifikaciju u svojoj arhitekturi ima dva konvolucijska sloja. Nakon drugog sloja izvršava se maksimalno sažimanje koje ravna i potpuno povezuje na skriveni sloj koji završava istim izlaznim slojem kao i duboka neuronska mreža. Konvolucijski slojevi programski su izvedeni kao jednodimenzionalni sa 32 filtera od kojih je svaki duljine 3. Parametar kojim ispunjavamo konvolucijski izlaz, *padding* dobiva vrijednost „same“. To znači da izlaz konvolucijskog sloja u odnosu na ulaz nije potrebno smanjiti. Funkcija aktivacije je ispravljačka. Broj neurona u skrivenom sloju je 16 uz postotak odbacivanja od 50%.



Slika 29. Arhitektura konvolucijske neuronske mreže za binarnu klasifikaciju

Programska izvedba binarne klasifikacije

> Uvoz biblioteka i učitavanje skupa podataka

> Označavanje nominalnih svojstava ulaznih podataka

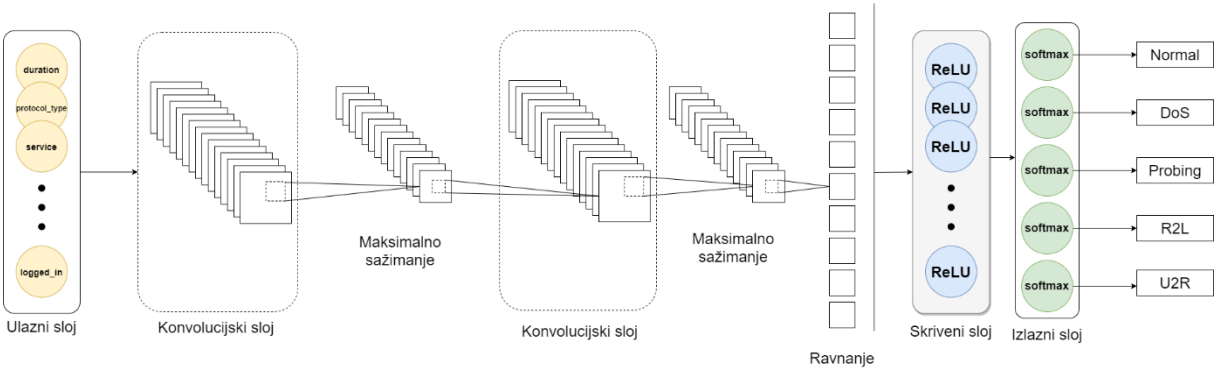
> Kodiranje izlaznih varijabli za binarnu klasifikaciju

> Skaliranje varijabli

```
X_train = np.reshape(X_train, (X_train.shape[0], X_train.shape[1], 1))
X_test = np.reshape(X_test, (X_test.shape[0], X_test.shape[1], 1))

from keras.preprocessing import sequence
from keras.models import Sequential
from keras.layers import Dense, Dropout, Conv1D, MaxPooling1D, Flatten

model = Sequential()
model.add(Conv1D(32, 3, padding = 'same', activation = 'relu'))
model.add(Conv1D(32, 3, padding = 'same', activation = 'relu'))
model.add(MaxPooling1D(pool_size = (2)))
model.add(Flatten())
model.add(Dense(16, activation = 'relu'))
model.add(Dropout(0.5))
model.add(Dense(1, activation = 'sigmoid'))

model.compile(loss = 'binary_crossentropy', optimizer = 'adam', metrics = [
'accuracy'])
model.fit(X_train, y_train, validation_data = (X_test, y_test), batch_size
= 256, epochs = 10)

y_pred = model.predict(X_test)
y_pred = (y_pred > 0.5)
```

> Matrica konfuzije za binarnu klasifikaciju

Konvolucijska neuronska mreža za višerazrednu klasifikaciju nakon svakog konvolucijskog sloja izvršava postupak maksimalnog sažimanja. Nakon drugog maksimalnog sažimanja podaci se ravnaju i prosljeđuju skrivenom sloju od 128 neurona uz postotak odbacivanja od 20%. Taj sloj se na kraju veže na izlazni sloj za višerazrednu klasifikaciju. Točnost je prikazana matricom konfuzije.



Slika 30. Arhitektura konvolucijske neuronske mreže višerazrednu klasifikaciju

Programska izvedba višerazredne klasifikacije

> Uvoz biblioteka i učitavanje skupa podataka
> Označavanje nominalnih svojstava ulaznih podataka
> Kodiranje izlaznih varijabli za višerazrednu klasifikaciju
> Skaliranje varijabli

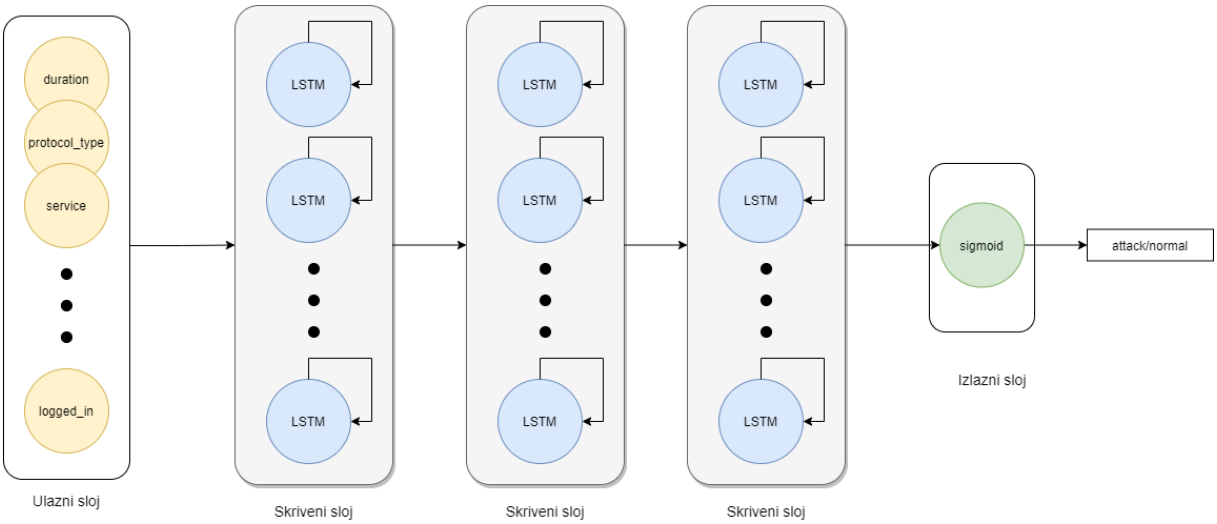
```
X_train = np.reshape(X_train, (X_train.shape[0], X_train.shape[1], 1))
X_test = np.reshape(X_test, (X_test.shape[0], X_test.shape[1], 1))

from keras.preprocessing import sequence
from keras.models import Sequential
from keras.layers import Dense, Dropout, Conv1D, MaxPooling1D, Flatten

model = Sequential()
model.add(Conv1D(128, 3, padding = 'same', activation = 'relu'))
model.add(MaxPooling1D(pool_size = (2)))
model.add(Conv1D(128, 3, padding = 'same', activation = 'relu'))
model.add(MaxPooling1D(pool_size = (2)))
model.add(Flatten())
model.add(Dense(128, activation = 'relu'))
model.add(Dropout(0.2))
```

```
model.add(Dense(5, activation = 'softmax'))


model.compile(loss = 'categorical_crossentropy', optimizer = 'adam', metrics
= ['accuracy'])
model.fit(X_train, y_train, validation_data = (X_test, y_test), batch_size =
256, epochs = 10)


y_pred = (model.predict(X_test) > 0.5).astype("float32")


y_test = np.argmax(y_test, axis=-1)
y_pred = np.argmax(y_pred, axis=-1)
```

> Matrica konfuzije za višerazrednu klasifikaciju

4.4. Povratna neuronska mreža

Arhitektura povratne neuronske mreže joj daje mogućnost spremanja stanja u nekom trenutku i korištenja tog stanja za računanje izlaza kojeg prosljeđuje na sljedeći skriveni sloj. Model povratne neuronske mreže za binarnu klasifikaciju sastoji se od tri skrivena LSTM sloja koji sadrže 16 memorijskih ćelija. Memorijske ćelije se promatraju kao neuroni. Prva dva sloja u programskoj izvedbi imaju parametar *return_sequences* postavljen na vrijednost „True“ jer se na sljedeći sloj mora proslijediti na izlaz sva skrivena stanja. Zadnji sloj taj parametar ima postavljen na vrijednost „False“ jer je za izlazni sloj bitno samo zadnje stanje. Funkcija aktivacije je *tanh.* Zadnji LSTM sloj se spaja na izlazni sloj za binarnu klasifikaciju. S obzirom na veličinu ulaza i izlaza odgovara strukturi više ulaza prema jednom izlazu.



Slika 31. Arhitektura povratne neuronske mreže za binarnu klasifikaciju

Programska izvedba binarne klasifikacije

> Uvoz biblioteka i učitavanje skupa podataka

> Označavanje nominalnih svojstava ulaznih podataka

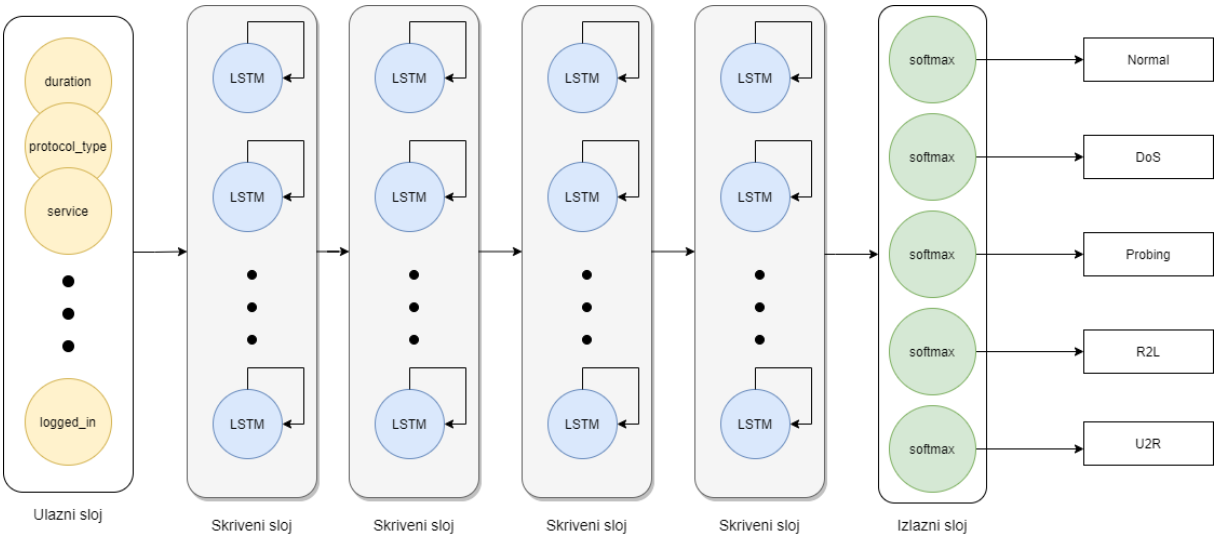
> Kodiranje izlaznih varijabli za binarnu klasifikaciju

> Skaliranje varijabli

```
X_train = np.reshape(X_train, (X_train.shape[0], 1, X_train.shape[1]))
X_test = np.reshape(X_test, (X_test.shape[0], 1, X_test.shape[1]))

from keras.preprocessing import sequence
from keras.models import Sequential
from keras.layers import Dense, Dropout, LSTM

model = Sequential()
model.add(LSTM(16, return_sequences = True))
model.add(Dropout(0.1))
model.add(LSTM(16, return_sequences = True))
model.add(Dropout(0.1))
model.add(LSTM(16, return_sequences = False))
model.add(Dropout(0.1))
model.add(Dense(1, activation = 'sigmoid'))
model.compile(loss = 'binary_crossentropy', optimizer = 'adam', metrics
 = ['accuracy'])
model.fit(X_train, y_train, validation_data = (X_test, y_test), batch_size =
256, epochs = 100)
```

> Matrica konfuzije za binarnu klasifikaciju

Arhitektura modela za višerazrednu klasifikaciju je izgrađena na četiri LSTM sloja. Svaki sloj sadrži 100 neurona uz postotak odbacivanja od 20%. Kao i kod modela za binarnu klasifikaciju svi LSTM slojevi osim zadnjeg imaju parametar *return_sequences* postavljen na vrijednost „*True*“.



Slika 32. Arhitektura povratne neuronske mreže za višerazrednu klasifikaciju

Programska izvedba

> Uvoz biblioteka i učitavanje skupa podataka

> Označavanje nominalnih svojstava ulaznih podataka

> Kodiranje izlaznih varijabli za višerazrednu klasifikaciju

> Skaliranje varijabli

```
X_train = np.reshape(X_train, (X_train.shape[0], 1, X_train.shape[1]))
X_test = np.reshape(X_test, (X_test.shape[0], 1, X_test.shape[1]))

from keras.preprocessing import sequence
from keras.models import Sequential
from keras.layers import Dense, Dropout, LSTM
model = Sequential()
model.add(LSTM(100, return_sequences = True))
model.add(Dropout(0.2))
model.add(LSTM(100, return_sequences = True))
model.add(Dropout(0.2))
model.add(LSTM(100, return_sequences = True))
model.add(Dropout(0.2))
model.add(LSTM(100, return_sequences = True))
model.add(Dropout(0.2))
model.add(LSTM(100))
model.add(Dropout(0.2))
model.add(Dense(5, activation = 'softmax'))
```

```
model.compile(loss = 'categorical_crossentropy', optimizer = 'adam', metrics
= ['accuracy'])
model.fit(X_train, y_train, validation_data = (X_test, y_test), batch_s
ize = 256, epochs = 10)

y_pred = (model.predict(X_test) > 0.5).astype("float32")

y_test = np.argmax(y_test, axis=-1)
y_pred = np.argmax(y_pred, axis=-1)
```

> Matrica konfuzije za višerazrednu klasifikaciju

Nakon prikazane arhitekture i programske izvedbe duboke, konvolucijske i povratne neuronske mreže potrebno je usporediti rezultate klasifikacija binarnih i višerazrednih modela kako bi se vidjelo koja mreža postiže najbolji rezultat.

5. Rezultati i analiza

Neuronske mreže svojom arhitekturom nalaze primjenu u određenim aplikacijskim domenama. U ovom radu je implementirana duboka, konvolucijska i povratna neuronska mreža za otkrivanje napada korištenjem skupa podataka NSL-KDD. Modeli su trenirani sa odgovarajućim funkcijama aktivacije i gubitka.

Programski izvedba je implementirana u alatu *Google Colab* koji za izvršavanje programskog koda koristi resurse *Google cloud* servisa „*Python 3 Google Compute Engine backend*“. Resursi se dodjeljuju po sesiji, a dostupni resursi su RAM memorija i disk. Maksimalna količina RAM memorije koja se može dodijeliti je 12,72 GB, a diska 107,77 GB. Za proces treniranja modela dodijeli se ~0,90 GB RAM memorije. Sa tom količinom RAM memorije proces treniranja modela duboke neuronske mreže za binarnu i višerazrednu klasifikaciju kroz 100 epoha traje ~100 s, a proces testiranja ~3 s. Za treniranje modela konvolucijske neuronske mreže za binarnu klasifikaciju kroz 10 epoha potrebno je ~260 s (4 min 20 s), a za testiranje ~5 s. Treniranje konvolucijskog modela za višerazrednu klasifikaciju kroz 10 epoha traje ~870 s (14 min 30 s), a testiranje ~10 s. Što se tiče procesa treniranja povratne neuronske mreže za binarnu klasifikaciju kroz 100 epoha, on traje ~300 s (5 min), a testiranje ~4 s, dok je za treniranje višerazredne klasifikacije kroz 10 epoha potrebno ~180s (3 min), a za testiranje ~8 s.

Svaki model postiže određenu točnost koja se analizira za modele binarne klasifikacije odvojeno od modela višerazredne klasifikacije. Osim točnosti analiziraju se i neka svojstva matrice konfuzije.

Osnovni elementi matrice konfuzije su broj ispravno pozitivnih (*eng. true positive, TP*), lažno pozitivnih (*eng. false positive, FP*), lažno negativnih (*eng. false negative, FN*) i ispravno negativnih (*eng. true negative, TN*) zapisa. Iz tih vrijednosti mogu se izračunati preciznost ($PPV = \frac{TP}{TP+FP}$) i stopa ispravno pozitivnih ($TPR = \frac{TP}{TP+FN}$).

Tablica 3. Matrice konfuzije za modele mreža binarne klasifikacije

Duboka			Konvolucijska			Povratna		
	attack	*normal*		*attack*	*normal*		*attack*	*normal*
attack	8527	610	*attack*	7307	718	*attack*	8990	721
normal	4306	9101	*normal*	5526	8993	*normal*	4490	8343

Tablica 4. Matrica konfuzije duboke mreže za višerazrednu klasifikaciju

	DoS	Probe	R2L	U2R	Normal
DoS	6265	160	2	0	1033
Probe	342	1480	0	0	599
R2L	7	6	224	0	2648
U2R	16	0	5	0	46
Normal	54	209	1	1	9446

Tablica 5. Matrica konfuzije konvolucijske mreže za višerazrednu klasifikaciju

	DoS	Probe	R2L	U2R	Normal
DoS	6403	30	0	0	1027
Probe	305	1487	57	0	572
R2L	1	12	255	0	2617
U2R	0	0	3	8	56
Normal	424	286	0	0	9001

Tablica 6. Matrica konfuzije povratne mreže za višerazrednu klasifikaciju

	DoS	Probe	R2L	U2R	Normal
DoS	6158	30	0	0	1272
Probe	261	1669	0	0	491
R2L	10	74	253	0	2548
U2R	15	0	20	1	31
Normal	515	205	2	0	8989

Tablica 3 prikazuje matrice konfuzije po modelu neuronske mreže za binarnu klasifikaciju, a u tablice 4, 5, 6 su matrice konfuzije su prikazane za svaki model neuronske mreže posebno. Pomoću vrijednosti tih tablica izračunata su još neka zanimljiva svojstva (*PPV, TPR*) koja su prikazana u tablicama 7 i 8.

Tablica 7. Usporedba binarnih klasifikatora

	Duboka (%)	Konvolucijska (%)	Povratna (%)
Točnost	78,19	72,30	76,89
PPV	93,32	91,05	92,04
TPR	66,76	56,94	65,01

U tablici 3 vidimo da najbolje rezultate postiže duboka neuronska mreža. Druga po redu je povratna neuronska mreža sa 76,89%. I na kraju je konvolucijska sa 72,30%.

Tablica 8. Usporedba višerazrednih klasifikatora

	Duboka (%)	Konvolucijska (%)	Povratna (%)
Točnost	77,42	76,09	75,72
DoS PPV	93,73	89,77	88,49
Probe PPV	79,78	81,93	84,38
R2L PPV	96,55	80,95	92,00
U2R PPV	0,00	100,00	100,00
DoS TPR	83,98	85,83	82,55
Probe TPR	61,13	61,42	68,93
R2L TPR	7,76	8,83	8,77
U2R TPR	0,00	11,94	1,49

Tablica 8 prikazuje točnost modela za višerazrednu klasifikaciju i vrijednosti svojstava *PPV, TPR* izračunatih za svaku klasu posebno.

Osim tehnika dubokog učenja također treba spomenuti da su na NSL-KDD skupu podataka trenirani i algoritmi strojnog učenja. Klasifikacija je napravljena algoritmima linearne regresije, stabla odlučivanja i slučajne šume. Linearnom regresijom se postiže točnost od 85,28%, stablom odlučivanja 99,78%, a algoritmom slučajne šume 99,24%.

6. Zaključak

Informacijska sigurnost ključno je svojstvo koje određuje kvalitetu informacijskog sustava. Konstantno poboljšavati mehanizme i procese za suzbijanje mogućih prijetnji nije nimalo lak zadatak jer napretkom tehnologije napreduju i prijetnje koje mogu učiniti veliku štetu informacijskom sustavu. Načinjena šteta na informacijskom sustavu direktno se odražava na poslovni sustav i njegove procese. Motivirajući se time računalna znanost ulaže velike napore kako bi se na što bolji način suprotstavila sigurnosnim prijetnjama i izazovima. Jedno od područja koje informacijsku sigurnost može dovesti na zavidnu razinu je umjetna inteligencija.

Umjetna inteligencija se kao pojam počinje spominjati još u 1940-tima, ali svoje zlatno razdoblje doživljava od 1956. do 1974. Unatoč istraživanjima tog razdoblja puni potencijal umjetne inteligencije se nije mogao provjeriti jer računalni resursi tada nisu bili dovoljno veliki da bi se mogao učitati veliki skup podataka koji bi umjetnu inteligenciju trenirao. Međutim računalni resursi danas su sposobni spremiti velike količine podataka što je zapravo idealno za učenje umjetne inteligencije. Time je umjetna inteligencija opet postala zanimljiva računalnoj znanosti koja umjetnu inteligenciju primjenjuje na razne aplikacijske domene. U domeni informacijske sigurnosti svoju primjenu nalazi u sustavima za otkrivanje napada. To su sustavi koji vrše dinamičku analizu mrežnog prometa i koristeći se algoritmima strojnog učenja ili tehnikama dubokog učenja otkrivaju radi li se o nekoj anomaliji u sustavu ili ne.

Tema ovog rada naglasak stavlja na tehnike dubokog učenja odnosno neuronske mreže. S gledišta arhitekture i specifičnosti neuronskih mreža podijeljene su na duboke, konvolucijske i povratne. Koristeći se svojstvima i arhitekturom pojedine izgrađeni su modeli za binarnu i višerazrednu klasifikaciju. Nakon treniranja i testiranja rezultati klasifikacija su smješteni u matricu konfuzije. Na osnovu tih rezultata može se zaključiti da za binarnu i višerazrednu klasifikaciju najbolje rezultate postiže duboka neuronska mreža. Modeli su trenirani sa NSL-KDD skupom podataka koji ima 41 svojstvo. Sva svojstva su bitna i sva su dio ulaznih podataka. To znači da bi iz svake mrežne aktivnosti trebalo izvući taj svojstva i proslijediti ih modelu kako bi napravio klasifikaciju i potencijalno otkrio upad ili anomaliju.

Popis literature

[1] Canadian Institute for Cybersecurity (bez dat.) NSL-KDD dataset [Na internetu]. Dostupno: https://www.unb.ca/cic/datasets/nsl.html [pristupano 24.08.2020.].

[3] Google Colab (bez dat.) Intro [Na internetu]. Dostupno: https://colab.research.google.com/notebooks/intro.ipynb [pristupano 24.08.2020.].

[4] TensorFlow (bez dat.) API [Na internetu]. Dostupno: https://www.tensorflow.org/ [pristupano 24.08.2020.].

[5] Keras (bez dat.) API [Na internetu]. Dostupno: https://keras.io/ [pristupano 24.08.2020.].

[6] Scikit learn (bez dat.) API [Na internetu]. Dostupno: https://scikit-learn.org/stable/modules/multiclass.html [pristupano 24.08.2020.].

[7] Zakon o informacijskoj sigurnosti. [Na internetu]. NN 79/07. Dostupno: https://www.zakon.hr/z/218/Zakon-o-informacijskoj-sigurnosti [pristupano 30.08.2020.].

[8] A. Sung, S. Mukkamala, “Identifying important features for intrusion detection using support vector machines and neural networks” in Symposium on Applications and the Internet, pp. 209–216. 2003.

[9] Barracuda (bez dat.) *What is a Intrusion Detection System?* [Na internetu]. Dostupno: https://www.barracuda.com/glossary/intrusion-detection-system [pristupano 30.08.2020.].

[10] John R. Vacca, *Computer and Information Security Handbook*, Morgan Kaufmann Publishers, 2009

[11] „Artifical Intelligence“ (bez dat.), u *Wikipedia, the Free Encyclopedia*. Dostupno: https://en.wikipedia.org/wiki/Artificial_intelligence [pristupano 04.09.2020.]

[12] Simon Haykin, *Neural Networks and Learning Machines*. SAD, Pearson Prentice Hall, 2009

[13] „Neuron“ (bez dat.), u *Wikipedia, the Free Encyclopedia*. Dostupno: https://en.wikipedia.org/wiki/Neuron [pristupano 04.09.2020.]

[14] „Theshold function“ (bez dat.), u *Github*. Dostupno: https://github.com/MachineLearner07/Basic-Machine-Learning-with-Python-and-R/issues/15 [pristupano 04.09.2020.]

[15] „Activation function“ (bez dat.), u *Medium*. Dostupno: https://medium.com/analytics-vidhya/activation-function-c762b22fd4da [pristupano 04.09.2020.]

[16] „Artificial Neural Networks - Backpropagation“ (13.09.2018), u *SuperDataScience* Dostupno: https://www.superdatascience.com/blogs/artificial-neural-networks-backpropagation/ [pristupano 04.09.2020.]

[17] Michael Nielsen (2019), *Using neural nets to recognize handwritten digits, Chapter 1* Dostupno: http://neuralnetworksanddeeplearning.com/chap1.html [pristupano 02.09.2020]

[18] „Learning process of a neural network“ (21.09.2018) u *Towards dana science.* Dostupno: https://towardsdatascience.com/how-do-artificial-neural-networks-learn-773e46399fc7 [pristupano 02.09.2020]

[19] Ethem Alpaydin (2009.), *Introduction to Machine Learning*, The MIT Press

[20] „The Ultimate Guide to Convolutional Neural Networks (CNN)“ (28.08.2018), u *SuperDataScience* Dostupno: https://www.superdatascience.com/blogs/the-ultimate-guide-to-convolutional-neural-networks-cnn [pristupano 05.09.2020.]

[21] Thomas Wood, „Softmax Function“ (bez dat.), u *DeepAI* Dostupno: https://deepai.org/machine-learning-glossary-and-terms/softmax-layer [pristupano 05.09.2020.]

[22] „The Ultimate Guide to Recurrent Neural Networks (RNN)“ (bez dat.), u *SuperDataScience* Dostupno: https://www.superdatascience.com/blogs/the-ultimate-guide-to-recurrent-neural-networks-rnn [pristupano 05.09.2020.]

[23] Christopher Olah, „Understanding LSTM Networks“ (27.08.2015), u *Github* Dostupno: http://colah.github.io/posts/2015-08-Understanding-LSTMs/ [pristupano 05.09.2020.]

[24] L.Dhanabal, S.P. Shantharajah, "A Study on NSL-KDD Dataset for Intrusion Detection System Based on Classification Algorithms", International Journal of Advanced Research in Computer and Communication Engineering, sve. 4, izdanje 6, pro. 2015, [Na internetu]. Dostupno: https://www.ijarcce.com/upload/2015/june-15/IJARCCE%2096.pdf [pristupano 06.09.2020].

[25] „Diagram with anyone, anywhere.“ (bez dat.) u *Diagrams.net*. Dostupno: https://www.diagrams.net/ [pristupano 06.09.2020.]

Popis slika

Popis tablica