Inteligentna dijagnostika kvarova rotacijske opreme male brzine vrtnje

Kežman, Domagoj

Master's thesis / Diplomski rad

2019

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Zagreb, Faculty of Mechanical Engineering and Naval Architecture / Sveučilište u Zagrebu, Fakultet strojarstva i brodogradnje**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:235:199959

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-04-20**

Repository / Repozitorij:

Repository of Faculty of Mechanical Engineering and Naval Architecture University of Zagreb

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

Domagoj Kežman

Zagreb, 2019. godina.

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

Mentori:

Prof. dr. sc. Dragutin Lisjak

Student:

Domagoj Kežman

Zagreb, 2019. godina.

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

Zahvaljujem se prof. dr. sc. Dragutinu Lisjaku na ukazanom povjerenju, pomoći i savjetima tijekom izrade rada,

Također bi se zahvalio kolegi asistentu Davoru Kolaru na literaturi i pomoći pruženoj tijekom izrade rada,

Zahvaljujem se svojoj obitelji, bez čije bi podrške i razumijevanja bilo puno teže privesti ovaj studij kraju.

Domagoj Kežman

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

Sveučilište u Zagrebu Fakultet strojarstva i brodogradnje	
Datum:	Prilog:
Klasa:	
Ur. broj:	

DIPLOMSKI ZADATAK

Student: **DOMAGOJ KEŽMAN** Mat. br.: 0035196776

Naslov rada na hrvatskom jeziku: **Inteligentna dijagnostika kvarova rotacijske opreme male brzine vrtnje**

Naslov rada na engleskom jeziku: **Intelligent fault diagnosis in low-speed rotating machinery**

Opis zadatka:

Analiza vibracija rotacijske opreme pri srednjim i velikim brzinama vrtnje danas praktički predstavlja industrijski standard održavanja po stanju. Rotacijska oprema male brzine vrtnje primjenjuje se u raznim industrijama kao što su čeličane, rudnici ugljena, energane, biološka postrojenja, razna oprema širokog spektra primjene npr. dizalice, bageri itd. Pri malim brzinama vrtnje, povećan je omjer između signala šuma i signala potrebnih za dijagnostiku, zbog čega je otežano izdvajanje karakterističnih značajki i analiza vibracijskog opterećenja. Jedno od mogućih rješenja navedenog problema može biti primjena dubokog strojnog učenja. U skladu s navedenim, u radu je potrebno:

1. Detaljno opisati strategiju održavanja po stanju.
2. Definirati eksperimentalni postav.
3. Prikupiti podatke karakterističnih tipova kvarova rotacijske opreme male brzine vrtnje.
4. Razviti računalni model procjene kvarova rotacijske opreme male brzine vrtnje primjenom dubokog strojnog učenja te evaluirati performanse na testnom skupu podataka.
5. Analizom dobivenih rezultata izvesti zaključke.

U radu je potrebno navesti korištenu literaturu i eventualno dobivenu pomoć.

Zadatak zadan:
02. svibnja 2019.

Rok predaje rada:
04. srpnja 2019.

Predviđeni datum obrane:
10. srpnja 2019.
11. srpnja 2019.
12. srpnja 2019.

Zadatak zadao:

prof. dr. sc. Dragutin Lisjak

Predsjednica Povjerenstva:

prof. dr. sc. Biserka Runje

SADRŽAJ

1. UVOD ... 1
2. ODRŽAVANJE ... 3
2.1. Korektivno održavanje ... 5
2.2. Preventivno održavanje po konstantnom ciklusu ... 5
2.3. Održavanje po stanju ... 7
2.4. Mjerenje i analiza vibracija ... 15
3. NEURONSKE MREŽE ... 19
3.1. Umjetne neuronske mreže ... 20
3.1.1. Učenje neuronske mreže ... 21
3.1.2. Aktivacijske funkcije ... 23
3.1.3. Optimizatori ... 26
3.1.4. Generaliziranje mreže ... 28
3.1.5. Algoritam povratnog prostiranja pogreške ... 30
3.2. Duboko strojno učenje ... 33
3.3. Smještaj dubokog strojnog učenja unutar područja umjetne inteligencije ... 34
3.4. Primjena dubokog strojnog učenja ... 39
4. KONVOLUCIJSKE NEURONSKE MREŽE ... 41
4.1. Konvolucija ... 44
4.2. Sloj sažimanja ... 47
4.3. Dopunjavanje ... 48
4.4. Algoritam povratnog prostiranja pogreške ... 49
5. OPTIMIZACIJA HIPERPARAMETARA MREŽE ... 52
6. EKSPERIMENTALNI POSTAV ... 57
6.1. Simulator kvarova SpectraQuest Simulator Expert ... 58
6.2. Troosni IEPE akcelerometar IMI Sensors 356B21 ... 61
6.3. National Instruments NI 9191+NI9234+LabView2014 ... 62
7. RAZVOJ RAČUNALNOG MODELA I REZULTATI ... 66
7.1. Programsko sučelje ... 66
7.2. Konvolucijska neuronska mreža i odabir hiperparametara ... 66
7.3. Rezultati konvolucijske neuronske mreže ... 72
8. ZAKLJUČAK ... 80
LITERATURA ... 82

POPIS SLIKA

Slika 1. Krivulja kade [5] 6
Slika 2. Podjela preventivnog održavanja [4] 7
Slika 3. Ilustracija principa održavanja po stanju [7] 9
Slika 4. Otkrivanje uzroka kvara analizom frekvencijskog spektra [6] 15
Slika 5. Kompresijski piezoelektrični akcelerometar [1] 17
Slika 6. Odnos vremenske i frekvencijske domene [1] 18
Slika 7. Usporedba biološkog (gore) i matematičkog (dolje) modela neurona [9] 19
Slika 8. Slojevi umjetne neuronske mreže [10] 21
Slika 9. Prikaz tipova generalizacije [12] 29
Slika 10. Prikaz najbolji rezultata na ImageNet-ovom natjecanju kroz godine [15] 34
Slika 11. Duboko učenje – izvlačenje značajki podataka [12] 37
Slika 12. Usporedba procesa učenja kod strojnog i dubokog učenja [19] 37
Slika 13. Odnos preciznosti i količine podataka kod dubokog i strojnog učenja [20] 39
Slika 14. Arhitektura LeNet-5 [20] 44
Slika 15. Pretvorba ulaza u izlaze u konvolucijskom sloju CNN-a [21] 45
Slika 16. Sažimanje uprosječavanjem i sažimanje maksimumom [21] 47
Slika 17. Algoritam povratnog prostiranja pogreške [22] 51
Slika 18. Dijagram tijeka eksperimenta 58
Slika 19. Simulator kvarova SpectraQuest Simulator Expert 59
Slika 20. IEPE akcelerometar IMI Sensors 356B21 62
Slika 21. National Instruments NI 9191+NI9234 62
Slika 22. Prikaz korisničkog sučelja u LabVIEW-u 64
Slika 23. Dio koda u kojem se zadaju hiperparametri mreže 70
Slika 24. Prikaz dijela koda u MATLAB-u koji definira slojeve mreže 72
Slika 25. Dijagram točnosti neuronske mreže (150 o/min) 73
Slika 26. Dijagram točnosti neuronske mreže (300 o/min) 74
Slika 27. Vrijeme učenja neuronske mreže (150 o/min) 75
Slika 28. Vrijeme učenja neuronske mreže (300 o/min) 75
Slika 29. Prikaz točnosti i gubitka mreže u 15 epoha 76
Slika 30. Različite varijante prikaza točnosti i gubitka [26] 77
Slika 31. Primjenjivost mreže na drugoj brzini vrtnje 78
Slika 32. Prikaz matrica zabune 79

POPIS TABLICA

Tablica 1. Prednosti i nedostaci preventivne strategije održavanja [4] 7
Tablica 2. Prednosti i nedostaci kontinuiranog i periodičkog nadziranja stanja [5] 10
Tablica 3. Prednosti i nedostaci održavanja po stanju [5] 14
Tablica 4. Aktivacijske funkcije [11] 24
Tablica 5. Usporedba 3 varijacije gradijentnog spusta [11] 27
Tablica 6. Usporedba prednosti i nedostataka strojnog i dubokog učenja [10] 38
Tablica 7. Moduli za simuliranje kvarova 60
Tablica 8. Količina i omjer podataka za trening, validaciju i test 67
Tablica 9. Veličine i broj jezgri po slojevima s obzirom na faktor k 67
Tablica 10. Hiperparametri konvolucijske neuronske mreže 69
Tablica 11. Struktura i parametri troslojne neuronske mreže (k=2) 70
Tablica 12. Točnost neuronske mreže 73
Tablica 13. Vrijeme učenja neuronske mreže 74
Tablica 14. Primjenjivost mreže na drugoj brzini vrtnje 78

POPIS OZNAKA

Oznaka	Jedinica	Opis
a_i		Vrijednost elementa u ulaznoj matrici
b		pomak
E		pogreška
f	Hz	frekvencija
$g^{(l)}$		aktivacijska funkcija u sloju l
i		indeks koji označava čvorove u prethodnom sloju *(l-1)*
j		indeks koji označava čvorove u trenutnom sloju l
$\tilde{J}(w)$		Funkcija gubitka nakon regularizacije
$J(w)$		Funkcija gubitka
$\nabla J(w_n)$		Gradijent funkcije gubitka
k_i		Vrijednost elementa u jezgri
n_B		Veličina mape značajki
n_K		Veličina jezgre
p		broj slojeva dopunjavanja
s		Korak jezgre
t	s	vrijeme
t_j		očekivani izlaz neurona
$w_{ij}^{(l)}$		težina koja spaja čvor i u sloju l-*1* i čvor j u sloju l
$w_{ij}^{(1)}$		Težina prije iteracije
$w_{ij}^{(2)}$		Težina nakon iteracije
w^T		transponirani vektor težina
x		vektor ulaza x
x_{opt}		Točka koja maksimizira ili minimizira funkciju cilja
y		Izlaz iz aktivacijske funkcije
$y_j^{(l)}$		aktivirani izlaz neurona j u sloju l
$z_j^{(l)}$		svi ulazi u čvor (neuron) j u sloju l
α		Koeficijent kod propusne zglobne aktivacijske funkcije
β		Koeficijent kod Swish aktivacijske funkcije
η		Stopa učenja
λ		Parametar regularizacije

SAŽETAK

Održavanje tehničkih sustava u industriji jedna je od najvažnijih djelatnosti svakog poduzeća. Kako se tehnologija razvija, tako i procesi održavanja postaju sve važniji te se ciljevi održavanja redefiniraju od otklanjanja kvarova nakon nastanka prema poduzimanju mogućih radnji za izbjegavanje nastanka kvara koji bi prouzročio havariju. Za to se može koristiti strategija održavanja po stanju, opisana u ovom radu. Održavanje po stanju je simulirano mjerenjem i prikupljanjem podataka o vibracijama rotacijske opreme male brzine vrtnje na simulatoru kvarova. Cilj rada je bio na temelju prikupljenih podataka sa simulatora kvarova razviti računalni model koji će procjenjivati potencijalne kvarove na rotacijskoj opremi male brzine vrtnje. Za modeliranje je korištena konvolucijska neuronska mreža. Promatralo se kako promjene određenih hiperparametara utječu na točnost i vrijeme treniranja mreže. Ukupno su istrenirane 144 konvolucijske neuronske mreže te točnost u najboljoj mreži iznosi 100 %, a u najlošijoj 51 %.

Ključne riječi: Održavanje po stanju, duboko učenje, konvolucijske neuronske mreže, strojno učenje, vibracije

SUMMARY

Maintenance of technical systems in the industry is one of the most important activities in a company. As technology develops, maintenance processes become more important because the maintenance goals are being redefined from eliminating failures after their occurrence to taking actions to avoid the occurrence of a fault that can cause breakdowns. To do this companies use condition based maintenance strategy described in this paper. Condition based maintenance was simulated by measuring and collecting vibration data of small speed rotation equipment on the fault simulator. The aim of this thesis was based on collecting the data from the failure simulator in order to develop a computer model that will evaluate potential faults in low-speed rotating machinery. A convolutional neural network was used for modeling. It was observed how changes in certain hyperparameters affect the accuracy and time needed to train the network. A total of 144 convolutional neural networks were trained and the accuracy in the best network was 100 % and in the worst 51 %.

Key words: condition based maintenance, deep learning, convolutional neural networks, machine learning, vibrations

1. UVOD

Količina digitalnih podataka koji se stvaraju u svijetu se udvostručuje svake dvije godine. Kako bi se iskoristila i obradila tolika količina podataka u industriji se razvijaju sve inteligentnije tehnike za njihovu analizu i obradu. Noviji pristup analizi podataka unutar područja umjetne inteligencije razvio se iz strojnog učenja, a zove se duboko učenje. Duboko učenje je koncept temeljen na neuronskim mrežama sa sposobnošću procesuiranja, učenja i predviđanja velikih količina podataka.

Ubrzani razvoj tehnologije u zadnjih dvadesetak godina otvorio je prostor iskorištavanju metoda strojnog učenja, kao što su duboke neuronske mreže, u postupcima održavanja u industrijskim poduzećima. U ovom radu će središte zanimanja biti na primjeni dubokog učenja na strategiju održavanja po stanju. Strategija održavanja po stanju se zasniva na nastojanjima da se uoče i uklone potencijalni kvarovi prije nego se dogodi havarija.

Duboko učenje je iskoristilo mnoštvo danas dostupnih podataka, sposobnost strojnog učenja da uči na temelju podataka i kapacitet dubokih neuronskih mreža da te iste podatke obradi te je postalo jedna od najpopularnijih disciplina u području umjetne inteligencije danas. U ovom radu je koncentracija na utjecaj i iskoristivost inteligentnih tehnika dijagnostike kvarova na održavanje u poduzeću.

U drugom poglavlju će se opisati opća podjela strategija održavanja. Objasnit će se koristi primjene strategije održavanja po stanju u industrijskom poduzeću, kao i načini mjerenja tih stanja.

U trećem poglavlju će se objasniti novi trend u strojnom učenju – duboko učenje. Usporedit će se sa strojnim učenjem te u kakvoj je vezi s umjetnom inteligencijom. Također će biti navedeni primjeri primjene.

Zatim, u četvrtom poglavlju slijedi objašnjenje kako su živčane stanice u mozgu doprinijele razvitku umjetnih neuronskih mreža. Objasnit će se: sastavni dijelovi svake neuronske mreže, proces koji se odvija kad mreža uči te kako se postiže zadovoljavajući rezultat na testnom skupu podataka.

U petom poglavlju bit će pobliže opisana neuronska mreža koja spada u skupinu najčešće korištenih – konvolucijska neuronska mreža. Opisat će se struktura mreže i način na koji joj svaki sloj doprinosi. Isto kao i kod umjetne neuronske mreže opisat će se proces učenja.

Šesto poglavlje se nastavlja na konvolucijsku neuronsku mrežu. U njemu se pobliže objašnjavaju veličine koje mreža ne može sama naučiti – hiperparametri. Za razliku od parametara koje mreža sama uči, kod hiperparametara glavnu ulogu ima analitičar jer ih on izabire. O iznosima hiperparametara ovisi uspjeh mreže. Kako bi se ubrzao i poboljšao njihov izbor te smanjila odgovornost analitičara, koriste se različite metode optimizacije koje će biti objašnjene u radu.

Sedmo poglavlje donosi opis eksperimentalnog postava na temelju kojeg su prikupljeni podaci za daljnju obradu.

Osmo poglavlje opisuje računalni model za predviđanje stanja i dobivene rezultate.

2. ODRŽAVANJE

Prema definiciji Europske federacije nacionalnih udruženja održavatelja (engl. *European Federation of National Maintenance Societies*, EFNMS), održavanje je funkcija poduzeća kojoj su povjerene stalna kontrola nad postrojenjima i obavljanje određenih popravaka i revizija, čime se omogućava stalna funkcionalna sposobnost i očuvanje proizvodnih i pomoćnih postrojenja te ostale opreme.

Kako bi se održavanje provelo na kvalitetan način u poduzeću potrebno je posjedovanje znanja u ovim poljima [1]:

- Inženjerstvo. Propadanje imovine ovisi jednim dijelom o dizajnu i proizvodnji iste. Loše konstruiran proizvod dovodi do niske pouzdanosti što zauzvrat dovodi do visoke potrebe za korektivnim postupcima održavanja. S druge strane ispravno konstruiran proizvod je pouzdaniji i manje sklon kvarovima.
- Znanost. Potrebno je poznavanje fizikalnih mehanizama koji imaju značajan utjecaj na propadanje opreme.
- Ekonomija. Troškovi održavanja mogu zauzimati značajan dio ukupnog operacijskog budžeta, ovisno o tipu industrije. Troškovi se mogu podijeliti na izravne (rad, materijal, itd.) i neizravne (posljedice kvarova).
- Pravo. Potrebno je formulirati ugovorom koja je strana odgovorna za koje postupke ako se za održavanje opreme provodi „out-sourcing“ ili ako se oprema održava na „leasing“.
- Statistika. Propadanje i kvarovi se događaju na neizvjestan način. Analiza podataka održavanja zato zahtjeva uporabu statističkih tehnika kako bi se izvukle informacije iz podataka.
- Operacijska istraživanja. Pružaju alate i tehnike za izradu modela, analizu i optimizaciju. Često analitički postupci nisu uspješni te je potrebno provesti simulaciju kako bi se odabrala najbolja strategija održavanja.
- Teorija pouzdanosti. Interdisciplinarna upotreba: vjerojatnosti, statistike, stohastičkih modela, inženjerstva i znanstvenog razumijevanja mehanizama po kojima se događaju kvarovi kako bi se utvrdila pouzdanost opreme.
- Informacijske tehnologije i računalna znanost. Rad i održavanje kompleksnih sustava generira velike količine podataka. Potrebno je stvoriti sustave koji spremaju i upravljaju podacima kako bi se iz podataka izvukle potrebne informacije. Računalna znanost daje

na korištenje široku lepezu tehnika umjetne inteligencije za rješavanje problema održavanja: rudarenje podataka, ekspertni sustavi, neuronske mreže, itd.

Suprotno ustaljenom mišljenju, uloga održavanja nije rješavanje zastoja u rekordnom vremenu, nego sprječavanje svih gubitaka uzrokovanih problemima s opremom ili sustavom. Misija odjela održavanja u organizaciji je postizanje i održavanje sljedećeg [2]:

- Optimalna dostupnost
- Optimalni radni uvjeti
- Maksimalna iskorištenost resursa za održavanje
- Optimalan radni vijek opreme
- Minimalan inventar rezervnih dijelova
- Sposobnost brze reakcije na nastali problem u sustavu

Pristup održavanju se bitno promijenio u posljednjem stoljeću. U samim začecima industrijske proizvodnje postojala je samo korektivno održavanje, što je značilo vraćanje radnog sustava u funkcionalno stanje provođenjem postupaka popravljanja i zamjenjivanja dijelova sustava koji su prouzročili kvar. Izvodili su ga obučeni tehničari i tad je održavanje smatrano „nužnim zlom“. U zadnjih 100 godina se fokus s održavanja nakon kvara premjestio na provođenje postupaka namijenjenih produljenju životnog vijeka opreme, nadziranja njenog stanja pa sve do predviđanja životnog vijeka opreme. U drugom svjetskom ratu dolazi do razvitka preventivne strategije održavanja. Preventivno održavanje uključuje dodatne početne troškove te se provodi samo u slučaju kad je krajnja korist veća od troškova. U to vrijeme dolazi do većeg fokusiranja na razmišljanje o održavanju proizvoda prilikom faze njegovog konstruiranja. Napretkom tehnologije kao što su: novi materijali, razvoj senzora za nadzor stanja, skupljanje i analiziranje podataka – ostvaruju se pretpostavke za daljnji razvoj održavanja kao što je strategija održavanja po stanju. Održavanje po stanju koristi podatke prikupljene senzorima s radnih sustava kako bi se isplaniralo koji dijelovi opreme će se zamijeniti te u kojem trenutku u vremenu će doći do tog postupka. Ova strategija dovodi do veće kvalitete proizvoda i zaustavljanja proizvodnje samo kad su razlozi za to potkrijepljeni informacijama.

Uloga održavanja u dugoročnoj profitabilnosti organizacije već je dugo prepoznata što je navelo znanstvenike i radnike u industriji na razvoj strategija održavanja koje doprinose dugoročnoj profitabilnosti. Strategije održavanje se mogu razvrstati u dvije osnovne kategorije [3]:

1. Korektivno održavanje (engl. *corrective maintenance*)
2. Preventivno održavanje (engl. *preventive maintenance*)

2.1. Korektivno održavanje

Korektivno održavanje je prva i najjednostavnija strategija koja se pojavila u održavanju tehničkih sustava. Temelji se na otklanjanju oštećenja i kvarova nakon što se oni pojave. Korištenjem ove strategije održavanja omogućava se provođenje proizvodnih procesa bez potrebe za čestim stajanjima uslijed zamijene ili popravka dijelova. No problem nastaje u trenutku kad se pojavi potreba za postupkom korektivnog održavanja. Kad dođe do potrebe za korektivnim održavanjem često to znači da je kvar katastrofalan jer zahvaća i ostale, zdrave komponente strojeva kao i druge opreme. Kvarovi su zbog nedostatka sustava za nadziranje iznenadni što dovodi do duljih vremena potrebnih za popravak i vraćanja opreme u rad. Uslijed toga, dolazi do većih poslovnih gubitaka. U današnje se vrijeme ova strategija održavanja koristi na strojevima na kojima je isplativije čekati njihov kvar pa ih ili popraviti ili zamijeniti, nego trošiti novce na nadziranje njihovog stanja. Koristi se npr. u postrojenjima u kojima postoji velika količina malih strojeva gdje je kvar stroja i njegovo izuzimanje iz rada normalna pojava.

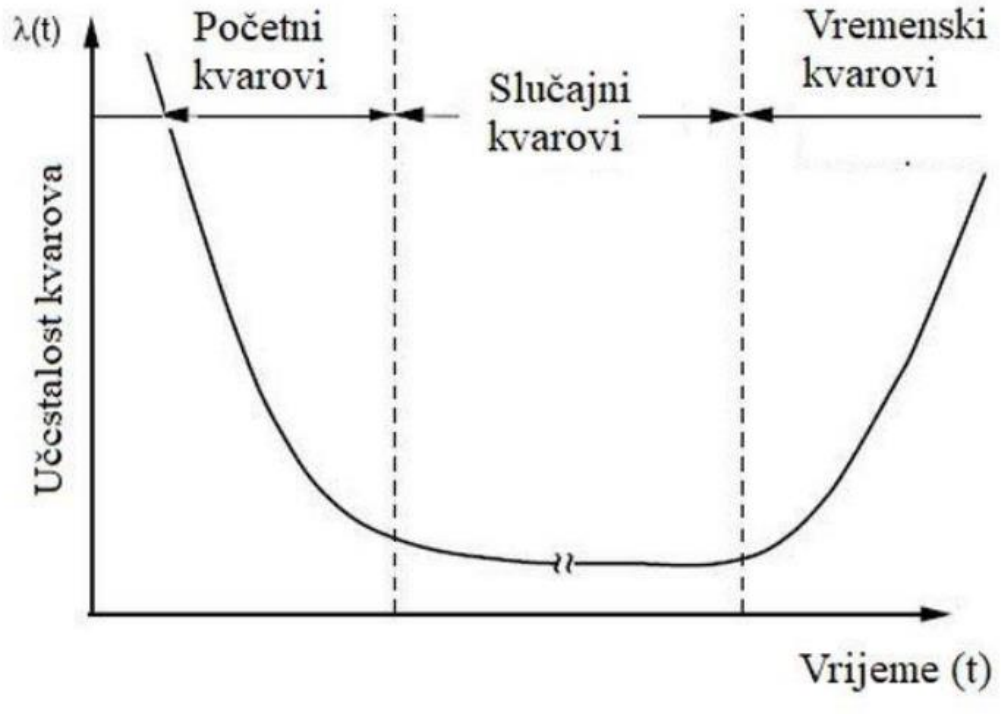
Nedostaci korektivnog održavanja su [4]:

- Čekanje na kvar predstavlja rizik kako po pitanju sigurnosti tako i oštećenja druge opreme te havarijskih kvarova
- Nema nadzora i pouzdanosti u radu postrojenja
- Veći proizvodni gubici u vremenu i sredstvima zbog neočekivanih i duljih zastoja
- Potreba osiguranja rezervne opreme
- Potreban veći broj zaposlenika na održavanju u rezervi i pripremi za kvar
- Dulje vrijeme popravka
- Veći zahtjevi za količinama rezervnih dijelova na skladištu

2.2. Preventivno održavanje po konstantnom ciklusu

S obzirom na troškove uslijed pojavljivanja neočekivanih zastoja u proizvodnji postojala je potreba za novom strategijom održavanja koja će težiti smanjivanju broja neočekivanih zastoja. Preventivno održavanje po konstantnom ciklusu se temelji na provođenju niza aktivnosti održavanja po unaprijed utvrđenom planu, prije nego što dođe do pojave kvara. Operacije održavanja se kod ove strategije planiraju i izvršavaju na temelju statističkih značajki dobivenih iz prošlih kvarova strojeva i njegovih komponenti. Intervali u kojima se provodi moraju biti kraći od očekivanih intervala rada stroja bez pojave kvara. Uobičajeno je uzeti interval unutar kojeg će se samo 1-2 % strojeva pokvariti. To za posljedicu ima činjenicu da je velika većina strojeva mogla biti u radu 2 ili 3 puta dulje nego što je to bila uslijed obavljanja postupaka

preventivnog održavanja [5]. Radi se zamjena dijelova stroja koji su vjerojatno još uvijek ispravni, što dovodi do gubitka uslijed neiskorištavanja punog radnog vijeka komponenti koje se zamjenjuju. Uslijed čestog zamjenjivanja starijih dijelova novima može doći do pojave početnih kvarova uzrokovanih tzv. „dječjim bolestima“ vidljivih na slici 1. To su kvarovi koji se pojavljuju na samom početku rada opreme, a nastali su uslijed: loše konstrukcije, loše montaže ili lošeg materijala.



Slika 1. Krivulja kade [5]

Krivulja kade opisuje učestalost kvarova opreme u vremenu. Početni kvarovi ili „dječje bolesti“ se javljaju pri samom začetku rada stroja. Zatim slijedi vrijeme normalnog rada u kojem se kvarovi ne javljaju često. Nakon određenog vremena se povećava vjerojatnost pojave kvarova te je prije toga potrebno izvršiti zamjenu komponente u kvaru.

Za razliku od korektivnog održavanja, kod preventivne strategije održavanja postoji manja šansa za velikim, katastrofalnim kvarovima jer se održavanje planira unaprijed. No većina strojnih dijelova se ne kvari u točno izračunatim intervalima, nego postoji velika varijacija u njihovom očekivanom radnom vijeku (npr. ležajevi). Varijacije se javljaju zbog različitih radnih uvjeta stroja u vremenu, ali ovise i o vrsti komponente koja se može pokvariti. Nemaju sve komponente stroja usku varijaciju očekivanog vijeka trajanja. Slijedom toga, isplativije bi bilo snimati stanje takve opreme te na temelju očitanih rezultata mjerenja donositi odluku o postupcima održavanja.

U tablici 1 su prikazane prednosti i nedostaci preventivne strategije održavanja:

Tablica 1. Prednosti i nedostaci preventivne strategije održavanja [4]

PREVENTIVNO ODRŽAVANJE	
PREDNOSTI	NEDOSTACI
Kvalitetnije upravljanje proizvodnim procesom	Mogućnost pojave oštećenja
Osigurana željena kvaliteta proizvoda	Visoka razina ranih kvarova
Povećanje produktivnosti proizvodnog sustava	Neiskorišteni raspoloživi resursi
Veća iskoristivost raspoloživih resursa	Visoki početni troškovi
Povećanje efikasnosti tehničkih sustava	Česti prekidi procesa eksploatacije sustava
Normiranje poslova održavanja	
Smanjenje prekovremenog rada	
Racionalno planiranje rezervnih dijelova	
Povećanje sigurnosti i bolja kontrola zagađenja okoliša	

Na slici 2 je prikazana podjela preventivne strategije održavanja. Dijeli se na: održavanje po konstantnom ciklusu, održavanje po stanju, proaktivno održavanje, kontrolne preglede.

Slika 2. Podjela preventivnog održavanja [4]

U idućem poglavlju bit će opisana primjena preventivne strategije održavanja po stanju. Ista strategija će kasnije biti primijenjena u eksperimentalnom dijelu rada.

2.3. Održavanje po stanju

Većina tehničkih sustava ne gubi svoje funkcionalne sposobnosti odjednom, nego je to kontinuirani proces. Oštećenja, kvarovi i havarije su posljedice laganog trošenja tijekom

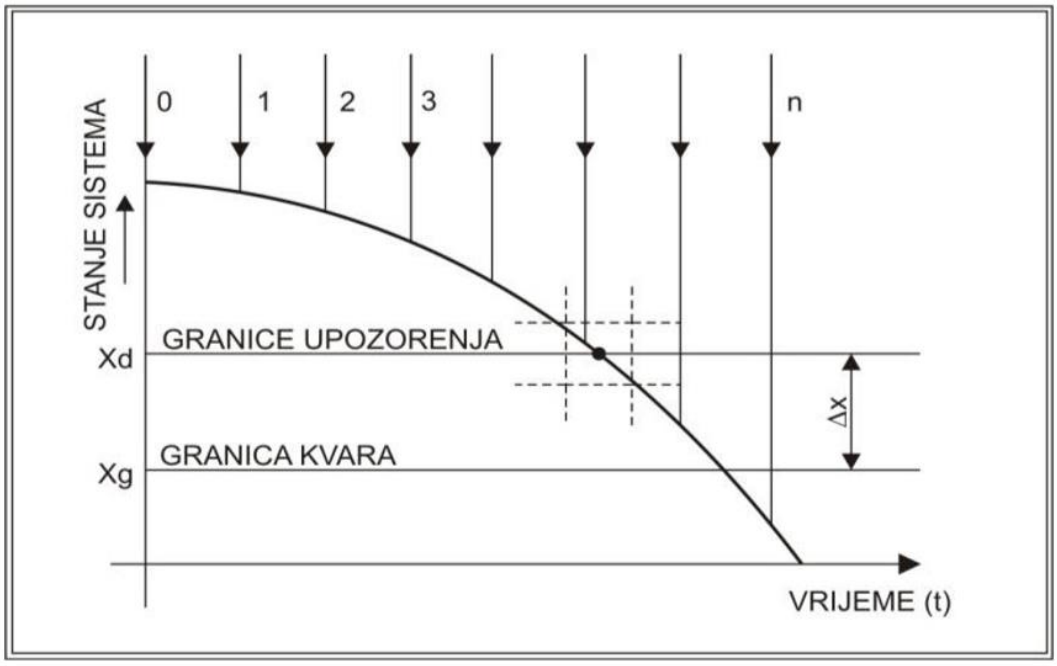
eksploatacije, a njihov nagovještaj se može pojaviti znatno ranije. Postepena degradacija tehničkih sustava se mjeri senzorima koji snimaju stanje sustava na temelju kojeg inženjer održavanja odlučuje postoji li potreba za popravkom ili sustav može i dalje nesmetano raditi. Potreba za popravkom se iskazuje vjerojatnošću pojave kvara. Ovakav pristup održavanju se naziva održavanje po stanju (engl. *condition based maintenance,* CBM).

Održavanje po stanju je dijagnostički proces kojim se određuje stanje ("zdravlje") svakog dijela tehničkog sustava kojeg se može mjeriti i čije se ponašanje može kontrolirati određenim parametrima. Cilj održavanja po stanju je razvijanje odgovarajućih metoda, postupaka i opreme za mjerenje određenih parametara radnog sustava koji ukazuju na pojavu odstupanja od normalnog rada, tj. očekivanu pojavu oštećenja ili kvara. [6]

Razlika u usporedbi s preventivnim održavanjem po konstantnom ciklusu je u tome što ne postoji unaprijed određeni raspored izvođenja popravaka nego to ovisi o stanju sustava. Održavanje po stanju se može smjestiti između preventivnog i korektivnog održavanja jer se teži postizanju maksimalno mogućeg vremenskog razmaka između popravaka, a da ne dođe do zastoja uslijed kvara.

Iako se ova strategija koristi zadnjih 30-40 godina, tek je u zadnje vrijeme došlo do bržeg razvoja, povodom razvoja tehnika za nadziranje stanja. U zadnjih 15 godina održavanje po stanju je prepoznato kao najbolja strategija održavanja za većinu slučajeva. Najveći uspjeh je isprva bio postignut u industrijama u kojima bi rad strojeva obuhvaćao dulje vremenske periode bez prekida pri konstantnoj brzini vrtnje s konstantnim opterećenjem. U novije vrijeme, razvojem tehnologije, nadziranje stanja opreme se može provoditi i na strojevima s promjenjivom brzinom i opterećenjem. [5]

Kod održavanja po stanju prate se parametri sustava te do intervencija dolazi kad iznos određenog parametra izlazi izvan granica koje predstavljaju funkcioniranje sustava u normalnom stanju. Na slici 3 je vidljiv prikaz stanja sustava u vremenu.



Slika 3. Ilustracija principa održavanja po stanju [7]

Stanje sustava u vremenu se može pratiti periodički (mjerenje parametara u određenim vremenskim intervalima) ili kontinuirano (parametri se mjere cijelo vrijeme rada stroja). Slika 3 prikazuje jedan od mogućih slučajeva zakonitosti promjene parametara stanja s dinamikom "provjere stanja", gdje su definirane i granice upozorenja (x_d) i granice kvara (x_g) koje se utvrđuju pokusima i prezentiraju se u normativno-tehničkoj dokumentaciji sustava. Granica upozorenja predstavlja tzv. dopuštenu vrijednost parametara stanja (x_d), a određuje se kao jedna od relevantnih pokazatelja modela održavanja po stanju s provjerom parametara. Granice upozorenja i kvara definiraju "signalizirajuću toleranciju" (Δx) koja određuje stupanj osjetljivosti odabrane dijagnostičke metode na parametar stanja i njegovu identifikaciju u skladu sa zakonom promjene stanja promatranog elementa. [7]

Stanje sustava se može pratiti kontinuirano i periodički. Kontinuirano praćenje se provodi cijelo vrijeme trajanja rada stroja, a koristi se na bitnijim i skupljim strojevima kod kojih bi posljedice kvarova mogle iznimno negativno utjecati na rad cjelokupnog sustava, rezultirajući katastrofalnim havarijama te samim time gubicima u proizvodnom ciklusu. Koristi se kako bi se zaustavio rad stroja uslijed naglih promjena parametara. Zbog toga reakcija sustava za nadziranje stanja mora biti brza. Kako bi se osigurala brzina reakcije, parametri se mjere u realnom vremenu te je zato njihova analiza jednostavnijeg oblika. To dovodi do toga da upozorenje o pojavi kvara dolazi tek nekoliko dana ili sati prije njegove pojave.

Periodičko praćenje stanja sustava se izvodi u određenim vremenskim intervalima. Koristi se na većem broju strojeva koji nisu toliko skupi kao oni kod kontinuiranog praćenja stanja. Puno je veća mogućnost uočavanja početnih grešaka na vrijeme pošto se podaci ne analiziraju kontinuirano, nego se može odvojiti vrijeme za njihovu analizu. Na taj se način planovi održavanja mogu planirati unaprijed kako bi se maksimizirala dostupnost opreme.

Osnovne prednosti i razlike ovih dvaju pristupa su vidljivi u tablici 2:

Tablica 2. Prednosti i nedostaci kontinuiranog i periodičkog nadziranja stanja [5]

<table>
<tr><th></th><th>PREDNOSTI</th><th>NEDOSTACI</th></tr>
<tr><td rowspan="3">Kontinuirano nadziranje stanja</td><td>Brza reakcija na naglu promjenu što uvelike doprinosi očuvanju bitne i skupe opreme</td><td>Konstantno nadziranje stanja stroja je skupo pa se pretvornici primjenjuju na samo najbitnije strojeve u postrojenju</td></tr>
<tr><td rowspan="2">Najbolji oblik zaštite protiv iznenadnih kvarova koji se ne mogu predvidjeti</td><td>Ako se za pretvornik uzme senzor za mjerenje zračnosti (engl. proximity probe), on mora biti ugrađen u stroj u fazi konstruiranja jer bi kasnija ugradnja bila teško moguća</td></tr>
<tr><td>Pošto reakcija mora biti brza, konstantno nadziranje se uglavnom zasniva na jednostavnijim parametrima kao što su: RMS ili vršne razine vibracije. Ti parametri u pravilu upozoravaju na pojavu kvara tek nekoliko sati ili dana prije pojave istoga, dok se korištenjem naprednijih dijagnostičkih tehnika moguće predvidjeti početnu pojavu kvara i do mjesec dana prije nego što se taj kvar pojavi</td></tr>
<tr><td rowspan="2">Periodičko nadziranje stanja</td><td>Puno niža cijena opreme za nadzor stanja</td><td rowspan="2">Postoji mogućnost da se iznenadni kvarovi ne uoče te se u slučajevima gdje su kvarovi nepredvidivi ova metoda ne koristi</td></tr>
<tr><td>Mogućnost za puno detaljnijom analizom stanja, što dovodi do prepoznavanja početnih pojava</td></tr>
</table>

	kvarova puno prije njihovog utjecaja na ispravan rad stroja. Posljedica tog je bolje planiranje održavanja	
	Primjenjuje se u slučajevima u kojima gubitak uslijed zastoja u proizvodnji nadilazi gubitak uslijed kupnje novog stroja	

Dijagnostika i prognostika su 2 važna aspekta u održavanju po stanju. Cilj dijagnostike je određivanje stvarnog stanja sustava bez potrebe za njegovim zaustavljanjem i demontiranjem.

Obuhvaća metode, postupke i sredstva za praćenje rada tehničkih sustava i njihovih komponenti, periodičkim ili kontinuiranim mjerenjem fizikalnih veličina od najvećeg značaja za rad i stanje opreme, te uspoređivanje izmjerenih veličina s utvrđenim graničnim vrijednostima normalnog rada, u cilju ocjene stanja opreme i donošenja odluka o daljnjim aktivnostima na njenom održavanju. [6]

Dijagnostika se bavi: detekcijom kvarova, njihovim izoliranjem i identifikacijom kad se dogode. Detekcija kvarova odgovara na pitanje funkcionira li sve po planu na nadgledanom sustavu. Cilj izolacije kvarova je locirati komponentu s greškom. Cilj identifikacije kvarova je odrediti prirodu kvara kad se dogodi. Prognostika se bavi predviđanjem kvara prije nego li se dogodi. Izvodi se kad se želi odrediti hoće li se dogoditi kvar u sustavu te kako bi se procijenilo kolika je vjerojatnost da će se kvar dogoditi te kad bi se on trebao dogoditi.

Prognostika je superiorna u odnosu na dijagnostiku jer može spriječiti neočekivane kvarove, smanjujući tako neplanirane troškove održavanja [3].

Kako bi se mogla osigurati dijagnostika i prognostika potrebno je provesti 3 ključna koraka [8]:

1. Prikupljanje podataka (engl. *data acquisition*)
2. Obrada podataka (engl. *data processing*)
3. Donošenje odluke o zahvatu održavanja (engl. *maintenance decision making*)

Prikupljanje podataka se odnosi na one podatke koji su relevantni za zdravlje sustava. To je postupak skupljanja i spremanja korisnih podataka s ciljanih tehničkih sustava. Prikupljeni podaci se mogu podijeliti u dvije kategorije [8]:

- Podaci o događajima (engl. *event data*). Podaci koji odgovaraju na pitanje zašto se dogodio zastoj u radu stroja (instaliranje, kvar, remont) ili koji se zahvat održavanja izvodio na njemu (manji popravak, preventivno održavanje, zamjena ulja, itd.). Uobičajeno zahtijeva ručno unošenje podataka u bazu podataka.

- Podaci dobiveni nadziranjem stanja (engl. *condition monitoring data*). Podaci dobiveni mjerenjem zdravstvenog stanja stroja pomoću raznih senzora (mikro-senzori, ultrasonički senzori, senzori za akustičnu emisiju, itd.). Ti podaci mogu dolaziti iz različitih mjerenja: vibracija, akustičnih podataka, analize ulja, temperature, tlaka, vlage, itd. Osnovna zamisao nadziranja stanja opreme je određivanje unutarnjeg stanja opreme dok je ona u radu. Stanje se može nadzirati kontinuirano i periodički.

Obrada podataka omogućuje rukovanje i analizu podataka prikupljenih sa senzora radi njihovog boljeg razumijevanja. Prvi korak nakon prikupljanja podataka je čišćenje podataka. Taj proces se izvodi zato što podaci, pogotovo oni o događajima, često sadrže greške. Pogreške u podacima mogu biti uzrokovane mnoštvom faktora, između ostalog: ljudskom nepažnjom, kao što je to slučaj kod podataka o događajima, ili greškama na senzorima, kao što je to slučaj kod podataka dobivenih nadziranjem stanja. Jednostavniji skupovi podataka se mogu čistiti ručno, dok se kod kompleksnijih skupova koriste grafičke metode vizualizacije podataka.

Nakon čišćenja podataka dolazi do njihovog analiziranja. Podaci prikupljeni u koraku prikupljanja podataka se mogu podijeliti u 3 kategorije [8]:

- Podaci vrijednosnog tipa. Podaci prikupljeni u određenom vremenu za vrijeme nadziranja stanja, npr. podaci za: analizu ulja, temperaturu, tlak i vlagu pripadaju u ovu skupinu podataka.

- Podaci valnog tipa. Podaci prikupljeni u određenom vremenu za vrijeme nadziranja stanja koji su u obliku vremenskih serija. U ovu skupinu podataka pripadaju podaci prikupljeni mjerenjem vibracija i akustičnih emisija.

- Podaci multidimenzijskog tipa. Podaci prikupljeni u određenom vremenu za vrijeme nadziranja stanja koji su u multidimenzionalnom obliku. Najčešći multidimenzionalni tip podataka su slike (npr. prikupljene termografijom, fotoaparatom, rendgenom, itd.).

Krajnji korak je donošenje odluke o zahvatu održavanja. Tehnike za podršku donošenja odluka su vrlo bitne pri odlučivanju o zahvatima održavanja. Mogu se podijeliti u dvije kategorije:

- Dijagnostika. Postupak povezivanja informacija dobivenih tijekom mjerenja s pojavljivanjem kvarova na strojevima. Ovaj postupak se još zove i prepoznavanje uzoraka (engl. *pattern recognition*). Može se provoditi ručno i automatski. Najpoznatiji pristupi rješavanju ovih problema su: statistički pristup, pristup rješavanja pomoću umjetne inteligencije, pristupi bazirani na modelu, itd.

- Prognostika. Najčešći pristupi rješavanju prognostičkih problema su: procjena preostalog životnog vijeka (engl. *remaining useful life*, RUL), procjena vjerojatnosti da će stroj raditi bez pojave kvara u određenom vremenu, itd.

Najčešće korištene tehnike pomoću kojih se provodi dijagnostika stanja tehničkih sustava su:

- Mjerenje i analiza vibracija. Stroj u normalnom stanju stvara signale ustaljenog iznosa. Kad dođe do promjene stanja stroja, tj. greške, signali koje stroj izdaje poprimaju nove vrijednosti, ovisno o tipu kvara.

- Termografija. Infracrvena termografija (ICT) je beskontaktna, nedestruktivna, brza i učinkovita metoda mjerenja energije zračenja tijela, odnosno objekta mjerenja. ICT se temelji na fizikalnoj činjenici da svako tijelo koje ima temperaturu iznad apsolutne nule emitira odnosno zrači elektromagnetske valove. Temperatura promatrane površine ovisi o mnogim parametrima kao što su : emisijski faktor, prividna temperatura okolnih objekata, stanje okoliša u kojem se objekt nalazi i slično. [6]

- Analiza lubrikanata. Lubrikant koji se nalazi u stroju u svom sastavu nosi informacije o mogućim kvarovima. Određuje se stanje (kvaliteta) ulja kako bi se odredilo je li ulje pogodno za daljnju uporabu. Rezultati analize ulja mogu također ukazati na uvjete trošenja dijelova koji su u kontaktu s uljem. Ulje u sebi može nositi krhotine elemenata strojeva. Po tim krhotinama se određuje je li se negdje dogodio kvar. Ako u stroju postoji više komponenti istog kemijskog sastava, često zna biti teško odrediti na kojoj se točno strojnoj komponenti dogodio kvar. Analiza lubrikanata se dijeli na: analizu strugotina, procedure spektrografske analize ulja (engl. *Spectrographic Oil Analysis Procedures*), ferogarafiju.
- Mjerenje akustičnih emisija. Povezano je s nadziranjem stanja pomoću vibracija uz fundamentalnu razliku da se ne montiraju na točno određenu komponentu kao vibracijski senzori nego osluškuju rad cijelog stroja.

- Analiza izvedbe. Kvarovi se predviđaju nadziranjem promjena određenih parametara, kao što su: protok, tlak, potrošnja električne energije, itd.

- Kontrola penetrantima. Osnova primjene leži u svojstvu tekućina temeljenih na lakim uljima (penetrantima) koja imaju sposobnost prodiranja i ispunjavanja sitnih, ”nevidljivih” šupljina (pukotina) u materijalu. Nakon toga se tekućina na pogodan način izvlači iz šupljine te ista postaje vidljiva. [6]

- Ultrazvuk. Ultrazvučna metoda ima široku primjenu. Koristi se za detekciju (procjenu) pukotina unutar materijala, ispitivanje zavarenog spoja, mjerenje dimenzija, karakterizaciju materijala, itd. Funkcionira tako da se energija zvuka unosi i širi kroz materijal u obliku vala. Kada signal dođe do neke vrste diskontinuiteta (npr. pukotine ili granice materijala različitih akustičnih impedancija) dio energije signala će se reflektirati natrag do površine. Reflektirani ultrazvučni signal se zatim pretvara u električni signal i šalje na obradu i na uređaju se prikazuje odziv. Za ultrazvučne pretvornike koriste se materijali koji imaju piezoelektrička svojstva. Najčešća frekvencijska područja uporabe ultrazvuka su između 20 kHz i 10 MHz. [6]

Prednosti i nedostaci strategije održavanja po stanju su vidljivi u tablici 3:

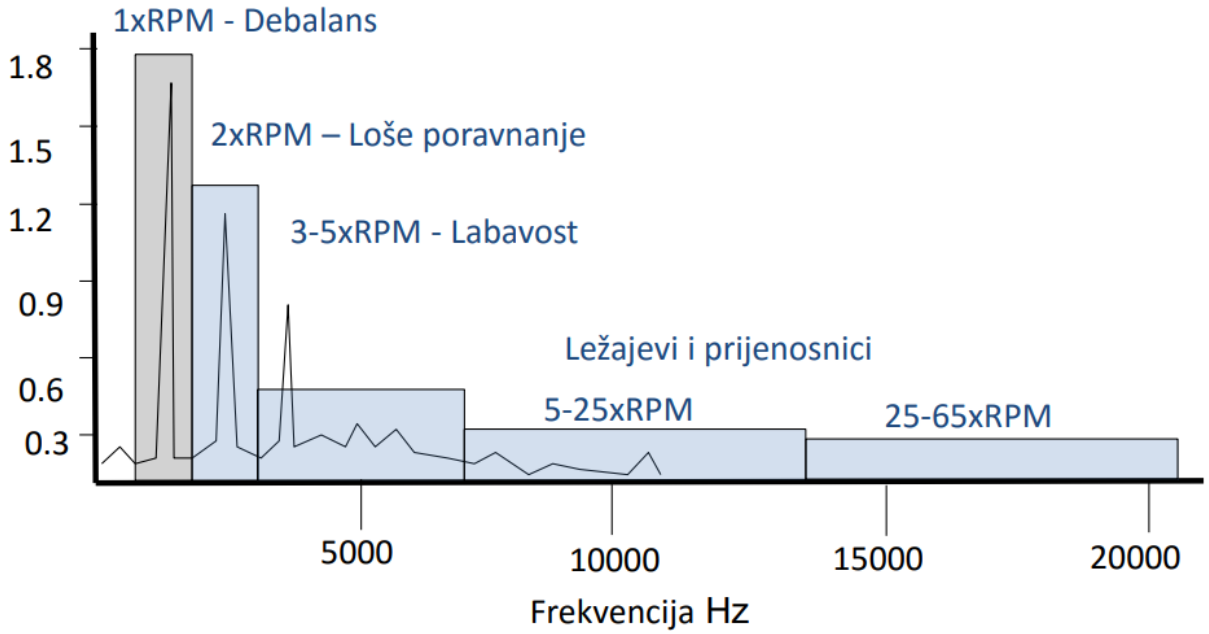
Tablica 3. Prednosti i nedostaci održavanja po stanju [5]

<table>
<tr><th colspan="2">ODRŽAVANJE PO STANJU</th></tr>
<tr><th>PREDNOSTI</th><th>NEDOSTACI</th></tr>
<tr><td>Porast sigurnosti</td><td>Dodatni napor menadžera za organiziranje i uvođenje održavanja po stanju</td></tr>
<tr><td>Povećanje proizvodnih količina u proizvodnom sustavu</td><td>Veliki vremenski razmak između vremena uvođenja i ostvarivanja koristi primjene održavanja po stanju</td></tr>
<tr><td>Porast raspoloživosti opreme i smanjenje poslova održavanja</td><td rowspan="2">Nesigurnost hoće li održavanje po stanju predvidjeti pogoršanje odnosno pojavu oštećenja</td></tr>
<tr><td>Poboljšanje kvalitete proizvoda</td></tr>
</table>

2.4. Mjerenje i analiza vibracija

Vibracija je oscilacija čiji je iznos parametar koji definira gibanje sustava. Oscilacija je promjena intenziteta neke veličine u odnosu na zadanu referentnu vrijednost, pri čemu se intenzitet naizmjenično mijenja iznad ili ispod referentne vrijednosti. Uzroci vibracija mogu biti: neravnoteža sustava, asimetričnost sustava, savijanje osovine/vratila, ekscentričnost sustava, vibracije kotrljajućih ležajeva, hrapavost dodirnih površina, oštećenja valjnih površina uslijed sklapanja i montaže, montaže, geometrijska nesavršenost, loše održavanje, itd. [6]

Čak i u dobrom stanju strojevi generiraju vibracije. Analiza vibracija je najčešće upotrebljavana metoda analize stanja stroja kod strategije održavanja po stanju. Omogućava trenutačno mjerenje stanja stroja, za razliku od npr. analize lubrikanata kod koje je potreban dulji vremenski period između izuzimanja uzorka i njegove analize. Tehnike mjerenja vibracija se koriste kako bi se otkrio: zamor materijala, habanje, neravnoteža, odstupanja, labavi sklopovi, itd. Analizirati se može stanje rotirajućih strojnih dijelova ili uređaja koji imaju ciklička ponavljanja, kao što su: ležajevi, vratila, zupčanici, rotirajuća električna polja, pumpe, motori, turbine, itd. Rad ovih strojeva otpušta energiju u obliku vibracija s frekvencijskim komponentama pomoću kojih se prati koji dijelovi tehničkog sustava ispuštaju te signale. Svaki tip kvara posjeduje karakterističnu komponentu vibracijskih frekvencija koje se mogu filtrirati i definirati (Slika 4).



Slika 4. Otkrivanje uzroka kvara analizom frekvencijskog spektra [6]

Na slici 4 je vidljivo kako se izvori grešaka na stroju mogu uočiti analiziranjem vibracija u frekvencijskoj domeni. Većina kvarova se odvija na karakterističnim iznosima frekvencija.

Osnovna dva načela za primjenu dijagnostike mjerenjem vibracija su [6]:

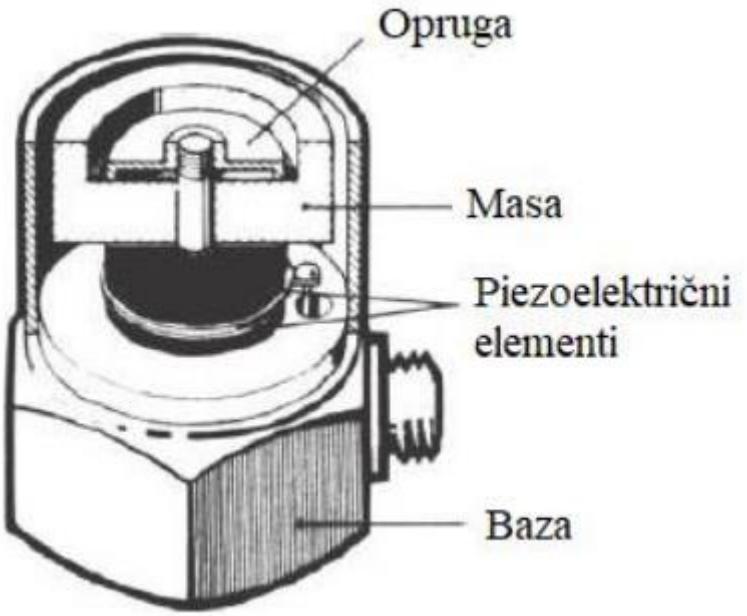
- Svaki tip kvara ima karakterističnu komponentu vibracijskih frekvencija koje se mogu filtrirati i definirati
- Amplituda svake pojedine vibracijske komponente će ostati konstantna ako dinamika tehničkog sustava ostane nepromijenjena

Ako se pojava povećanih vibracija ne otkrije na vrijeme dolazi do posljedica kao što su: povećan utrošak energije, povećan zamor materijala i kraći životni vijek opreme, štetno djelovanje na okolinu, trošenje, itd.

Za snimanje vibracija na tehničkom sustavu i njihovo pretvaranje u električne signale koriste se senzori, odnosno pretvornici (engl. *transducers*). Pretvornici mjere sva 3 parametra u kojima se vibracija može mjeriti: pomak, brzina, akceleracija. Najčešće korištene vrste pretvornika su [1]:

- Senzori za mjerenje zračnosti (engl. *proximity probes*). Mjere relativnu udaljenost između vrha senzora i mjerene površine.
- Mjerač brzine (engl. *velocity transducers*). Daje signal proporcionalan apsolutnoj brzini.
- Dvostruki vibracijski senzori (engl. *dual vibration probes*). Vibracije vratila se uobičajeno mjere senzorima za mjerenje zračnosti, ali oni daju rezultate gibanja relativne u odnosu na kućište. Kako bi se snimilo apsolutno gibanje vratila, potrebno je zbrojiti iznose relativnog i apsolutnog gibanja, što ovi senzori čine. Sastoje se od senzora za mjerenje zračnosti i seizmičkih senzora.
- Laserski vibrometar (engl. *laser vibrometer*). Funkcionira na principu Dopplerovog efekta tako da se laserska zraka usmjeri na vibrirajuću površinu od koje se odbija te se dobiva iznos amplitude i frekvencije. Beskontaktna metoda.
- Akcelerometri (engl. *accelerometers*). Pretvornici koji proizvode signal proporcionalan iznosu akceleracije. Akcelerometar je najčešće korišten pretvornik za mjerenje vibracija opreme.

Daleko najčešće korišteni su piezoelektrični akcelerometri. Koriste se piezoelektričnim svojstvima određenih kristala i keramika te proizvode električni napon proporcionalan naprezanju. Na slici 5 se vidi primjer kompresijskog akcelerometra kod kojeg su piezoelektrični elementi smješteni između mase u senzoru i baze mjernog elementa.



Slika 5. Kompresijski piezoelektrični akcelerometar [1]

Nakon što se baza akcelerometra poveže s vibrirajućim elementom, masa prati gibanje baze preko piezoelektričnih elemenata koji imaju funkciju krute opruge. Promjena inercije mase deformira piezoelektrične elemente proizvodeći pritom naprezanje proporcionalno promjenama u akceleraciji. Piezoelektrični elementi proizvode električni napon proporcionalan akceleraciji koji se izražava u $pC/(ms^2)$. Ovaj iznos se prevodi u napon pomoću pojačala.

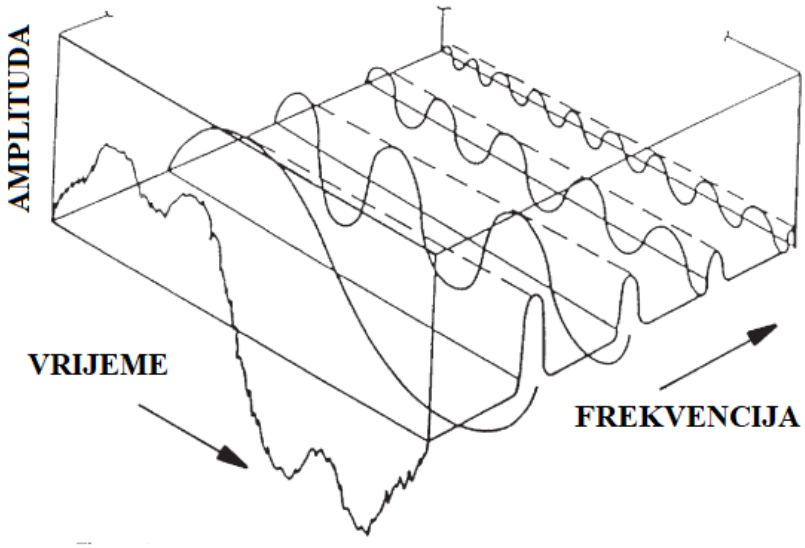
Jedna od najvećih prednosti akcelerometra je ekstremno širok raspon amplitude i frekvencije kojeg je u stanju mjeriti. Tipičan iznos amplitude koju može snimiti iznosi 160 dB (10^8:1), dok tipična frekvencija koju može obuhvatiti iznosi od 1 Hz do 20 kHz. [1]

Da bi se ispravno izmjerili signali s mjerenog sustava potrebno je ostvariti mehanički kontakt između mjerene baze i akcelerometra. Metode montaže akcelerometra su [6]:

- Pomoću klina
- Lijepljenjem
- Magnetom
- Bez fiksne montaže (držanje rukom, s pomoćnim držačima)

Mjerenjem vibracijskih signala akcelerometrom dobivaju se vibracijski podaci u vremenskoj domeni. Profil vibracija zabilježen u vremenskoj domeni valovitog je oblika, a sastoji se od zbroja karakterističnih komponenti sinusoidalnog oblika svakog dijela tehničkog sustava u radu. Tako npr. kod valjnog ležaja, svaka komponenta tog ležaja (unutarnja staza, vanjska staza, valjni elementi) za vrijeme rotacije vratila dolazi u kontakt s drugim dijelovima u određenim

vremenskim intervalima. Svaka komponenta ležaja će se dodirivati pri specifičnim frekvencijama i proizvoditi valove određene amplitude. Kad se svi valovi koje proizvode različite strojne komponente zbroje u vremenskoj domeni dobiva se kompleksan vibracijski profil iz kojeg je komplicirano očitati stanje stroja. Ako se dinamika tehničkog sustava ne mijenja, svaka komponenta proizvodi valoviti signal koji se ponavlja u vremenu. Pošto su signali konstantni u vremenu, iznosi ukupnih vibracija tehničkog sustava u vremenu nisu toliko korisni za analizu jer je cilj otkriti na kojim frekvencijama dolazi do povišenih iznosa vibracija kako bi se po iznosu frekvencije utvrdilo koja komponenta sustava uzrokuje povišene vibracije. Da bi se to otkrilo radi se transformacija vibracijskih signala iz vremenske u frekvencijsku domenu pomoću matematičkog procesa: „Fourierova transformacija“. Na slici 6 je vidljiv grafički prikaz te transformacije.



Slika 6. Odnos vremenske i frekvencijske domene [1]

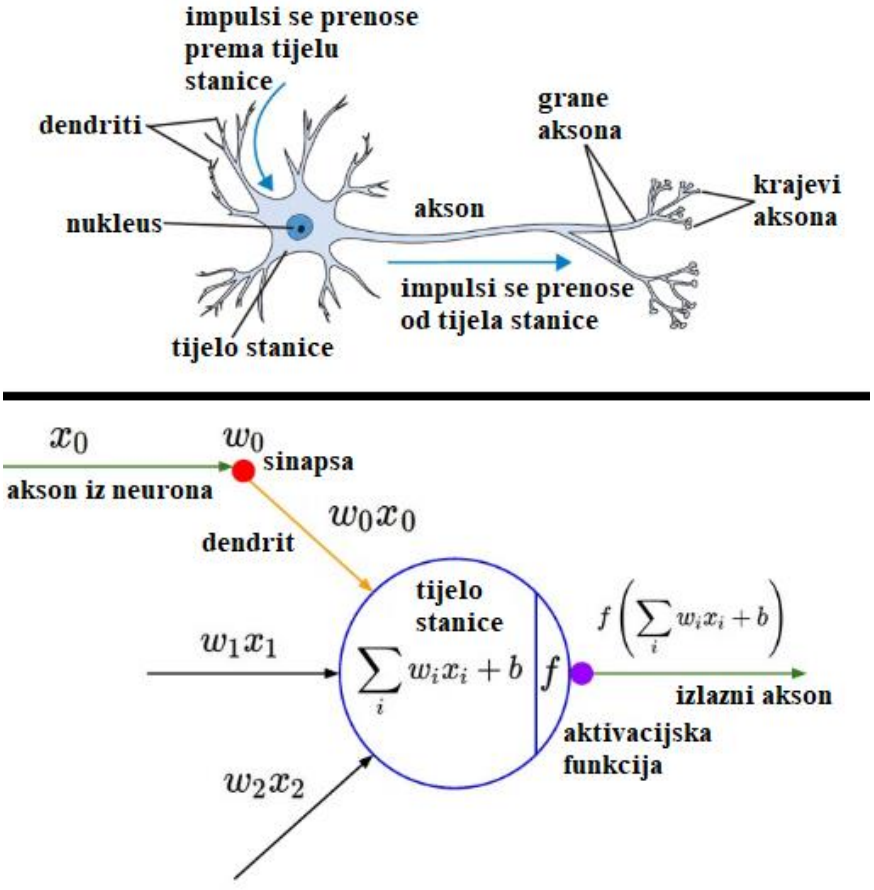
U vremenskoj domeni na horizontalnoj osi se nalazi vrijeme, a u frekvencijskoj domeni se nalazi frekvencija vibracija. U oba slučaja se na ordinatnoj osi prikazuje iznos amplitude. Vidljivo je kako je iznos amplitude vibracijskog signala u vremenskoj domeni ovisan o iznosi vibracijskih signala svih ostalih komponenti sustava. U frekvencijskoj domeni je složeni spektar vibracija iz vremenske domene pretvoren u diskretne komponente. Na taj se način može očitati na kojoj se frekvenciji javlja povišeni signal te se na temelju toga otkriva komponenta koja ga uzrokuje, pošto svaka komponenta vibrira na određenoj frekvenciji.

3. NEURONSKE MREŽE

Duboko učenje je podskup strojnog učenja te koristi algoritme inspirirane strukturom i funkcijom neuronskih mreža u mozgu. Biološka neuronska mreža je dakle temelj na kojem se zasniva duboko učenje. Umjetne neuronske mreže (engl. *artificial neural networks,* ANN) simuliraju rad bioloških mreža u mozgu.

Neuron ili živčana stanica je osnovna jedinica živčanog sustava. Građen je od tijela stanice u kojoj se nalazi jezgra (nukleus) i od mnoštva dendrita i aksona. U ljudskom mozgu se nalazi oko 100 vrsta neurona. Njihov ukupan broj iznosi oko 10^{11}, a svaki neuron je prosječno povezan s 10^4 neurona.

Na slici 7 se može vidjeti usporedba biološkog i matematičkog modela neurona.



Slika 7. Usporedba biološkog (gore) i matematičkog (dolje) modela neurona [9]

Svaki neuron prima ulazne signale iz svojih dendrita i proizvodi izlazni signal koji se dalje kreće pomoću aksona. Akson se nakon nekog vremena grana i povezuje pomoću sinapsi s dendritima drugih neurona. U računskom prikazu neurona, signal koji putuje aksonom (x_0) se

množi s w_0 na sinapsi. Jačina sinapse (težina *w*) je parametar koji se može naučiti te kontrolira utjecaj jednog neurona na drugi. Dendriti prenose signal do tijela stanice, gdje se zbrajaju. Ako je suma veća od određene granične vrijednosti (engl. *treshold*) neuron se upali i šalje signal dalje preko aksona.

3.1. Umjetne neuronske mreže

Umjetna neuronska mreža je tehnika strojnog učenja koja oponaša ljudski mozak s ciljem simuliranja postupka učenja kod čovjeka.

Ovaj rad će se baviti konvolucijskim neuronskim mrežama, no iako su među najčešće korištenima, konvolucijske su mreže samo jedna od mnogo različitih vrsta neuronskih mreža.

Neke od poznatijih vrsta umjetnih neuronskih mreža su:

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

Umjetna neuronska mreža se sastoji od skupa povezanih jedinica zvanih neuroni (čvorovi) koji su organizirani u slojeve. Neuroni su mjesta u neuronskoj mreži u kojima se izvršavaju računske operacije. U njih ulaze izlazni signali iz prijašnjeg sloja pomnoženi određenim koeficijentom, koji se u neuronskim mrežama naziva težina. Težine mogu smanjiti ili povećati iznos svakog ulaza u neuron. Nakon što su u neuron ušli svi izlazi iz prijašnjeg sloja pomnoženi svojim težinama, oni se sumiraju te se na tu sumu dodaje pomak (engl. *bias*):

$$z = w^T x + b \tag{3.1}$$

w^T – transponirani vektor težina

x – vektor ulaza x

b – pomak

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

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

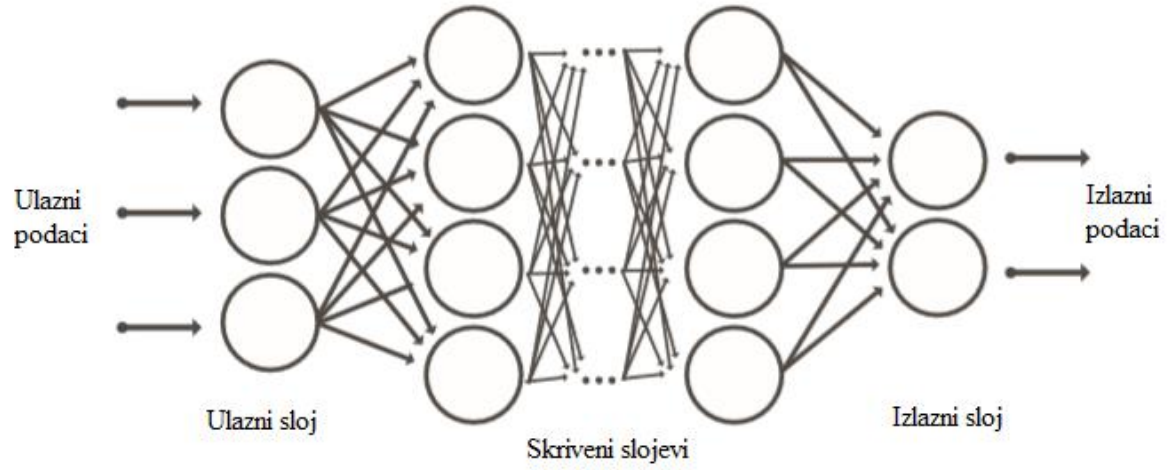
$$y = f(z) = f(w^T x + b) \tag{3.2}$$

Ovisno o aktivacijskoj funkciji koja se koristi ovisi hoće li se taj neuron i u kolikoj mjeri aktivirati i sudjelovati u idućim slojevima. U svakom se sloju nalazi određeni broj neurona. Slojevi (Slika 8) se dijele na:

1.) Ulazni sloj (engl. *input layer*). Svaki čvor predstavlja jednu komponentu ulaznih podataka. Npr. kod slučaja u kojem se na temelju ulaznih podataka kao što su: visina, težina, mišićna masa želi odrediti spol osobe, ti ulazni podaci (visina, težina, mišićna masa) bi bili čvorovi koji čine ulazni sloj

2.) Skriveni sloj (engl. *hidden layer*). Za svaki se sloj proizvoljno odabere broj čvorova.

3.) Izlazni sloj (engl. *output layer*). Koliko postoji mogućih klasa toliko treba postojati i čvorova u izlaznom sloju.



Slika 8. Slojevi umjetne neuronske mreže [10]

Kad se na temelju prolaska ulaznih podataka kroz skrivene slojeve dobije određeni izlaz u izlaznom sloju mreže, taj se izlaz uspoređuje s izlazom kojeg je mreža trebala dati te se računa pogreška pomoću funkcije gubitka. Na temelju te pogreške mreža uči ažurirajući pritom svaku težinu.

3.1.1. Učenje neuronske mreže

Neuronska mreža uči u dvije faze:

1) Unaprijedna faza (engl. *forward pass*)
2) Povratna faza (engl. *backward pass*)

Unaprijedna faza kod nadziranog učenja se računa:

- U neuronsku mrežu se postave ulazni podaci. Svaki uzorak na ulazu posjeduje svoju labelu, tj. točan odgovor koji bi mreža trebala izračunati.
- Ulazni podaci prolaze kroz neuronsku mrežu te preko njenih skrivenih slojeva, međusobno povezanih težinama, dolaze do izlaznog sloja u kojem se izračunava rezultat. Rezultat se izražava vjerojatnostima.
- Rezultat kojeg je mreža izračunala se uspoređuje s rezultatom koji je trebala izračunati i računa se pogreška. Pogreška se izražava funkcijom gubitka.

Te 3 točke spadaju u unaprijednu fazu učenja. Druga, povratna faza učenja se odvija nakon nje. Nakon što je izračunata funkcija gubitka dolazi do ažuriranja težina optimizacijskom metodom gradijentnog spusta čiji su sastavni dijelovi: određena težina, stopa učenja, gradijent. Gradijent računa promjenu funkcije gubitka s obzirom na promjenu veličine neke težine. U izlaznom sloju nastoji povisiti vrijednost neurona koji predstavljaju točan odgovor, a sniziti vrijednosti neurona koji predstavljaju netočan odgovor. To će se postići korištenjem algoritma povratnog prostiranja pogreške koji će biti objašnjen u tijeku rada. Ukratko, algoritam ažurira težine tako da računa pogrešku u zadnjem sloju te ju lančanim pravilom propagira prema unatrag računajući parcijalne derivacije pogreške s obzirom na: izlaz neurona, ulaz neurona, težine.

Učenje mreže je zapravo pokušaj optimizacije modela. Cilj je učenja pronaći vrijednosti težina koje točno povezuju ulazne s izlaznim vrijednostima. Težine se optimiraju korištenjem optimizacijskih algoritama koji za cilj kod neuronskih mreža imaju minimiziranje funkcije gubitka.

Pogreška je razlika između one vrijednosti koju mreža predviđa za svaki čvor u izlaznom sloju i vrijednosti koju bi trebala predviđati (labele) da rezultat bude točan. Gubitak se računa pomoću funkcije gubitka. Postoji više različitih funkcija gubitka. Njihov odabir ovisi o vrsti problema, npr. nastoji li se riješiti problem regresije ili klasifikacije. Za problem klasifikacije se najčešće koristi unakrsna provjera (engl. *cross validation*) dok se za problem regresije najčešće koriste: srednja kvadratna pogreška (engl. *mean square error* - MSE), korijen srednje kvadratne pogreške (engl. *root mean square error* – RMSE), srednja apsolutna pogreška (engl. *mean absolute error* – MAE).

Jedan prolazak svih podataka kroz neuronsku mrežu zove se epoha. Kroz mrežu mogu prolaziti uzorci pojedinačno jedan za drugim, ali isto tako za vrijeme postupka učenja neuronske mreže može proći i više ulaznih podataka istovremeno. Količina uzoraka koji prolaze kroz mrežu istovremeno određuje se veličinom zasebnih skupova podataka (engl. *batch*). *Batch* predstavlja broj uzoraka koji će u istom trenutku proći kroz mrežu. Veličinu, a samim time i broj zasebnih skupova podataka određuje analitičar. Što je zasebni skup podataka veći, to treniranje mreže kraće traje, ali treba imati na umu da je za veće *batch*-eve potrebno snažnije računalo te da model slabije generalizira na test skupu. Broj zasebnih skupova podataka se izračuna tako da se broj svih trening uzoraka podijeli s veličinom jednog zasebnog skupa podataka. Ako je broj uzoraka 1500, a veličina „*batch*-a" je 50, onda je količina „*batch*-eva" jednaka 30. Veličina jednog *batch*-a predstavlja hiperparametar neuronske mreže. Hiperparametri će biti objašnjeni kasnije u radu.

Kroz model prolaze 3 skupa podataka:

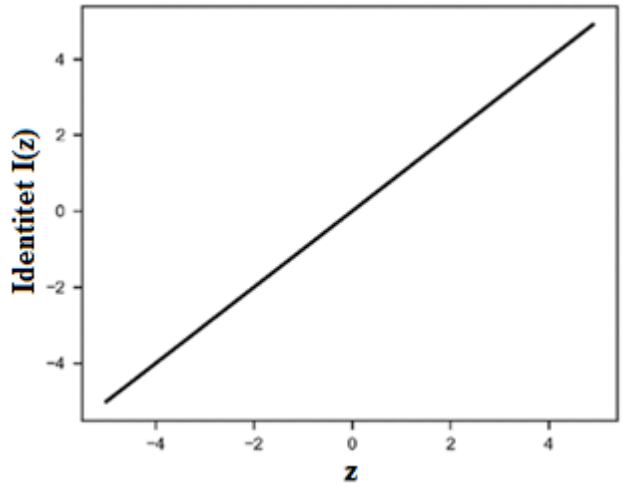
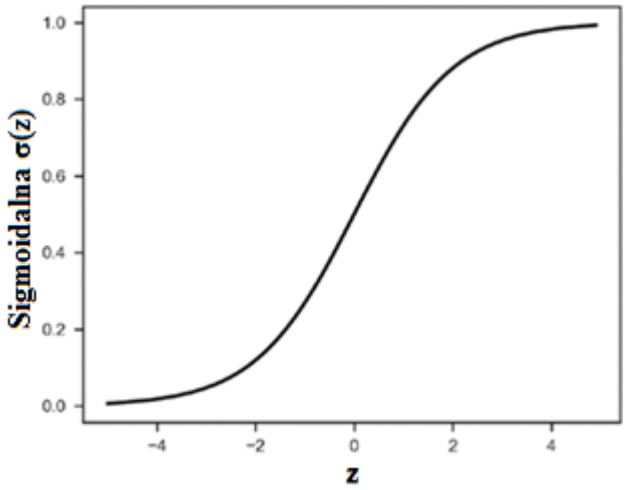
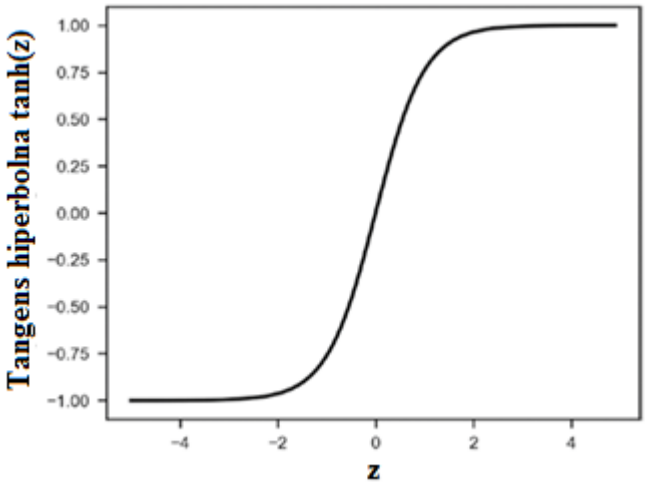
1.) Trening skup. Podaci nad kojima model uči mijenjajući pritom iznose svojih parametara. Kod nadziranog učenja su labelirani. Nakon procesa učenja neuronska mreža iskazuje razinu točnosti i pogreške.
2.) Validacijski skup. Podaci različiti od trening skupa podataka. Validacijski skup je podskup podataka koji se koristi za izbor hiperparametara. Podaci iz ovog skupa prolaze kroz istu neuronsku mrežu koju treniraju podaci iz trening skupa te se točnost i pogreška iskazuju u definiranom periodu. Omogućuje ocjenjivanje koliko dobro model u trenutnoj fazi učenja generalizira na skupu na kojem nije učen.
3.) Test skup. Daje konačnu provjeru koliko dobro model generalizira.

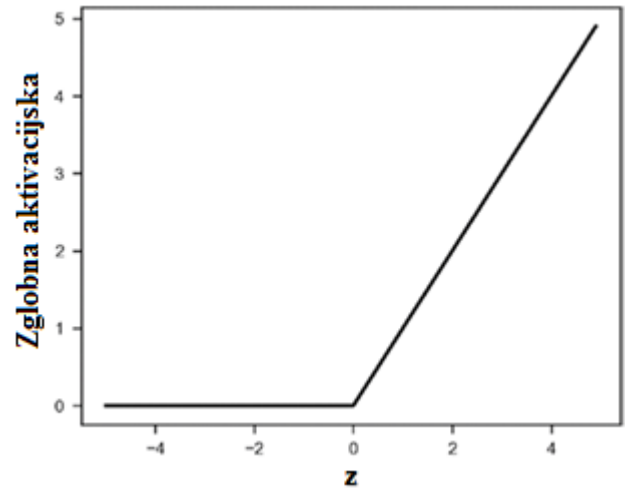
Uobičajeno se za trening skup podataka uzima 70-80 % ukupnih podataka, za validacijski skup oko 5%, a za test skup 15-20 %.

3.1.2. Aktivacijske funkcije

U umjetnoj neuronskoj mreži, aktivacijska funkcija je funkcija koja mapira ulaze čvorova u odgovarajuće izlaze, tako da se izlazom neurona uvijek smatra vrijednost funkcije z nakon aktivacije. Koriste se u skrivenim slojevima i u izlaznom sloju. Većina aktivacijskih funkcija su nelinearne i odabrane su na ovaj način kako bi se mogli razvrstati podaci koji se ne mogu odvojiti linearnim metodama. Nelinearne aktivacijske funkcije omogućuju neuronskim mrežama izračunavanje složenijih funkcija. U tablici 4 su navedene neke od najčešće korištenih aktivacijskih funkcija u neuronskim mrežama:

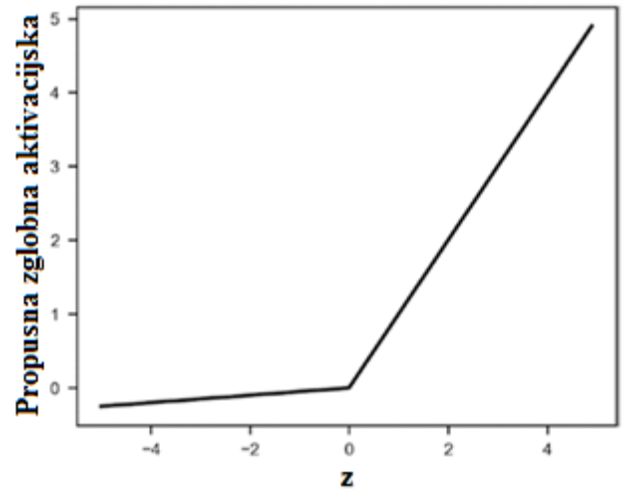
Tablica 4. Aktivacijske funkcije [11]

Graf funkcije	Naziv funkcije
	Identitet funkcija (engl. *Identity function*): $f(z)=I(z)=z$ (3.3)
	Sigmoidalna funkcija (engl. *Sigmoid function*): $f(z)=\sigma(z)=\frac{1}{1+e^{-z}}$ (3.4)
	Tanh, Tangens hiperbolna aktivacijska funkcija (engl. *Hyperbolic Tangent Activation*, Tanh): $f(z)=tanh\,(z)$ (3.5)



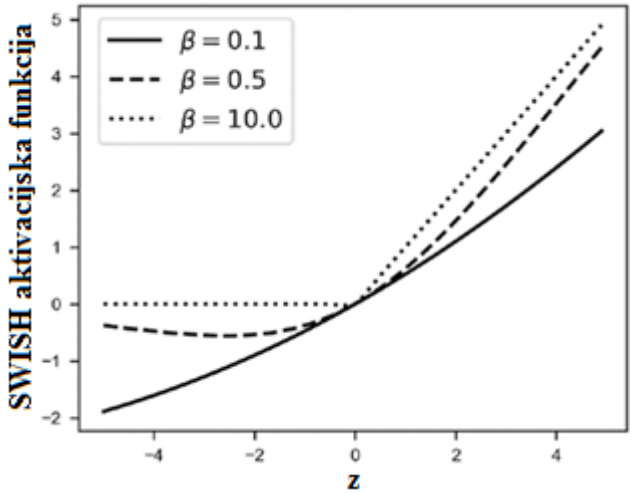
Zglobna aktivacijska funkcija (engl. *Rectified Linear Unit* - ReLU):

$$f(z)=max\ (0,z) \qquad (3.6)$$



Propusna zglobna aktivacijska funkcija (engl. *Leaky Rectified Linear Unit*):

$$f(z)=\begin{cases}\alpha z\ za\ z\leq 0\\ z\ za\ z\geq 0\end{cases} \qquad (3.7)$$



Swish aktivacijska funkcija (engl. *Swish Activation Function*):

$$f(z)=z\sigma(\beta z) \qquad (3.8)$$

Identitet funkcija je osnovna aktivacijska funkcija. Kao izlaz daje nepromijenjeni ulaz. Sigmoidalna funkcija je jedna od najčešće korištenih aktivacijskih funkcija. Na temelju grafa je zamjetno kako za bilo koji ulaz *x* daje rezultate između 0 i 1. Zbog toga se ova funkcija koristi za modele koji na izlazu mreže izražavaju vjerojatnost nekog događaja. Tanh funkcija je slična sigmoidalnoj funkciji, s rasponom za *y* od -1 do 1. Koristi ju se umjesto sigmoidalne kad se žele

dobiti i negativni i pozitivni izlazi iz neurona. Najčešće se koristi kod povratnih neuronskih mreža (engl. *Recurrent Neural Networks*, RNN) za zadatke raspoznavanja govora i obrade prirodnog jezika (engl. *Natural Language Processing*, NLP). No problem koji se javlja kod sigmoidalne i tahn funkcije je problem eksplodirajućeg i nestajućeg gradijenta. Taj problem se rješava upotrebom zglobne aktivacijske funkcije. Zglobna aktivacijska funkcija se koristi kod gotovo svih konvolucijskih neuronskih mreža. Manje je računalno zahtjevna nego sigmoidalna ili tanh jer uključuje jednostavnije matematičke operacije. Zglobna aktivacijska funkcija pretvara sve negativne brojeve u 0, a pozitivne ostavlja onakvima kakvi su bili. To znači da neće svi neuroni u sloju biti aktivirani. Ovo smanjuje računalnu zahtjevnost te ubrzava učenje mreže. Nedostatak zglobne aktivacijske funkcije su upravo negativne vrijednosti koje su poprimile iznos nula. Ti neuroni postaju neaktivni što stvara problem kod ažuriranja težina pomoću gradijenta u povratnoj fazi učenja. Propusna zglobna aktivacijska funkcija rješava taj problem jer ima blagi nagib u području od -∞ do 0, tako da gradijent neće poprimiti vrijednost nula. Zbog toga je sporija od zglobne aktivacijske funkcije.

Swish aktivacijska funkcija do sad nije bila toliko primjenjivana. Studije su pokazale da zamjenjivanje aktivacijskih funkcija ReLU-a Swishom poboljšava točnost klasifikacije na ImageNet-u za 0,9%. [11]

3.1.3. Optimizatori

Funkcija gubitka u zadatku nadziranog učenja uspoređuje i mjeri razliku između predviđenog rezultata i pravog rezultata. Postoji više funkcija gubitka. Zadatak je analitičara odabrati koju funkciju gubitka koristiti. Funkcija gubitka se za vrijeme povratne faze učenja nastoji smanjiti korištenjem određene optimizacijske metode ili optimizatora. Optimizacijske metode ažuriraju težine modela na takav način da funkciju gubitka približavaju što je moguće bliže njenoj minimalnoj vrijednosti. Kako bi se to postiglo računa se gradijent te se težine ažuriraju u negativnom smjeru gradijenta jer je cilj smanjiti funkciju gubitka, a ne ju povećati. U neuronskim mrežama se najčešće koristi optimizacijska metoda gradijentnog spusta (engl. *gradient descent*). No, gradijentni spust nije jedina optimizacijska metoda koja se može koristiti u neuronskim mrežama. Postoje i druge optimizacijske metode temeljene na gradijentu kao što su: Momentum, Nesterov ubrzani gradijent, Adagrad, Adam, RMSprop, itd. Gradijentni spust je numerička funkcija. Ovisno o veličini seta podataka nad kojim se ažuriraju težine gradijentni spust se dijeli na:

- Stohastički gradijentni spust (engl. *stochastic gradient descent*, SGD). Težine se ažuriraju nakon prolaska svakog uzorka kroz mrežu. Algoritam je puno sporiji od ostala

dva, ali je zato najtočniji. Metoda je zbog svog osciliranja otpornija na zapinjanje u lokalnim minimumima.

- Gradijentni spust s mini-grupama (engl. *mini-batch gradient descent*). Trening set se podijeli u više manjih grupa podataka koje se zatim prosljeđuju kroz mrežu. Težine se ažuriraju nakon svakog prolaska tih manjih skupova podataka (engl. *batch*). Ovo je najčešće korištena optimizacijska metoda u dubokom učenju zbog svoje brzine i točnosti.
- Standardni (grupni) gradijentni spust (engl. *batch gradient descent*). Težine se ažuriraju na kraju svake epohe tek kad se svakom uzorku zasebno izračuna promjena težina. Metoda se po brzini može mjeriti s metodom gradijentnog spusta s mini-grupama, ali je njen utjecaj na smanjenje funkcije gubitka lošiji. Podložna zapinjanju u lokalnim optimumima.

U tablici 5 se može vidjeti usporedba ovih triju metoda gradijentnog spusta nakon provedenih 100 epoha na nekom zadatku.

Tablica 5. Usporedba 3 varijacije gradijentnog spusta [11]

Oblik gradijentnog spusta	Vrijeme u radu	Vrijednost funkcije gubitka	Točnost
Standardni (grupni) gradijentni spust	2,5 min	0,323	16%
Gradijentni spust s mini-grupama	2,5 min	0,14	66%
Stohastički gradijentni spust	35 min	0,094	80%

Kako bi gradijentni spust s mini-grupama postigao jednaku vrijednost funkcije gubitka kao i stohastički gradijentni spust bilo bi potrebno provesti 450 epoha ili trenirati mrežu približno 11 minuta, što bi značilo da je gradijentnom spustu s mini-grupama bilo potrebno 31% vremena potrebnog stohastičkom gradijentnom spustu da postigne isti rezultat. [11]

Formula za izračun novih težina korištenjem stohastičkog gradijentnog spusta dana je u nastavku:

$$w_{12}^{2}=w_{12}^{1}-\eta \times \nabla J(w_n) \tag{3.9}$$

Formula (3.9) se čita:

nova težina = stara težina - (stopa učenja * gradijent)

1.) Stara težina je iznos težine koja povezuje određena dva čvora, a korištena je u prijašnjoj epohi.
2.) Stopa učenja određuje koliki korak treba poduzeti u smjeru minimuma, uglavnom iznosi od 0.01 do 0.0001.
3.) $\nabla J(w_n)$ je gradijent funkcije gubitka. To je vektor kojemu su komponente parcijalne derivacije funkcije gubitka u odnosu na sve komponente vektora težina w, a dobiva se tako da se izračuna promjena funkcije gubitka s obzirom na promjenu određene težine. Izračun gradijenta se izvodi pomoću algoritma povratnog prostiranja pogreške (engl. *backpropagation*). Gradijent kazuje u kojem se smjeru gubitak kreće prema maksimumu. Zato se u formuli koristi minus – kako bi se gradijentni spust kretao u suprotnom smjeru od maksimuma.

Pošto se gradijent računa za svaku težinu, tako će njegov iznos uvijek ovisiti o iznosu te težine. Težine se mogu iterativno ažurirati sa: svakim uzorkom, svakom mini-serijom, svakom epohom. Svakom iteracijom bi se težine trebale pomicati bliže svojim optimalnim vrijednostima. S iteracijama se može stati kad promjene gradijenta postanu neznačajne. Ti izračuni mogu biti dugotrajni pa se uglavnom unaprijed odredi broj iteracija.

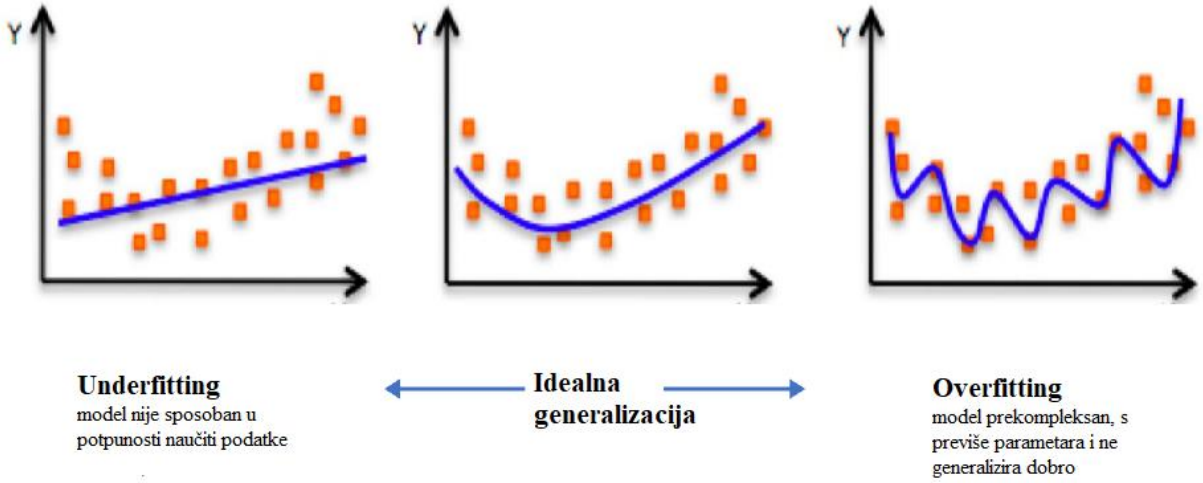
3.1.4. Generaliziranje mreže

Za neuronsku se mrežu kaže da dobro generalizira kad joj se daju novi podaci koji pripadaju istim klasama kao i oni iz trening skupa te ih ona ispravno klasificira. Postoje dva problema generalizacije:

1.) Prenaučenost (engl. *overfitting*). Do prenaučenosti ili overfittinga mreže dolazi kad je mreža izvrsno naučila na trening skupu, ali nije u stanju dobro klasificirati nikad viđene podatke iz test skupa. Prenaučenost se smanjuje:
   - Povećanjem količine i diverzifikacijom podataka
   - Rezanjem, rotacijom, povećavanjem, zaokretanjem podataka (engl. *data augmentation*)
   - Uklanjanjem nekih slojeva i neurona u mreži
   - Ignoriranjem nekih čvorova u slojevima za vrijeme treninga (engl. *dropout*)
2.) Podnaučenost (engl. *underfitting*). Do podnaučenosti ili underfittinga dolazi kad model ne može dovoljno dobro klasificirati ni podatke na kojima je treniran. Točnost na trening skupu je niska, a gubitak visok.

   Podnaučenost se smanjuje:

- Povećanjem kompleksnosti modela: povećanjem broja slojeva i broja neurona u mreži, mijenjanjem rasporeda slojeva.
- Dodavanjem više značajki u ulazni sloj.
- Smanjivanjem *dropout*-a.



Slika 9. Prikaz tipova generalizacije [12]

Na slici 9 se nalaze 3 dijagrama, s lijeva na desno: dijagram podnaučene mreže, dijagram idealno trenirane mreže, dijagram prenaučene mreže. Ta tri dijagrama prikazuju istreniranost mreže na trening podacima. Lijeva mreža nije u stanju usvojiti obrasce ponašanja podataka u trening skupu, srednja mreža dobro aproksimira podatke, a desna mreža je iznimno dobro naučila na trening skupu podataka zbog čega neće biti u stanju ispravno predviđati nove podatke iz testnog skupa.

Prenaučenost se može smanjiti upotrebom metoda regularizacije. Regularizacija smanjuje kompleksnost mreže te time postiže da model bolje generalizira na test setu podataka. Zbog regularizacije točnost trening seta opada, ali cilj su i onako dobri rezultati na test setu. Primjenjuje se uglavnom na funkciji gubitka, tako što kažnjava visoke vrijednosti težina. Postoji mnoštvo metoda regularizacije, a najpoznatije su:

- L2 regularizacija. Smanjuje sposobnost modela prilagođavanju kompliciranim setovima podataka za vrijeme treninga. Djeluje tako da se funkciji gubitka $J(w)$ dodaje vrijednost $\lambda R(w)$, koja se naziva penal. Penal ima zadatak smanjiti kapacitet mreže kako bi se mreža prilagodila kompliciranijim setovima podataka iz test seta. Kapacitet opisuje mogućnost prilagođavanja različitim podacima. Nova funkcija gubitka s L2 regularizacijom bi iznosila:

$$\tilde{J}(w)=J(w)+\frac{\lambda}{2m}|w|^2 \tag{3.10}$$

λ je hiperparametar, w je vektor svih težina u mreži, uključujući i pragove.

- L1 regularizacija. Funkcionira po istom principu kao i L2 regularizacija. Formula za izračun funkcije gubitka iznosi:

$$\tilde{J}(w)=J(w)+\frac{\lambda}{m}||w|| \tag{3.11}$$

- Isključivanje (engl. *dropout*). Isključivanje je metoda regularizacije koja u svakoj iteraciji uklanja različite čvorove, s ciljem smanjenja prenaučenosti u test skupu podataka. To zapravo znači da se u svakoj iteraciji trenira druga mreža. Dok mreža uči metodom isključivanja, validacijski set podataka koristi mrežu sa svim čvorovima.

3.1.5. Algoritam povratnog prostiranja pogreške

Već je ranije objašnjeno na koji način funkcioniraju unaprijedna i povratna faza. U ovom će se poglavlju detaljnije opisati povratna faza, tj. matematički će se opisati kako izračunati promjenu funkcije gubitka s obzirom na promjenu težine u neuronskoj mreži. To se postiže primjenom algoritma povratnog prostiranja pogreške (engl. *backpropagation*). *Backpropagation* je alat koji algoritam gradijentnog spusta koristi kako bi se izračunao gradijent funkcije gubitka. Kreće se u smjeru od izlaza prema ulazu neuronske mreže. Temelj ovog algoritma je činjenica da izlazne vrijednosti nekog sloja zavise od iznosa težina prijašnjeg sloja. Prvo se računa pogreška zadnjeg sloja, zatim se sve pogreške zadnjeg sloja zbroje i propagiraju na predzadnji sloj, i tako sve do prvog sloja u mreži. Ovaj se pristup naziva ulančavanje unatrag. Računa se koliko je svaka težina utjecala na pogrešku.

Za opis algoritma povratnog prostiranja pogreške koristit će se sljedeći izrazi:

$y_j^{(l)}$ – aktivirani izlaz neurona *j* u sloju *l*

$z_j^{(l)}$ – svi ulazi u čvor (neuron) *j* u sloju *l*

$g^{(l)}$ – aktivacijska funkcija u sloju *l*

$w_{ij}^{(l)}$ – težina koja spaja čvor *i* u sloju *l-1* i čvor *j* u sloju *l*

i – indeks koji označava čvorove u prethodnom sloju *(l-1)*

j – indeks koji označava čvorove u trenutnom sloju *l*

E – pogreška, izračunava se izrazom: $E=\frac{1}{2}\sum_j\left(y_j-t_j\right)^2$

t_j – očekivani izlaz neurona

Opisat će se put od pogreške u izlaznom sloju, do parcijalne derivacije pogreške u odnosu na promjenu težine.

Prvo se izražava pogreška trenutnog sloja:

$$\frac{\partial E}{\partial z_j} = \frac{\partial E}{\partial y_j} \times \frac{\partial y_j}{\partial z_j} \tag{3.12}$$

$\frac{\partial E}{\partial y_j}$ je oznaka za parcijalnu derivaciju pogreške *E* po izlazu neurona j.

$$\frac{\partial E}{\partial y_j} = \frac{1}{2} \times \frac{\partial}{\partial y_j}\left(y_j - t_j\right)^2 = \frac{1}{2} \times 2 \times \left(y_j - t_j\right) = (y_j - t_j) \tag{3.13}$$

$\frac{\partial y_j}{\partial z_j}$ je oznaka za parcijalnu derivaciju izlaza neurona j po ulazu neurona j. Izraz će se izvesti za sigmoidalnu aktivacijsku funkciju:

$$y_j = g(z_j) = \left(\frac{1}{1+e^{-z_j}}\right) \tag{3.14}$$

$$\begin{aligned} \frac{\partial y_j}{\partial z_j} &= \frac{\partial}{\partial z_j}\left(\frac{1}{1+e^{-z_j}}\right) = \frac{-1(-e^{-z_j})}{(1+e^{-zj})^2} = \\ &= \frac{1}{1+e^{-z_j}} \frac{e^{-z_j}}{1+e^{-z_j}} = y_j \times \frac{e^{-z_j}}{1+e^{-z_j}} = y_j \frac{(1+e^{-z_j})-1}{1+e^{-z_j}} = \\ &= y_j \times \frac{1+e^{-z_j}}{1+e^{-z_j}} \times \frac{-1}{1+e^{-z_j}} = y_j(1-y_j) \end{aligned} \tag{3.15}$$

Sada se kombinacijom izraza u (3.13) i (3.15) dobiva:

$$\frac{\partial E}{\partial z_j} = (y_j - t_j) \times y_j \times (1-y_j) \tag{3.16}$$

Sad kad je izračunata pogreška trenutnog sloja može se krenuti na računanje greške prethodnog sloja. Potrebno je odrediti sumu utjecaja svih neurona u prethodnom sloju. Parcijalna derivacija pogreške po izlaznim neuronima prethodnog sloja se računa:

$$\frac{\partial E}{\partial y_j} = \sum_j \frac{\partial E}{\partial z_j} \times \frac{\partial z_j}{\partial y_j} = \sum_j w_{ij} \frac{\partial E}{\partial z_j} \tag{3.17}$$

w_{ij} se dobio iz: $z_j = \sum w_{ij} \times y_i$

Svrha algoritma povratnog prostiranja pogreške je određivanje parcijalne derivacije pogreške po težinama:

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial z_j} \frac{\partial z_j}{\partial w_{ij}} = \frac{\partial E}{\partial z_j} \times y_i \tag{3.18}$$

Sada se težine mogu ažurirati. Ažuriraju se ovisno o stopi učenja, kako je prikazano u jednadžbi:

$$w_{ij}^{(2)} = w_{ij}^{(1)} - \eta \times \frac{\partial E}{\partial w_{ij}} \tag{3.19}$$

$w_{ij}^{(1)}$ – iznos težine w_{ij} na početku

$w_{ij}^{(2)}$ – iznos težine w_{ij} nakon prve iteracije

Izrazom (3.19) je definiran algoritam učenja mreže.

Povratno prostiranje pogreške ima svoje nedostatke kad se obavlja na mrežama s puno slojeva. Kod takvih mreža težine koje se nalaze u prvim slojevima se ili ne mijenjaju nikako ili se mijenjaju previše. Kad se težine na početku mreže svakom epohom ažuriraju za jako male iznose radi se o problemu nestajućeg gradijenta (engl. *vanishing gradient problem*), a kad se težine jako mijenjaju radi se o problemu eksplodirajućeg gradijenta (engl. *exploding gradient problem*). Težinama koje zahvati problem nestajućeg gradijenta će biti potreban velik broj iteracija da postignu svoju optimalnu vrijednost. Kod problema eksplodirajućeg gradijenta je upitno uopće može li težina ikada poprimiti zadovoljavajući iznos. Do ovog problema dolazi zbog pravila ulančavanja unatrag koje može ili jako smanjiti ili jako povećati gradijent krećući se od izlaza prema ulazu mreže.

Što je težina bliže ulazu mreže, bit će potrebno više operacija množenja kako bi se ulančavanjem unatrag došlo do parcijalne derivacije pogreške s obzirom na tu težinu. Ako su parcijalne derivacije u tim umnošcima mali brojevi (manji od 1), onda će se povećanjem broja operacija množenja doći do jako malog broja. Ako su parcijalne derivacije brojevi veći od 1 onda će parcijalna derivacija pogreške poprimati sve veći iznos sa svakim idućim slojem.

Problem nestajućeg i eksplodirajućeg gradijenta se može prikazati pomoću jednadžbe stohastičkog gradijentnog spusta za ažuriranje težina: $w_{ij}^{(2)} = w_{ij}^{(1)} - \eta \times \frac{\partial E}{\partial w_{ij}}$

Da bi se dobio gradijent u početnim slojevima mreže potrebno je pomnožiti veći broj veličina: $\frac{\partial E}{\partial w_{ij}} = e \times d \times c \times b \times a$. Slova *e, d, c, b, a* predstavljaju parcijalne derivacije u svakom sloju mreže. Sad se može dobiti uvid koliko bi iznosio taj gradijent u slučaju da je svaka od veličina (*e, d, c, b, a*) veća ili manja od 1. Da su veličine, tj. parcijalne derivacije, manje od 1, njihov bi krajnji iznos bio jako mali broj. Taj mali broj (gradijent) bi se množio sa stopom učenja koja je isto mali broj. Sve to bi se oduzelo od određene težine. Promjena te težine bi stoga bila slabo zamjetna. Do suprotne pojave dolazi kod eksplodirajućeg gradijenta kad su elementi umnoška

(*e, d, c, b, a*) brojevi veći od 1 te je samim time i gradijent velik broj. Postoji mogućnost da se težina nikad ne bi mogla ažurirati na zadovoljavajući način jer bi pomak uvijek preskakao optimalnu točku uslijed prevelikog gradijenta.

Do problema nestajućeg gradijenta najčešće dolazi zbog upotrebe sigmoidalne aktivacijske funkcije jer njena derivacija uvijek iznosi manje od 0,25. Zato se kod konvolucijskih neuronskih mreža koristi ReLU funkcija. Korištenjem te aktivacijske funkcije je manje vjerojatno da će doći do problema nestajućeg gradijenta jer je derivacija njenih pozitivnih vrijednosti uvijek jednaka 1. U borbi protiv problema eksplodirajućeg i nestajućeg gradijenta se još mogu koristiti:

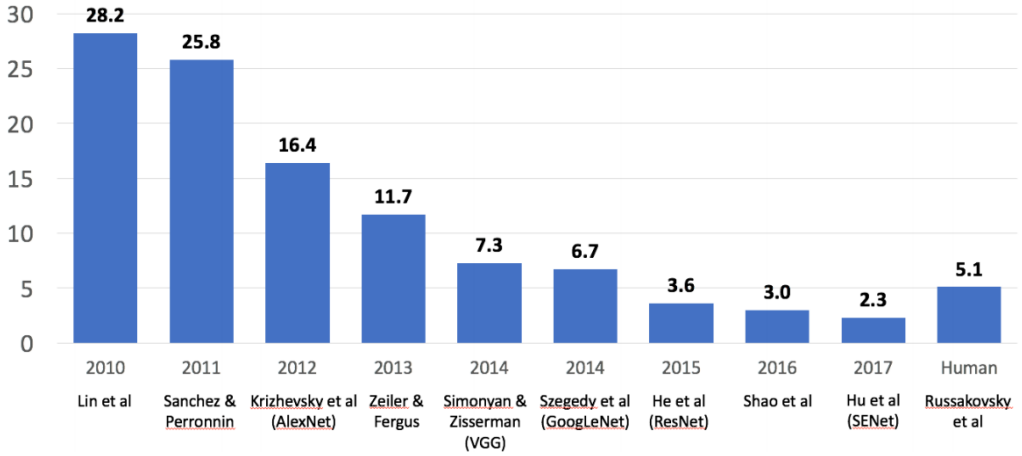
- Adaptivne stope učenja (engl. *adaptive learning rates*)
- Metode konjugiranog gradijenta (engl. *conjugate gradient methods*)
- Normalizacija mini-skupa podataka (engl. *batch normalization*)
- Regularizacijske tehnike
- Smanjivanje broja slojeva u mreži

3.2. Duboko strojno učenje

Duboko učenje, koje se često u literaturi može pronaći i pod nazivom duboke neuronske mreže, je grana strojnog učenja. Zasnovano je na neuronskim mrežama koje se sastoje od: ulaznog, izlaznog i više skrivenih slojeva te za cilj ima iterativno učenje iz podataka. Duboko učenje pokušava simulirati način na koji ljudski mozak funkcionira kako bi se mogli riješiti apstraktni i slabo definirani zadaci. Duboka neuronska mreža se od klasične neuronske mreže razlikuje po tome što sadrži više od jednog skrivenog sloja. Funkcionira tako da svaki skriveni sloj uzima kao ulaz rezultate iz prijašnjeg sloja te ih obrađuje i prosljeđuje pomoću težina idućem sloju u mreži. Što se više slojeva nalazi u dubokoj neuronskoj mreži, to je ona sposobna rješavati kompleksnije zadatke. Svaki idući skriveni sloj prepoznaje kompleksnije značajke.

Duboko učenje je koncept koji omogućuje računalu izgradnju kompliciranih modela na temelju jednostavnijih koncepata, kao što su: rubovi, krugovi i razni jednostavni oblici. To je pristup strojnom učenju koji se temelji na poznavanju ljudskog mozga, statistike i primijenjene matematike. Posljednjih godina se dogodio veliki rast u popularnosti ovog pristupa i njegovoj korisnosti, temeljen uglavnom na sve snažnijim računalima, velikim količinama podataka i raznim tehnikama za učenje dubokih neuronskih mreža. Primjena dubokog učenja je profitabilna te se njime koriste najveće svjetske tehnološke kompanije: Google, Microsoft, Facebook, Apple, itd. [13]

Svoj meteorski rast doživljava 2012. godine kad je konvolucijska neuronska mreža osvojila po prvi put ImageNet-ovo natjecanje u kojem je bilo potrebno naučiti mrežu klasificirati skoro 1,5 milijuna slika u 1000 različitih klasa. Krizhevsky-ev Alexnet [14] je snizio grešku mreže s 26,1% iz prethodne godine na 15,3% (Slika 10). Od tad, ovo natjecanje svake godine osvajaju duboke neuronske mreže.



Slika 10. Prikaz najbolji rezultata na ImageNet-ovom natjecanju kroz godine [15]

Na slici 10 je vidljiv značajan pad pogreške mreže između 2011. i 2012. godine. 2015. godine je mreža uspjela nadmašiti ljudsku sposobnost klasifikacije.

3.3. Smještaj dubokog strojnog učenja unutar područja umjetne inteligencije

Duboko učenje se ubrzano počelo razvijati zadnjih 10 godina. Dio je većeg područja nazvanog strojno učenje, a strojno učenje je podskup umjetne inteligencije.

Umjetna inteligencija je grana računalne znanosti koja se bavi simulacijom inteligentnog ponašanja u računalima. Pojam je 1956. prvi počeo koristiti John McCarthy s američkog MIT-a (Massachusetts Institute of Technology). Umjetna inteligencija je pokušaj oponašanja načina na koji čovjek razmišlja. Pomoću inteligentnih softvera se nastoji automatizirati rad, raspoznati govor ili slike, dijagnosticirati bolesti u medicini te doprinijeti znanstvenim istraživanjima. Kad se umjetna inteligencija tek razvila, bila je u stanju riješiti probleme koji su teški ljudima, ali jednostavni računalima. To su problemi koji se mogu opisati matematičkim pravilima. Pravi izazov za umjetnu inteligenciju je zapravo bio rješavati zadatke koji su ljudima jednostavni za izvršiti, ali kompliciran za formalno opisati. To su tipovi problema koje čovjek rješava intuitivno – kao što su razumijevanje izgovorenih riječi ili raspoznavanje slika. Poteškoće nastaju zbog potrebe za točno opisanim uputama. Zato se razvilo područje unutar umjetne inteligencije nazvano strojno učenje.

Strojno učenje stječe znanje izvlačenjem uzoraka iz podataka. Pripada u skup umjetne inteligencije i njime se omogućava učenje sustava temeljeno na podacima, umjesto na programiranju. Strojno učenje se može usporediti s djetetom koje tek kad dođe na svijet ne zna ništa o njemu, no s vremenom ono prilagođava svoje shvaćanje svijeta na temelju iskustva.

Definirao ga je Tom M. Mitchell 1959. godine: „Za računalni program se može reći da uči iz iskustva E za neki zadatak T i s nekom mjerom učinkovitosti P, ako se učinkovitost na zadatku T, mjerena učinkovitošću P, poboljšava s iskustvom E.“ [16]

Neuronske mreže, kojima će se baviti ovaj rad, su samo jedna metoda strojnog učenja. Strojno se učenje prema funkciji i formi dijeli na sljedeće metode: [17]

- Regresijske metode (engl. *Regression*)
- Instanca-bazirane metode (engl. *Instance-based Methods*)
- Regularizacijske metode (engl. *Regularization Methods*)
- Stabla odlučivanja (engl. *Decision Tree Learning*)
- Bayesian metode (engl. *Bayesian Methods*)
- Kernel metode (engl. *Kernel Methods*)
- Klaster metode (engl. *Clustering Methods*)
- Asocijativna pravila (engl. *Association Rule Learning*)
- Neuronske mreže (engl. *Neural Networks*)
- Metode redukcije dimenzija (engl. *Dimensionality Reduction*)
- Orkestrirane metode (engl. *Ensemble Methods*).

Koncept strojnog učenja se zasniva na iterativnom smanjivanju pogreške. Proces učenja započinje inicijalnim rješenjem koje je uobičajeno pogrešno. Rješenje koje je dao algoritam strojnog učenja se uspoređuje s onim koje je trebao dati (engl. *ground truth*), računa se pogreška i pomoću nje se prilagođava algoritam. Postupak se dalje ponavlja sve dok se ne postigne zadovoljavajuća razina pogreške. Nakon učenja, algoritam se testira na test skupu pomoću kojeg se ocjenjuje njegova valjanost.

Strojno učenje neuronske mreže dijeli se na 3 vrste:

1) Nadzirano učenje. Za vrijeme učenja u neuronsku mrežu ulaze podaci s već označenim točnim, ali skrivenim odgovorima. Mreža daje više ili manje dobre rezultate te ih uspoređuje s rezultatima koje bi trebala davati i u postupku mijenja težine kako bi se u idućoj iteraciji približila točnom rezultatu. Nakon određenog broja iteracija učenje se

zaustavlja i izračunava se točnost trenirane mreže tako što se gleda za koliko je ulaznih podataka mreža dala točno rješenje. Iteracija je korak u učenju mreže u kojem se podešavaju težine. Težine se mogu podešavati pojedinačno ili grupno. Nakon procesa učenja provodi se testiranje mreže u kojem se u mrežu ubacuju novi podaci i očekuje se da mreža klasificira nove podatke u klase pomoću parametara koje je naučila u trening skupu.

2) Nenadzirano učenje. Kod ovog tipa učenja nijedan ulazni podatak u trening skup ne ulazi sa svojom labelom ili točnim odgovorom. Pošto mreža ne može usporediti ulaze s izlazima, ona grupira podatke na temelju značajki za koje pretpostavlja da su bitne.
3) Polunadzirano učenje. Služi se kombinacijom nadziranog i nenadziranog učenja. Njime se koristi kad neki podaci imaju označene odgovore, a neki ne. Prvo se mreža uči na označenim (labeliranim) podacima, a kad se naučila na njima u mrežu se ubacuju neoznačeni podaci koji su slični testnom skupu podataka u nadziranom tipu učenja. Neoznačenim podacima se pogađaju klase. Nakon toga ispravno označeni podaci i podaci koje je pogađala neuronska mreža ponovno prolaze kroz neuronsku mrežu koja na njima još jednom uči. Polunadzirano učenje se može koristi u slučaju kad je količina neoznačenih podataka, kao i vrijeme koje je potrebno uložiti u njihovo označavanje – prevelika.

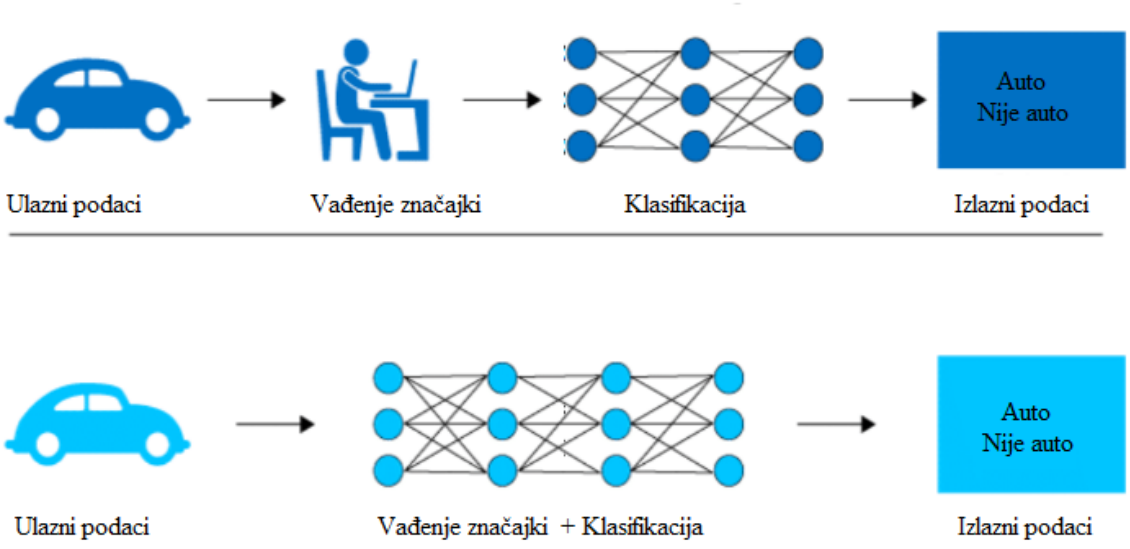
U zadnje vrijeme se polje strojnog učenja počelo ubrzano razvijati, a zasluge ubrzanom razvitku se pripisuju: [18]

- Procesori postaju sve moćniji
- Troškovi upravljanja i pohranjivanja velikih količina podataka dramatično opadaju. Nova, inovativna rješenja doprinijela su bržoj izvedbi i sposobnosti analiziranja velikih setova podataka.
- Mogućnost preraspodjele računalne obrade na više računala ubrzala je obradu i analizu velikih količina podataka
- Postoji mnogo komercijalnih setova podataka na kojima se može vršiti analiza, uključujući: podatke o vremenu, podatke iz društvenih mreža, podatke iz medicine...
- Velika internetska zajednica. Algoritmi strojnog učenja su dostupni kroz velike „open-source“ zajednice na internetu. Stoga, postoji mnogo izvora i knjižnica koje su pojednostavnile razvoj.

- Nije potrebno biti znanstvenik o podacima kako bi se strojno učenje primijenilo u pojedinoj industriji.

Slika 11. Duboko učenje – izvlačenje značajki podataka [12]

Na slici 11 se nalazi grafički prikaz dubokog učenja u odnosu na strojno učenje i umjetnu inteligenciju. Umjetna inteligencija je najširi pojam koji obuhvaća svako oponašanje ljudskog ponašanja pomoću računala. Strojno učenje je dio umjetne inteligencije koji za cilj ima učenje na temelju iskustva. Kod strojnog učenja ne postoji niz egzaktnih „*if-then*“ pravila, već algoritam učenja kroz više iteracija pokušava poboljšati performanse. Kod dubokog učenja se od analitičara ne očekuje izvlačenje značajki, nego algoritam na temelju velike količine ulaznih podataka prepoznaje koje su značajke bitne za određivanje rješenja.



Slika 12. Usporedba procesa učenja kod strojnog i dubokog učenja [19]

Na slici 12 se vidi kako je kod strojnog učenja potrebno bolje poznavanje samog problema od strane analitičara jer se od njega zahtijeva ručno isprobavanje različitih značajki, dok algoritam

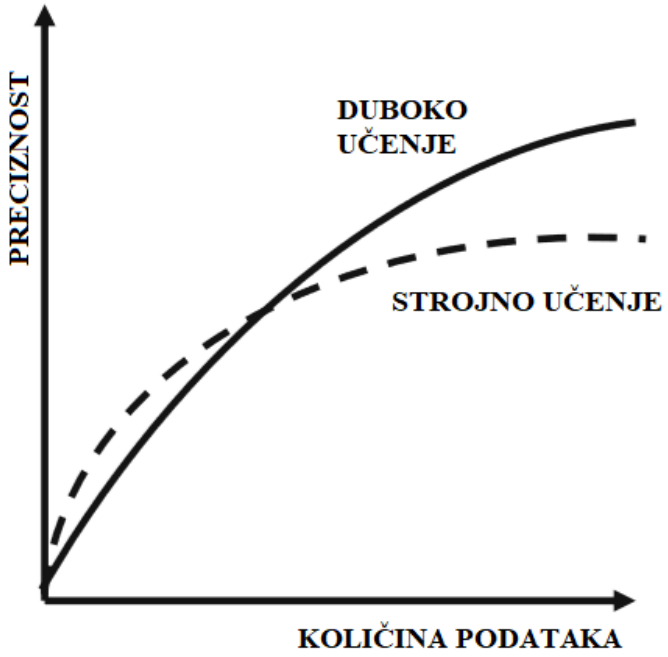
dubokog učenja sam vadi značajke i daje informacije o podacima, na temelju velike količine podataka.

U tablici 6 se mogu vidjeti ključne razlike između koncepata strojnog i dubokog učenja.

Tablica 6. Usporedba prednosti i nedostataka strojnog i dubokog učenja [10]

	PREDNOSTI	NEDOSTACI
Strojno učenje	Dobri rezultati s malim setovima podataka	Potrebno je isprobavati različite značajke i klasifikatore kako bi se postigli najbolji rezultati
	Brzo treniranje modela	Točnost ograničena
Duboko učenje	Automatski uči značajke i klasifikatore	Zahtijeva veće skupove podataka
	Točnost potencijalno neograničena	Računalno zahtjevno

Za strojno učenje se odlučuje ako se ne posjeduje veliki set podataka kao ni dovoljno dobro računalo za njihovu obradu. Duboko učenje često zahtjeva, ovisno o problemu koji se rješava, stotine tisuća ili milijune slika kako bi se postigao optimalan rezultat. Dokaz tome se vidi na slici 13 koja ilustrira kako se s dovoljno velikom količinom podataka s algoritmima dubokog učenja mogu postići bolji rezultati nego sa strojnim učenjem.



Slika 13. Odnos preciznosti i količine podataka kod dubokog i strojnog učenja [20]

Duboko učenje je zahvaljujući povećanju moći računala i sniženju njihove cijene postalo široko primjenjivo. Glavni pokretač napretka je povećanje računalne memorije, koje sa sobom nosi mogućnost učitavanja velikih setova podataka. Preciznost dubokog učenja raste s povećanjem broja ulaznih podataka.

3.4. Primjena dubokog strojnog učenja

Primjenom koncepta dubokog učenja došlo je do velikih pomaka u području umjetne inteligencije (raspoznavanje lica, autonomna vozila, prevođenje teksta u govor). To su prepoznale i iskoristile velike svjetske tvrtke kao što su npr. Google, koji pomoću dubokog učenja poboljšava svoju tražilicu i Amazon koji ga je implementirao u svog glasovnog asistenta Alexu.

Duboko učenje se može primijeniti na:

- Raspoznavanju lica
- Klasifikaciji slika
- Raspoznavanju govora
- Pretvaranju teksta u govor
- Raspoznavanju rukopisa
- Medicinska dijagnostika
- Autonomna vozila

- Raspoznavanje anomalija (bankovno poslovanje)
- Reklame na internetu
- Predviđanje (cijene dionica, sportski rezultati, rezultati izbora,...)
- Google pretraživanje
- Filtriranje nepoželjne elektronske pošte
- Održavanje strojeva

4. KONVOLUCIJSKE NEURONSKE MREŽE

Konvolucijska neuronska mreža je duboka neuronska mreža specijalizirana za raspoznavanje slika. Oponaša način na koji vizualni korteks mozga obrađuje i raspoznaje slike. Nosi taj naziv jer u svojoj strukturi posjeduje barem jedan sloj koji obavlja operaciju konvolucije.

Konvolucijske neuronske mreže svoj izvor imaju u istraživanjima Hubela i Wiesela na vizualnom korteksu mačke. [20]

Hubel i Wiesel su 1959. godine proveli niz istraživanja na živoj mački kojima su dokazali kako određeni dijelovi vizualnog polja pobuđuju određene neurone u mozgu mačke. Vizualni korteks mačke se sastoji od malih regija stanica koje su osjetljive na određene predjele u njenom vizualnom polju. Koje će se stanice pobuditi ovisi o obliku i orijentaciji predmeta u vizualnom polju. Tako vertikalni rubovi u vizualnom polju pobuđuju jedne neuronske stanice, a horizontalni rubovi uzrokuju pobuđivanje drugih neuronskih stanica. Stanice su povezane slojevitom arhitekturom, a to je otkriće dovelo do pretpostavke da sisavci koriste te različite slojeve za konstruiranje dijelova slike na različitim razinama apstrakcije. Konvolucijske neuronske mreže su nastale na tom primjeru. One uočavaju primitivnije oblike u ranijim skrivenim slojevima, a složenije oblike u kasnijim.

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

Konvolucijske mreže su razvijene 1980-ih no nakon toga su bile zaboravljene te su doživjele svoj povratak u središte zanimanja nedavnim ubrzanim razvojem računalne tehnologije.

Od 2012. godine konvolucijske mreže se počinju ubrzano razvijati te su preuzele vodeću ulogu u vizijskim sustavima. Danas najbolje konvolucijske mreže premašuju ljudsku izvedbu, što se smatralo nemogućim prije samo nekoliko desetljeća. [21]

Slojevi koji čine konvolucijsku neuronsku mrežu su:

- Ulazni sloj (engl. *input layer*)
- Konvolucijski sloj (engl. *convolutional layer*)
- ReLU sloj

- Sloj sažimanja (engl. *pooling layer*)
- Potpuno povezani sloj (engl. *fully-connected layer*)
- Izlazni sloj (engl. *output layer*)

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

Konvolucijski sloj zadužen je za obavljanje operacija konvolucije. Konvolucija je matematička operacija dviju funkcija kako bi se dobila treća funkcija koja opisuje kako oblik jedne funkcije utječe na drugu. Za operaciju konvolucije su zadužene jezgre dimenzija F×F od kojih svaka prelazi preko ulazne matrice, obavljajući pritom operaciju konvolucije. Broj jezgri određuje broj mapa značajki na izlazu iz konvolucijskog sloja. Mapa značajki je matrica koja nastaje kao rezultat primjenjivanja jezgre na prijašnjem sloju. Prvi konvolucijski slojevi prepoznaju jednostavnije oblike (rubove, boje) dok su slojevi dublje u mreži u stanju prepoznati kompliciranije oblike.

ReLU sloj najčešće dolazi nakon konvolucijskog, pa se često zna i pribrojiti konvolucijskom sloju. Čini ga funkcija koja sve negativne vrijednosti pretvara u 0. Iako se mogu koristiti i ostale aktivacijske funkcije, ReLU je toliko često korišten u konvolucijskim mrežama da se i sam sloj zove po toj funkciji.

Sloj sažimanja dolazi nakon konvolucijskog i ReLU sloja. Sažimanje ne sadrži parametre koje mreža mora naučiti. Sloj sažimanja se kreće isto kao i jezgra po ulaznoj matrici nekim korakom *s* i veličinom receptivnog polja *F* te se kao izlaz dobiva manja matrica u odnosu na ulaznu. Sloj sažimanja smanjuje prenaučenost mreže, ali i memorijske zahtjeve jer smanjuje broj parametara.

Ova 3 sloja se mogu ponavljati i do nekoliko puta. Nakon tih slojeva u mreži dolazi potpuno povezani sloj. Njega sačinjavaju neuroni povezani sa svim neuronima iz prijašnjeg sloja mreže.

Osnovna razlika između neurona u konvolucijskom sloju i potpuno povezanom sloju je ta da su neuroni u konvolucijskom sloju povezani svaki sa svojim receptivnim poljem u prijašnjem sloju, a u potpuno povezanom sloju je svaki neuron povezan sa svakim iz prijašnjeg sloja. Također, u konvolucijskom se sloju javlja dijeljenje težina.

Zbog svoje potpune povezanosti sloj sadrži velik broj parametara. Ovaj sloj se isto može ponoviti nekoliko puta. U posljednjem, potpuno povezanom sloju, koji je ujedno i izlazni sloj nalazi se aktivacijska funkcija koja računa rezultat. Bira se ovisno o tipu problema kojeg se

nastoji riješiti (klasifikacija ili regresija). Ako se rješava problem klasifikacija onda se najčešće koristi softmax funkcija. Softmax funkcija daje vjerojatnosti za svaku klasu (4.1):

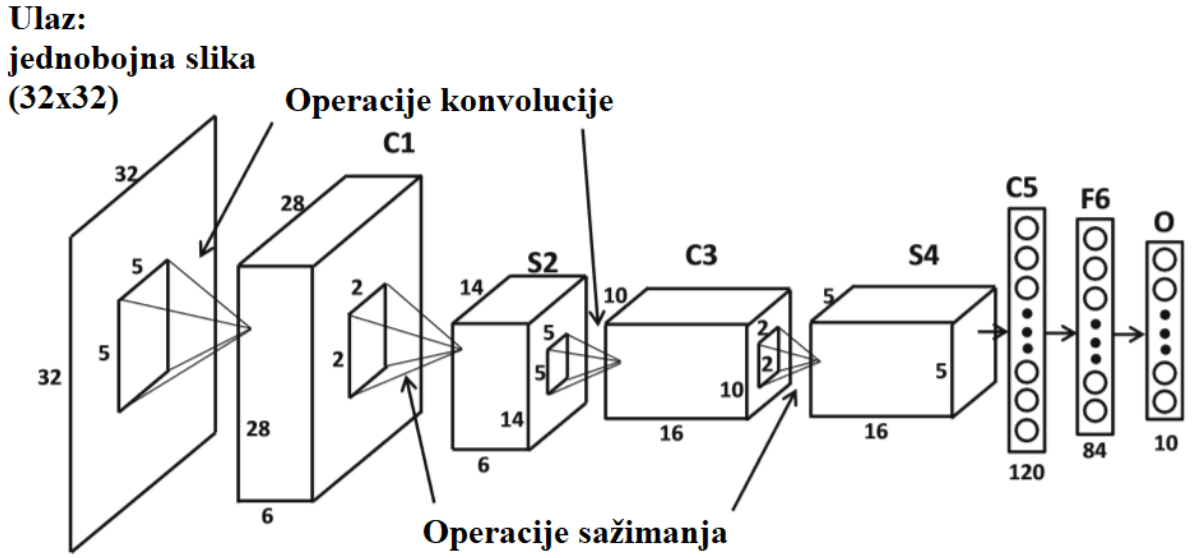
$$L_i = -log\left(\frac{e^{f_{y_i}}}{\sum_j e^{f_j}}\right) \tag{4.1}$$

Svaki neuron predstavlja po jednu klasu. Zbroj svih vrijednosti neurona u posljednjem sloju iznosi 1. Softmax sloj kao funkciju gubitka koristi funkciju gubitka u obliku unakrsne entropije (4.2) (engl. *cross-entropy*) za zadatak binarne klasifikacije, a za zadatak višeklasne klasifikacije ta se jednadžba pretvara u oblik (4.3)(3.3).

$$E(w) = -\sum_{n=1}^{N} \{t_n ln y_n + (1 - t_n) ln(1 - y_n)\} \tag{4.2}$$

$$E(w) = -\sum_{n=1}^{N}\sum_{k=1}^{K} \{t_{nk} ln y_k(x_n, w)\} \tag{4.3}$$

Na slici 14 je prikaz jedne od prvih konvolucijskih neuronskih mreža – LeNet-5. LeNet-5 se koristio za prepoznavanje ručno pisanih brojeva, najčešće od strane banaka. Konvolucijske neuronske mreže se nisu značajno promijenile od tad. Najveća je razlika u većem broju slojeva i različitim aktivacijskim funkcijama. Mreža se sastojala od 2 konvolucijska sloja, 2 sloja sažimanja te od 3 potpuno povezana sloja. Na slici se ne vidi aktivacijski sloj, ali on se zapravo nalazi nakon svakog sloja sažimanja. LeNet-5 je koristio sigmoidalnu aktivacijsku funkciju, dok danas većina konvolucijskih neuronskih mreža koristi ReLU funkciju. Na kraju mreže se nalaze 3 potpuno povezana sloja, od kojih zadnji pomoću softmax funkcije daje rezultat, tj. izražava vjerojatnost za svaku znamenku od 0 do 9. Softmax funkcija se koristi na izlazu u slučaju kad postoji više klasa te je za svaku klasu potrebno odrediti vjerojatnost da je ta klasa ispravan odgovor. Zbroj vjerojatnosti svih klasa iznosi 1.



Slika 14. Arhitektura LeNet-5 [20]

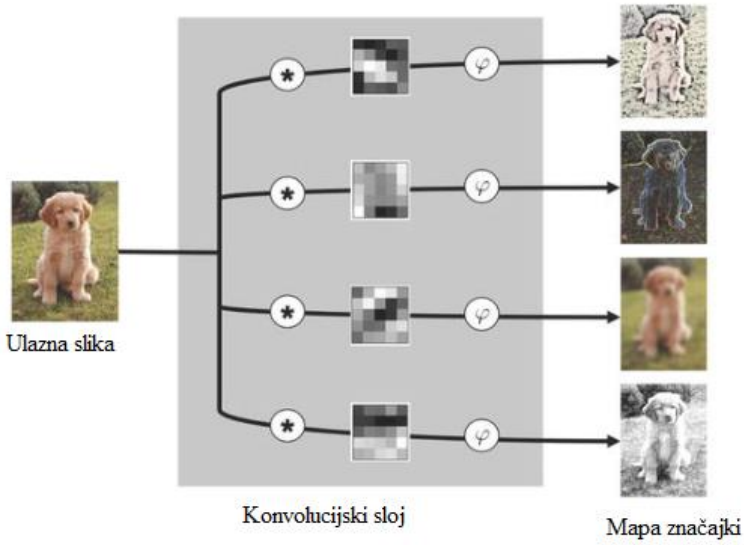
U idućim će se potpoglavljima objasniti temeljne operacije u konvolucijskoj neuronskoj mreži: konvolucija, sažimanje, dopunjavanje te algoritam povratnog prostiranja pogreške koji je nešto različit u odnosu na umjetnu neuronsku mrežu.

4.1. Konvolucija

Konvolucijski sloj funkcionira tako da pomoću jezgri (engl. *filters, kernels*) pretvara ulazne slike u izlazne, tzv. mape značajki (engl. *feature maps*). Proces se sastoji od toga da se na ulaznu matricu primijeni unaprijed određeni broj jezgri, na svaku se jezgru doda pomak (engl. *bias*) te na kraju nelinearna aktivacijska funkcija kako bi se dobila izlazna matrica (mapa značajki). Veličina jezgre se definira visinom i širinom njene matrice. U većini se slučajeva za jezgre koriste manje matrice dimenzija 3x3 ili 5x5. Svaki element unutar jezgre, odnosno njene matrice je jedna težina. Jezgra veličine 5x5 bi tako imala 25 težina. Jezgre su uobičajeno manje od ulaznih matrica, a njihov broj u konvolucijskom sloju odgovara broju mapa značajki koje će taj sloj iznjedriti. Operacija konvolucije se izvodi primjenjujući formulu (4.4):

$$\begin{pmatrix} a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 \\ a_7 & a_8 & a_9 \end{pmatrix} \times \begin{pmatrix} k_1 & k_2 & k_3 \\ k_4 & k_5 & k_6 \\ k_7 & k_8 & k_9 \end{pmatrix} = \sum_{i=1}^{9} a_i \times k_i \qquad (4.4)$$

U jednadžbi (4.4) prva matrica predstavlja ulaznu matricu, a druga matrica predstavlja jezgru. Produkt ovog umnoška je vrijednost koja se u mapi značajki naziva neuron ili piksel. Mapa značajki se sastoji od neurona, a njihov je broj jednak broju elemenata u mapi značajki pomnožen s brojem jezgri.



Slika 15. Pretvorba ulaza u izlaze u konvolucijskom sloju CNN-a [21]

Na slici 15 je prikazana pretvorba ulazne slike u 4 mape značajki. U konvolucijskom se sloju nalaze 4 jezgre od kojih je svaka zadužena za sebi svojstvenu pretvorbu ulaza u izlaz. Svaka jezgra ima svoju aktivacijsku funkciju.

Jezgre u konvolucijskom sloju su matrice kvadratnog oblika s nasumično odabranim veličinama u njima. Mreža uči tako da se vrijednosti u jezgrama iterativno mijenjaju korištenjem algoritma povratnog prostiranja pogreške. U konvolucijskim neuronskim mrežama vrijednosti u jezgrama su analogne težinama u običnim neuronskim mrežama. Jezgre su uglavnom puno manje od matrice koju obrađuju. Jezgre otkrivaju značajke. Npr. jezgra $\begin{pmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{pmatrix}$ otkriva vertikalne rubove, dok jezgra $\begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{pmatrix}$ služi za otkrivanje horizontalnih rubova.

$$\begin{bmatrix} 1 & 1 & 1 & 3 \\ 4 & 6 & 4 & 8 \\ 30 & 0 & 1 & 5 \\ 0 & 2 & 2 & 4 \end{bmatrix} \times \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 7 & 5 & 9 \\ 4 & 7 & 9 \\ 32 & 2 & 5 \end{bmatrix}, \begin{bmatrix} 5 & 7 & 7 \\ 36 & 4 & 9 \\ 0 & 3 & 7 \end{bmatrix} \quad (4.5)$$

U jednadžbi (4.5) je prikazan račun koji se zove konvolucija. Na ulazu u konvolucijski sloj se nalazi slika koju predstavlja matrica dimenzija 4x4. U konvolucijskom se sloju nalaze 2 jezgre, obje dimenzija: 2x2. Množenjem svake jezgre u konvolucijskom sloju s ulaznom matricom dobiju se 2 mape značajki. Množenje se odvija tako da jezgra prelazi preko ulazne matrice, u njoj se povezuje s lokalnom receptivnom regijom koja je jednaka dimenzijama jezgre i obavlja operaciju konvolucije kao u jednadžbi (4.4). Nakon prve operacije konvolucije dobije se vrijednost jednog neurona u mapi značajki. Nakon što je izračunat prvi neuron, jezgra se

nastavlja kretati u desno po ulaznoj matrici nekim korakom *s* (engl. *stride*) te se spaja sa sljedećom lokalnom receptivnom regijom kako bi izračunala vrijednost drugog neurona u mapi značajki. Jezgra se kreće po ulaznoj matrici s lijeva na desno te se tako i rezultati konvolucije zapisuju u mapu značajki. Kad je jezgra, krećući se korakom *s* u desno došla do desnog ruba matrice, spušta se za korak *s* prema dolje te opet s lijeva na desno računa vrijednosti neurona u mapi značajki. Objašnjenje procesa pretvorbe ulazne matrice u mapu značajki za jednadžbu (4.5) izgleda ovako: Prva mapa značajki se dobila tako da je ulazna matrica dimenzija 4×4 konvoluirala s prvom jezgrom dimenzija 2×2.

$$\begin{bmatrix} 1 & 1 & 1 & 3 \\ 4 & 6 & 4 & 8 \\ 30 & 0 & 1 & 5 \\ 0 & 2 & 2 & 4 \end{bmatrix} \times \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 7 & 5 & 9 \\ 4 & 7 & 9 \\ 32 & 2 & 5 \end{bmatrix} \tag{4.6}$$

Konvolucija započinje tako da se jezgra $\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$ pomnoži s gornjom lijevom pod-matricom jednake veličine $\begin{pmatrix} 1 & 1 \\ 4 & 6 \end{pmatrix}$, čime se dobije broj 7 u elementu (1,1) mape značajki**.**

Zatim se jezgra pomiče jedan korak u desno te se pod-matrica $\begin{pmatrix} 1 & 1 \\ 6 & 4 \end{pmatrix}$ iz ulazne matrice množi s istom jezgrom i dobiva se broj 5 u idućem neuronu. Operacija konvolucije za drugu pod-matricu ulazne matrice, koristeći brojeve, izgleda ovako: 1*1 + 1*0 + 6*0 + 4*1 = 1+4 = 5

Jezgra se u idućem koraku ponovno kreće u desno za jedan korak i množi se matricom $\begin{pmatrix} 1 & 3 \\ 4 & 8 \end{pmatrix}$ te se dobiva vrijednost 9 u mapi značajki. Pošto je jezgra došla do desnog ruba ulazne matrice ona se pomiče opet na njen početak, ali se spušta za jedno polje. Množi se s pod-matricom $\begin{pmatrix} 4 & 6 \\ 30 & 0 \end{pmatrix}$.

$$\begin{bmatrix} 1 & 1 & 1 & 3 \\ 4 & 6 & 4 & 8 \\ 30 & 0 & 1 & 5 \\ 0 & 2 & 2 & 4 \end{bmatrix} \times \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 7 & 5 & 9 \\ 4 & 7 & 9 \\ 32 & 2 & 5 \end{bmatrix} \tag{4.7}$$

U mapi značajki u jednadžbi (4.7) zatamnjen je element s brojem 32 kako bi se ukazalo da je broj 32 najveća vrijednost u dobivenoj mapi značajki. 32 je najveća vrijednost u mapi značajki zato što se jezgra $\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$ u tom neuronu podudara s pod-matricom $\begin{pmatrix} 30 & 0 \\ 0 & 2 \end{pmatrix}$ ulazne matrice. Mapa značajki ima velike vrijednosti tamo gdje se vrijednosti jezgre poklapaju s receptivnim poljem. U ovom slučaju su i jezgra i pod-matrica dijagonalni, što je naznačeno u mapi značajki većom vrijednosti na tom mjestu.

U opisanom slučaju jezgra se kretala po ulaznoj matrici korakom s=1. No taj korak može biti i veći, npr. 2 ili 3. Kad bi korak iznosio 2 to bi značilo da se jezgra nakon što obavi prvu operaciju u gornjem lijevom uglu, seli 2 mjesta u desno za iduću operaciju. U ovom bi slučaju korak s=2 rezultirao mapom značajki veličine 2x2 sa samo 4 vrijednosti unutar nje.

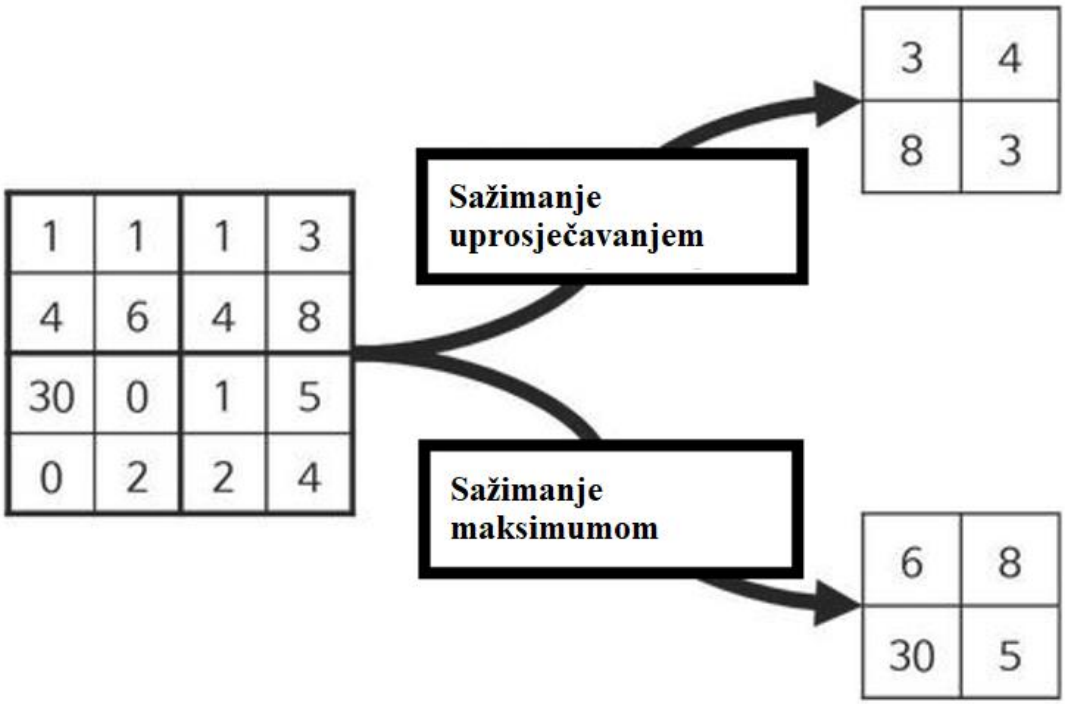
Formula za veličinu mape značajki glasi:

$$n_B = \left(\frac{n_A - n_K}{s} + 1\right) \tag{4.8}$$

Ulazna matrica veličine $n_A \times n_A$, jezgra veličine $n_K \times n_K$ i korak veličine s daju izlaznu matricu (mapu značajki) veličine $n_B \times n_B$. Korak je veličina o kojoj će ovisiti za koliko će se polja jezgra pomicati, horizontalno i vertikalno, po ulaznoj matrici.

4.2. Sloj sažimanja

Sažimanje (engl. *pooling*) se koristi kako bi se smanjila vjerojatnost pojave prenaučenosti, ali i kako bi se smanjenjem dimenzija mapa značajki smanjilo opterećenje za računalo tijekom procesa učenja. Sloj sažimanja smanjuje ulaznu matricu tako što od susjednih piksela izuzima one piksele s najvećom vrijednosti ili računa njihovu srednju vrijednost. Broj piksela iz kojih se izuzima vrijednost može varirati ovisno o veličini matrice koja vrši sažimanje i njenom koraku. Proces sažimanja će se opisati pomoću slike 16.



Slika 16. Sažimanje uprosječavanjem i sažimanje maksimumom [21]

Na slici 16 se vide dva prijašnje spomenuta tipa sažimanja: sažimanje uprosječavanjem (engl. *mean pooling)* i sažimanje maksimumom (engl. *max pooling*). Analitičar odabire veličinu

hiperparametara u sloju sažimanja: veličinu jezgre i korak jezgre. Veličina jezgre predstavlja veličinu matrice koja će vršiti sažimanje, a korak jezgre korak pomicanja te matrice po ulaznoj matrici. Kod sažimanja uprosječavanjem ulaznu se matricu podijelilo na 4 jednaka dijela (veličina jezgre 2x2, korak jezgre 2) i iz svakog se uzela srednja vrijednost, npr. za prvi kvadrat srednja vrijednost glasi:

$$\frac{a_{11}+a_{12}+a_{21}+a_{22}}{n} \tag{4.9}$$

Primjenom ove formule za prvi kvadrant bi se dobila vrijednost 3.

Veličina matrice $n_B \times n_B$ nakon sažimanja se dobije korištenjem jednadžbe (4.8) gdje je n_A veličina jedne dimenzije ulazne matrice, n_K veličina jedne dimenzije kvadratne jezgre koja se kreće po ulaznoj matrici i vrši sažimanje, a s iznos koraka kojim se ta jezgra kreće po ulaznoj matrici:

Sažimanjem se gubi točna lokacija neke značajke, ali ona u konvolucijskim neuronskim mrežama ni nije toliko bitna koliko je bitna njena relativna pozicija u odnosu na druge značajke.

Kod sažimanja maksimumom na slici 16 umjesto srednje vrijednosti iz svakog od 4 kvadrata vadila se najveća brojčana vrijednost.

4.3. Dopunjavanje

U prijašnjem primjeru konvolucije primijećeno je kako je matrica značajki koja je nastala procesom konvolucije manjih dimenzija od ulazne matrice. Ulazna matrica je bila veličine 4x4, a dobivena izlazna matrica je dimenzija 3x3. Stoga, kako bi se očuvala veličina izlazne matrice jednaka onoj na ulazu, potrebno je provesti proces dopunjavanja (engl. *padding*). Dopunjavanje se izvodi kako bi veličina matrice nakon konvolucijske obrade bila jednaka veličini matrice prije konvolucijske obrade. Provodi se tako da se prije samog procesa konvolucije matricu okruži nulama oko svakog ruba. Broj nula koje će se dodati oko ulazne matrice se može izraziti korištenjem formule:

$$n_B = \frac{n_A + 2p - n_K}{s} + 1 \tag{4.10}$$

gdje p izražava broj potrebnih slojeva s nulama kako bi matrica nakon procesa konvolucije bila jednakih dimenzija ulaznoj matrici. U formuli je n_B veličina mape značajki, n_A veličina ulazne matrice, n_K veličina jezgre.

Operacija dopunjavanja sprječava gubljenje informacija koje ulazne matrice imaju na svojim rubovima, jer da nema dopunjavanja jezgra koja prelazi preko ulazne matrice bi preko nekih piksela prešla samo jednom, a preko nekih više puta. Prešla bi više puta preko piksela koji nisu

na rubu slike, a rubne piksele ne bi bila u mogućnosti na jednak način obuhvatiti. Time bi došlo do gubitka informacija s rubova ulazne matrice. Bez dopunjavanja bi mreže s puno slojeva nakon samo par operacija konvolucije imale značajno smanjene mape značajki.

Postoje 2 tipa dopunjavanja:

- Valjano (engl. *valid padding*). Ulazne matrice se ne dopunjavaju nulama pa se mape značajki nakon konvolucije smanjuju.
- Nepromijenjeno (engl. *same padding*). Ulazna matrica se dopunjava nulama tako da mapa značajki bude jednakih dimenzija nakon konvolucije kao što je bila i ulazna matrica.

4.4. Algoritam povratnog prostiranja pogreške

Povratno prostiranje pogreške u konvolucijskim mrežama se razlikuje od povratnog prostiranja pogreške kroz klasičnu neuronsku mrežu. Razlika je u tome što su iste vrijednosti težina podijeljene na sve lokalne receptivne regije u ulaznoj matrici.

$A^{(l-1)}$ je izlazna mapa značajki konvolucijskog sloja *(l-1).* Djelovanjem jezgre iz aktivacijskog sloja l na mapu značajki $A^{(l-1)}$ dobije se mapa značajki $A^{(l)}$. U nastavku će se koristiti oznake:

J – funkcija gubitka

Cilj je smanjiti pogrešku mreže tako što će se izračunati vrijednost gradijent funkcije pogreške J u smjeru težina $W^{(l)}$ i pomaka $b^{(l)}$ u sloju l neuronske mreže. U unaprijednoj fazi konvolucijski slojevi računaju iznose mapa značajki A. One propagiraju kroz iduće slojeve mreže, sve dok ne dođu do izlaznog sloja u kojem se očitava pogreška. U jednadžbi (4.11) se nalazi parcijalna derivacija funkcije pogreške J po težini, u jednadžbi (4.12) parcijalna derivacija pogreške J po ulazu, a u jednadžbi (4.13) parcijalna derivacija pogreške J po mapi značajki.

$$dW^{(l)} = \frac{\partial J}{\partial W^{(l)}} \tag{4.11}$$

$$dZ^{(l)} = \frac{\partial J}{\partial Z^{(l)}} \tag{4.12}$$

$$dA^{(l)} = \frac{\partial J}{\partial M^{(l)}} \tag{4.13}$$

Upotrebom lančanog pravila parcijalna derivacija funkcije pogreške J po $W^{(l)}$ iznosi:

$$\frac{\partial J}{\partial W^{(l)}_{c,i,j,k}} = \frac{\partial J}{\partial Z^{(l)}_{m,n,c}} \frac{\partial Z^{(l)}_{m,n,c}}{\partial w^{(l)}_{c,i,j,k}} \tag{4.14}$$

Iz prethodne jednadžbe deriviranjem $Z^{(l)}$ po $W^{(l)}$ se dobiva:

$$dW^{(l)}_{c,i,j,k} = \sum_{m=0}^{M^{(l)}-1} \sum_{n=0}^{N^{(l)}-1} dZ^{(l)}_{m,n,c} A^{(l-1)}_{i+m,j+n,k} \tag{4.15}$$

Iz prethodne jednadžbe se računa $dZ^{(l)}$:

$$dZ^{(l)}_{i,j,k} = \frac{\partial J}{\partial Z^{(l)}_{i,j,k}} = \frac{\partial J}{\partial A^{(l)}_{i,j,k}} \frac{\partial A^{(l)}_{i,j,k}}{\partial Z^{(l)}_{i,j,k}} = \frac{\partial J}{\partial A^{(l)}_{i,j,k}} \frac{\partial}{\partial Z^{(l)}_{i,j,k}} g^{(l)}\left(Z^{(l)}_{i,j,k}\right) = \tag{4.16}$$

$$= dA^{[l]}_{i,j,k} g^{[l]\prime}(Z^{[l]}_{i,j,k})$$

U prethodnoj jednadžbi $dA^{(l)}$ je dobivena iz prethodnog sloja propagacijom unatrag.

Sada slijedi definiranje gradijenta $dA^{(l-1)}$ koji će se propagirati prethodnom sloju mreže. Dobiva se:

$$dA^{(l-i)}_{i,j,k} = \frac{\partial J}{\partial A^{(l-i)}_{i,j,k}} = \sum_{\substack{p,m \\ p+m=i}} \sum_{\substack{r,n \\ r+n=j}} \sum_{C=0}^{C^{(l-1)}} W^{(l)}_{c,m,n,k} dZ^{(l)}_{p,r,c} \tag{4.17}$$

Gdje $i=0,..., M^{(l-1)} - 1$, $j=0,...,N^{(l-1)} - 1$ *i* $k=0,...C^{(l-1)} - 1$. Sad se još trebaju izračunati gradijenti pomaka $db^{(l)}$:

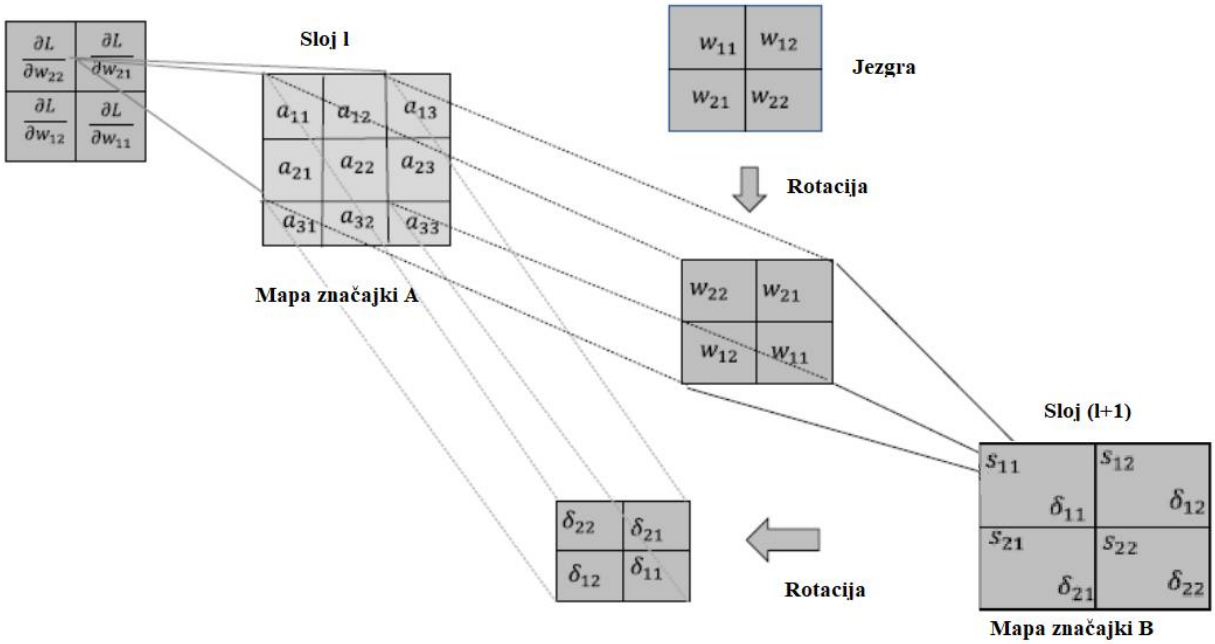
$$dB^{(l)}_{i,j,k} = \sum_{m=0}^{M^{(l)}-1} \sum_{n=0}^{N^{(l)}-1} \sum_{k=0}^{C^{(l)}-1} dZ^{(l)}_{m,n,k} \tag{4.18}$$

Sad kad je definiran izračun gradijenata $dW^{(l)}$ i $db^{(l)}$, ažuriraju se parametri modela $W^{(l)}$ i $b^{(l)}$ u ovisnosti o stopi učenja η:

$$W^{(l)}_{c,i,j,k} \leftarrow W^{(l)}_{c,i,j,k} - \eta \frac{\partial L}{\partial W^{(l)}_{c,i,j,k}} \tag{4.19}$$

$$b^{(l)}_{i,j,k} \leftarrow b^{(l)}_{i,j,k} - \eta \frac{\partial L}{\partial b^{(l)}_{i,j,k}} \tag{4.20}$$

Na slici 17 je grafički prikazano povratno prostiranje pogreške u konvolucijskoj neuronskoj mreži.



Slika 17. Algoritam povratnog prostiranja pogreške [22]

Neuroni u konvolucijskim neuronskim mrežama dijele težine, za razliku od ranije opisane obične neuronske mreže kod koje svaki neuron ima samo sebi svojstvene težine. U konvolucijskom sloju ulazna se matrica povezuje s mapom značajki putem jezgre. Sve vrijednosti unutar mape značajki su nastale množenjem iste jezgre s ulaznom matricom što znači da svaki neuron u nekoj mapi značajki dijeli isti set težina s ostalim neuronima u toj mapi značajki. Dijeljenje težina povećava općenitost naučene značajke i poboljšava sposobnost generalizacije jer na istu jezgru dolaze različiti podaci. Tako mreža uči manji broj parametara. Jezgra utječe na sve neurone u mapi značajki. Slijedom toga, prilikom računanja algoritmom povratnog prostiranja pogreške potrebno je računati vrijednosti grešaka i promjene težina za sve neurone u mapi značajki te na kraju sumirati njihov utjecaj. Dijeljenje težina se svodi na to da svaka težina u jezgri pridonosi svakom neuronu u mapi značajki. Stoga će svaka promjena težine u jezgri utjecati na sve izlazne neurone.

5. OPTIMIZACIJA HIPERPARAMETARA MREŽE

Postoje 2 vrste parametara u neuronskim mrežama:

- Parametri koje mreža uči (engl. *learnable parameters*). Varijable koje mreža uči za vrijeme treninga pomoću algoritma povratnog prostiranja pogreške. To su težine i pragovi. Broj parametara se povećava s povećanjem broja neurona i slojeva.
- Hiperparametri (engl. *hyperparameters*). Varijable koje se zadaju prije samog procesa učenja te se ne uče naknadno. Potrebno ih je odrediti prije nego se u neuronsku mrežu unesu podaci za učenje.

Hiperparametri se dijele u 3 kategorije: [11]

1. Hiperparametri s kontinuiranim realnim brojevima. Mogu poprimiti bilo koju vrijednost: stopa učenja η, parametar regularizacije λ.
2. Hiperparametri s diskretnim vrijednostima, ali teoretski mogu poprimiti beskonačan broj vrijednosti: broj skrivenih slojeva, broj neurona u svakom sloju, broj epoha
3. Hiperparametri koji su diskretne vrijednosti i sadrže konačan broj mogućnosti: optimizator, aktivacijska funkcija, metoda smanjivanja stope učenja (engl. *learning rate decay method*)

Hiperparametri u konvolucijskim neuronskim mrežama su:

- Broj jezgri. Svaki konvolucijski sloj sadrži otprije odabrani broj jezgri. Broj jezgri utječe na broj mapa značajki u svakom sloju. Težine u njima se mijenjaju kako mreža uči.
- Veličina jezgre. Dimenzije jezgre se definiraju širinom i visinom njene matrice. Jezgre su kvadratne matrice uobičajeno dimenzija: 3x3 ili 5x5. Svako polje unutar jezgre predstavlja jednu težinu te se odabirom veličine jezgre odabire i broj težina u njoj. Npr. jezgra dimenzija 5x5 imala bi 25 težina. U slučaju da se izabere premala veličina jezgre, mreža vjerojatno neće biti sposobna prenijeti sve bitne informacije u procesu konvolucije, a prevelika jezgra usporava proces učenja.
- Dopunjavanje (engl. *padding*). Povećavaju se dimenzije ulazne matrice dodajući nule oko rubova kako bi se dobila mapa značajki jednakih dimenzija kao i ulazna matrica. Kod ovog se hiperparametra odlučuje hoće li se dopunjavanje koristiti ili neće (valjano ili nepromijenjeno dopunjavanje), a broj slojeva s nulama (p) se izračunava iz formule (4.8) za dopunjavanje.

- Korak *s* (engl. *stride*). Broj piksela za koje se potrebno pomaknuti u vertikalnom i horizontalnom smjeru kod operacija konvolucije i sažimanja.
- Redoslijed i broj slojeva. Empirijski je pokazano da mreže s više slojeva zahtijevaju mnogo manji broj neurona kako bi postigli iste rezultate i obično generaliziraju bolje do tad ne viđene podatke. [11]
- Stopa učenja η. Najčešće se odabire u iznosu između 0.01 do 0.0001. Jedan od ključnih hiperparametara mreže. O njemu ovisi konvergencija mreže. Stopa učenja se može i dinamički mijenjati sa svakom iteracijom. U tom slučaju se pojavljuju još dodatni hiperparametri:

a) Stubišna metoda smanjivanja stope učenja (engl. *staircase decay*):
$\eta = \begin{cases} 2 \text{ za } j<4 \\ 0.4 \text{ za } j \geq 4 \end{cases}$ Ovdje bi hiperparametri bili: broj iteracija nakon kojih se stopa učenja mijenja i iznos promijene stope učenja η.

b) Korak metoda (engl. *step decay*): $\eta = \frac{\eta_0}{\frac{j}{D+1}}$, η_0 je izraz za prijašnju stopu učenja, *D* je hiperparametar čija se veličina odabire, *j* je redni broj iteracije.

c) Inverzno-vremenska metoda (engl. *inverse time decay*): $\eta = \frac{\eta_0}{1+vj}$, *v* je hiperparametar.

- Aktivacijska funkcija. Neke od najpoznatijih su: sigmoidalna, tangens hiperbolna aktivacijska funkcija, zglobna aktivacijska funkcija, itd. Uobičajeno je da se kroz cijelu mrežu koristi ista aktivacijska funkcija.
- Broj epoha. Epohu sačinjava jedan prolaz svih uzoraka kroz neuronsku mrežu za vrijeme treninga. Nakon svake epohe se može vidjeti pogreška koju mreža daje. Manjim brojem epoha se skraćuje vrijeme učenja, ali i točnost naučene mreže.
- Veličina mini grupe podataka (engl. *batch size*). O ovom hiperparametru ovisi koliko će uzoraka od jednom proći kroz mrežu. Dobar izbor veličine mini grupe znatno ubrzava učenje mreže, ne gubeći previše na njenoj točnosti. Na temelju veličine jednog *batcha* može se odrediti i broj *batch*-eva, što je opisano u prijašnjim cjelinama. Ako se koristi algoritam gradijentni spust s mini grupama, težine će se ažurirati nakon svake mini grupe.
- Parametri regularizacije. Odabir iznosa λ u slučaju da se radi o L2 ili L1 regularizaciji. Postotak isključivanja (engl. *dropout*) se odabire ako se odlučuje za tu metodu

regularizacije. Uglavnom se odabire postotak isključivanja od 20% do 50%. Ima jako dobre rezultate na velikim mrežama.

- Optimizacijska metoda. Nije nužno uvijek koristiti optimizator gradijentni spust. Alternative su: Adam (engl. *Adaptive Moment estimation*), RMSProp (engl. *Root Mean Square Propagation*), itd.
- Inicijalizacija težina (engl. *weight initialization methods*). Potrebno je biti oprezan kod inicijaliziranja težina kako ne bi došlo do spore konvergencije neuronske mreže u samom početku. Pragovi se smiju inicijalizirati nulom. Preporučena inicijalizacija je da se uzimaju vrijednosti iz uniformne distribucije u rasponu $(-r, r)$ gdje je $r=\sqrt{6/(n_{in}+n_{out}}$ za aktivacijsku funkciju hiperbolnog tangensa odnosno $r=\sqrt{6/(n_{in}+n_{out}}$ za logističku funkciju gdje n_{in} i n_{out} predstavljaju broj ulaznih i izlaznih neurona za određeni sloj. [22]
- Funkcija gubitka. Ovisno o problemu (klasifikacija ili regresija) i ulaznim podacima odabire se funkcija gubitka. Unakrsna provjera se najčešće koristi za klasifikaciju, a za regresiju se koriste: srednja kvadratna pogreška, korijen srednje kvadratne pogreške, srednja apsolutna pogreška, itd.
- Sloj sažimanja. Odabire se kojom će metodom neuronska mreža obavljati sažimanje nad mapama značajki i veličina matrice n_K*n_K koja će vršiti sažimanje. Najčešće korištene metode su: sažimanje maksimumom i sažimanje usrednjavanjem.

Optimizacijom hiperparametara nastoji se odabrati set hiperparametara koji će najbolje utjecati na učenje neke neuronske mreže. Problem optimizacije hiperparametara nastoji se pomoću optimizacijskih metoda premjestiti s čovjeka na računalo jer je čovjek sklon pogreškama i nailazi na teškoće pri shvaćanju višedimenzionalnih problema. Optimizacija hiperparametara spada u skup problema crne kutije (engl. *black-box problem*) jer se povezanost izbora hiperparametara s izvedbom mreže koja koristi te hiperparametre ne može opisati matematičkom formulom. Zbog činjenice da se ne mogu izračunati derivacije funkcije, optimizacijske metode kao što su metoda gradijentnog spusta ili Newtonova metoda se ne mogu koristiti. Pravilan izbor hiperparametara ima ključnu ulogu u smanjenju vremena učenja i poboljšanju točnosti mreže.

4 najpoznatije računalne metode za optimiziranje hiperparametara su:

- Mrežno pretraživanje (engl. *Grid search*)
- Nasumično pretraživanje (engl. *Random search*)
- Bayes optimizacija (engl. *Bayesian optimization*)

- Genetički algoritam (engl. *Genetic algorithm*)

Hiperparametre može mijenjati i analitičar, ali vrlo je mala vjerojatnost da će postići optimalan izbor hiperparametara, pogotovo ako se radi o višedimenzionalnom problemu.

U ovom radu, koristit će se tehnika mrežnog pretraživanja za optimizaciju hiperparametara modela. Mrežno pretraživanje svodi se na pretraživanje svih mogućih kombinacija zadanih hiperparametara. Svaka vrijednost *x* je jednako udaljena od prethodne i sljedeće. Što se uzme manji razmak između *x*-ova, to će mreža imati veću vjerojatnost doći blizu maksimumu, ali će za posljedicu imati dulje trajanje učenja. Valja imati na umu da utjecaj hiperparametara na mrežu zavisi o vrijednostima svih ostalih hiperparametara. Ako se odabire između n broja hiperparametara to problem mrežnog pretraživanja čini n-dimenzionalnim. Kako bi se dobio dovoljno dobar rezultat potrebno je izvesti poveći broj ispitivanja. Mrežno pretraživanje garantira optimalan pronalazak najbolje kombinacije zadanih hiperparametara. Posljedica je velika vremenska zahtjevnost jer je potrebno izračunati sve kombinacije.

Algoritam mrežnog pretraživanja se dijeli u 3 koraka:

1. Definiranje n-dimenzionalne mreže. Svaka dimenzija korespondira jednom hiperparametru.
2. Za svaku dimenziju se odabire raspon mogućih iznosa tog hiperparametra. Npr. za stopu učenja se nastoji odabrati najbolju od n vrijednosti: $\eta = n * 10^{-2}$ (n=1,...100) ili za veličinu mini-grupe podataka se nastoji odabrati koliko uzoraka će sačinjavati jednu mini-grupu: 10, 20, 30, 40, 50, itd.
3. Pretraživanje svih mogućih vrijednosti funkcija i izbor najbolje

Metoda ne reagira dobro na porast broja hiperparametara.

U prijašnjem je poglavlju nabrojano 13 hiperparametara. U slučaju da se primjenom mrežne tehnike nastoji ispitati koji bi iznosi hiperparametara bili najbolji za neuronsku mrežu te se biraju vrijednosti za samo njih 6 dobio bi se sljedeći rezultat:

- Stopa učenja = $n*10^{-2}$ (za n=1,...,100) – 10 vrijednosti stope učenja
- Broj konvolucijskih slojeva: 1,2,3,4,5 - 5 vrijednosti
- Veličina jezgri u konvolucijskim slojevima: 3x3 ili 5x5 - 2 vrijednosti
- Korak s = 1,2 - 2 vrijednosti
- Dopunjavanje = da/ne - 2 vrijednosti
- Veličina mini grupe podataka = 10,20,30,40 – 4 vrijednosti

Ovo bi značilo da bi mreža u slučaju da se želi dobiti optimalan odnos parametara iznad trebala istrenirati:

$$10 * 5 * 2 * 2 * 2 * 4 = 1600 \text{ puta}$$

Ovakav pristup „pokušaj i pogreška“ bi se u optimizaciji hiperparametara mogao koristiti u situaciji kad ne postoji prevelik izbor hiperparametara. S obzirom na to da u primjeru iznad broj hiperparametara sugerira kako bi za njihov optimalan izbor bilo dovoljno 1600 ponavljanja, to takav pristup čini ne praktičnim. Zato su znanstvenici pribjegli metodama koje će optimirati hiperparametre na jednostavniji i brži način. Osim navedene tehnike, za optimizaciju hiperparametara se može koristiti i nasumično pretraživanje, ali i naprednije tehnike, kao što su genetski algoritam ili u zadnje vrijeme Bayesova optimizacija. Ipak, zbog jednostavnosti primjene, mrežno pretraživanje ostaje najkorištenija tehnika optimizacije hiperparametara.

6. EKSPERIMENTALNI POSTAV

Praktična faza izrade rada sastojala se od 2 cjeline:

1. Prikupljanje kvarova na simulatoru kvarova u laboratoriju za održavanje
2. Priprema podataka, učenje i testiranje mreže

U ovom poglavlju biti će opisano kako su prikupljani podaci za potrebe razvijanja računalnog modela za procjenu kvarova rotacijske opreme male brzine vrtnje. Implementirana je strategija održavanja po stanju. Dijagnostika se vršila metodom mjerenja i analize vibracija.

Proces procjene stanja opreme vibrodijagnostikom se sastoji od sljedećih koraka [6]:

1. Snimanje vibracija opreme (mjerni pretvornici (senzori))
2. Konverzija vibracijskog u električni signal (analizator /konverteri)
3. Transformacija električnog signala
4. Prikaz prikupljenih podataka i informacija (softver)
5. Spremanje i dokumentiranje podataka i informacija
6. Dijagnostika / Inteligentna dijagnostika (ekspert/software)

Snimanje vibracija opreme i konverzija vibracijskog u električni signal se vršila pomoću 3-osnog IEPE akcelerometra PCB Piezotronics. Transformacija signala se vršila pomoću sustava za prikupljanje podataka NI 9191+NI9234. Prikaz prikupljenih podataka i informacija te njihovo spremanje i dokumentiranje se napravilo u softverskom paketu LabVIEW, a sustav za inteligentnu dijagnostiku kvarova se izradio u MATLAB-u.

Podaci su analizirani i testirani korištenjem duboke konvolucijske neuronske mreže. Kako bi se zadovoljila potreba algoritma dubokog učenja za velikim količinama podataka prikupljeno je ukupno 24000 zapisa. Na simulatoru kvarova SpectraQuest Simulator Expert simuliran je rad rotacijske opreme pri malim brzinama vrtnje:

- 150 okretaja/minuti
- 300 okretaja/minuti

Pri svakoj brzini vrtnje simulirano je sljedećih 8 tipova stanja stroja:

- Normalno stanje (engl. *normal state*, NS)
- Debalans (engl. *imbalance rotor fault*, IMRF)
- Ekscentrični rotor (engl. *eccentric rotor fault*, ERF)
- Nagnuti rotor (engl. *cocked rotor fault*, CRF)
- Kvar vanjske staze kotrljanja (engl. *outer race bearing fault*, ORBF)

- Kvar unutarnje staze kotrljanja (engl. *inner race bearing fault*, IRBF)
- Kvar valjnog elementa (engl. *ball bearing fault*, BBF)
- Kombinirani kvar ležaja (engl. *combinated bearing fault*, CBF)

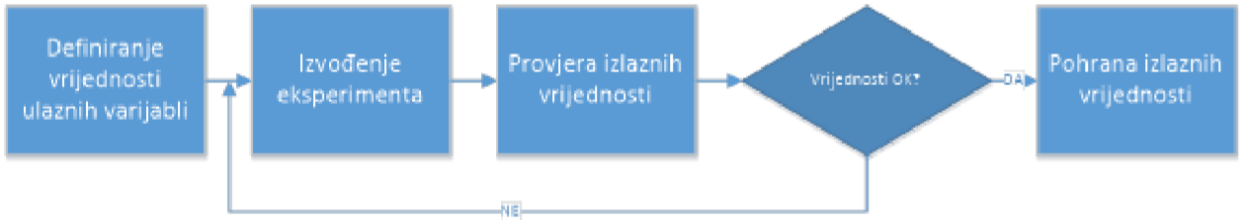
Od navedenih 8 stanja, prvo je stanje normalno stanje, a ostalih 7 su kvarovi simulirani pomoću modula na simulatoru. Pri svakoj brzini vrtnje prikupljeno je 12000 zapisa, što dovodi do brojke od 1500 zapisa za svaki simulirani tip stanja. Količina podataka koju je potrebno prikupiti za uspješno rješavanje pojedinog problema dubokog učenja ne može se sa sigurnošću znati, pa se količina podataka koja se prikuplja uglavnom odabire temeljem iskustva. Vrijedi pravilo da što je zadatak kompleksnije prirode, biti će potreban veći set podataka za njegovo rješavanje.

Vibracijski signali su za vrijeme vrtnje bilježeni preko troosnog senzora (x, y, z, osi) koji se nalazio na kućištu iznad jednog od 2 ležaja vratila. Pomoću softverske platforme LabVIEW na računalu je namješten broj mjerenja u sekundi u iznosu od 12800 Hz. Taj broj označava količinu stanja koja senzor izmjeri u jednoj sekundi. Od tih se mjerenja dobije graf s amplitudom na ordinati i vremenom, koje iznosi 1 sekundu, na apscisi. Svaki graf u trajanju od jedne sekunde predstavlja jedan zapis, odnosno ulaz u konvolucijsku neuronsku mrežu.

Za provođenje eksperimenta i snimanje mjerenih podataka na računalo korištena je sljedeća oprema:

- Simulator kvarova SpectraQuest Simulator Expert
- Troosni IEPE akcelerometar IMI Sensors 356B21
- National Instruments NI 9191+NI9234+LabView2014

Eksperiment se provodi na način definiran dijagramom tijeka za svaku kombinaciju ulaznih varijabli na slici 18.



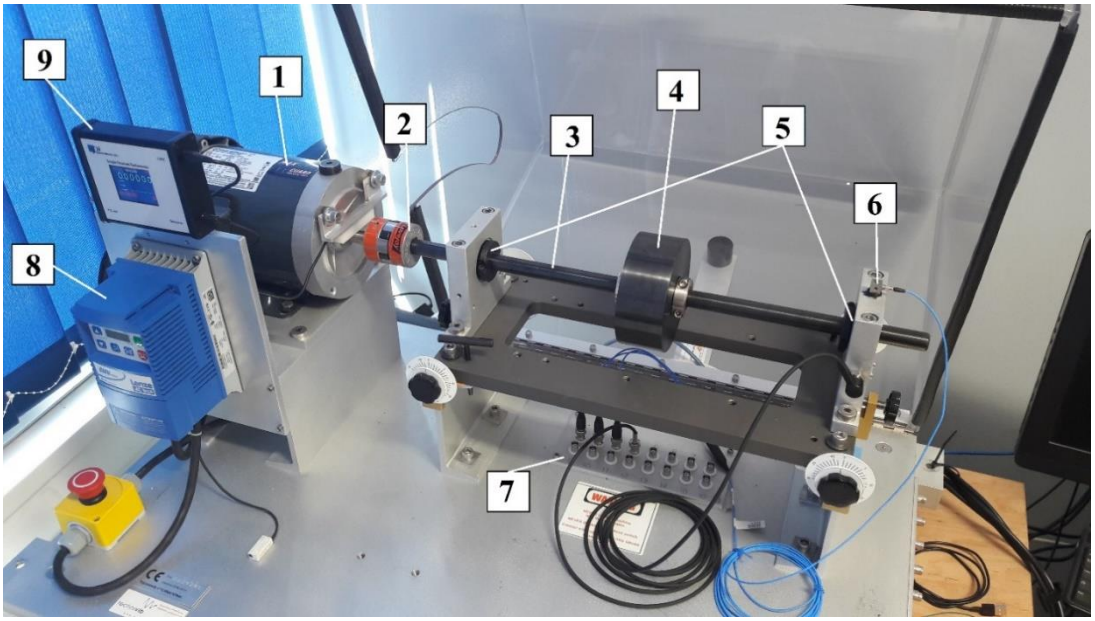
Slika 18. Dijagram tijeka eksperimenta

6.1. Simulator kvarova SpectraQuest Simulator Expert

Simulator kvarova SpectraQuest Simulator Expert je uređaj koji omogućava oponašanje strojne opreme kako bi se bolje razumjela pojava različitih vibracijskih signala. S ovim strojem se u dobro kontroliranim uvjetima mogu razviti i poboljšati vještine potrebne za dijagnozu

industrijske opreme. Nije moguće provesti ovakvu vrstu ispitivanja u industrijskom postrojenju bez ometanja proizvodnog procesa, što bi dovelo do negativnog utjecaja na razinu proizvodnosti i dobiti u poduzeću jer je za provođenje ovakve vrste eksperimenata potrebno uzastopno pokretanje i zaustavljanje strojeva. Simulator omogućava vanmrežno osposobljavanje osoblja i eksperimentiranje koje će zauzvrat smanjiti zastoje u proizvodnji. [23]

Pomoću ovog uređaja se mogu simulirati različitih tipova kvarova pri različitim brzinama vrtnje.



Slika 19. Simulator kvarova SpectraQuest Simulator Expert

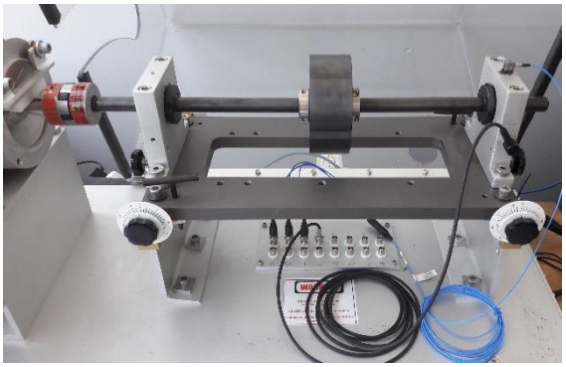
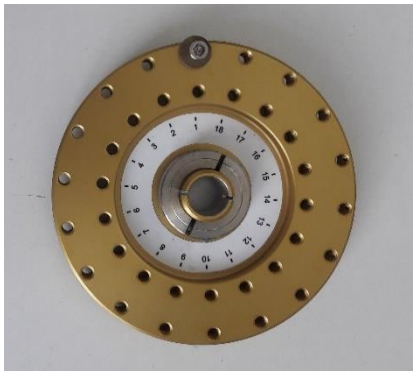
Na slici 19 je prikazan simulator kvarova SpectraQuest Fault Simulator Expert koji se nalazi u Laboratoriju za održavanje na Fakultetu strojarstva i brodogradnje u Zagrebu. Glavne dijelove ovog simulatora čine:

1. Trofazni elektromotor s 1 konjskom snagom. Broj okretaja po minuti može iznositi od 0 do 6000.
2. Kandžasta spojka s gumenim uloškom.
3. Vratilo. Na vratilo se stavljaju različiti moduli, ovisno o tipu kvara koji se nastoji simulirati.
4. Uteg od 5kg.
5. Valjni ležajevi
6. Senzor: 3-osni akcelerometar IEPE PCB 356B21
7. Ploča sa 16 BNC priključaka. Nalazi se ispod vratila, a povezana je sa 17 BNC priključaka na rubu postolja preko kojih se povezuje uređaj za prikupljanje podataka.

8. Frekventni regulator.
9. Ugrađeni tahometar. Instrument za mjerenje brzine vrtnje osovine, vratila ili nekog drugog rotirajućeg tijela.

Simulator kvarova opremljen je trofaznim motorom sa frekvencijskim pretvaračem snage 0,75kW, ugrađenim digitalnim tahometrom i modulom za simuliranje debalansa. Za potrebe eksperimenta, a u cilju simuliranja pojedinih tipova kvarova koristili su se moduli iz tablice 7.

Tablica 7. Moduli za simuliranje kvarova

1. Normalno stanje (NS)	
2. Debalans (IMRF)	
3. Ekscentrični rotor (ERF)	

4. Nagnuti rotor (CRF)	
5. Kvar ležaja • Kvar vanjske staze kotrljanja (ORBF) • Kvar unutarnje staze kotrljanja (IRBF) • Kvar valjnog elementa (BBF) • Kombinirani kvar ležaja (CBF)	

6.2. Troosni IEPE akcelerometar IMI Sensors 356B21

Karakteristike troosnog IEPE akcelerometra prikazanog na slici 20:

- IEPE
- 1,02 mV/g
- ±500g
- Y i Z os 2-10 000 Hz, X os 7000 Hz

Slika 20. IEPE akcelerometar IMI Sensors 356B21

6.3. National Instruments NI 9191+NI9234+LabView2014

Karakteristike sustava za prikupljanje podataka (Slika 21):

- 4 kanalni analogni (+/- 5V) sustav
- 51,2 kS/s/kanalu
- LAN
- WLAN
- Prikupljanje podataka kontrolirano programiranim sustavom unutar okruženja LabView2014

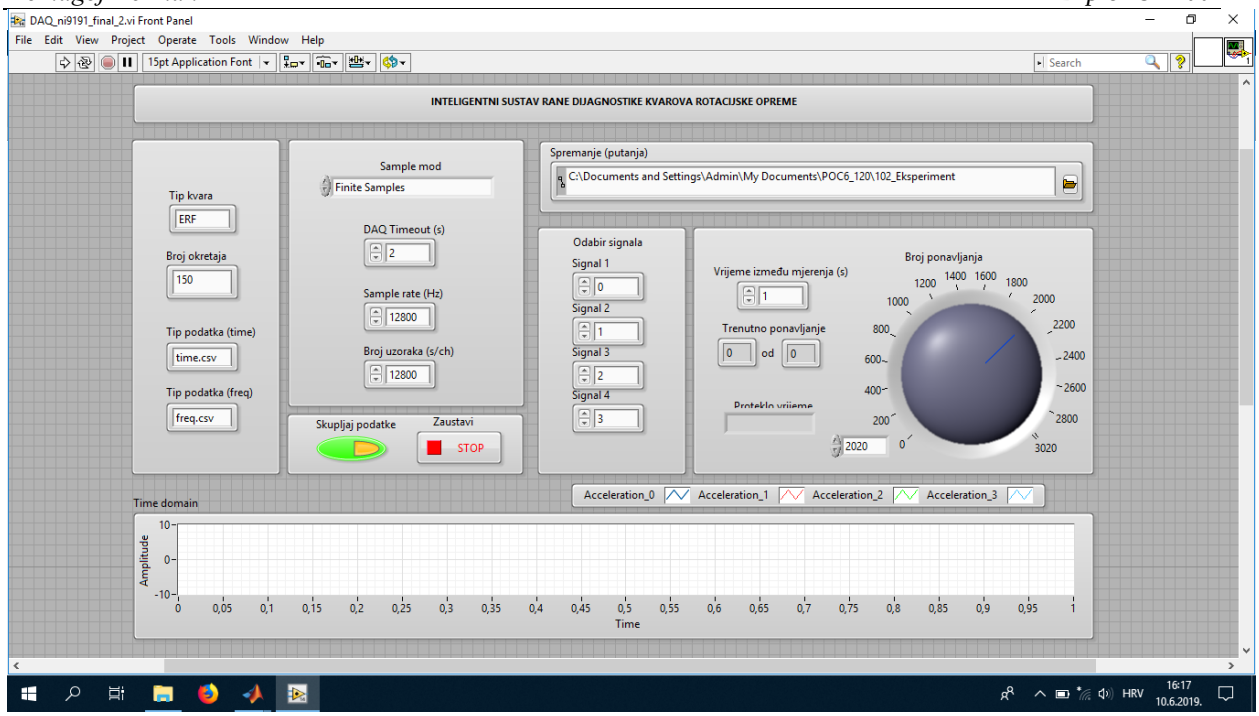
Slika 21. National Instruments NI 9191+NI9234

Prikupljanje podataka (engl. *data acquisition*, DAQ) sa senzora je izvršeno pomoću modula za prikupljanje podataka NI 9191+NI9234 (Slika 21) napravljenog od strane tvrtke National Instruments. Modul prima signale s troosnog senzora preko 4 ulazna BNC konektora. Modul kontrolira vrijeme, sinkronizaciju i prijenos podataka od senzora do računala. Na računalu se rezultati očitani senzorom prikazuju pomoću softverskog paketa LabVIEW.

LabVIEW, skraćeno od „Laboratory Virtual Instrument Engineering Workbench“, je platforma, mjerni softverski paket i razvojno okruženje za grafičko programiranje razvijen od tvrtke National Instruments (NI). Programiranje se izvodi spajajući funkcionalne blokove vodovima kroz koje prolaze podaci te se po tome razlikuje od suvremenog programiranja kao što su C, C++, ili Java, u kojem se programira unošenjem tekstualnih kodova. Program kreiran u LabVIEW-u sastoji se od jednog ili više „virtualnih instrumenata“ (VI). Izgledom i načinom rada oponašaju stvarne instrumente pa se zato tako nazivaju. Koristeći LabVIEW, korisnik stvara točno onu vrstu virtualnog instrumenta koji je potreban, a pritom je puno jeftiniji od standardnih instrumenata koji se koriste. Način na koji se izvršava program u softveru LabVIEW određen je strukturom blok dijagrama koji su linijama povezani. Svaki blok ima svoju funkciju i njihovim povezivanjem stvara se logika izvršavanja. VI se sastoji od dva glavna dijela: [24]

- Korisničkog sučelja (engl. *Front panel*)
- Blok dijagrama (engl. *Block diagram*)

Za potrebe ovog rada je korišteno korisničko sučelje „Inteligentni sustav rane dijagnostike kvarova rotacijske opreme“ (Slika 22) razvijeno za potrebe projekata unutar Laboratorija za održavanje. Na korisničkom sučelju se zadaju: vrijeme između mjerenja, broj ponavljanja mjerenja, stopa uzorkovanja, broj uzoraka, naziv mapa za spremanje, itd. Rezultati, odnosno vibracijski profili za svaku os mjerenja, se prikazuju uživo pomoću dijagrama koji pokazuju amplitudu u vremenskoj i frekvencijskoj domeni.



Slika 22. Prikaz korisničkog sučelja u LabVIEW-u

Parametri eksperimenta su:

- Frekvencija uzorkovanja F_s = 12800 Hz
- Nyquistova frekvencija F_{Ny} = 0.5 * F_s = 0.5*12800= 6400 Hz
- Vrijeme uzorkovanja T_s = 1s
- Broj uzorkovanja u svakom mjerenju N_s = F_{Ny} * T_s = 12800
- Razmak između mjerenja: 2s
- Ukupan broj mjerenja za pojedini set ulaznih parametara: 1500

Dobiveni izlazni podaci sa troosnog akcelerometra spremaju se na računalo u .csv (engl. *comma separated value*) obliku u vremenskoj i frekvencijskoj domeni. Naziv spremljene datoteke sastoji se od:

a) Podatak o tipu kvara
   - Normal NS
   - Debalans IMRF
   - Ekscentar ERF
   - Cocked CRF
   - Ležaj ORBF
   - Ležaj IRBF
   - Ležaj BBF

- Ležaj CBF

b) Podatak o broju okretaja
    - 150 o/min
    - 300 o/min

c) Podatak o vremenskoj ili frekvencijskoj domeni
    - Time
    - Freq

d) Podatak o mjernoj iteraciji

Zapis u spremljenoj datoteci vremenske domene sadrži sljedeća polja:

- Podatak u vremenu (Timestamp)
- Snaga vibracija (RMS) – X os
- Snaga vibracija (RMS) – Y os
- Snaga vibracija (RMS) – Z os
- Tip kvara, broj okretaja i broj mjerenja

Zapis u spremljenoj datoteci frekvencijske domene sadrži sljedeća polja:

- Trenutna frekvencija
- Snaga vibracija (RMS) – X os
- Snaga vibracija (RMS) – Y os
- Snaga vibracija (RMS) – Z os
- Tip kvara, broj okretaja, broj mjerenja

7. RAZVOJ RAČUNALNOG MODELA I REZULTATI

7.1. Programsko sučelje

Algoritam konvolucijske neuronske mreže je izrađen u programu MATLAB u kojeg se uvoze podaci sa senzora koji su prethodno prilagođeni za obradu u MATLAB-u.

Program MATLAB služi za rješavanje različitih matematičkih problema, te čitav niz izračunavanja i simulacija vezanih uz obradu signala, upravljanje, regulaciju i identifikaciju sustava. Prva verzija MATLAB-a, jednostavni matrični laboratorij (Matrix Laboratory), napisana je krajem 1970. godine na sveučilištima University of New Mexico i Stanford University s ciljem primjene u matričnoj teoriji, linearnoj algebri i numeričkoj analizi. Svi podaci u MATLAB-u tretiraju se kao matrice čije dimenzije nije potrebno čuvati kao posebne varijable. Danas svojstva MATLAB-a daleko prelaze originalni "matrični laboratorij". Radi se o interaktivnom sustavu i programskom jeziku za opća tehnička i znanstvena izračunavanja. MATLAB je također zamišljen kao sustav u kojem korisnik na jednostavan način može graditi svoje vlastite alate i biblioteke te modificirati postojeće. U tu svrhu se koristi jednostavni programski jezik. [25]

U MATLAB-u je napisan kod za učenje, validaciju i testiranje konvolucijske neuronske mreže. Zadatak konvolucijske neuronske mreže je samostalno učenje značajki vibracijskih signala prikupljenih na simulatoru kvarova. Mreža će učiti na podacima u vremenskoj domeni. Naučena mreža će nakon učenja napraviti predikciju preostalih podataka koji nisu korišteni za učenje. Za svaki zapis će mreža predvidjeti kojem od 8 stanja (IMRF, ERF, CRF, ORBF, IRBF, BBF, CBF) taj zapis pripada. Svaki je podatak, ovisno o svom stanju, spremljen u jednu od 8 različitih mapa. Mape su grupirane u veće mape koje se odnose na brzine vrtnje (150 i 300). Algoritam se poziva na mapu brzine vrtnje, koristeći 8 mapa unutar nje kao labele pri učenju mreže.

7.2. Konvolucijska neuronska mreža i odabir hiperparametara

Nakon što je u poglavlju 6 opisano prikupljanje 12000 ulaznih podataka za svaku brzinu vrtnje, ti se podaci postavljaju kao ulazi u konvolucijsku neuronsku mrežu u programu MATLAB. Omjer podataka koji će pripadati trening, validacijskom i test skupu vidljiv je u tablici 8.

Tablica 8. Količina i omjer podataka za trening, validaciju i test

Količina podataka za jednu brzinu vrtnje	Broj podataka	trening 80%	validacija 5%	test 15%
BBF	1500	1200	75	225
CBF	1500	1200	75	225
CRF	1500	1200	75	225
ERF	1500	1200	75	225
IMRF	1500	1200	75	225
IRBF	1500	1200	75	225
NS	1500	1200	75	225
ORBF	1500	1200	75	225
UKUPNO	12000	9600	600	1800

U ovom se radu promjenom određenih hiperparametara konvolucijske neuronske mreže nastoji odrediti koliki utjecaj imaju pojedini hiperparametri na točnost i vrijeme učenja. Hiperparametri će se ispitati metodom mrežnog optimiziranja hiperparametara navedenom u radu u poglavlju 5. Hiperparametri koji se mijenjaju su:

- Veličina jezgri u prva dva konvolucijska sloja
- Broj jezgri
- Broj slojeva konvolucijske neuronske mreže

U kreiranoj konvolucijskoj mreži se broj i veličina jezgri po slojevima računaju ovisno o iznosu faktora k, kako je opisano u tablici 9.

Tablica 9. Veličine i broj jezgri po slojevima s obzirom na faktor k

Faktor k= [2 4 8 16]						
k= [2 4 8 16]	1. sloj	2. sloj	3. sloj	4. sloj	5. sloj	6. sloj
broj jezgri	k	4k	4k	4k	4k	4k
veličina jezgre	[2k 1]	[k/2 1]	[4 1]	[4 1]	[4 1]	[4 1]

k=2	1. sloj	2. sloj	3. sloj	4. sloj	5. sloj	6. sloj
broj jezgri	2	8	8	8	8	8
veličina jezgre	[4 1]	[1 1]	[4 1]	[4 1]	[4 1]	[4 1]
k=4	1. sloj	2. sloj	3. sloj	4. sloj	5. sloj	6. sloj
broj jezgri	4	16	16	16	16	16
veličina jezgre	[8 1]	[2 1]	[4 1]	[4 1]	[4 1]	[4 1]
k=8	1. sloj	2. sloj	3. sloj	4. sloj	5. sloj	6. sloj
broj jezgri	8	32	32	32	32	32
veličina jezgre	[16 1]	[4 1]	[4 1]	[4 1]	[4 1]	[4 1]
k=16	1. sloj	2. sloj	3. sloj	4. sloj	5. sloj	6. sloj
broj jezgri	16	62	62	62	62	62
veličina jezgre	[32 1]	[8 1]	[4 1]	[4 1]	[4 1]	[4 1]

U tablici je vidljivo da kako raste k, tako raste i broj jezgri u svakom konvolucijskom sloju, ali i njihova veličina u prva dva sloja. Veličina jezgri se ne mijenja u zadnja 4 sloja, što je označeno žutom bojom. U radu su ispitane neuronske mreže s jednim do šest konvolucijskih slojeva. Postupak je tekao tako da je prvo naučena mreža sa svih 6 slojeva, zatim je uklonjen 6. sloj kako bi se učila mreža s 5 slojeva. Nakon toga je uklonjen 5. sloj kako bi se učila mreža s 4 sloja. Zadnja mreža koja je naučena je mreža sa samo jednim, prvim konvolucijskim slojem. Svaka od tih 6 neuronskih mreža je trenirana s 4 vrijednosti k (2, 4, 8, 16), što dovodi do broja od 24 neuronske mreže za jednu brzinu vrtnje. Svojevoljno je izabran broj od 3 ponavljanja učenja svake mreže, kako bi se dobila što manja odstupanja. Zatim je izračunat prosjek točnosti te 3 mreže, što znači da su za jednu brzinu vrtnje naučene 72 konvolucijske neuronske mreže. Kako je u radu cilj ispitivanje rezultata kod dviju brzina vrtnje (150 o/min i 300 o/min), to

dovodi do konačnog broja od 144 naučene konvolucijske neuronske mreže. Cjelokupan proces učenja je trajao 82 sata i 19 minuta na računalnoj infrastrukturi s grafičkim procesorom GeForce 740M s 2 GB radne memorije te 384 CUDA jezgre s taktom od 980 MHz.

Ostali hiperparametri su konstantni i posjeduju vrijednosti navedene u tablici 10.

Tablica 10. Hiperparametri konvolucijske neuronske mreže

HIPERPARAMETAR	IZNOS
Stopa učenja, η	0.009, periodički opada tijekom treninga
Broj epoha	15
Veličina mini grupe podataka	64
Optimizacijska metoda	Stohastički gradijentni spust
Funkcija gubitka	Softmax
Dopunjavanje	DA
Korak s kod konvolucije	1
Korak s kod sažimanja	2
Tip sažimanja	Sažimanje maksimumom
Aktivacijska funkcija	ReLU
Početna stopa učenja	0.009
Momentum	0.8
Drop Period	10, broj epoha nakon kojih se ažurira stopa učenja
Drop Factor	0.1, faktor kojim se stopa učenja množi tijekom učenja mreže
Učestalost validacije	10, svakih 10 iteracija se provjerava rezultat na validacijskom skupu podataka
Regularizacija	L2
Faktor regularizacije	0.0001

Normalizacija mini grupe podataka	DA

Izbor i iznos hiperparametara se navodi u dijelu koda u kojem se opisuju opcije učenja mreže (Slika 23). Ostali hiperparametri, kao veličina i broj jezgre ili broj slojeva i dr., se navode u području u kojem se opisuje struktura slojeva.

```
76
77 - options = trainingOptions('sgdm','MaxEpochs',15, ...
78       'Momentum',0.8,...
79       'InitialLearnRate', 0.009, ...
80       'LearnRateSchedule', 'piecewise', ...
81       'LearnRateDropFactor', 0.1, ...
82       'LearnRateDropPeriod', 10, ...
83       'MiniBatchSize', 64,...
84       'ValidationData',validData2, ...
85       'ValidationPatience',Inf, ...
86       'ValidationFrequency',10);
```

Slika 23. Dio koda u kojem se zadaju hiperparametri mreže

Pošto neuronska mreža sa 6 slojeva posjeduje identična zadnja 4 sloja, u tablici 11 će se prikaz organizacije slojeva i iznosi njihovih parametara opisati strukturom troslojne neuronske mreže za k=2.

Tablica 11. Struktura i parametri troslojne neuronske mreže (k=2)

SLOJ	PARAMETRI I VELIČINA SLOJA
Ulazni sloj	[12800 x 1 x 3]
1. Konvolucijski sloj	2 jezgre veličine: [4 x 1 x 3] , Korak s=[1 1], Veličina sloja: [12800 x 1 x 2]
Batch normalizacija 1	Normalizacija
Aktivacijski sloj 1	ReLU
Sloj sažimanja 1	Sažimanje maksimumom: [2 1] Korak s=[2 2] Veličina sloja: [6400 x 1 x 2]

2. Konvolucijski sloj	8 jezgri veličine: [1 x 1 x 2] , Korak s=[1 1], Veličina sloja: [6400 x 1 x 8]
Batch normalizacija 2	Normalizacija
Aktivacijski sloj 2	ReLU
Sloj sažimanja 2	Sažimanje maksimumom: [2 1] Korak s=[2 2] Veličina sloja: [3200 x 1 x 8]
3. Konvolucijski sloj	8 jezgri veličine: [4 x 1 x 8] , Korak s=[1 1], Veličina sloja: [3200 x 1 x 8]
Batch normalizacija 3	Normalizacija
Aktivacijski sloj 3	ReLU
Sloj sažimanja 3	Sažimanje maksimumom: [2 1] Korak s=[2 2] Veličina sloja: [1600 x 1 x 8]
Potpuno povezani sloj	8 neurona
Izlazni sloj	klasifikacija

Strukturu neuronske mreže u tablici 11 sačinjavaju 3 skupa slojeva s potpuno povezanim slojem i izlaznim slojem na kraju mreže. Ulazni sloj je dimenzija [12800 x 1 x 3] jer je akcelerometar bilježio 12800 mjerenja u sekundi što sačinjava jedan zapis. Dubina ulaznog sloja iznosi 3 jer je akcelerometar mjerio vibracije u 3 osi (x, y i z os). Prvi skup slojeva započinje konvolucijskim slojem s 2 jezgre veličine [4 x 1 x 3]. Uslijed koraka $s = 1$ dobiva se mapa značajki dimenzija [12800 x 1 x 2]. Zatim slijedi sloj batch normalizacije. Taj sloj se koristi kad se želi ubrzati učenje neuronske mreže i smanjiti osjetljivost na inicijalizaciju mreže. Normalizira svaki ulazni kanal u mini-grupama oduzimanjem od svake vrijednosti prosjek mini-grupe i diljenjem sa standardnom devijacijom te mini-grupe. Nakon toga slijedi

aktivacijski sloj s ReLU funkcijom. Prvi skup slojeva završava slojem sažimanja u kojem prozor koji sažima maksimumom iznosi [2 1], a korak *s* iznosi [2 2]. Veličina mape značajki nakon prvog sloja sažimanja je [6400 x 1 x 2]. Nakon toga slijedi drugi i treći skup slojeva. Prvi konvolucijski slojevi uče jednostavnije značajke, dok su oni kasniji sposobni naučiti kompleksnije. Nakon 3. sloja sažimanja u mreži se javlja potpuno povezani sloj sa 8 izlaza. U izlaznom sloju se obavlja klasifikacija pomoću softmax funkcije.

Na slici 24 je vidljiv kod mreže iz tablice 11 napisan u MATLAB-u.

```
layers = [ ...
    imageInputLayer([12800 1 3], 'Name','input')
    convolution2dLayer(kernel_1_size,number_of_kernels_1, 'Name','conv_1','Padding','same')
    batchNormalizationLayer('Name','batch_1')
    reluLayer('Name','relu_1')
    maxPooling2dLayer([2 1],'Stride',2, 'Name','maxPool_1')
    convolution2dLayer(kernel_2_size,number_of_kernels_2,'Name','conv_2','Padding','same')
    batchNormalizationLayer('Name','batch_2')
    reluLayer('Name','relu_2')
    maxPooling2dLayer([2 1],'Stride',2,'Name','maxPool_2')
    convolution2dLayer([4 1],number_of_kernels_2,'Name','conv_3','Padding','same')
    batchNormalizationLayer('Name','batch_3')
    reluLayer('Name','relu_3')
    maxPooling2dLayer([2 1],'Stride',2,'Name','maxPool_3')


    fullyConnectedLayer(8, 'Name','fc')
    softmaxLayer('Name','softMax')
    classificationLayer('Name','classOutput') ];
```

Slika 24. Prikaz dijela koda u MATLAB-u koji definira slojeve mreže

7.3. Rezultati konvolucijske neuronske mreže

Nakon što su u MATLAB-u naučene 144 neuronske mreže, napravljena je predikcija tih mreža na 15% podataka (testni skup) koji su do tog trenutka bili neiskorišteni. Izračunata je točnost (engl. *accuracy*) svake od tih mreža. Točnost neuronske mreže se računa po formuli:

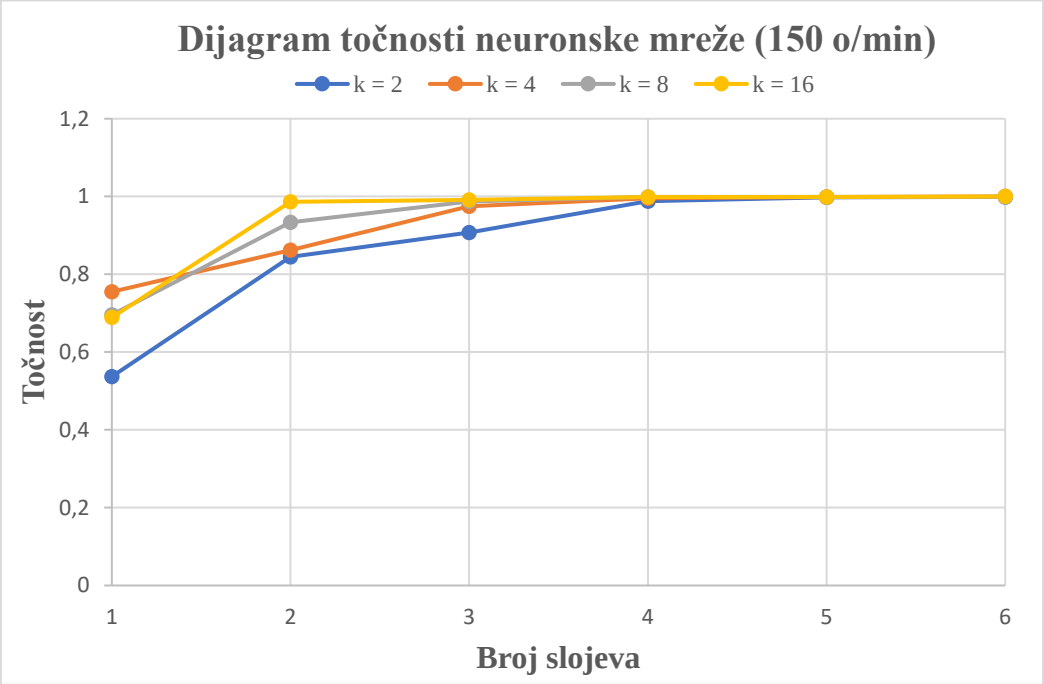
$$Točnost = \frac{Broj\ primjera - Broj\ netočnih\ predviđanja}{Broj\ primjera}$$

Pošto je svaka neuronska mreža s određenim setom hiperparametara učena 3 puta, u tablici 12 je izražen prosjek 3 vrijednosti točnosti koju je dala svaka od tih mreža za određeni broj slojeva i vrijednost faktora k.

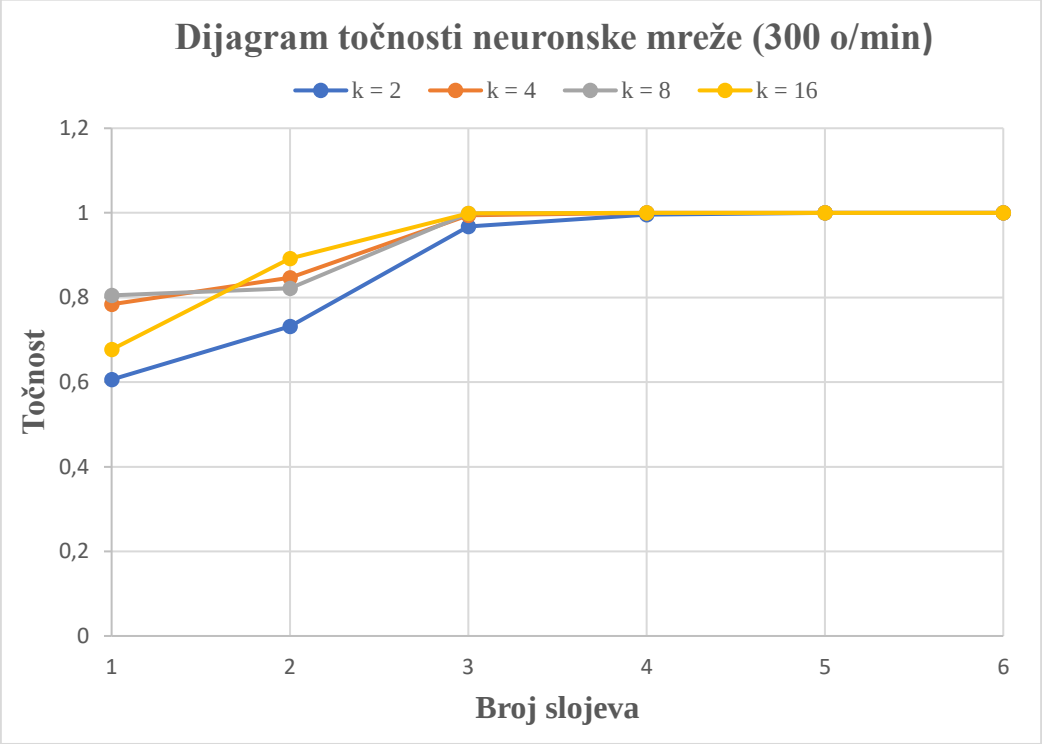
Tablica 12. Točnost neuronske mreže

150 o/min		Faktor k / Točnost				300 o/min		Faktor k / Točnost			
		2	4	8	16			2	4	8	16
Broj konvolucijskih slojeva	1	0,537	0,755	0,696	0,689	Broj konvolucijskih slojeva	1	0,606	0,784	0,805	0,677
	2	0,845	0,862	0,934	0,986		2	0,732	0,847	0,822	0,892
	3	0,907	0,975	0,987	0,991		3	0,968	0,994	0,999	0,999
	4	0,988	0,995	0,999	0,999		4	0,996	1,000	1,000	1,000
	5	0,998	0,999	0,999	0,999		5	1,000	1,000	1,000	1,000
	6	0,999	1,000	0,999	1,000		6	1,000	1,000	1,000	1,000

Tablica 12 je grafički prikazana dijagramima na slikama 25 i 26. Na njima je vidljivo da s porastom broja slojeva i broja i veličine jezgri raste i točnost svake mreže.



Slika 25. Dijagram točnosti neuronske mreže (150 o/min)



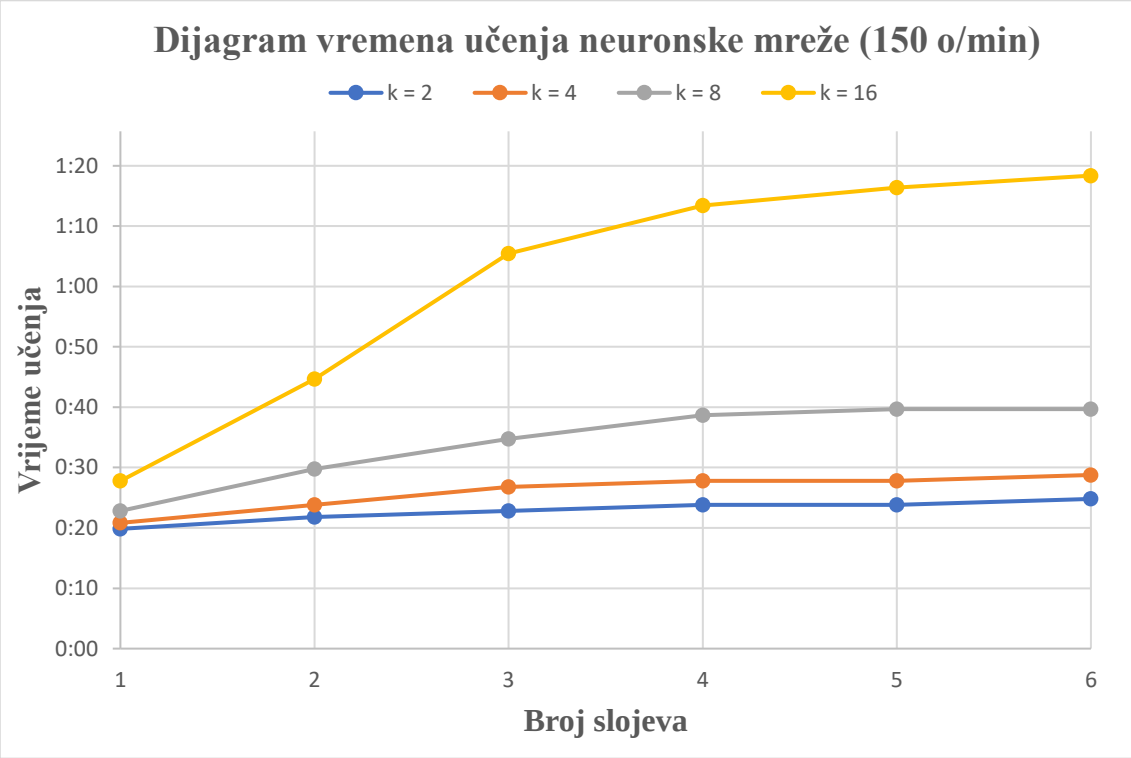
Slika 26. Dijagram točnosti neuronske mreže (300 o/min)

Može se zamijetiti kako bi većina mreža s 3 konvolucijska sloja u ovom slučaju dala iznimno dobre rezultate, a ako se nastoji postići apsolutna točnost na test skupu podataka onda bi se primjenile mreže s 5 ili 6 slojeva, ovisno o brzini vrtnje. No jedna od bitnijih stavki u svakom procesu je njegovo trajanje. U tablici 13 su vidljiva vremena potrebna za učenje i testiranje svake mreže na setu od 12000 podataka. U ovoj je tablici zamjetan trend porasta vremena učenja porastom broja slojeva i faktora k.

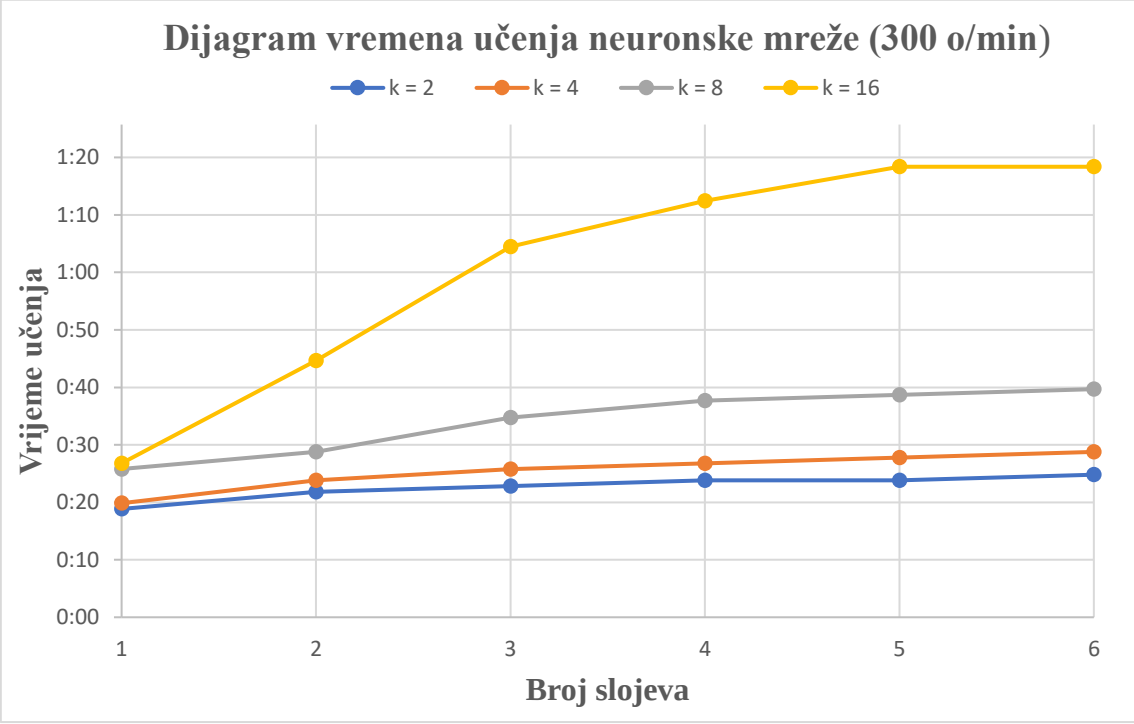
Tablica 13. Vrijeme učenja neuronske mreže

150 o/min		Faktor k / Vrijeme učenja				300 o/min		Faktor k / Vrijeme učenja			
		2	4	8	16			2	4	8	16
Broj konvolucijskih slojeva	1	0:20	0:21	0:23	0:28	Broj konvolucijskih slojeva	1	0:19	0:20	0:26	0:27
	2	0:22	0:24	0:30	0:45		2	0:22	0:24	0:29	0:45
	3	0:23	0:27	0:35	1:06		3	0:23	0:26	0:35	1:05
	4	0:24	0:28	0:39	1:14		4	0:24	0:27	0:38	1:13
	5	0:24	0:28	0:40	1:17		5	0:24	0:28	0:39	1:19
	6	0:25	0:29	0:40	1:19		6	0:25	0:29	0:40	1:19

Povećanjem veličine jezgre, broja jezgri i broja slojeva u mreži raste i broj parametara koje mreža mora naučiti te je rast vremena potrebnog za učenje mreže logičan. Na slici 27 i slici 28 grafički su prikazani podaci iz tablice 13 u obliku dijagrama.

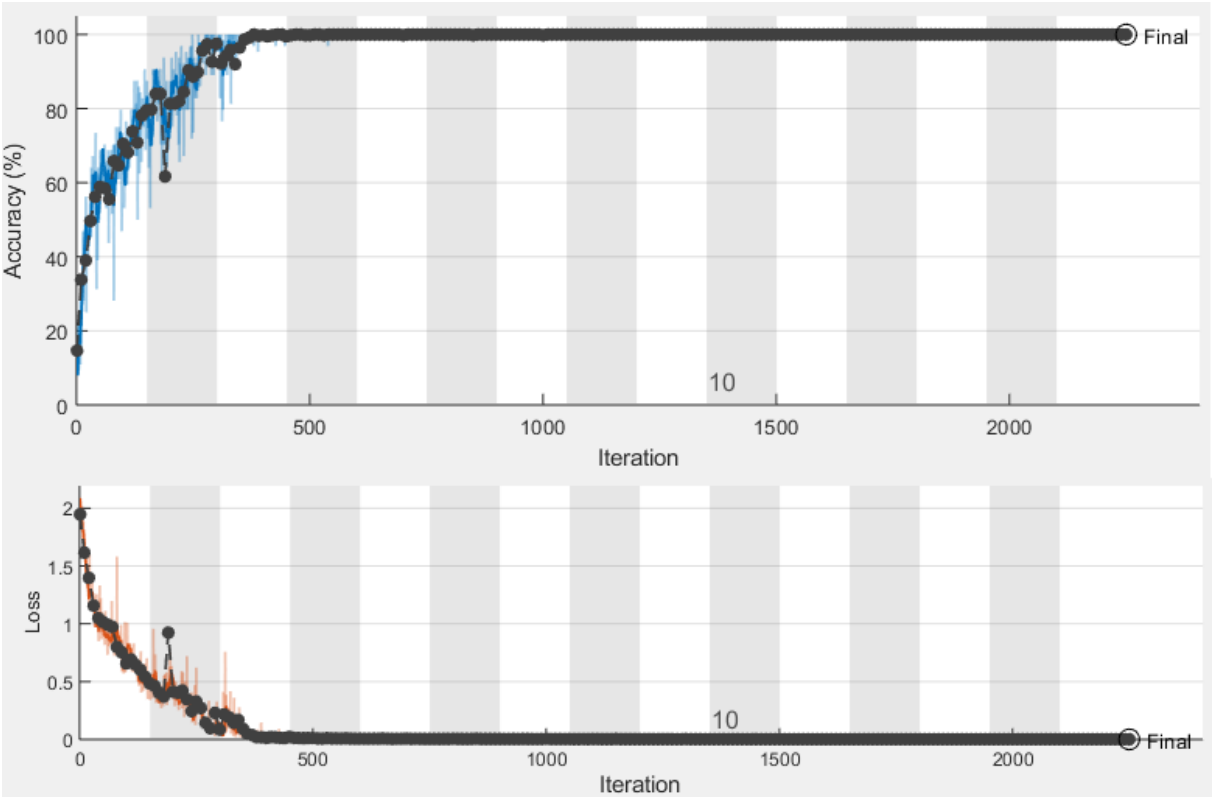


Slika 27. Vrijeme učenja neuronske mreže (150 o/min)



Slika 28. Vrijeme učenja neuronske mreže (300 o/min)

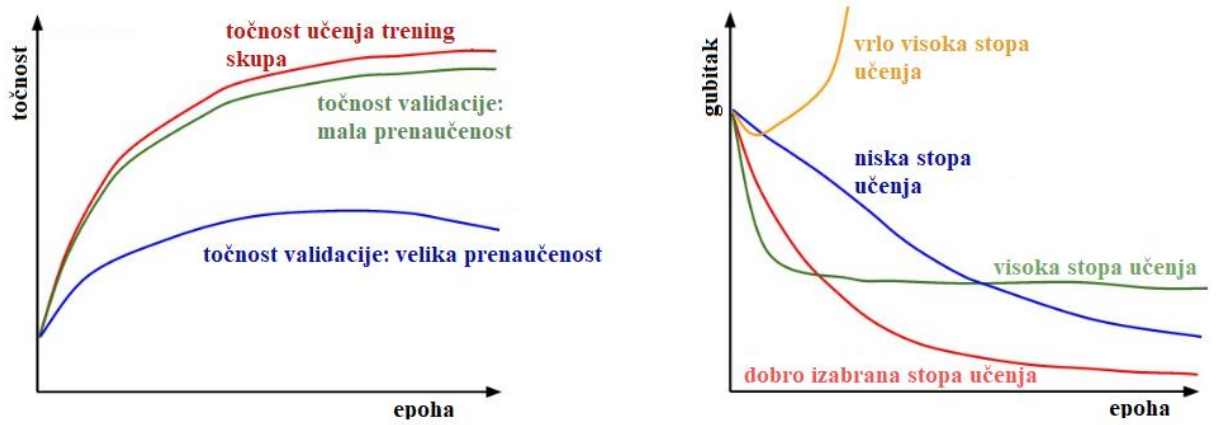
Zamjetna je velika razlika u vremenu potrebnom za učenje mreže k=16 i ostalih mreža. Na slici 29 je prikazan grafički prikaz kretanja točnosti i gubitka za mrežu učenu pri podacima prikupljenim na 300 o/min, s faktorom k=16 i 6 slojeva. Točnost te mreže iznosi 100 %.



Slika 29. Prikaz točnosti i gubitka mreže u 15 epoha

Vidljivo je kako su bile potrebne samo 3 epohe da mreža nauči klasificirati bez greške na trening skupu podatak. Stoga bi se u ovom slučaju broj epoha potrebnih za učenje mogao i smanjiti.

Slika 30 prikazuje kako se može na temelju oblika grafova točnosti i gubitka odrediti je li stopa učenja pre visoka ili pre niska te hoće li doći do prenaučenosti pri testiranju na test skupu podataka.



Slika 30. Različite varijante prikaza točnosti i gubitka [26]

Prikaz točnosti ukazuje na to hoće li doći do prenaučenosti prilikom testiranja na test skupu podataka. Na to ukazuje razmak između točnosti na trening skupu podataka i validacijskom skupu podataka. Ako je razmak velik onda mreža ne pogađa dobro na test skupu podataka jer je predobro naučila na trening skupu podataka.

Kod gubitka, graf prikaza niske stope učenja podsjeća na pravac, a ukoliko se odabere vrlo visoka stopa učenja onda neće doći do konvergencije. Visoke stope učenja će brže smanjiti gubitak na početku, ali se nakon toga njihov gubitak ne smanjuje.

Na temelju slike 29 se može zaključiti kako je stopa učenja dobro izabrana te da neće doći do prenaučenosti, što rezultati i potkrjepljuju.

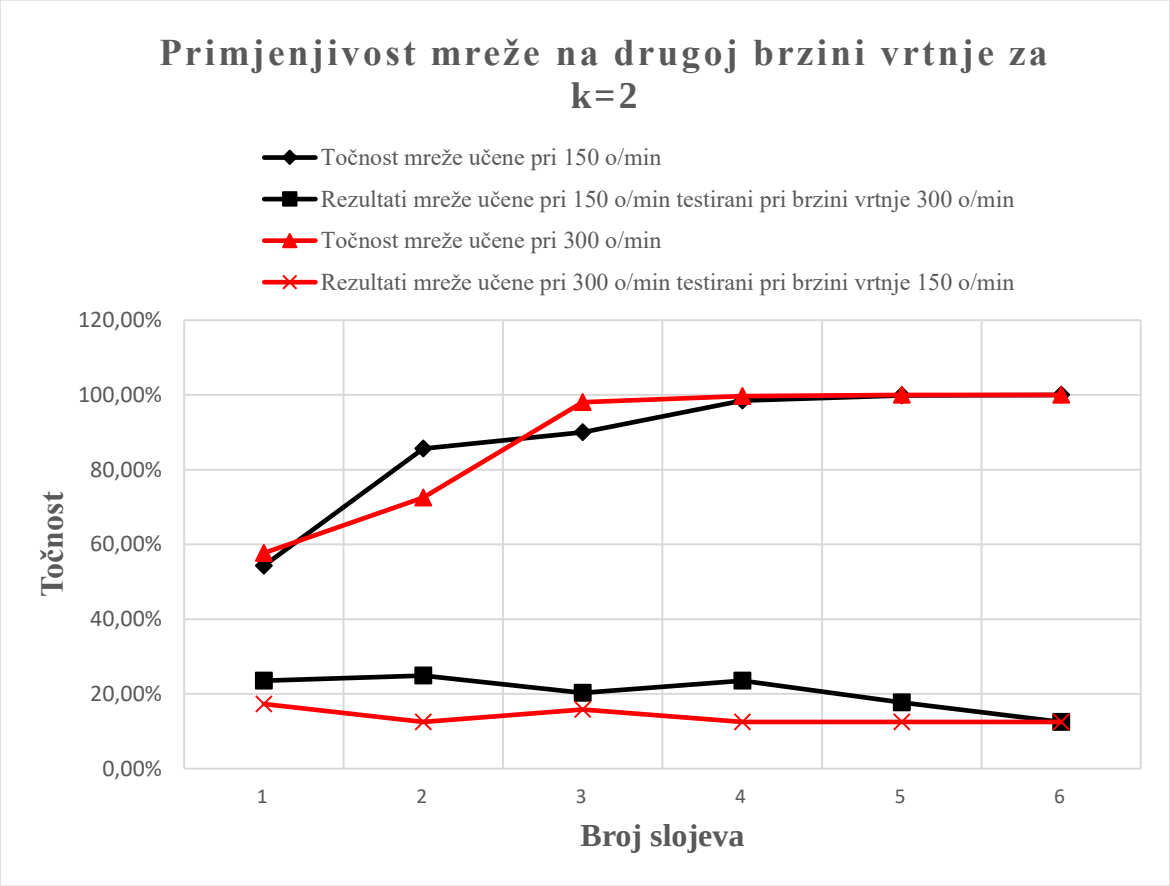
U proizvodnom prostrojenju uvjeti u kojima se izvodi mjerenje nisu uvijek konstantni u vremenu. Stavka koja se npr. može mijenjati u vremenu na stroju je brzina vrtnje. Zbog toga je provedeno ispitivanje primjenjivosti konvolucijske neuronske mreže koja je učena na jednoj brzini vrtnje na test podatke prikupljene pri drugoj brzini vrtnje. U ovom slučaju to znači da je uzeto nekoliko mreža naučenih na brzini vrtnje od 150 o/min i testirano na podacima snimljenim pri brzini vrtnje u iznosu od 300 o/min i obratno. Broj klasa koje se testiraju je i dalje jednak 8.

Točnost u tablici 14 i točnost u tablici 12 nisu identične pošto je u tablici 12 ispisan prosjek točnosti 3 naučene mreže, a u tablici 14 je uzeta druga mreža po točnosti za taj sloj. Ispitani su rezultati mreža s jednim do šest slojeva, pri faktoru k=2 i zapisani u tablici 14.

Tablica 14. Primjenjivost mreže na drugoj brzini vrtnje

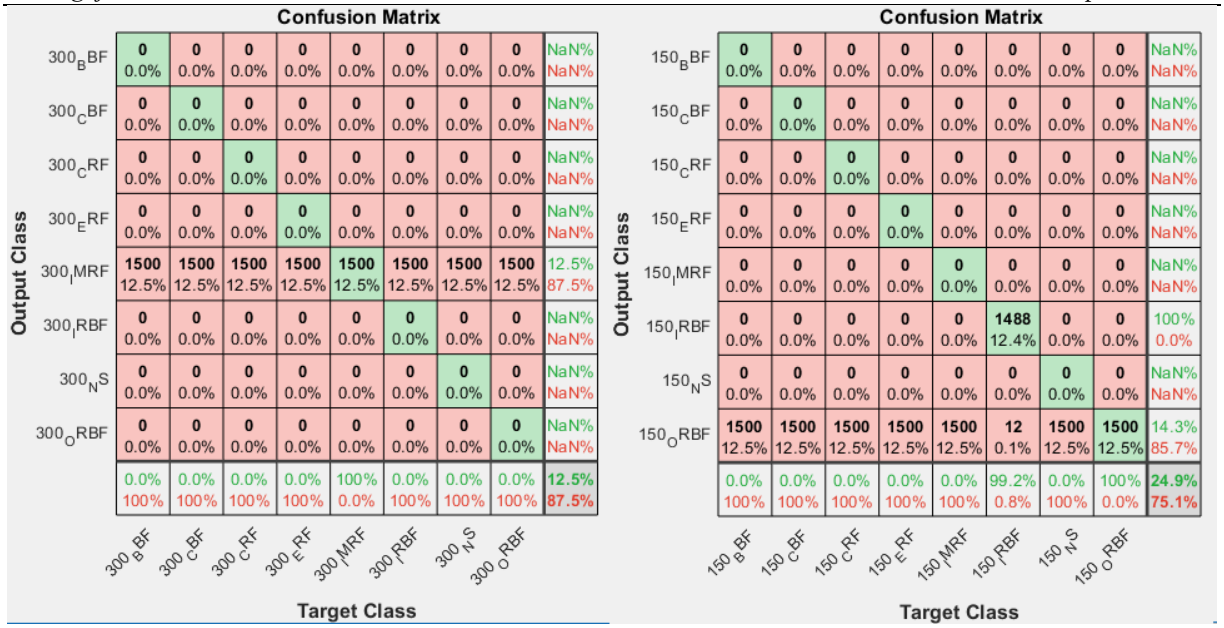
Broj konvolucijskih slojeva	Točnost mreže učene pri 150 o/min	Rezultati mreže (150 o/min) testirani pri brzini vrtnje 300 o/min	Točnost mreže učene pri 300 o/min	Rezultati mreže (300 o/min) testirani pri brzini vrtnje 150 o/min
1	54,3 %	23,5 %	57,7 %	17,3 %
2	85,67 %	24,9 %	72,5 %	12,5 %
3	90 %	20,3 %	98,1 %	15,8 %
4	98,5 %	23,5 %	99,7 %	12,5 %
5	99,83 %	17,7 %	100 %	12,5 %
6	100 %	12,5 %	100 %	12,5 %

Iz tablice 14 je vidljivo kako nije moguće implementirati konvolucijsku neuronsku mrežu naučenu pri jednoj brzini vrtnje na podatke prikupljene pri drugoj brzini vrtnje. Na slici 31 su grafički prikazani rezultati iz tablice 14.



Slika 31. Primjenjivost mreže na drugoj brzini vrtnje

Detaljnija analiza rezultata iz tablice 14 se može vidjeti na slici 32 na kojoj su prikazane matrice zabune za rezultate najlošije testirane mreže (crveno) iz tablice 14 (točnost=12,5 %) i najbolje (zeleno) testirane mreže (točnost=24,9 %).



Slika 32. Prikaz matrica zabune

Iz matrice zabune za mrežu učenu pri 300 o/min (crveno) je vidljivo kako ta mreža sve uzorke klasificira pod IMRF klasu (debalans) te stoga točnost iznosi 1/8 ili 12,5 %. Iz matrice zabune za mrežu s 2 sloja (zeleno) učenu pri 150 o/min (točnost 24,9%) te testiranu na podacima pri 300 o/min je vidljivo da mreža sve uzorke klasificira pod ORBF (kvar vanjske staze kotrljanja), osim za IRBF (kvar unutarnje staze kotrljanja) za koji vjerojatnost da će biti ispravno pogođen kad se kvar dogodi iznosi 99,2 %. Iz ovoga se može zaključiti kako mreža učena pri jednoj brzini vrtnje ispravno klasificira samo podatke učene pri istoj brzini vrtnje.

8. ZAKLJUČAK

U radu je prikazan razvoj računalnog modela za procjenu kvarova rotacijske opreme pri malim brzinama vrtnje u iznosu od 150 i 300 o/min. Nastojalo se dokazati kako se konvolucijske neuronske mreže, koje se uglavnom koriste pri obradi slika, mogu koristiti i za obradu vibracijskih signala prikupljenih pomoću akcelerometra u postupcima održavanja po stanju. Na simulatoru kvarova je simulirano 8 različitih stanja rotacijske opreme od kojih je jedno normalno, a ostalih 7 stanja predstavlja različite kvarove na opremi. Akcelerometar je prikupljao 12800 podataka u sekundi u 3 osi, što je značilo da će svaka slika ili zapis biti vektor dimenzija [12800 x 1 x 3]. U MATLAB-u je izrađena konvolucijska neuronska mreža koja je učila, validirala i testirala ukupno 24 000 prikupljenih zapisa. Vršila se predikcija kojoj od 8 klasa pripada svaki zapis iz test skupa podataka.

Ispitivan je utjecaj različitih iznosa određenih hiperparametara (broj jezgri, veličina jezgri, broj slojeva) na točnost mreže. Za izbor hiperparametara je primijenjena optimizacijska metoda mrežnog pretraživanja. Konvolucijske neuronske mreže su učene i validirane na 85 % ukupnog skupa podataka te testirane na 15 %. Rezultati koji su dobiveni su vidljivi u tablici 12. Na temelju tih rezultata je donesen zaključak kako povećanje broja i veličine jezgri te broja slojeva u mreži doprinosi povećanju točnosti mreže, ali i vremenu potrebnom da se provede proces učenja. Na temelju napisanog u poglavlju 5. u kojem su predstavljene 4 metode optimizacije hiperparametara, kao moguće unaprjeđenje se preporučuje isprobavanje i usporedba primjene ostalih metoda optimizacije hiperparametara kao što su: nasumično pretraživanje, Bayesova optimizacija i genetički algoritam.

Također ispitana je mogućnost primjene konvolucijske neuronske mreže naučene na podacima prikupljenima pri jednoj brzini vrtnje za predviđanje klasa na podacima prikupljenima pri drugoj brzini vrtnje. Rezultati koji su dobiveni dovode do zaključka da takva primjena nije moguća. Stoga se postavlja pitanje primjenjivosti ove tehnike predviđanja rezultata u proizvodnim uvjetima u nekom poduzeću u slučaju da je dinamika sustava promjenjiva.

Prediktivna tehnika opisana u radu zahtijeva velike količine podataka prikupljene za normalno stanje, ali i za stanje kad je stroj u kvaru. Problem pri primjeni u realnim uvjetima je taj što bi bilo potrebno mnogo vremena za čekanje da se dogodi kvar kako bi se prikupili podaci koji bi ga u budućnosti signalizirali. Druga varijanta bi bila izoliranje stroja iz proizvodnog sustava te simuliranje svih vrsta kvarova čime bi se ostvarili gubici u proizvodnji uslijed izuzimanja stroja iz rada, a i postoji mogućnost oštećenja ostalih dijelova stroja. Uslijed činjenice da je zahtjevno

prikupiti podatke za različite vrste kvarova tehničkog sustava jer je stroj za vrijeme rada uglavnom u normalnom stanju dolazi se do zaključka da je implementacija modela nadziranog učenja teško primjenjiva u proizvodnji. Ipak, učenjem u simulacijskim uvjetima moguće je kreirati modele koji se lakše mogu implementirati u realnim uvjetima.

Na kraju, može se zaključiti kako je tehnika dubokog učenja korištenjem konvolucijskih neuronskih mreža primjenjiva u dijagnosticiranju i predviđanju kvarova u tehničkim sustavima kad postoji dovoljna količina podataka za svaku vrstu kvara, ali za donošenje konačnog suda o njenoj primjenjivosti u industriji bilo bi potrebno provesti ispitivanje u proizvodnim uvjetima.

U prvoj fazi, takvo ispitivanje može uključivati procjenu stanja opreme na temelju odstupanja od normalnog stanja tj. primjenu nenadziranog učenja za otkrivanje anomalija ili primjenu nadziranog učenja na temelju konvolucijske neuronske mreže sa 2 klase.

LITERATURA

[1] Kobbacy A.H. Prabhakar Murthy D.N. Complex System Maintenance Handbook. Springer-Verlag London Limited. 2008.

[2] Mobley R. Keith: „An introduction to predictive maintenance“

[3] Ahmad R, Kamaruddin S. An overview of time-based and condition-based maintenance in industrial application. Computers & Industrial Engineering 63 (2012) 135–149.

[4] D. Lisjak: „Bilješke s predavanja kolegija: Održavanje“. 2015./16. ak. Godina

[5] Doko M. Primjena metoda strojnog učenja u održavanju. Zagreb: Sveučilište u Zagrebu, Fakultet strojarstva i brodogradnje; 2018.

[6] D. Lisjak: „Bilješke s predavanja kolegija: Dijagnostika u Održavanju“. 2016./17. ak. Godina

[7] Kondić V, Horvat M, Maroević F. Primjena dijagnostike kao osnove održavanja po stanju na primjeru motora osobnog automobila. Tehnički glasnik 7, 2013.

[8] Jardine A. K. S., Lin D., Banjevic D. A review on machinery diagnostics and prognostics implementing condition-based maintenance. Mechanical Systems and Signal Processing 20 (2006) 1483–1510

[9] http://cs231n.github.io/neural-networks-1/ [pristupljeno 15.6.2019.]

[10] https://www.mathworks.com/content/dam/mathworks/tag-team/Objects/d/80879v00_Deep_Learning_ebook.pdf [pristupljeno 15.2.2019.]

[11] Michelucci, U. Applied Deep Learning: A Case-Based Approach to Understanding Deep Neural Networks. Dübendorf, Švicarska: Apress; 2018.

[12] MIT's introductory course on deep learning methods and applications. Dostupno na: http://introtodeeplearning.com/materials/2019_6S191_L1.pdf [pristupljeno 1.3.2019.]

[13] Goodfellow I. Bengio Y. Courville A. Deep Learning. MIT Press. 2016. https://www.deeplearningbook.org/

[14] Krizhevsky A. Sutskever I. Hinton G.E. ImageNet Classification with Deep Convolutional Neuraln Networks. Advances in neural information processing systems. Siječanj 2012.

[15] http://cs231n.stanford.edu/slides/2019/cs231n_2019_lecture01.pdf [pristupljeno 12.6.2019.]

[16] Mitchell M. Machine Learning, 1998. Izdavač: McGraw-Hill Science/Engineering/Math; 1997.

[17] Ćorić I. Metode strojnog učenja u predviđanju profitabilnosti kupaca [doktorski rad]. Mostar: Sveučilište u Mostaru, Ekonomski fakultet; 2017. Sveučilište u Splitu: Ekonomski fakultet 2017.

[18] Hurwitz J, Kirsch D. Machine learning for dummies. St. Hoboken: John Wiley & Sons, Inc.; 2018.

[19] https://deeplearning.mit.edu/ [pristupljeno 15.2.2019.]

[20] Aggarwal, Charu C. Neural Networks and Deep Learning: A Textbook. Izdavač: Springer International Publishing AG, part of Springer Nature; 2018.

[21] Kim, P.: „MATLAB Deep Learning - With Machine Learning, Neural Networks and Artificial Intelligence“, 2017

[22] Smolčić D. Raspoznavanje objekata konvolucijskim neuronskim mrežama. Zagreb: Sveučilište u Zagrebu, Fakultet elektrotehnike i računarstva. lipanj 2015.

[23] https://spectraquest.com/machinery-fault-simulator/details/mfs/ [pristupljeno 2.6.2019.]

[24] http://repozitorij.fsb.hr/9009/1/%C5%BDupan_2019_diplomski_finalno.pdf [pristupljeno 2.6.2019.]

[25] https://www.fer.unizg.hr/_download/repository/matlab_upute.pdf [pristupljeno 2.6.2019.]

[26] http://cs231n.github.io/neural-networks-3/ [pristupljeno 24.6.2019.]