Izrada hidrološkog modela rijeke Bednje baziranog na povratnoj neuronskoj mreži

Varjačić, Dora

SVEUČILIŠTE U ZAGREBU

GEOTEHNIČKI FAKULTET

DORA VARJAČIĆ

IZRADA HIDROLOŠKOG MODELA RIJEKE BEDNJE BAZIRANOG NA POVRATNOJ NEURONSKOJ MREŽI

DIPLOMSKI RAD

VARAŽDIN, 2023.

Sazivam članove ispitnog povjerenstva

za 25.09.2023. u 9 sa

Obranu ovog rada kandidat će vršiti i pred ispitnim povjerenstvom u Varaždinu

Varaždin, 11.09.2023.

Predsjednik
ispitnog povjerenstva:

Prof.dr.sc. Sanja Kovač

Članovi povjerenstva

1) Doc.dr.sc. Djevad Oskoruš
2) Dr.sc. Karlo Lebovar
3) Prof.dr.sc. Lanka Brodić
4) Doc.dr.sc. Jelena Loborec

SVEUČILIŠTE U ZAGREBU

GEOTEHNIČKI FAKULTET

DORA VARJAČIĆ

IZRADA HIDROLOŠKOG MODELA RIJEKE BEDNJE BAZIRANOG NA POVRATNOJ NEURONSKOJ MREŽI

DIPLOMSKI RAD

KANDIDAT:

Dora Varjačić

MENTORICA:

Doc. dr. sc. Dijana Oskoruš

NEPOSREDNI VODITELJ:

Dr. sc. Karlo Leskovar

VARAŽDIN, 2023.

IZJAVA O AKADEMSKOJ ČESTITOSTI

Izjavljujem i svojim potpisom potvrđujem da je diplomski rad pod naslovom:

Izrada hidrološkog modela rijeke Bednje baziranog na povratnoj neuronskoj mreži

rezultat mog vlastitog rada koji se temelji na istraživanjima te objavljenoj i citiranoj literaturi te je izrađen pod mentorstvom **doc.dr.sc. Dijane Oskoruš.**

Izjavljujem da nijedan dio rada nije napisan na nedozvoljen način, odnosno da je prepisan iz necitiranog rada te da nijedan dio rada ne krši bilo čija autorska prava. Izjavljujem također, da nijedan dio rada nije iskorišten za bilo koji drugi rad u bilo kojoj drugoj visokoškolskoj, znanstvenoj ili obrazovnoj ustanovi.

U Varaždinu, 31.08.2023.

Dora Varjačić

(Ime i prezime)

Dora Varjačić

(Vlastoručni potpis)

IZJAVA MENTORA O POSTOTKU SLIČNOSTI DIPLOMSKOG RADA S VEĆ OBJAVLJENIM RADOVIMA

Izjavljujem i svojim potpisom potvrđujem da je diplomski rad pod naslovom:

Izrada hidrološkog modela rijeke Bednje baziranog na povratnoj neuronskoj mreži

pregledan anti-plagijat programskim paketom PlagScan te da postotak sličnosti cjelovitog diplomskog rada, s već objavljenim radovima, ne prelazi 20%, kao i da pojedinačni postotak sličnosti diplomskog rada sa svakom literaturnom referencom pojedinačno ne prelazi 5%.

U Varaždinu, 31.08.2023.

Doc.dr.sc. Dijana Oskoruš

(Mentor)

(Vlastoručni potpis)

Prvenstveno zahvaljujem svojoj mentorici doc. dr. sc. Dijani Oskoruš što je potaknula moj interes za polje hidrologije te na strpljenju i podršci tijekom izrade diplomskog rada.

Također, velika zahvala neposrednom voditelju dr. sc. Karlu Leskovaru na prenesenom znanju, idejama i savjetima te što je uvijek bio na raspolaganju za sva moja pitanja.

Na kraju, zahvaljujem svojoj obitelji i prijateljima na podršci tijekom svih godina studiranja.

SAŽETAK

IME I PREZIME AUTORA: DORA VARJAČIĆ

NASLOV RADA: Izrada hidrološkog modela rijeke Bednje baziranog na povratnoj neuronskoj mreži

Hidrologija je znanost koja proučava distribuciju i kretanje vode na površini i ispod površine Zemlje te u njezinoj atmosferi. Informacije dobivene analizom ponašanja i kretanja vode kroz hidrološki ciklus od velike su važnosti u područjima kao što su upravljanje vodnim resursima, upravljanje rizicima od opasnosti povezanih s vodom, projektiranje infrastrukture, proizvodnja električne energije i zaštita okoliša.

Hidrološko modeliranje moćan je alat koji omogućava simulaciju i razumijevanje interakcija između različitih komponenti hidrološkog ciklusa, a odabir vrste modela ovisi o ciljevima istraživanja, dostupnim podacima, računalnim resursima i karakteristikama hidrološkog sustava koji se modelira.

U hidrološko modeliranje sve se više integrira umjetna inteligencija, kako bi se prevladala ograničenja klasičnih modela te poboljšala njihova točnost i mogućnosti predviđanja. Jedno od potpolja umjetne inteligencije su umjetne neuronske mreže, koje se koriste za određivanje nelinearnih odnosa i vremenskih ovisnosti u hidrološkim podacima.

U ovom radu korišten je programski jezik Python za razvoj i implementaciju hidrološkog modela baziranog na povratnoj neuronskoj mreži. Ova specifična arhitektura neuronske mreže odabrana je zbog svoje sposobnosti hvatanja uzoraka i vremenskih ovisnosti u nizovima podataka.

Izrađen je model sliva Bednje, koji ima bujični režim ovisan o količini oborina. Model predviđa protoke na hidrološkoj postaji Ludbreg, na temelju prethodnih podataka o oborinama i protocima s uzvodnih postaja.

KLJUČNE RIJEČI: hidrologija, sliv Bednje, hidrološko modeliranje, povratna neuronska mreža

ABSTRACT

NAME AND SURNAME of the AUTHOR: DORA VARJAČIĆ

TITLE: The development of a hydrological model of the Bednja River based on a recurrent neural network

Hydrology is a science that studies distribution and movement of water below and above the Earth's surface and in its atmosphere. The information obtained from the analysis of the behaviour and movement of water through the hydrological cycle is of great importance in areas such as water resources management, risk management of water-related hazards, infrastructure design, electricity generation and environmental protection.

Hydrological modelling is a powerful tool that enables simulation and understanding of interactions between different components of the hydrological cycle, and the choice of a model type depends on the research objectives, available data, computer resources and characteristics of the hydrological system that is being modelled.

Artificial intelligence has been increasingly integrated into hydrological modelling, in order to overcome the limitations of classic models and enhance their accuracy and predictive capabilities. One of the subfields of artificial intelligence are artificial neural networks, which are used to determine nonlinear relationships and time dependencies in hydrological data.

In this paper, the Python programming language was used for the development and implementation of a hydrological model based on a recurrent neural network. This specific neural network architecture was chosen for its ability to capture patterns and temporal dependencies in data sets.

A model of the Bednja basin was created, which has a flood regime dependent on the amount of precipitation. The model predicts flows at the Ludbreg hydrological station, based on previous data on precipitation and flows from upstream stations.

KEYWORDS: hydrology, hydrological modelling, Bednja basin, recurrent neural network

SADRŽAJ

1. UVOD

Hidrologija je znanost koja se bavi proučavanjem vode, njene pojave, vremenske i prostorne distribucije i svojstava na Zemlji. Uključuje promatranje, mjerenje i analizu različitih hidroloških procesa koji su dio hidrološkog ciklusa. Hidrološki ciklus je kontinuirani proces koji opisuje kretanje i kruženje vode na Zemlji. To uključuje vodu koja se nalazi na površini Zemlje, ispod površine Zemlje i u atmosferi. Voda u hidrološkom ciklusu mijenja svoja agregatna stanja te se može naći u plinovitom, tekućem i krutom stanju, prolazeći pri tom kroz brojne fizikalne procese, kao što su: precipitacija, evaporacija, transpiracija, kondenzacija, sublimacija, infiltracija itd. Hidrološki ciklus je od temeljne važnosti za održavanje ravnoteže vode na Zemlji, održavanje života i utjecaj na vremenske i klimatske obrasce, a glavni pokretač ciklusa je Sunce [1].

Hidrolozi prikupljaju podatke korištenjem terenskih mjerenja, nadzornih mreža i tehnika daljinskog opažanja kako bi razumjeli ponašanje vode u različitim okruženjima. Važnost hidrologije upravo je u razumijevanju i praćenju spomenutih hidroloških procesa koje nam zatim, kroz različite alate i uz prikupljene podatke, omogućava [2]:

- Upravljanje vodnim resursima
- Upravljanje poplavama, sušama i drugim opasnostima vezanim za vodu
- Upravljanje kvalitetom vode i kontrolu onečišćenja
- Zaštitu okoliša
- Projektiranje i upravljanje infrastruktirom
- Razumijevanje klimatskih promjena
- Urbano planiranje i razvoj

Hidrološko modeliranje jedan je od alata koji se koristi kako bi poboljšali svoje razumijevanje hidroloških procesa, istražili različite scenarije i napravili predviđanja o dostupnosti vode, rizicima od poplava i drugim povezanim pojavama. Konstruiranjem matematičkog prikaza hidrološkog sustava, moguće je simulirati i analizirati odgovor sustava na različite ulazne podatke, kao što su obrasci oborina ili promjene u korištenju zemljišta [3].

Postoje različite vrste hidroloških modela koji se koriste za simulaciju i predstavljanje različitih aspekata hidrološkog sustava. Ovi modeli mogu varirati od jednostavnih

empirijskih modela do složenijih fizički utemeljenih modela [4]. S obzirom da je u posljednjih nekoliko desetljeća upotreba umjetne inteligencije doživjela porast u velikom broju znanstvenih područja, njena upotreba proširena je i na područje hidrološkog modeliranja [5].

Umjetna inteligencija je široko polje računalne znanosti koje se fokusira na stvaranje inteligentnih sustava sposobnih za simulaciju inteligencije nalik ljudskoj. To je najširi pojam koji se koristi za klasifikaciju sustava koji oponašaju ljudsku inteligenciju, a koristi se za razvoj algoritama, tehnika i sustava koji omogućuju računalima obavljanje zadataka kao što su percepcija, rasuđivanje, učenje i donošenje odluka. Umjetna inteligencija može se implementirati različitim metodama, uključujući strojno učenje i neuronske mreže. Strojno učenje je potpolje umjetne inteligencije koje se usredotočuje na razvoj algoritama i modela koji omogućuju računalima da uče i donose predviđanja ili odluke bez eksplicitnog programiranja. Algoritmi strojnog učenja uče obrasce i odnose iz podataka i koriste se tim znanjem za informirana predviđanja ili poduzimanje radnji. Umjetne neuronske mreže potpolje su strojnog učenja i u središtu su algoritama dubokog učenja. Njihov naziv i struktura inspirirani su ljudskim mozgom, oponašajući način na koji biološki neuroni šalju signale jedni drugima [6].

Hidrološki modeli koji koriste algoritme umjetne inteligencije u određenim aspektima nadilaze ograničenja tradicionalnih modela te kao takvi mogu biti moćni alati za dobivanje informacija do kojih bi inače bilo vrlo komplicirano ili nemoguće doći. Neuronske mreže su se pokazale posebno učinkovite u hidrološkom modeliranju zbog svoje sposobnosti hvatanja složenih odnosa u podacima i rukovanja nelinearnostima [7].

U ovom radu opisan je općeniti princip rada umjetnih neuronskih mreža, posebno povratnih neuronskih mreža, te proces izrade i rezultati hidrološkog modela baziranog na povratnoj neuronskoj mreži, dizajniranoj u programskom jeziku Python. Izrađen je hidrološki model za područje sliva Bednje, koji na temelju vrijednosti oborina i protoka izmjerenih u određenom prošlom razdoblju predviđa buduće protoke na hidrološkoj postaji Ludbreg. Nakon pregleda i procjene učinkovitosti dobivenog modela, razmotrene su prednosti i nedostaci ovakvog pristupa hidrološkom modeliranju.

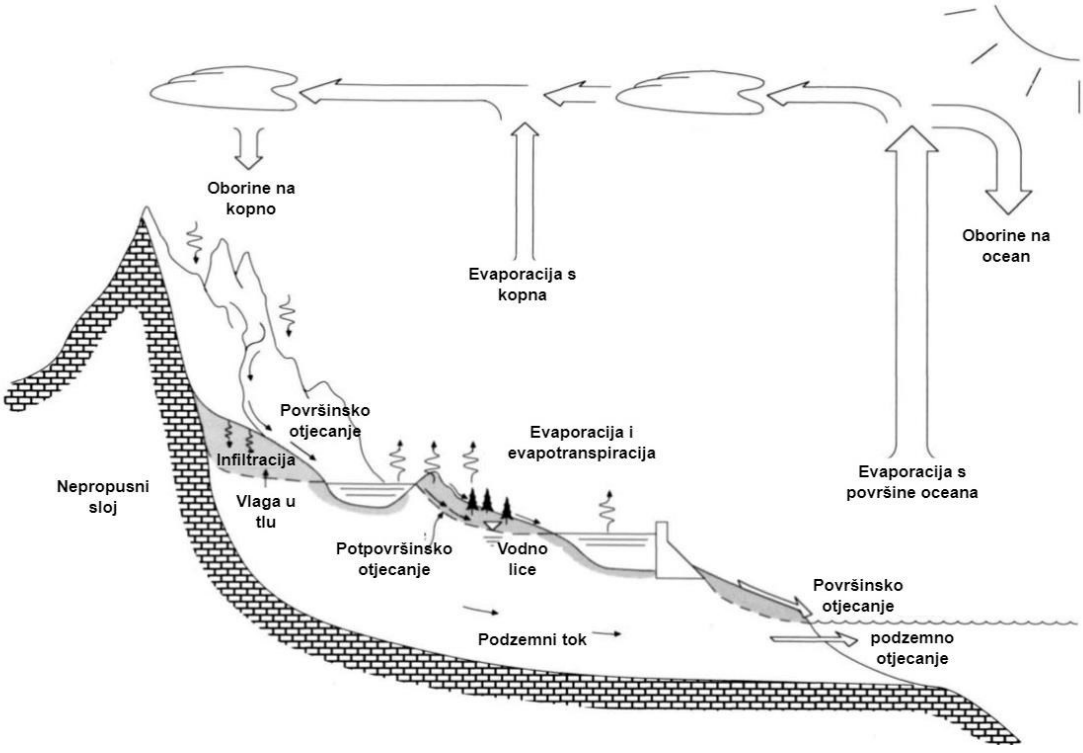
2. PREGLED DOSADAŠNJIH ISTRAŽIVANJA

2.1. Hidrologija i hidrološko modeliranje

2.1.1. Hidrologija i hidrološki ciklus

Chow i sur. u udžbeniku „Applied Hydrology“ navode kako je hidrologija, koja tretira sve faze vode na Zemlji, predmet od velike važnosti za ljude i njihov okoliš, a praktične primjene hidrologije nalaze se u zadacima kao što su: projektiranje i rad hidrotehničkih građevina, vodoopskrba, pročišćavanje i odvodnja otpadnih voda, navodnjavanje, odvodnja, proizvodnja hidroenergije, kontrola poplava, plovidba, kontrola erozije i taloženja sedimenta, kontrola saliniteta, smanjenje onečišćenja, rekreacijska uporaba vode i ribolov te zaštita divljih životinja [8].

Voda na Zemlji nalazi se na njezinoj površini, u litosferi i atmosferi. Sva ta voda povezana je procesima u hidrološkom ciklusu (Slika 1.), koji predstavlja središnji fokus hidrologije.



Slika 1. Hidrološki ciklus [8]

Hidrološki ciklus nema početak i kraj, već se sastoji od različitih procesa, koji se odvijaju simultano i kontinuirano: isparavanjem voda s površine Zemlje i s vodenih površina ulazi u atmosferu, u atmosferi se vodena para prenosi i diže sve dok, uslijed

hlađenja, ne dođe do kondenzacije i ponovnog prelaska na kopno ili vodenu površinu u obliku oborina. Oborina se može akumulirati u snježnom pokrivaču i ledenjacima, može postati dio površinskog toka ili se infiltrirati kroz tlo u podzemlje te kasnije izbijati na površinu kroz izvore te formirati površinsko otjecanje. Pod djelovanjem sunčeve energije i biljaka proces evapotranspiracije ponovo pokreće uvijek istu količinu vode koja cirkulira na Zemlji i njenoj atmosferi [1,8].

Razumijevanje hidrološkog ciklusa omogućava donošenje informiranih odluka i razvoj strategija za rješavanje sadašnjih i budućih izazova povezanih s vodom te je ključno za održivo upravljanje vodama, očuvanje okoliša, ublažavanje katastrofa i osiguranje dobrobiti ljudske populacije i ekosustava.

2.1.2. Hidrološko modeliranje

„Hidrološko modeliranje može se definirati kao karakterizacija stvarnih hidroloških značajki i sustava korištenjem fizičkih modela malih razmjera, matematičkih analoga i računalnih simulacija“ [9]

Hidrološko modeliranje je proces simulacije i predviđanja ponašanja vode u hidrološkim sustavima, kao što su rijeke, jezera, slivovi i podzemni tokovi, korištenjem matematičkih i računalnih tehnika, a uključuje opisivanje složenih interakcija između različitih hidroloških komponenti (oborine, isparavanje, infiltraciju, otjecanje, protok).

„Hidrološki model je, jednostavno rečeno, pojednostavljeni prikaz sustava iz stvarnog svijeta“ [10]

„Najbolji model je onaj koji daje rezultate bliske stvarnosti uz korištenje najmanje parametara i složenosti modela.“[4]

Rezultat hidrološkog modeliranja je hidrološki model. Prema tome, **hidrološki model** je matematički ili računalni prikaz hidrološkog sustava koji simulira kretanje i ponašanje vode u određenom području ili slivu. Važnost hidrološkog modeliranja leži u njegovoj sposobnosti da pruži uvid u hidrološke procese i ponašanje vodnih resursa i time podrži donošenje odluka u upravljanju i planiranju vezanom uz vodu.

Hidrološki modeli mogu varirati u svojoj namjeni i složenosti, a njihova klasifikacija nije egzaktna te u literaturi nailazimo na različite definicije. Shaw i sur. u „Hydrology in practice“, govoreći o modelima na razini sliva, navode podjelu hidroloških modela u dvije velike grupe [1]:

- **Deterministički modeli** – nastoje simulirati fizičke procese u slivu koji su uključeni u prijelaz oborine u površinski tok
- **Stohastički modeli** – opisuju hidrološke vremenske nizove nekoliko mjerenih varijabli (oborina, isparavanje, protok), uključujući distribucije vjerojatnosti,

,dok najuspješnijom metodom smatraju kombinaciju ova dva pristupa.

Prema Chow i sur. hidrološki modeli mogu se podijeliti na [8]:

- **Fizički modeli** – uključuju skalirane modele, koji predstavljaju sustav smanjen u mjerilu, te analogne modele, kod kojih se koristi drugi fizički sustav čija su svojstva slična promatranom sustavu
- **Apstraktni modeli** – predstavljaju sustav u matematičkom obliku, koristeći skup jednadžbi koje povezuju ulazne i izlazne varijable. Te varijable mogu biti funkcije prostora i vremena, ili slučajne varijable koje nemaju fiksnu vrijednost u određenoj točki u prostoru i vremenu (nego su opisane distribucijama vjerojatnosti). U tom slučaju ovi modeli mogu se dodatno podijeliti na:
    - *Deterministički modeli* – ne uzimaju u obzir slučajnost, pa određeni ulazni podatak uvijek daje isti izlaz (daju prognoze)
    - *Stohastički modeli* – imaju rezultate koji su barem djelomično slučajni (daju predviđanja)

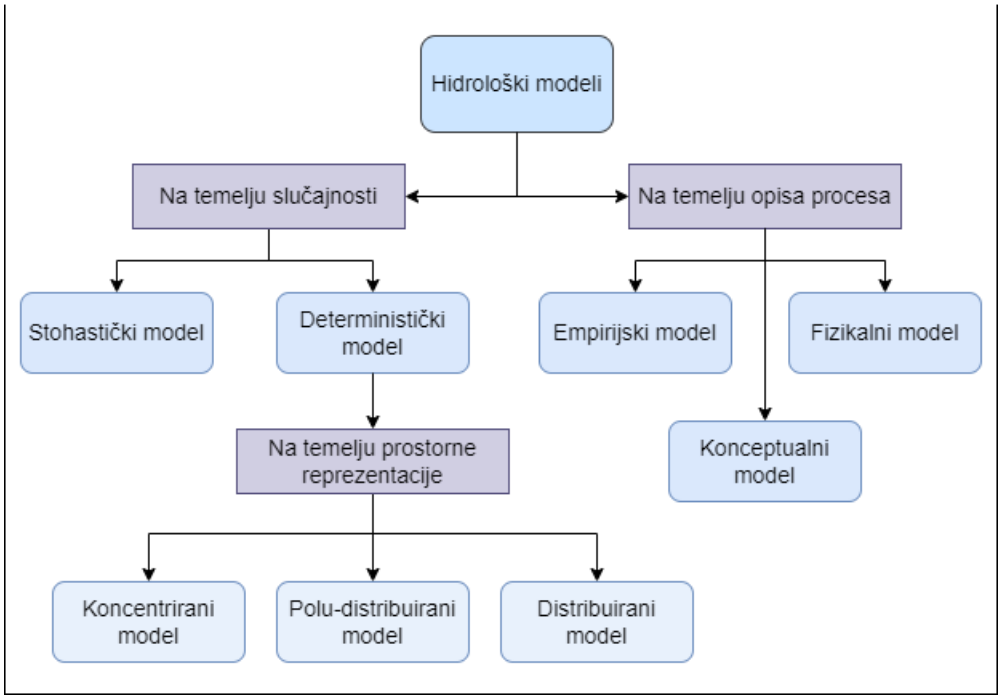
Jedna od najvažnijih klasifikacija hidroloških modela je na [4]:

- **Empirijski modeli** – temelje se na statističkim odnosima izvedenim iz postojećih mjerenih podataka, te ne predstavljaju eksplicitno fizičke procese koji se odvijaju unutar promatranog hidrološkog sustava
- **Konceptualni modeli** – predstavljaju hidrološki sustav koristeći pojednostavljene prikaze uključenih fizičkih procesa, a temelje se na konceptualnom razumijevanju kretanja vode kroz sliv. Promatrani sustav dijele na komponente (oborine, evapotranspiracija, infiltracija, površinsko otjecanje itd.), od kojih svaku predstavljaju matematičke jednadžbe
- **Fizikalni modeli** – predstavljaju matematički idealiziranu reprezentaciju hidroloških procesa u promatranom sustavu, temeljenu na zakonima fizike

Hidrološki modeli se dalje klasificiraju na [3]:

- **Koncentrirani (homogeni)** – sliv tretiraju kao jednu jedinicu, ne daju informacije o prostornoj distribuciji ulaza i izlaza već simuliraju samo bruto prostorno prosječan odziv sliva
- **Distribuirani** - predstavljaju sliv kao sustav međusobno povezanih podsustava – i vertikalno i horizontalno, pa se ovakvi modeli mogu smatrati skupom podslivova raspoređenih u nizu ili kao razgranata mreža

Na Slika 2. prikazana je detaljna podjela hidroloških modela prema [3]:



Slika 2. Klasifikacija hidroloških modela (prema [3])

Razvoj hidroloških modela oduvijek je bio potaknut potrebom za rješavanjem inženjerskih problema. Prvim priznatim hidrološkim modelom smatra se Racionalna metoda, čiji je koncept razvio Mulvaney 1850. godine, kao odgovor na potrebu za dizajniranjem sustava odvodnje, a koja je u upotrebi i danas [11,12].

Od razvoja prvog modela do danas hidrološki modeli značajno su napredovali, od jediničnog hidrograma, čiji je koncept prvi predložio Sherman 1932. godine, preko ulaz/izlaz (empirijskih) modela do današnjih DBM (Data Based Mechanistic) ili hibridnih modela te modela baziranih na umjetnoj inteligenciji. Taj napredak najvećim dijelom

možemo pripisati razvoju snage i mogućnosti računala te većoj dostupnosti prostorno – vremenskih podataka [11,12].

Neki od učestalo korištenih hidroloških modela danas su semi-distribuirani model TOPMODEL, fizikalni modeli kao što su MIKE SHE, DHSVM, KINEROS, CATFLOW, HEC-HMS i VELMA, te hibridni SWAT (Soil and Water Assessment Tool) [13,14].

U posljednjih nekoliko desetljeća razvijeno je i usavršeno mnogo hidroloških modela, a kako bi se odlučili za korištenje nekog od njih hidrolozi moraju dobro poznavati njihove karakteristike. Zbog prirode predviđanja okoliša, odnosno njegove heterogenosti, ne postoji univerzalno najbolji model. Za svako područje i namjenu modela, postoji više metoda modeliranja koje je moguće primijeniti. Odabir korištenog modela stoga ovisi o promatranom području, željenim rezultatima, potrebnoj složenosti modela, a možda i najviše o dostupnim podacima i ostalim resursima [15].

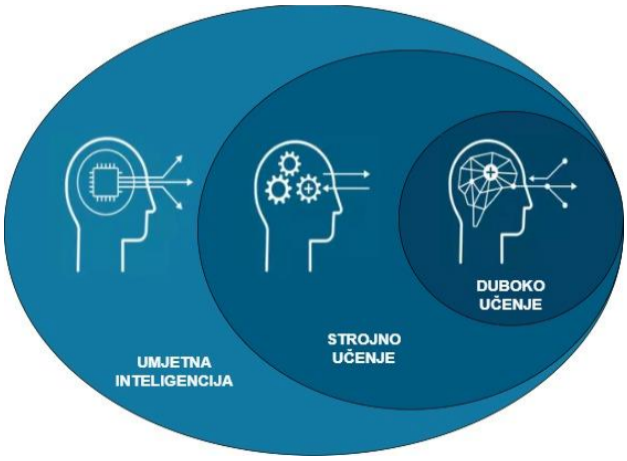
2.1.3. Upotreba umjetnih neuronskih mreža u hidrološkom modeliranju

Od sredine 20. stoljeća upotreba umjetne inteligencije porasla je u različitim inženjerskim područjima i granama znanosti, što je dovelo do promjena i napredaka gotovo u svakom polju, uključujući i hidrologiju. Primjena algoritama umjetne inteligencije sve je veća u hidrološkom modeliranju zbog njihove sposobnosti prevladavanje izazova koje predstavljaju velika količina podataka, visoka složenost, dinamika, nelinearnost i nestacionarnost uočene u hidrološkim procesima [5].

„*To (umjetna inteligencija) je znanost i inženjerstvo stvaranja inteligentnih strojeva, posebno inteligentnih računalnih programa. To je povezano sa sličnim zadatkom korištenja računala za razumijevanje ljudske inteligencije, ali AI se ne mora ograničiti na metode koje su biološki vidljive.*“ [16]

Umjetna inteligencija (eng. *Artificial Intelligence*/AI) odnosi se na razvoj računalnih sustava koji u kombinaciji s robusnim skupovima podataka omogućavaju rješavanje problema koji obično zahtijevaju ljudsku inteligenciju. AI obuhvaća različite tehnike i pristupe za oponašanje ljudskih kognitivnih sposobnosti i rješavanje složenih problema te ima brojne primjene u raznim industrijama i znanstvenim područjima, uključujući: automatsko prepoznavanje govora, računalni vid, primjenu u korisničkim službama, preporuke kod internetske kupovine i oglašavanja, pretraživanje interneta, poboljšanje ciber-sigurnosti, izgradnju pametnih gradova, automatizaciju i robotiku [6,17].

Uz pojam umjetne inteligencije, često se spominju strojno učenje (eng. *Machine Learning*) i duboko učenje (eng. *Deep Learning*), te se ovi pojmovi ponekad smatraju sinonimima, iako to nije u potpunosti točno. Najširi pojam je umjetna inteligencija, strojno učenje je njezino potpolje, dok je duboko učenje potpolje strojnog učenja (Slika 3.). Strojno učenje usmjereno je na razvoj algoritama koji mogu učiti iz podataka i donositi predviđanja ili odluke bez da su za to eksplicitno programirani. Duboko učenje odnosi se na razvoj algoritama koji uče na način inspiriran strukturom i funkcioniranjem ljudskog mozga – umjetnih neuronskih mreža [18].



Slika 3. Odnos umjetne inteligencije i strojnog i dubokog učenja

Umjetna neuronska mreža (eng. *Artificial Neural Network*/ANN) je računalni model inspiriran funkcioniranjem bioloških neuronskih mreža u ljudskom mozgu, koje se sastoje od milijardi međusobno povezanih neurona. Ovaka model svojevrstan je empirijski model i primjenjiv u hidrologiji kada imamo dovoljno veliku količinu dostupnih podataka i relativno kompleksne procese u promatranom hidrološkom sustavu [19,20].

ANN su svoju primjenu u hidrologiji našle u različitim područjima, od kojih su neka: prognoza oborina, modeliranje kišnog otjecanja, modeliranje protoka, modeliranje podzemnih voda, modeliranje pronosa sedimenta, kvalitete vode, hidroloških vremenskih nizova, modeliranje utjecaja klimatskih promjena, rad vodonosnika, ekomodeliranje itd. [21,22]

ANN u hidrološkom modeliranju prvi su koristili French i sur. 1992. godine za izradu modela prognoze oborina u vremenu i prostoru [23]. Nakon toga je primjena ANN u hidrologiji kontinuirano u porastu. Halff i sur. 1993. koristili su trostruku ANN za predviđanje hidrograma [24], 1995. godine Kothayari je predložio jednostavnu ANN za

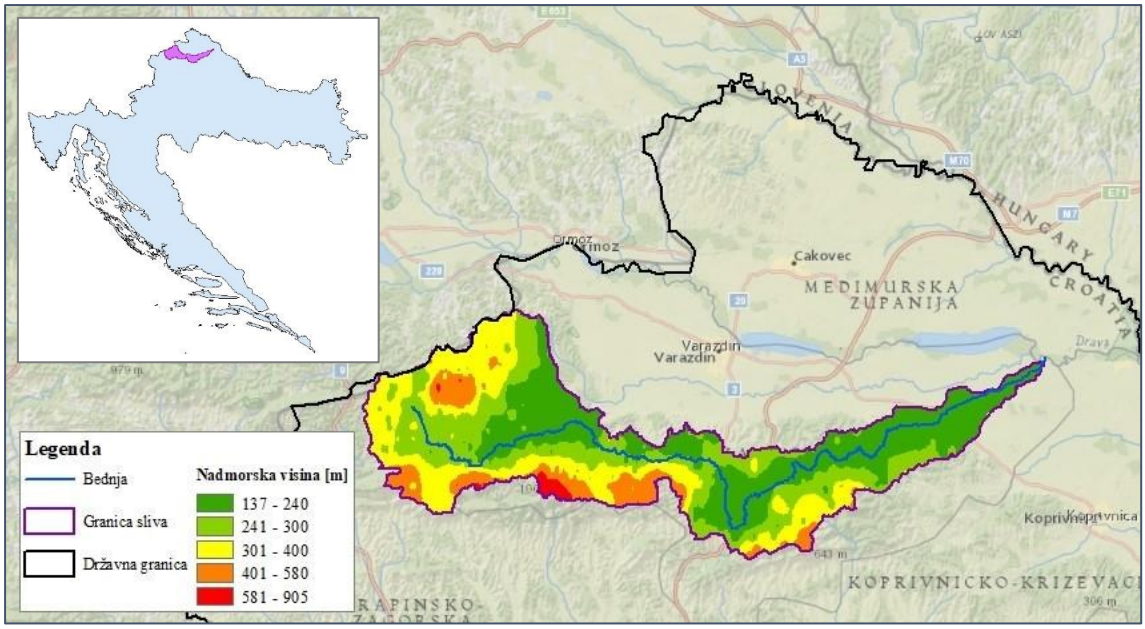
procjenu srednjeg mjesečnog otjecanja [7], dok su Raman i Sunil Kumar 1995. predložili isti ANN model za procjenu mjesečne količine padalina [25].

Ako proučavamo posebno modele predviđanja protoka, u mnogim istraživanja korišten je uzročno-posljedični oblik modeliranja protoka, gdje su povijesna opažanja uzročnih varijabli (oborine, temperatura, sadržaj vlage u tlu, isparavanje, otjecanje), pojedinačno ili u kombinaciji, korištena za predviđanje protoka [26]. S druge strane, u nekim modelima su za predviđanja protoka korištene samo opažene prethodne vrijednosti protoka (npr. modeliranje jednovarijantnih vremenskih nizova podataka) [27]. Također, postoji velik broj studija kod kojih su uspoređivani rezultati i mogućnosti modela temeljenih na ANN i tradicionalnih statističkih metoda [28]. Kod ovakvih istraživanja, uglavnom se pokazivalo kako ANN imaju bolju sposobnost predviđanja, a posebno je izražena njihova nadmoć nad modelima koji koriste linearnu regresiju, što i nije čudno s obzirom na prirodnu nelinearnost u hidrološkim procesima, koje regresijske metode ne mogu točno predstaviti. U mnogim istraživanjima korišteni su hibridni modeli, u kojima su ANN i neka od tradicionalnih metoda korišteni zajedno kako bi međusobno nadoknadili moguća ograničenja. Sivakumar, Jayawardena i Fernando kombinirali su ANN s metodom rekonstrukcije faznog prostora za predviđanje protoka rijeke [29]. Jain i Kumar koristili su kombinaciju ANN-ova i tradicionalnih tehnika vremenskih nizova za mjesečno predviđanje protoka [7,28].

3. ZNAČAJKE SLIVA I RIJEKE BEDNJE

Bednja je najduža rijeka čiji su izvor i ušće u Hrvatskoj, a njezin sliv smješten je u sjeverozapadnoj Hrvatskoj (Slika 4.). Desna je pritoka rijeke Drave pa pripada dunavskom, odnosno crnomorskom slivu.

Bednja izvire na nadmorskoj visini od 300 m kroz više raštrkanih malih izvora, dok stalan tok tvori podno Maceljske gore, u blizini naselja Brezova Gora. Teče gotovo cijelim tokom u smjeru zapad-istok. U blizini Novog Marofa, kod naselja Presečno, tok Bednje zaokreće prema jugu, okolo Varaždinsko-topličkog gorja, pa opet skreće u smjeru sjevera te se vraća u smjer zapad-istok. Kod naselja Mali Bukovec Bednja utječe u rijeku Dravu (na 136 m nadmorske visine) [30,31].



Slika 4. Položaj i oblik sliva Bednje

3.1. Hidrogeološke značajke

Ukupna dužina toka Bednje je oko 106 km. Slivno područje Bednje površine je otprilike 596 km^2, a može se podijeliti na dva dijela. Visinski dio zauzima 480 km^2, tj. oko 70% sliva, dok nizinski dio zauzima preostalih 30%. Aluvijalnu ravnicu Bednje između izvorišnog područja i ušća karakteriziraju suženja s proširenjima, koja čine Bednjansko polje, Lepoglavsko-ivanečko polje, Margečansko-završko-podevčevsko polje, Novomarofsko-ključko polje, Topličko-tuheljsko polje, Ludbreško-kapelsko-bukovačko polje, itd. [30,31].

Na kretanje vode u slivu Bednje uglavnom utječu geološke strukture, kao što su rasjedi i druge formacije. Lepoglavska sinklinala formirana je u srednjem pliocenu, nakon taloženja rhomboidea slojeva. Ona je za vrijeme gornjeg pliocena i donjeg pleistocena predstavljala depresiju ispunjenu slatkom vodom, a nakon izdizanja je formirano korito Bednje. Bednja u gornjem toku teče paralelno s osi Lepogravske sinklinale, koja je dio tektonske jedinice horst Ravna gora, a zatim naglo skreće prema jugu i kod Margečana presjeca trijaske tufove. Dalje prema istoku ponovno teče paralelno s osi antiklinale do rasjeda Ključ-Črešnjevo. Skretanje toka Bednje uzrokovano je diferencijalnim kretanjima blokova tijekom pleistocena i holocena [32].

Sliv Bednje prema hidrogeološkim značajkama možemo podijeliti na tri dijela. Prvi dio čini planinsko područje (Ivanščica, Kalničko gorje i Ravna gora), sastavljeno najvećim dijelom od karbonatnih stijena, posebice dolomita, nastalih tijekom mezozoika.

Procijenjeno je da je karbonatni vodonosnik na ovom području debeo nekoliko stotina metara, a propusnost stijena kreće se od 3% do 25%. Iako neki hidrogeološki aspekti ostaju nedovoljno proučeni, poznato je da bore i rasjedi igraju značajnu ulogu u strukturnim značajkama ovog dijela slivnog područja. Drugi dio čine sedimentni kompleksi iz tercijarnog razdoblja, nastali erozijom i ispiranjem starijih stijena. Treći dio obuhvaća nizinsko područje uz rijeku Bednju i sastoji se od naslaga kvartarne starosti. Ovo područje je važno jer se u njemu talože šljunci, pijesci i gline holocenske starosti, tvoreći dio aluvija Varaždinske županije. Ti aluvijalni sedimenti stvaraju vodonosnike međuzrnske poroznosti, heterogenog sastava, male debljine te niske propusnosti (zbog prevladavanja sitnozrnatih čestica). Prihranjivanje podzemne vode prvenstveno se događa infiltracijom oborina i procjeđivanjem iz vodotoka [33,34].

3.2. Hidrološke značajke

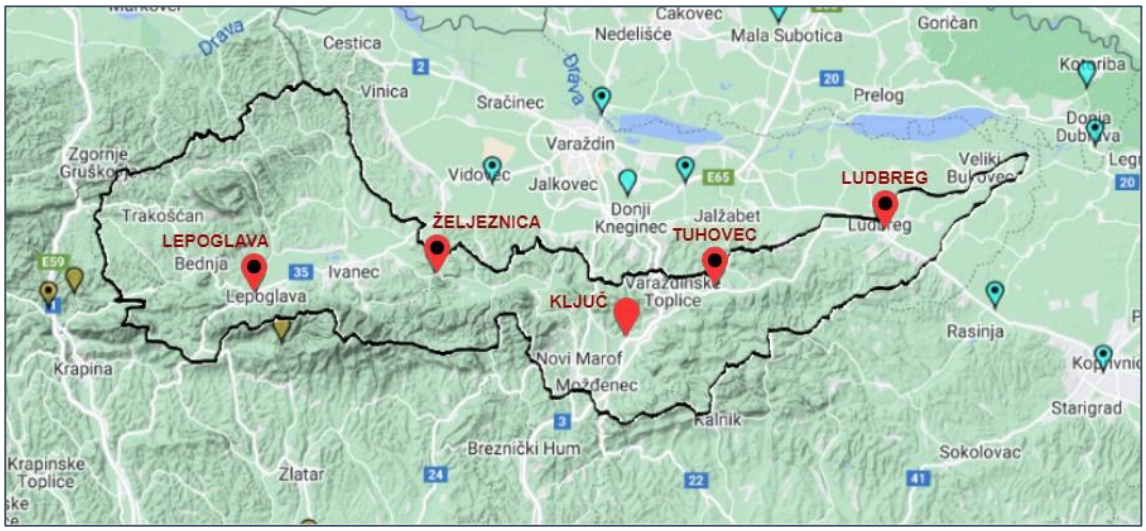
Slivno područje Bednje vrlo je izduženo. Lepezasto je prošireno u izvorišnom području (širine do 29 km), nakon čega se sužava prema području ušća (širine 4 km). Prosječna širina sliva je 5,8 km. Uzdužni profil rijeke Bednje uglavnom označava nizinsku rijeku s minimalnim visinskim promjenama. No, korito rijeke omeđeno je planinama i brežuljcima, koji čine i razvodnicu sliva, pa je cjelokupno područje sliva pretežito brdsko. Tok Bednje dijeli se na: gornji tok (do ušća Železnice), srednji tok (do ušća Velinečkog potoka) i donji tok do ušća u Dravu [33,35].

Bednja ima peripanonski kišno-snježni (bujični) režim, koji ovisi o količini oborina (i topljenju snijega), s maksimumom protoka u proljeće (ožujak - travanj) i čestim plavljenjem doline [36]. Nepovoljan raspored oborina i njihovo otjecanje s padina Ivančice, Ravne Gore i Kalničkog gorja u korito rijeke u kombinaciji s visokim intenzitetom oborina, uzrokuju visoku i brzu koncentraciju toka u gornjem toku te naglo plavljenje doline [34,37].

Na području gornjeg i srednjeg toka korito Bednje građeno je od materijala bogatog glinom, pomiješanog s naslagama pijeska i prašine. S obzirom na osjetljivost ovog materijala na eroziju, brz dotok vode dovodi do povećanja profila korita i presijecanja meandara. Donji tok rijeke Bednje obuhvaća ravničarsko područje, a erozijski procesi aktivni su na cijelom slivu [34].

Osnovni hidrološki podaci dobivaju se motrenjima (mjerenja i opažanja) na hidrološkim postajama mreže površinskih i mreže podzemnih voda na prostoru Republike Hrvatske, koje su u nadležnosti Državnog Hidrometeorološkog Zavoda (DHMZ) [38].

Na rijeci Bednji smješteno je pet hidroloških postaja površinskih voda, a to su redom: Lepoglava, Željeznica, Ključ, Tuhovec i Ludbreg. Postaja ključ je limnigrafska, dok su ostale četiri postaje s automatskom dojavom vodostaja u realnom vremenu (AD). Na svakoj od postaja mjere se vodostaji i protoci, a na postaji Željeznica mjere se još i koncentracija i nanos [38]. Lokacije postaja prikazane su na Slici 5.. U Tablici 1. navedeni su osnovni statistički pokazatelji podataka o dnevnim protocima s tri hidrološke postaje s kojih su korišteni podaci za izradu hidrološkog modela u ovom radu.



Slika 5. Hidrološke postaje na području sliva Bednje [38]

Tablica 1. Osnovni statistički pokazatelji dnevnih protoka za razdoblje od 1.1.2010. – 31.12.2020.

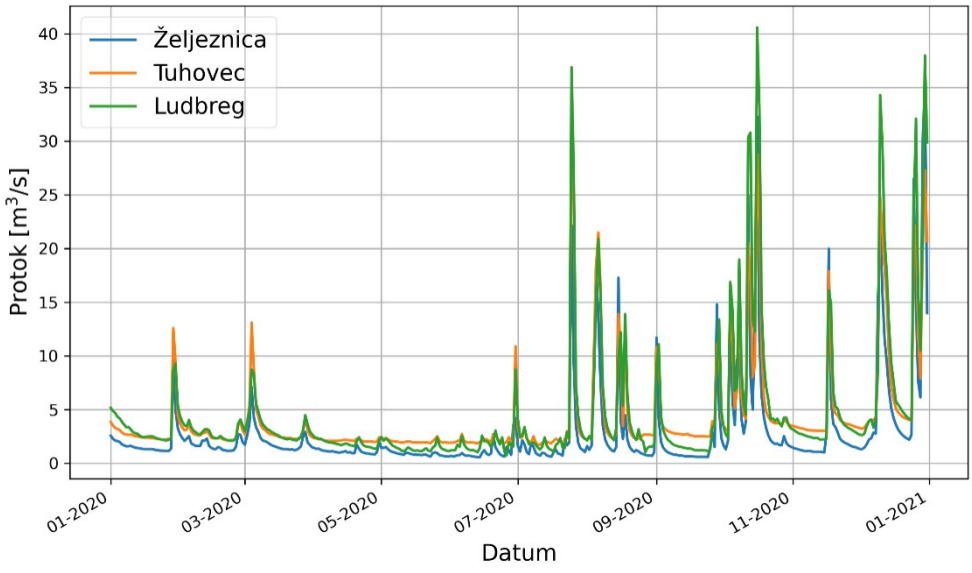
	Broj podataka	**Aritmetička sredina [m³/s]**	**Standardna devijacija [m³/s]**	**Min. [m³/s]**	**25%**	**50%**	**75%**	**Maks. [m³/s]**
Željeznica	4018	3,75	7,33	0,27	0,84	1,53	3,21	123,00
Tuhovec	4018	5,92	10,22	0,82	1,75	2,59	5,27	103,00
Ludbreg	4018	6,75	11,53	0,74	1,85	3,08	6,36	116,00

Za izradu modela u ovom radu korištene su vrijednosti protoka izmjerenih na postajama Željeznica, Tuhovec i Ludbreg. U Tablici 1. navedeni su osnovni statistički pokazatelji dnevnih protoka s tih postaja. Hidrološke postaje navedene su redom, od

najuzvodnije do najnizvodnije. Vidljivo je kako srednja vrijednost protoka raste pomicanjem nizvodno, od 3,75 [m^3/s] na Željeznici do 6,75 [m^3/s] na postaji Ludbreg. Ovakvi rezultati statističke analize su očekivani i u skladu s opisanim oblikom sliva i posljedičnom dinamikom u njemu. Vrijednosti standardne devijacije na sve tri postaje su relativno visoke, što potvrđuje bujični režim rijeke. Standardne devijacije također rastu pomicanjem nizvodno, ukazujući na veću varijabilnost vrijednosti protoka uzrokovanu naglim porastima koncentracije toka pri većem intenzitetu padalina, opet vezanu za oblik sliva.

Pregledom vrijednosti 25, 50 i 75 percentila uočavamo da su vrijednosti 75 percentila za sve tri postaje blizu vrijednosti aritmetičkih sredina, što ponovo ukazuje na postojanje naglih i značajnih porasta vrijednosti protoka u područjima s relativno niskim bazičnim protocima, koje podižu vrijednost aritmetičke sredine.

Na Slici 6. prikazan je hidrogram protoka sa sve tri promatrane postaje za razdoblje od 1.1.2020. – 31.12.2020. S obzirom da bi prikaz svih dostupnih podataka (za razdoblje od 11 godina) bio nepregledan, za primjer je odabrana najrecentnija godina. Vidljiv je općenit porast vrijednosti protoka nizvodno. Vršne vrijednosti javljaju se gotovo istovremeno, s malim pomakom nizvodno, a najviše su uglavnom kod najnizvodnije postaje.



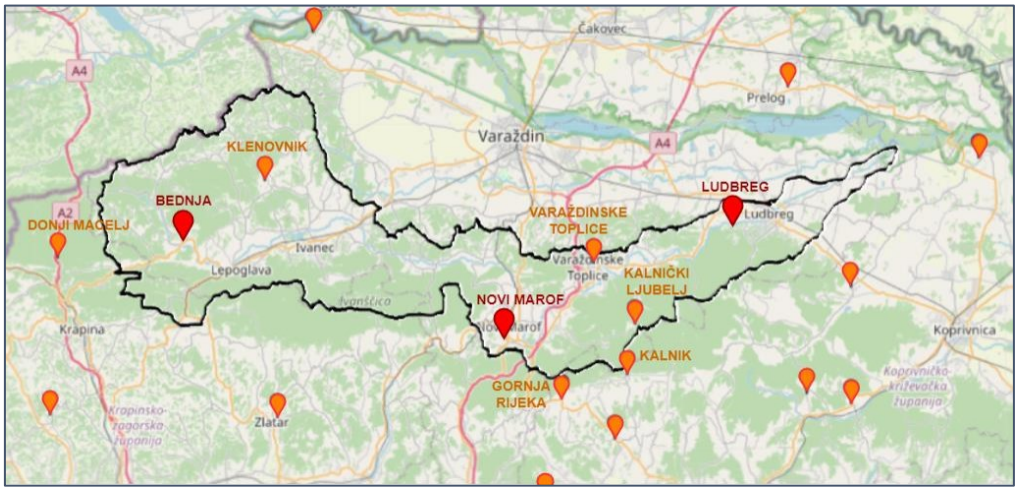
Slika 6. Protoci na hidrološkim postajama u razdoblju od 1.1.2010. – 31.12.2020.

3.3. Meteorološke značajke

Područje sliva Bednje pripada u šire Varaždinsko područje te, kao i cijela sjeverozapadna Hrvatska, općenito spada u umjereno toplu i kišnu klimu, s najvećim intenzitetom padalina u zimskom dijelu godine. Najtopliji mjeseci su ljetni mjeseci (srpanj, kolovoz), dok su najhladniji zimski mjeseci (siječanj, veljača). Ovo područje ima kontinentalni tip godišnjeg hoda oborina. Tipičan je sjeveroistočni vjetar koji najčešće puše u zimskom dijelu godine donoseći hladno i vedro vrijeme. Najvjetrovitije razdoblje godine je proljeće (istočnjak), a najvjetrovitije područje su vrhovi Ivančice. No, zbog okruženosti planinama i brežuljcima, temperature na samom području sliva često se znatno razlikuju od onih u obližnjim gradovima (Varaždinu ili Krapina). Takav smještaj, pri određenim uvjetima, prilikom spuštanja hladnog zraka niz padine uzrokuje temperaturnu inverziju i niže temperature zraka, posebno u zimskim mjesecima [39].

Na području sliva nalaze se tri klimatološke postaje, a s mjerenjima su počele: Bednja – 2006., Novi Marof - 2006., Ludbreg – 1981.. Najniže i najviše izmjerene temperature (od početka mjerenja) na navedenim postajama su: Bednja -23,8 C° (9.2.2012.) i 37,9 C° (8.8.2013.), Novi Marof – 25,0 C° (8.1.1985.) i 38,0 C° (5.8.2012.), Ludbreg -26,6 C° (8.1.1985.) i 38,2 C° (24.8. 2012.), uz napomenu da ne postoje podaci za dijeli niz [38].

Na slivnom području Bednje nalazi se šest meteoroloških postaja (Slika 7.): tri klimatološke postaje (KMP) Bednja, Novi Marof i Ludbreg i tri kišomjerne postaje (KŠP): Klenovnik, Varaždinske Toplice i Kalnički Ljubelj. U blizini sliva još je nekoliko kišomjernih postaja, čiji su podaci o oborinama relevantni za njegove meteorološke značajke [38,40].



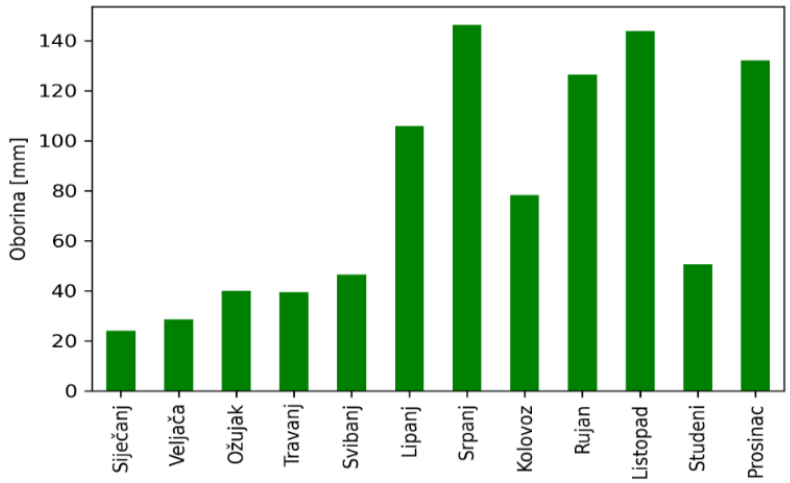
Slika 7. Meteorološke postaje na području sliva Bednje [38]

Tablica 2. Osnovni statistički pokazatelji dnevnih oborina za razdoblje 1.1.2010. – 31.12.2020.

	Broj podataka	Aritmetička sredina [mm]	Standardna devijacija [mm]	Min. [mm]	25%	50%	75%	Maks. [mm]
Bednja	4018	2,92	7,58	0,0	0,0	0,0	1,10	80,00
Gornja Rijeka	4018	2,73	7,37	0,0	0,0	0,0	1,10	114,00
Klenovnik	4018	2,99	7,65	0,0	0,0	0,0	1,38	94,70
Kalnički Ljubelj	4018	3,02	7,92	0,0	0,0	0,0	1,50	89,30
Novi Marof	4018	2,48	6,49	0,0	0,0	0,0	1,00	68,10
Varaždinske Toplice	4018	2,80	7,60	0,0	0,0	0,0	1,00	88,80

U Tablici 2. navedeni su osnovni statistički pokazatelji podataka o dnevnim oborinama s meteoroloških postaja korištenih za izradu modela u ovom radu. Vidljivo je da su srednje vrijednosti oborina sa svih postaja međusobno slične, s najnižom vrijednosti od 2,48 [mm] na postaji Novi Marof i najvišom vrijednosti od 2,99 [mm] na postaji Klenovnik. Općenito je na području gornjeg sliva nešto veća srednja količina oborina. Vrijednosti standardne devijacije pokazuju visoku varijabilnost u podacima, što je očekivano s obzirom da je vrijednost za dane bez kiše 0 (ujedno i minimalna vrijednost). Zbog toga što je više od polovice dana u godini bez oborina iznosi 25 i 50 percentila su 0.

Na Slici 8. prikazan je godišnji hod oborina za 2020. na klimatskoj postaji Bednja. Količine oborina zabilježene u 2020. godini generalno su niže od prethodnih godina, ali je uobičajen hod oborina na području sliva Bednje i dalje vidljiv. Može se primijetiti poklapanje većih količina oborina sa za zabilježenim vršnim protocima u istom razdoblju, prikazanima na Slici 6..



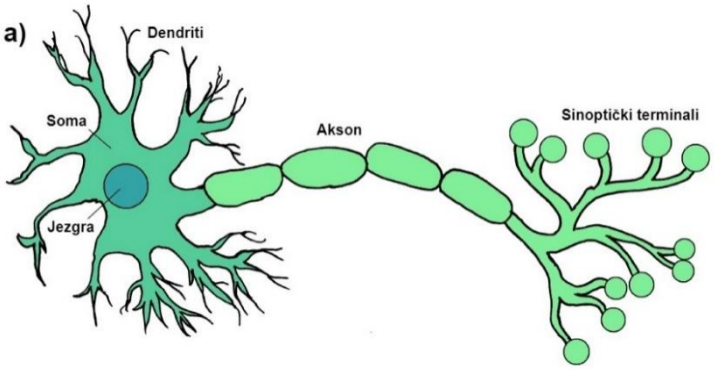
Slika 8. Godišnji hod oborina na klimatološkoj postaji Bednja za razdoblje od 1.1.2020. – 31.1.2020.

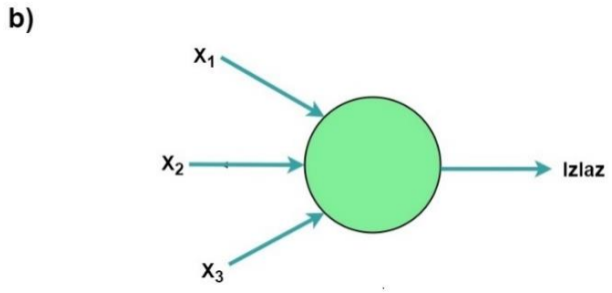
4. METODOLOGIJA

4.1. Umjetne neuronske mreže

Prethodno je već spomenut pojam umjetne neuronske mreže (eng. *Artificial Neural Network*/ANN), kao grane umjetne inteligencije, odnosno temeljnog algoritma dubokog učenja. Generalni cilj razvoja umjetne inteligencije jest što bolja imitacija ljudskog mozga – njegove inteligencije, sposobnosti učenja i donošenja odluka, pa čak i kreativnosti, u umjetnom obliku [19].

Ljudski mozak sastoji se od velikog broja živčanih stanica, neurona (Slika 9.,a)), koji pri obradi primljenih informacija rade paralelno. Neuroni su međusobno povezani sinapsama te pomoću električnih impulsa kroz njih prenose informacije s jednog neurona na sljedeći, sve do mozga, koji zatim razmišlja, zaključuje, donosi odluke itd. Umjetne neuronske mreže inspirirane su upravo stvarnom strukturom i funkcioniranjem ljudskog mozga, posebno međusobno povezanom mrežom neurona koji obrađuju i prenose informacije [19,41].





Slika 9. Usporedba ljudskog neurona i perceptrona [41]

Umjetne neuronske mreže sastoje se od više međusobno povezanih jedinica (čvorova), zvanih **umjetni neuroni**, koji su organizirani u slojeve. Umjetni neuroni u mreži analogni su biološkim neuronima u mozgu, na način da primaju ulaze od drugih neurona ili vanjskih izvora, izvode proračune na tim ulazima i generiraju izlazni signal. Izlaz jednog neurona postaje ulaz za drugi, omogućavajući protok informacija kroz mrežu, pa su tako veze između neurona u neuronskoj mreži slične sinapsama u mozgu, gdje se informacije prenose s jednog neurona na drugi [19,42].

Iako su neuronske mreže inspirirane mozgom, one su ipak vrlo pojednostavljena apstrakcija. Mozak je neizmjerno složen, s milijardama neurona i zamršenim obrascima povezivanja, dok neuronske mreže obično imaju puno manji broj umjetnih neurona. Neuronske mreže imaju za cilj obuhvatiti neke od temeljnih principa neuronske obrade, ali ne repliciraju u potpunosti zamršenost bioloških neuronskih mreža.

4.1.1. Povijest umjetnih neuronskih mreža

Povijest neuronskih mreža seže do sredine 20. stoljeća, sa značajnim razvojem koji se dogodio tijekom godina. 1943. godine Warren McCulloch i Walter Pitts predložili su neuronski model koji je predstavljao pojednostavljeni računalni model rada neurona u ljudskom mozgu, a postavio je temelj za umjetne neuronske mreže [42,43].

Potaknut tim modelom, Frank Rosenblatt razvio je perceptron (Slika 9., b)). Perceptron je temeljni koncept u teoriji neuronskih mreža i građevni blok za složenije arhitekture neuronskih mreža. To je vrsta umjetnog neurona koji se može koristiti za zadatke binarne klasifikacije. Cilj mu je oponašati ponašanje jednog biološkog neurona, koji prima ulazne informacije, primjenjuje na njih težinske faktore i proizvodi izlaz na temelju aktivacijske funkcije [42,43].

Osnovna struktura perceptrona (na primjeru perceptrona s tri ulazne vrijednosti, Slika 9.) sastoji se od [42]:

- **Ulazi** (eng. *Inputs*), x_1, x_2, x_3 - svaki je povezan s pripadajućim težinskim faktorom, koji određuje njegovu važnost
- **Težinski faktori** (eng. *Weights*), w_1, w_2, w_3 - svakom ulazu pridružen je faktor koji odražava njegov doprinos konačnom rezultatu. Obično se inicijaliziraju nasumično i ažuriraju tijekom procesa učenja.
- **Težinska suma** (eng. *Weighted Sum*), $\sum x_j w_j$
- **Granična vrijednost** (eng. *Treshold Value*) **ili pristranost**, b (eng. *Bias*) – određuje da li je izlazna vrijednost 0 ili 1, odnosno koliko je lako perceptronu doći do izlaza 1, te vrijedi: b = - granična vrijednost
- **Izlaz**
    - Algebarski opis pomoću granične vrijednosti:

$$Izlaz = \begin{cases} 0 \; ako \; \sum_j w_j x_j \leq granična \; vrijednost \\ 1 \; ako \; \sum_j w_j x_j > granična \; vrijednost \end{cases} \quad (1)$$

    - Algebarski opis pomoću pristranosti:

$$Izlaz = \begin{cases} 0 \; ako \; \sum_j w_j x_j + b \leq 0 \\ 1 \; ako \; \sum_j w_j x_j + b > 0 \end{cases} \quad (2)$$

Korištenjem različitih težinskih faktora i pristranosti mogu se napraviti različiti modeli, čak i s ovim jednostavnim tipom umjetnog neurona. Naravno, perceptron nije potpuni model koji oponaša ljudsko odlučivanje, ali se kao jedinice mogu koristiti za vaganje različitih vrsta dokaza kako bi se donosile odluke, pa tako onda složena mreža perceptrona može donositi dosta suptilne odluke. Danas je uobičajenije koristiti druge modele neurona, a jedan od najčešće korištenih modela neurona je sigmoidni neuron.

U daljnjem povijesnom razvoju ANN, 1974. godine Paul Werbos predstavio je algoritam povratne propagacije, koji je omogućio neuronskim mrežama da učinkovito prilagođavaju svoje težinske faktore i uče iz podataka za obuku [44]. Od tada su

neuronske mreže nastavile napredovati, došlo je do razvoja novih tehnika i različitih naprednih tipova ANN [20].

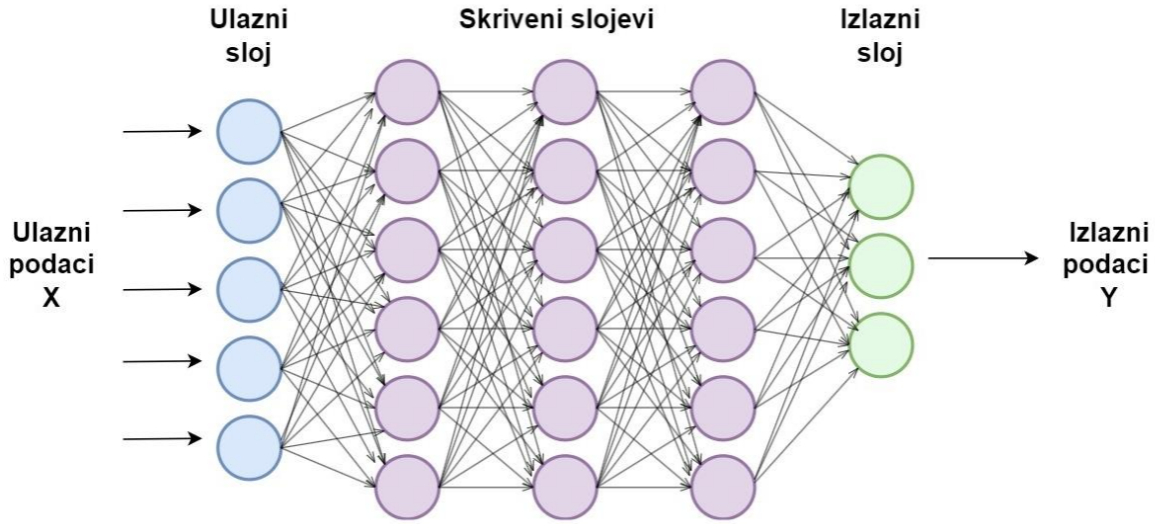
U sljedećem potpoglavlju način na koji funkcionira umjetna neuronska mreže objašnjen je na primjeru unaprijedne neuronske mreže.

4.1.2. Princip rada neuronske mreže

Unaprijedna neuronska mreža (eng. *Feed-forward Neural Network*) ili višeslojni perceptron (eng. *Multilayer Perceptrone*) je najjednostavniji tip neuronske mreže, no svejedno je dovoljno kompleksna da bismo na njenom primjeru prikazali općeniti princip rada svih neuronskih mreža. Kao što i sam naziv daje naslutiti, ovo je vrsta umjetne neuronske mreže u kojoj informacije teku prema naprijed, od ulaznog sloja do izlaznog sloja bez ikakvih ciklusa ili petlji [7,21]. Već je ranije navedeno da su neuronske mreže algoritmi koji se koriste u dubokom učenju, a važno je istaknuti da se „duboko" u dubokom učenju odnosi na dubinu slojeva u neuronskoj mreži, odnosno da se tek neuronska mreža koja se sastoji od više od tri sloja (uključujući ulaze i izlaze) može smatrati algoritmom dubokog učenja [6].

Osnovna jedinica neuronske mreže je **neuron**. Neuroni u mreži organizirani su u slojeve, a arhitektura napredne neuronske mreže (Slika 10.) sastoji se od tri vrste slojeva [42]:

- **Ulazni sloj** - odgovoran je za primanje ulaznih podataka i njihovo prosljeđivanje sljedećem sloju. Svaki neuron u ulaznom sloju predstavlja značajku ili atribut ulaza, dakle njihov broj ovisi o broju ulaznih podataka
- **Skriveni slojevi** – međuslojevi između ulaznog i izlaznog sloja. Sastoje se od više neurona koji izvode proračune na ulaznim podacima. Svaki neuron prima ulaze iz prethodnog sloja, primjenjuje težinske faktore i pristranosti na njih i prosljeđuje rezultat kroz aktivacijsku funkciju da proizvede izlaz. Broj skrivenih slojeva i broj neurona u svakom sloju može varirati ovisno o složenosti problema.
- **Izlazni sloj** - završni sloj mreže, odgovoran za proizvodnju željenog izlaza ili predviđanja. Broj neurona u izlaznom sloju ovisi o prirodi problema.



Slika 10. Arhitektura unaprijedne neuronske mreže (prema [20])

Informacije kroz mrežu prolaze s ulazne na izlaznu stranu. Neuroni u jednom sloju povezani su s onima u sljedećem, ali ne i međusobno. Ulazni podaci prvog (ulaznog) sloj su ulazne poznate varijable relevantne za rješavanje problema, stoga je ulazni sloj transparentan i sredstvo pružanja informacija mreži. Posljednji (izlazni) sloj sastoji se od vrijednosti koje je predvidjela mreža i stoga predstavlja izlaz modela. Optimalan broj skrivenih slojeva i broj neurona u svakom skrivenom sloju obično se određuje metodom pokušaja i pogreške. Neuroni unutar susjednih slojeva mreže potpuno su povezani vezama. Svakoj vezi se dodjeljuje težinski faktor (w), koji predstavlja relativnu snagu veze dvaju neurona [42,45].

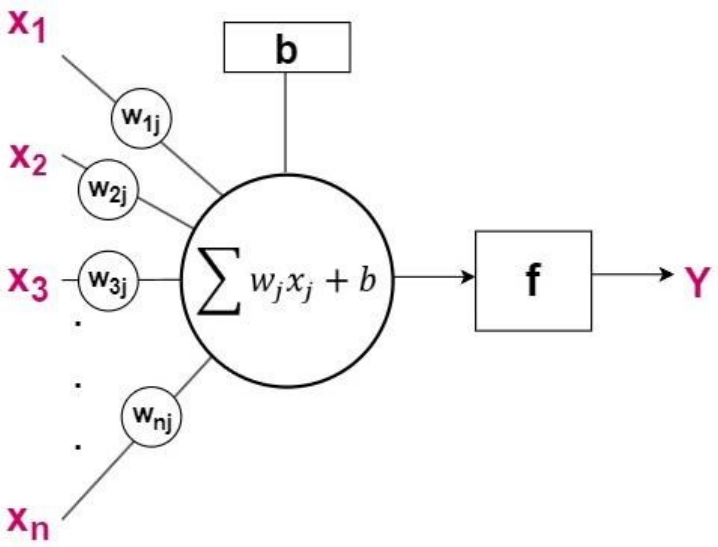
Na temelju ulaznih podataka (x) i težinskih faktora (w) dodijeljenih vezama između neurona, u svakom čvoru računa se težinska suma (S_{ij}):

$$S_{ij} = \sum_{i=1}^{n} w_i x_j \tag{3}$$

Ta suma, s dodanom pristranosti (b) zatim prolazi kroz aktivacijsku ili prijenosnu funkciju, čime dobivamo izlaz iz neurona (y):

$$y = f\left(\sum_{i=1}^{n} w_i x_j + b\right) \tag{4}$$

Na ovaj način, računaju se **aktivacije**, odnosno izlazi u svakom neuronu. Izlazi (y) iz jednog sloja predstavljaju ulaze u sljedeći sloj te se proces ponavlja do kad se ne dođe do vrijednosti u zadnjem, izlaznom sloju [42,46]. Na Slici 11. prikazana je aktivacija jednog neurona.



Slika 11. Aktivacija jednog neurona (prema [19])

4.1.3. Aktivacijske funkcije

Aktivacija neurona je vrijednost koja je u njemu „pohranjena“ i koja odlučuje o tome koliki utjecaj taj neuron ima na one u sljedećim slojevima neuronske mreže. Ako je aktivacija neurona 0, tada je on neaktivan i nema utjecaj na izračune u neuronima u sljedećem sloj. Ako je aktivacija neurona veća od 0, on je aktivan i ulazi u izračune u sljedećem sloju. Veći utjecaj imaju neuroni s višom vrijednosti aktivacije i obratno [42,43].

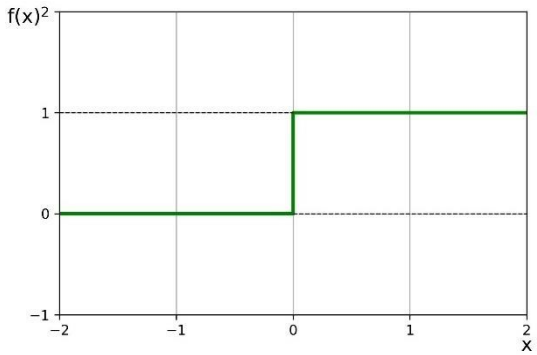
Aktivacijska (prijenosna) funkcija je iznimno važna značajka neuronskih mreža. Ona određuje da li će se neuron aktivirati ili ne, odnosno veličinu njegove aktivacije. Najjednostavnija aktivacijska funkcija je već navedena težinska suma ulaza u neuron, koju koristi jednoslojna neuronska mreža ADALINE (Adaptive Linear Element) [46]. No, u današnjim neuronskim mrežama uglavnom se koriste aktivacijske funkcije koje na ulazne podatke primjenjuju određenu nelinearnu transformaciju. Korištenje takvih funkcija omogućava neuronskim mrežama da uče učinkovitije od modela linearne regresije i obavljaju mnogo kompleksnije zadatke [20,43].

U nastavku su navedene i ukratko objašnjene najčešće korištene aktivacijske funkcije [42,47,48]:

1. Step funkcija

Izlaz funkcije je 1 ako je ulaz veći od granične vrijednosti (npr. t = 0). U suprotnom je izlaz 0. Dakle, neuron je neaktivan ako je izlaz 0, a aktiviran ako je izlaz 1. Ovakva funkcija koristi se u mreži s perceptronima, za pretvorbu izlaza u binarni oblik.

$$f(x) = \begin{cases} 1 \; ako \; x \; \geq t \\ 0 \end{cases} \tag{5}$$

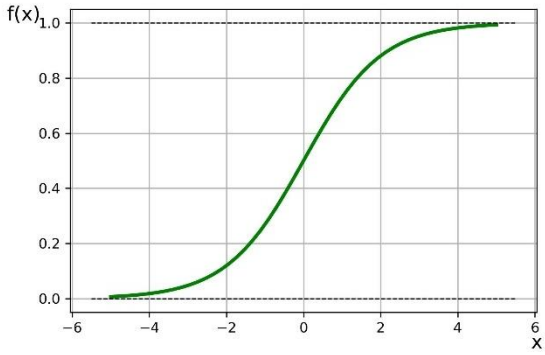


Slika 12. Grafički prikaz step funkcije

2. Sigmoidna funkcija

Ovo je vrsta logističke aktivacijske funkcije koja ulazne vrijednosti preslikava u raspon od 0 do 1. Kada je x veliki pozitivan broj, eksponencijalni član u nazivniku postaje vrlo velik, a sigmoidna funkcija se približava vrijednosti blizu 1. Suprotno, kada je x veliki negativan broj, eksponencijalni član postaje vrlo malen, a sigmoidna funkcija približava se vrijednosti blizu 0. Sigmoidna funkcija se često koristi kao aktivacijska funkcija u izlaznom sloju problema binarne klasifikacije, gdje je cilj predvidjeti jednu od dvije klase. Također se može koristiti u skrivenim slojevima neuronske mreže, iako su druge aktivacijske funkcije stekle veću popularnost zbog svojih boljih performansa u arhitekturama dubokog učenja. Jedan od nedostataka sigmoidne funkcije je mogućnost pojave nestajućeg gradijenta, koji je objašnjen u jednom od sljedećih poglavlja.

$$f(x) = \frac{1}{1 + e^{-1}} \tag{6}$$

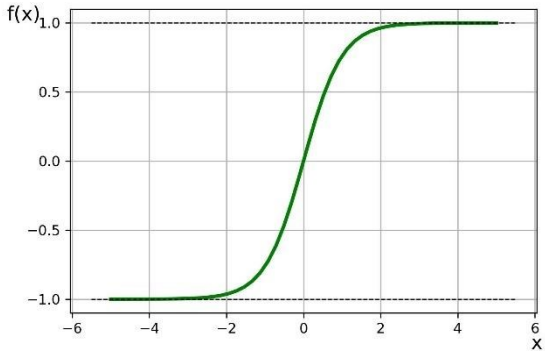


Slika 13. Grafički prikaz sigmoidne funkcije

3. TanH

Tangentna hiperbolična funkcija ustvari je skalirana i pomaknuta sigmoidalna funkcija, pa preslikava ulazne vrijednosti u raspon od -1 do 1. Često se koristi kao aktivacijska funkcija u skrivenim slojevima neuronskih mreža, a osobito je korisna kad se radi s ulazima koji imaju negativne i pozitivne vrijednosti budući da bilježi i pozitivne i negativne odnose u podacima. U usporedbi sa sigmoidnom funkcijom, TanH funkcija ima strmiji gradijent, što može olakšati učenje u neuronskim mrežama. No, kod ekstremnih vrijednosti ulaza također može doći do problema nestajućeg gradijenta, što ju čini manje prikladnom za vrlo duboke mreže.

$$f(x) = \frac{2}{1 + e^{-2x}} - 1 \tag{7}$$



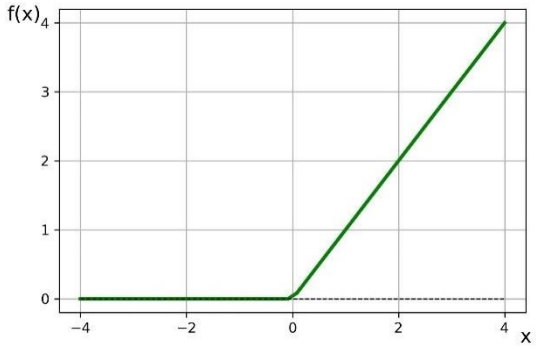
Slika 14. Grafički prikaz TanH funkcije

4. ReLU (eng. *Rectified Linear Unit*)

Ako je ulaz pozitivan ili nula, ReLU vraća vrijednost ulaza. Ako je ulaz negativan, ReLU vraća 0. Vrlo je popularna u dubokom učenju zbog svoje jednostavnosti, koja ju čini računalno učinkovitom za računanje i omogućuje mreži da brzo uči. Kod ove funkcije ne javlja se problem nestajućeg gradijenta, stoga se obično koristi u skrivenim slojevima

dubokih neuronskih mreža, ali dolazi do problema „umiranja ReLU“. Naime, za neke ulaze, gradijent može biti nula, uzrokujući da neuron postane neaktivan i "umre" tijekom faze obuke. Jednom kad se neuron zaglavi u ovom stanju, više ne pridonosi procesu učenja. S druge strane, ovaj problem u nekim slučajevima može biti i prednost jer smanjuje složenost mreže.

$$f(x) = max(0, x) \tag{8}$$

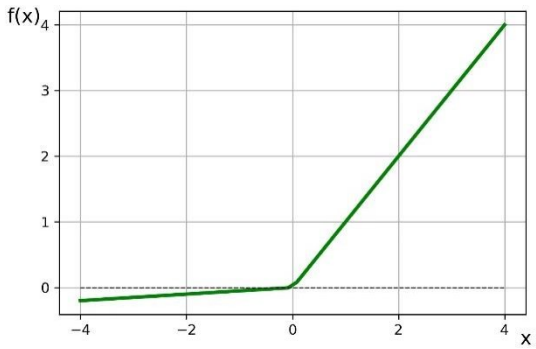


Slika 15. Grafički prikaz ReLU funkcije

5. Leaky ReLU

Ova funkcija je varijacija ReLU funkcije, kojom se pokušava riješiti problem „umiranja ReLU“. Leaky ReLU na pozitivne ulazne vrijednosti djeluje jednako kao i ReLU, dok negativne ulazne vrijednosti ne postavlja na 0, već se one množe s koeficijentom a (empirijska mala vrijednost). To omogućava pozitivan (mali) gradijent i za negativne ulaze, što pomaže neuronskoj mreži da nastavi s učenjem i ažuriranjem težinskih faktora, čak i za negativne ulazne vrijednosti.

$$f(x) = \{^{x \; ako \; x \; \geq 0}_{a \cdot x} \tag{9}$$

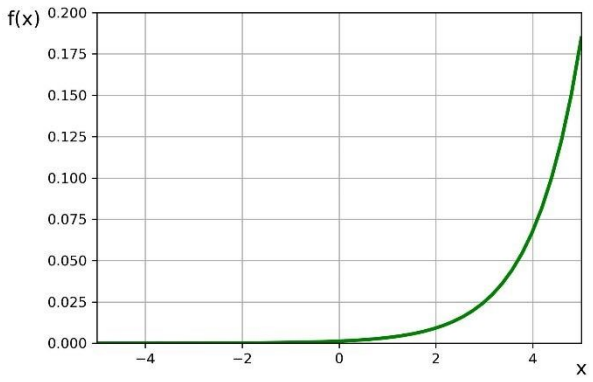


Slika 16. Grafički prikaz leaky ReLU funkcije

6. Softmax funkcija

Softmax ili normalizirana eksponencijalna funkcija na svaki ulazni element primjenjuje eksponencijalnu funkciju te ga normalizira dijeljenjem sa sumom svih tih eksponenata. Ona ulazne vrijednosti preslikava u raspon između 0 i 1, na način da je zbroj svih izlaznih vrijednosti jednak 0. Drugim riječima, rezultat primjene Softmax funkcije je raspodjela vjerojatnosti. Ova funkcija često se koristi u izlaznom sloju mreže kod rješavanje problema višeklasne klasifikacije.

$$f(x) = \frac{e^{x_i}}{\sum e^{x_j}} \tag{10}$$



Slika 17. Grafički prikaz Softmax funkcije

4.1.4. Kako neuronska mreža uči

Jednostavne neuronske mreže (npr. one sastavljene od perceptrona) konstruiraju se na način da obavljaju određeni zadatak, koristeći zadanu logičku funkciju. Tada točno znamo što i kako mreža radi. Kod složenijih neuronskih mreža koriste se i složenije aktivacijske funkcije (npr. sigmoidalna) te se obično gubi nadzor nad načinom kako ona obrađuje podatke. U tom slučaju neuronska mreža uči kroz postupak učenja ili treniranja [19].

Dva su glavna pristupa u obučavanju neuronskih mreža, a razlikuju se po tome kako su podaci o obuci označeni ili strukturirani [17]:

- **Nadzirano učenje** (eng. supervised learning) - uključuje obuku neuronske mreže pomoću označenih podataka, gdje je svaka ulazna vrijednost povezana s odgovarajućom izlaznom ili ciljnom vrijednosti. Cilj je prilagoditi parametre mreže

na takav način da ona nauči preslikavati ulazne podatke u točan izlaz, a najčešće se koristi u rješavanju zadataka klasifikacije i regresije

- **Nenadzirano učenje** (eng. unsupervised learning) - uključuje obuku neuronske mreže na neoznačenim podacima, pri čemu mreža uči izdvajati uzorke, strukture ili prikaze iz podataka, bez poznatih izlaznih vrijednosti. Cilj obuke temelji se na inherentnim svojstvima podataka i skrivenim uzorcima u njima, a koristi se za grupiranje, asocijacije i smanjenje dimenzionalnosti.

U ovom radu naglasak je na nadziranom učenju mreže. Dakle, procesu optimizacije parametara mreže (težinskih faktora i pristranosti) korištenjem označenog skupa podataka. U nastavku je opisan takav proces obuke neuronske mreže po koracima:

1. Inicijalizacija parametara

Inicijalizacija parametara odnosi se na proces postavljanja početnih vrijednosti parametara mreže. Pravilna inicijalizacija parametara je važna jer može utjecati na brzinu konvergencije i ukupnu izvedbu neuronske mreže tijekom obuke. Odgovarajuća metoda inicijalizacije pomaže u sprječavanju problema kao što su nestajući gradijenti ili eksplodirajući gradijenti, koji mogu ometati proces učenja. Neke od često korištenih metoda su [49]:

- **Zero inicijalizacija** - pristranosti se inicijaliziraju s 0, a težinski faktori se inicijaliziraju s nasumičnim brojevima
- **Nasumična inicijalizacija** – težinski faktori i pristranosti inicijaliziraju se nasumičnim vrijednostima izvučenim iz određene distribucije. Uobičajeno korištene distribucije uključuju Gaussovu (normalnu) distribuciju ili uniformnu distribuciju. Ovakva inicijalizacija pomaže u uvođenju raznolikosti u početne parametre i sprječava da mreža zapne u lokalnim minimumima

2. Unaprijedna propagacija (eng. *Forward Propagation/ Forward Pass*)

Unaprijedna propagacija je proces u kojem se ulazni podaci obrađuju kroz neuronsku mrežu radi stvaranja predviđanja ili izlaza. To uključuje pomicanje ulaza prema naprijed kroz slojeve mreže, primjenu težinskih faktora i pristranosti te aktivacijskih funkcija za proizvodnju izlaza u svakom sloju.

Izračun aktivacija u neuronima prikazan je shematski na Slici 11. i matematički u sljedećem izrazu [42]:

$$a_j^{(L)} = \sigma(w_{j,0} \cdot a_0^{(L-1)} + w_{j,1} \cdot a_1^{(L-1)} + \cdots + w_{j,n} \cdot a_n^{(L-1)} + b_0) \quad (11)$$

Pri čemu je:

- $a_j^{(L)}$- aktivacija jotog neurona u L-tom sloju
- $w_{j,0}$ - težinski faktor veze između jotog neurona u prethodnom sloju i prvog neurona u L-tom sloju
- b_0- pristranost prvog neurona u L sloju
- σ - sigmoidna aktivacijska funkcija

Možemo zapisati i kao [42]:

$$z_j^{(L)} = w_{j,0} \cdot a_0^{(L-1)} + w_{j,1} \cdot a_1^{(L-1)} + \cdots + w_{j,n} \cdot a_n^{(L-1)} + b_0 \quad (12)$$

$$a_j^{(L)} = \sigma(z_j^{(L)}) \quad (13)$$

Kompaktniji zapis ovog izraza pomoću tenzora [42]:

$$\sigma\left(\begin{bmatrix} w_{0,0} & w_{0,0} & \cdots & w^{0,0} \\ w_{0,0} & w_{0,0} & \cdots & w^{0,0} \\ \vdots & \vdots & \ddots & \vdots \\ w_{0,0} & w_{0,0} & \cdots & w^{0,0} \end{bmatrix} \begin{bmatrix} a_0^{(0)} \\ a_0^{(0)} \\ a_0^{(0)} \\ [a_0^{(0)}] \end{bmatrix} + \begin{bmatrix} b_0 \\ b_0 \\ b_0 \\ b_0 \end{bmatrix}\right) \quad (14)$$

3. Izračun funkcije pogreške (eng. *Cost/Loss function*)

Nakon što ulazni podaci unaprijednom propagacijom dođu do posljednjeg (izlaznog) sloja mreže, izlaz mreže uspoređuje se s ciljnim vrijednostima pomoću funkcije pogreške (C). Funkcija pogreške je matematička mjera koja kvantificira razliku između predviđenih izlaza mreže i pravih ciljnih vrijednosti iz podataka za obuku mreže. Ona predstavlja izvedbu mreže ili odstupanje između predviđenih i željenih izlaza, odnosno govori o tome koliko dobro mreža predviđa izlazne vrijednosti, tj. obavlja svoj zadatak [42].

Izbor funkcije pogreške ovisi o konkretnom zadatku i vrsti podataka koji se predviđaju, a jedna od najjednostavnijih i najčešće korištenih funkcija je srednja kvadratna pogreška (eng. *Mean Squared Error*/MSE). MSE se često koristi za zadatke regresije, u kojima mreža ima cilj predvidjeti kontinuirane vrijednosti. Ova funkcija izračunava razliku između predviđenih i stvarnih vrijednosti, kvadrira tu razliku, a zatim uzima prosjek svih

kvadrata razlika. Rezultat je jedna vrijednost koja predstavlja ukupnu razliku između predviđenih i stvarnih vrijednosti, tj. prosječna pogreška svih primjera za obuku [42,50].

$$C = \frac{1}{n}\sum_{i=1}^{n}(a_j^{(L)} - y_j)^2 \tag{15}$$

Pri čemu je:

- C - funkcija pogreške (MSE)
- n - broj primjera za obuku
- $a_j^{(L)}$- predviđena vrijednost izlaza
- y_j - stvarna vrijednost izlaza

Predviđene vrijednosti izlaza mreže u stvari su vrijednosti aktivacije neurona u posljednjem sloju. Tada je prema formuli za izračun aktivacije (Izraz 11) vidljivo da su ulazne vrijednosti funkcije pogreške sve ulazne vrijednosti te svi težinski faktori i pristranosti u mreži. Funkcija pogreške govori o preciznosti predviđanja mreže, što znači da veća vrijednost funkcije pogreške znači lošiji rad mreže i obratno. Stoga je cilj obuke neuronske mreže upravo minimiziranje funkcije pogreške, na način da se izmjenjuje tj. prilagođava parametre mreže. To se obično postiže upotrebom optimizacijskih algoritama poput gradijentnog spuštanja, za iterativno prilagođavanje mrežnih parametara kroz proces povratne propagacije.

4. Metoda gradijentnog spusta (eng. *Gradient Descent*) i povratna propagacija (eng. *Backpropagation*)

Nakon inicijalizacije nasumičnih parametara mreže, unaprijedne propagacije i izračuna funkcije pogreške želimo minimalizirati iznos funkcije pogreške, tj. želimo prilagoditi njezine parametre kako bismo se što više približili njenom minimumu. Kod jednostavnijih funkcija (npr. funkcija jedne varijable) minimum se može eksplicitno izračunati traženjem točke u kojoj je njena derivacija jednaka nuli. Kod kompleksnijih funkcija s više varijabli, kakve koriste duboke neuronske mreže, izračun minimuma nije toliko jednostavan [19].

Ako znamo da je gradijent funkcije (∇f) vektor koji predstavlja brzinu promjene funkcije s obzirom na svaku njezinu ulaznu varijablu te pokazuje smjer najstrmijeg rasta funkcije u određenoj točki u njezinoj domeni, tada negativan gradijent pokazuje smjer

najstrmijeg (najbržeg) pada funkcije. Veličina vektora gradijenta predstavlja strmost funkcije u određenoj točki. Veća magnituda označava strmiju padinu, dok manja magnituda odgovara ravnijem području [42].

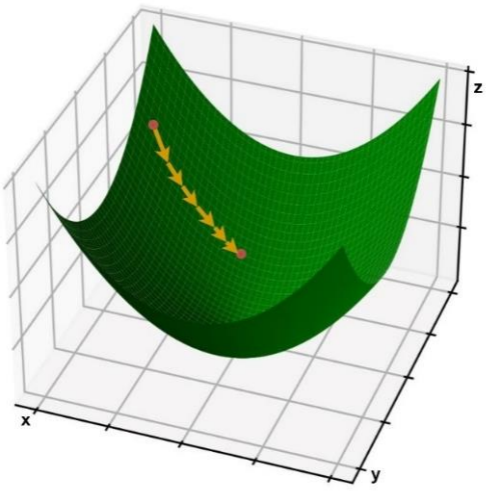
Izraz za gradijent funkcije $f(x_1, x_2, \ldots x_n)$:

$$\nabla f = (\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \ldots, \frac{\partial f}{\partial x_n}) \tag{16}$$

Gradijent je temeljni koncept u algoritmima za optimizaciju, jer pruža informacije o smjeru i veličini najvećeg rasta ili pada. On vodi traženje minimuma iterativnim podešavanjem parametara funkcije u smjeru negativnog gradijenta. Taj postupak naziva se **metoda gradijentnog spusta**. U svakom koraku ove metode računa se gradijent funkcije pogreške te se zatim, izmjenom parametara mreže, radi mali korak u smjeru negativnog gradijenta. Ovaj postupak se ponavlja do kad ne dobijemo dovoljno mali iznos funkcije pogreške [19,42].

Na Slici 18. prikazano je kako nam vektor (negativnog) gradijenta daje informaciju o tome koje parametre treba izmijeniti i u kojem smjeru, kako bi doprinos minimiziranju funkcije pogreške bio što veći. Magnituda svake komponente vektora gradijenta govori o tome koliko je funkcija pogreške osjetljiva na izmjenu te komponente, tj. koliko utječe na pomak prema minimumu, dok predznak govori o smjeru u kojem moramo napraviti izmjenu. Prikazan je i pojednostavljen primjer funkcije pogreške i smjer kojim idu koraci pri gradijentnom spustu. Naravno, stvarne funkcije pogreške u modelima uglavnom su složenije i zahtjevnije za izračune.

$$W = \begin{bmatrix} 3{,}56 \\ 1{,}27 \\ -1{,}56 \\ \vdots \\ 2{,}18 \\ -1{,}40 \\ 1{,}22 \end{bmatrix} \qquad -\nabla C(W) = \begin{bmatrix} 0{,}12 \\ -0{,}55 \\ 0{,}78 \\ \vdots \\ 0{,}30 \\ -0{,}19 \\ 0{,}65 \end{bmatrix}$$



Slika 18. Primjer vektora koji sadrži parametre mreže i pripadajućeg vektora gradijenta (gore) i primjer funkcije pogreške i smjera gradijentnog spuštanja (dolje) [autorske slike]

Algoritam koji omogućava učinkovit izračun gradijenata naziva se **povratna propagacija**. Naziv potiče od toga što se nakon unaprijedne propagacije i izračuna funkcije pogreške, ta pogreška propagira nazad kroz mrežu i kroz svaki neuron kako bi se pomoću nje prilagodili parametri mreže [20,42,51].

Na primjeru najjednostavnije mreže s jednim neuronom u svakom sloju može se dobro predstaviti algoritam povratne propagacije. U prvom koraku se na temelju poznatih parametara iz prethodnog sloja mreže, računa pogreška u neuronu izlaznog sloja (C_0) [42]:

$$C_0 = (a^{(L)} - y)^2 \tag{17}$$

Primjenom pravila za derivaciju kompozicije (lančano pravilo) računaju se derivacije potrebne za izračun gradijenta:

$$\frac{\partial C_0}{\partial w^{(L)}} = \frac{\partial z^{(L)}}{\partial w^{(L)}} \cdot \frac{\partial a^{(L)}}{\partial z^{(L)}} \cdot \frac{\partial C_0}{\partial a^{(L)}} \tag{18}$$

Uvrštavanjem Izraza 11, 12 i 15 slijedi:

$$\frac{\partial C_0}{\partial w^{(L)}} = a^{(L-1)} \cdot \sigma'(z^{(L)}) \cdot 2(a^{(L)} - y) \tag{19}$$

Kako bi se dobio prosjek svih primjera za obuku računa se suma:

$$\frac{\partial C_0}{\partial w^{(L)}} = \frac{1}{n}\sum_{k=o}^{n-1}\frac{\partial C_k}{\partial w^{(L)}} \tag{20}$$

Prikazan je izračun samo jedne komponente vektora gradijenta. Postupak je analogan i za ostale parametre mreže, pa vektor gradijenta izgleda:

$$\nabla C = \begin{matrix} \frac{\partial C_0}{\partial w^{(1)}} \\ \frac{\partial C_0}{\partial b^{(1)}} \\ \vdots \\ \frac{\partial C_0}{\partial w^{(L)}} \\ [\frac{\partial C_0}{\partial b^{(L)}}] \end{matrix} \tag{21}$$

Kod mreža s više neurona u slojevima izračun ostaje vrlo sličan. Ono što se mijenja je parcijalna derivacija C_0 po jednoj od aktivacija u prethodnom sloju, koja je u tom slučaju suma s obzirom na to da sada svaka aktivacija ima utjecaj na više neurona u promatranom sloju, koji imaju utjecaj na funkciju pogreške:

$$\frac{\partial C_0}{\partial a_k{}^{(L-1)}} = \sum_{j=0}^{n_L-1}\frac{\partial z_j{}^{(L)}}{\partial a_k{}^{(L-1)}}\cdot\frac{\partial a_j{}^{(L)}}{\partial z_j{}^{(L)}}\cdot\frac{\partial C_0}{\partial a_j{}^{(L)}} \tag{22}$$

Iz navedenih proračuna možemo vidjeti koliko je funkcija pogreške „osjetljiva“ na promjene težinskih faktora i pristranosti, ali i na promjene aktivacije iz prethodnog sloja ($a_k{}^{(L-1)}$). Iako na te aktivacije ne možemo direktno utjecati, možemo nastaviti primjenjivati lančano pravilo unazad kroz mrežu i vidjeti koliko je funkcija pogreške osjetljiva na promjene prethodnih težinskih faktora i pristranosti (koji utječu na $a_k{}^{(L-1)}$).

Nakon svakog izračuna gradijenta, radi se korak u smjeru minimuma funkcije pogreške, odnosno ažuriraju se težinski faktori i pristranosti.

5. Iteracija

Uzastopnim izvođenjem unaprijedne propagacije, izračuna funkcije pogreške, povratne propagacije i ažuriranja gradijenta, mreža uči minimizirati funkciju pogreške i poboljšati svoju izvedbu na podacima za obuku. Kriterij zaustavljanja može biti

maksimalan broj ponavljanja iste ili bliske vrijednosti, postizanje željene razine točnosti ili neki drugi kriteriji konvergencije.

6. Validacija i testiranje

Tijekom ili nakon obuke, izvedba mreže procjenjuje se na zasebnom validacijskom skupu podataka kako bi se odredila njezina sposobnost generalizacije. To pomaže provjeriti je li mreža naučila smislene obrasce iz podataka o obuci i je li potrebno dalje ažurirati parametre. Nakon toga, mreža se testira na zasebnom testnom skupu podataka kako bi se procijenila njezina izvedba na podacima koje još nije vidjela [52].

4.2. Tipovi dubokih neuronskih mreža

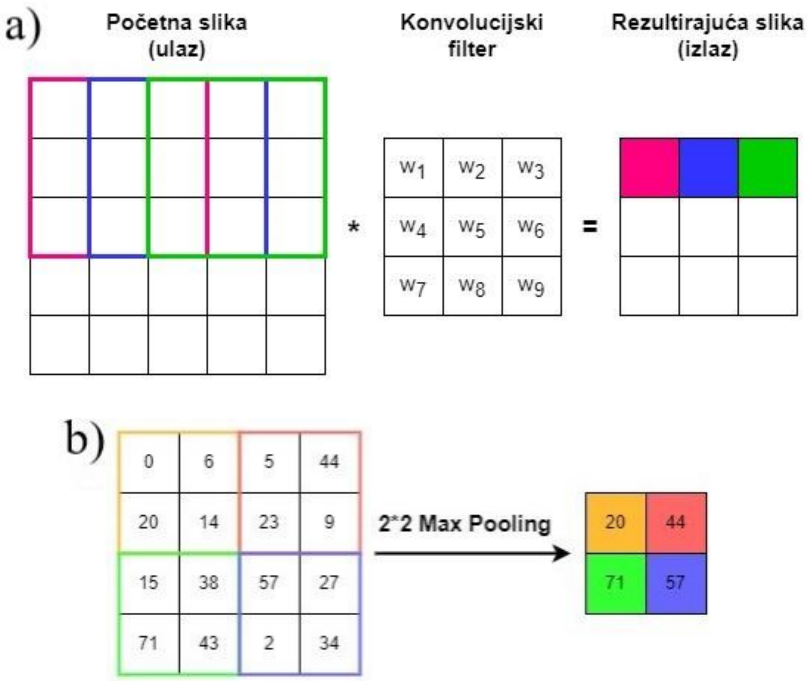
Prema arhitekturi mreže, odnosno prema načinu na koji su neuroni međusobno povezani i organizirani, možemo izdvojit na nekoliko najčešće korištenih tipova neuronskih mreža [42]:

4.2.1. Unaprijedne neuronske mreže

U prethodnim poglavljima princip rada neuronske mreže prikazan je na primjeru unaprijedne neuronske mreže. To je najjednostavniji tip mreže u kojem se ulazni podaci kreću od ulaznog sloja do izlaznog, bez povratnih veza prema neuronima iz prethodnih slojeva, a najčešće se koriste za zadatke kao što su klasifikacija i regresija. Arhitektura ovakve mreže već je objašnjena i prikazana na Slici 10.

4.2.2. Konvolucijske neuronske mreže

Konvolucijske mreže (eng. *Convolutional neural networks*) najsličnije su običnim (unaprijednim) mrežama. Koriste se za rad s podacima u obliku mreža, kao što su slike (mreža piksela), a razlika je u tome što sadrže konvolucijske slojeve. Unutar konvolucijskih slojeva se primjenjuju operacije konvolucije na ulazne podatke, što uključuje klizanje malog filtra preko ulazne mreže te množenje i zbrajanje po elementima (Slika 19.,a)). Ova operacija omogućava mreži izdvajanje lokalnih uzoraka i prostornih odnosa iz podataka. Uz konvolucijske, mreža sadrži i Pooling slojeve, koji se koriste za smanjenje dimenzija ulazne mreže (Slika 19.,b)). Time se smanjuje broj parametara koje mreža treba naučiti i količina računanja koja se u njoj izvode [53].



Slika 19. Operacije u slojevima konvolucijske neuronske mreže: a) Konvolucijski sloj, b) Pooling sloj [autorska slika]

Ove mreže obično se koriste u prepoznavanju slika, detekciji objekata i zadacima obrade prirodnog jezika [53].

4.2.3. Povratne neuronske mreže

Povratne neuronske (eng. *Recurrent neural networks*/RNN) mreže uglavnom su namijenjene za rad s nizovima podataka, kao što su tekstualni podaci i vremenski nizovi podataka. Takvi podaci razlikuju se od drugih vrsta podataka u smislu da se, dok se za sve značajke tipičnog skupa podataka može pretpostaviti da ne ovise o redoslijedu, to ne može pretpostaviti za ovakav skup podataka. Povratne neuronske mreže koriste povratne veze unutar mreže kako bi osigurale pamćenje prošlih informacija, tj. izlaz iz prethodnog koraka koristi se kao ulaz u trenutni korak, za razliku od već navedenih tipova mreža u kojima su svi ulazi i izlazi neovisni jedni o drugima [20,54].

Neki od zadataka za koje se ovakve neuronske mreže koriste su: jezično modeliranje i generiranje teksta, prepoznavanje govora, strojno prevođenje, prepoznavanje slika, prepoznavanje lica i predviđanje vremenskih nizova. Povratne neuronske mreže pronašle su primjenu u hidrološkom modeliranju zbog svoje sposobnosti rukovanja nizovima podataka i hvatanja vremenskih ovisnosti, koje su ključne za razumijevanje složene i dinamične prirode hidroloških sustava.

Arhitektura RNN

Glavna i najvažnija značajka RNN-ova je njihovo skriveno stanje. Skriveno stanje odnosi se na unutarnju memoriju mreže, koja hvata i pohranjuje informacije iz prethodnih koraka u nizu. Može se smatrati znanjem ili kontekstom koji RNN prenosi naprijed dok obrađuje niz podataka. U svakom vremenskom koraku (t) RNN uzima ulaz i ažurira svoje skriveno stanje na temelju trenutnog ulaza i skrivenog stanja iz prethodnog vremenskog koraka (t-1). Skriveno stanje u vremenskom koraku t tada se koristi za donošenje predviđanja odluka ili se prenosi na sljedeći vremenski korak [55].

Na Slici 20. prikazana je osnovna arhitektura RNN mreže koja obrađuje niz podataka ($x(t)=x_1, x_2,...x_\tau$) s vremenskim korakom od 1 do τ. Lijeva strana prikazuje kompaktni zapis arhitekture mreže, dok je na desnoj strani ona „odmotana“ kako bi se jasnije prikazao princip rada. Ulazni podaci (x) i pripadajući parametri prolaze kroz aktivacijsku funkciju (kao što je ReLU) te se računa skriveno stanje (h), prema izrazu [54,55]:

$$h_t = f_1(u \cdot x_t + w \cdot h_{t-1} + b_h) \tag{23}$$

Pri čemu je:

- h_t- skriveno stanje (aktivacija neurona)
- f_1 - aktivacijska funkcija
- x_t- ulaz u mrežu
- h_{t-1}- prethodno skriveno stanje
- u – težinski faktor veze ulaza i skrivenog stanja
- w – težinski faktor veze prethodnog i trenutnog skrivenog stanja
- b_h - pristranost

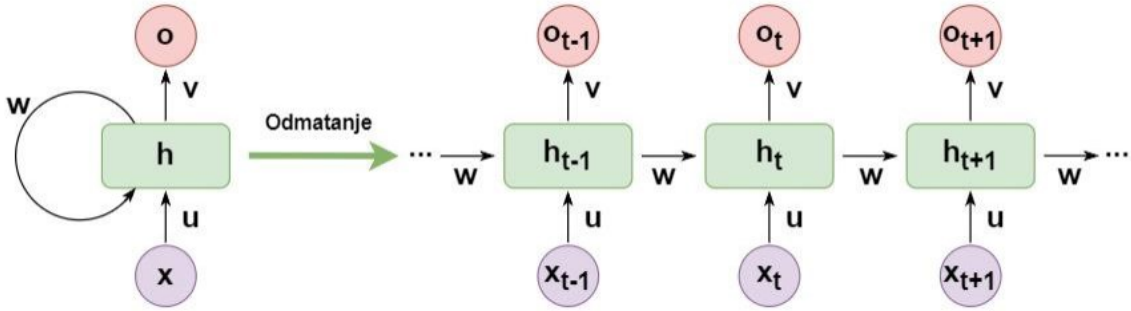
Ovakva mreža koristi iste težinske faktore za svaki element niza (da daje isti rezultat izvođenjem istog zadatka na svim ulazima ili skrivenim slojevima), smanjujući time broj parametara i dopuštajući modelu generalizaciju na nizove različitih duljina. Izlazi iz mreže računaju se prema izrazu [54–56]:

:

$$o_t = f_2(v \cdot h_t + b_o) \tag{24}$$

Pri čemu je:

- o_t- izlaz iz mreže
- f_2 - aktivacijska funkcija
- h_t- skriveno stanje
- v - težinski faktor veze skrivenog stanja i izlaza
- b_o – pristranost



Slika 20. Arhitektura RNN (prema: [57])

Učenje povratne neuronske mreže

Na primjeru unaprijedne mreže prikazano je kako neuronska mreža uči, pri čemu ključnu ulogu ima algoritam povratne propagacije. Proces učenja RNN je vrlo sličan, a razlika je u tome što ona za učenje koristi modificiranu varijantu tog algoritma – **povratnu propagaciju kroz vrijeme** (eng. *Backpropagation Through Time*/BPTT) [58].

U RNN-ovima skriveno stanje u svakom vremenskom koraku ovisi o trenutnom ulazu i skrivenom stanju iz prethodnog vremenskog koraka. Tako se varijable izračunavaju jedna po jedna, određenim redoslijedom (h_{t-1}, h_t, h_{t+1}, itd.). To stvara lančanu strukturu u kojoj se skrivena stanja povezuju tijekom vremena. Stoga se u BPTT povratna propagacija primjenjuje unazad kroz sva ta skrivena stanja uzastopno (Slika 21.). U tom slučaju vrijedi: C_t ovisi o varijabli h_t, koja ovisi o h_{t-1} itd., a svaka od tih varijabli je funkcija težinskog faktora w i prethodnog skrivenog stanja [55,58].

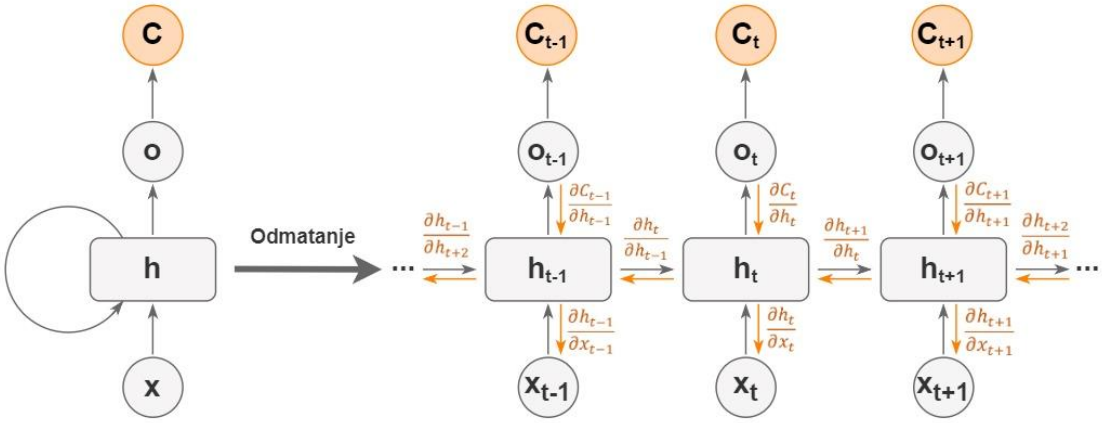
Izračun funkcije pogreške radi se za svaki vremenski korak posebno, te sumira:

$$\frac{\partial C}{\partial w} = \sum_{t=1}^{T} \frac{\partial C_t}{\partial w} \tag{25}$$

Proračun gradijenata radi se za svaki vremenski korak, prema lančanom pravilu:

$$\frac{\partial C_t}{\partial w} = \frac{\partial C_t}{\partial o_t} \cdot \frac{\partial o_t}{\partial h_t} \cdot \frac{\partial h_t}{\partial h_{t-1}} \cdot \frac{\partial h_{t-1}}{\partial h_{t-2}} \cdots \frac{\partial h_0}{\partial w} \tag{26}$$

Tako se BPTT razlikuje od tipične povratne propagacije samo u činjenici da se pogreške u svakom vremenskom koraku zbrajaju kako bi se izračunala ukupna pogreška.



Slika 21. Povratna propagacije kroz vrijeme u RNN [58]

Prednosti i nedostaci implementacije RNN

Ovisno o tipu podataka i zadatku koji mreža treba obavljati, povratne neuronske mreže imaju brojne prednosti u odnosu na unaprijedne mreže, a najvažnije od njih su [20,59]:

- Izvrsnost u obradi nizova podataka, kao što su vremenski ovisni nizovi, tekst i govor. Mogu uhvatiti vremenske ovisnosti i kontekst, što ih čini prikladnima za zadatke koji zahtijevaju razumijevanje takvih nizova.
- Unutarnja memorija - omogućava održavanje informacija iz prethodnih koraka, što ih čini odličnima za zadatke koji zahtijevaju razumijevanje povijesnih informacija.
- Mogućnost obrade ulaza različitih duljina - ova fleksibilnost osobito je korisna u zadacima poput obrade prirodnog jezika, gdje rečenice ili dokumenti mogu imati različite duljine.
- Mogućnost predviđanja vremenskih serija, kao što je predviđanje cijena dionica, vremena ili potražnje, gdje su povijesni obrasci i trendovi ključni za točna predviđanja.

Iako su RNN mnogo učinkovitije od unaprijednih mreža kad se radi o nizovima podataka, također imaju određene nedostatke odnosno ograničenja. Neki od njih su [60]:

- Računalna složenost - ove mreže mogu biti računalno skupe za obuku i zahtijevaju značajne resurse, za duge nizove podataka ili duboke arhitekture. Priroda RNN-ova

ograničava paralelizaciju, što može usporavati procesuiranje većih skupova podataka.

- Kratkoročno pamćenje - standardni RNN-ovi imaju ograničenja u zadržavanju informacija tijekom dugih razdoblja, što dovodi do neučinkovitog hvatanja odnosa ili ovisnosti između događaja koji su odvojeni značajnim brojem vremenskih koraka. Problem nastaje, između ostalog, zbog pojave nestajućeg gradijenta, a posebno je izražen kod strojnog prevođenja, prepoznavanja govora i predviđanja vremenskih nizova (gdje je hvatanje dugoročnih ovisnosti ključno za točna predviđanja).

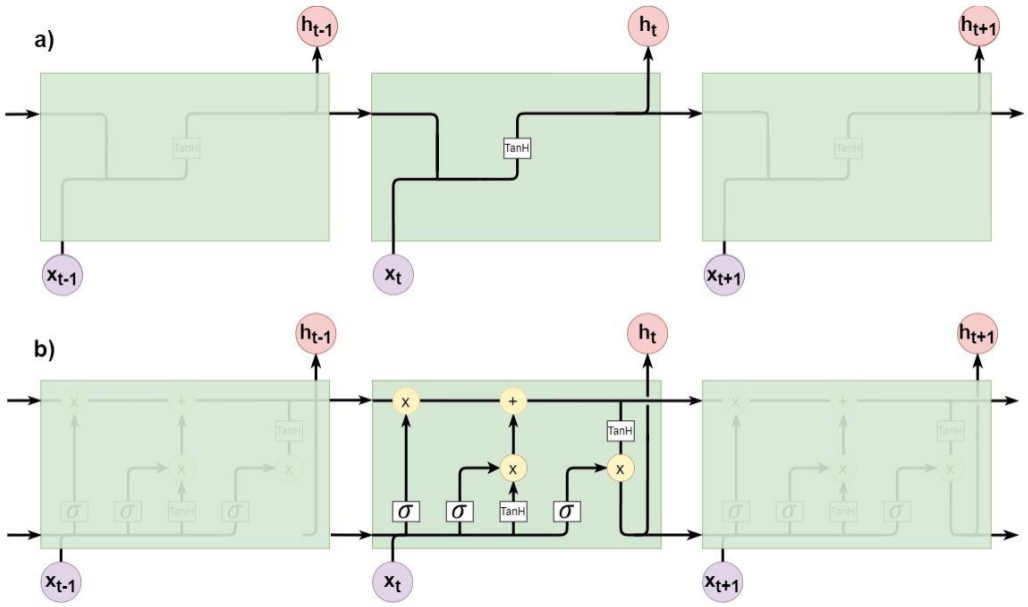
- Problemi s gradijentom:
    - **Eksplodirajući gradijent** (eng. *Exploding Gradient*) - gradijenti parametara modela postaju iznimno veliki tijekom povratne propagacije. Kada gradijenti postanu vrlo veliki, optimizacijski algoritam može premašiti optimalne vrijednosti parametara, uzrokujući preveliko ažuriranje težinskih faktora modela. To može dovesti do divergencije umjesto do konvergencije, što znači da se funkcija gubitka može povećavati umjesto da se smanji, a izvedba modela može se pogoršati, a ne poboljšati tijekom vremena. Ovaj problem često se javlja u dubokim neuronskim mrežama s mnogo slojeva, gdje se gradijenti mogu umnožiti kroz više slojeva, uzrokujući njihov eksponencijalan rast, kao i u RNN pri obradi dužih nizova podataka i korištenju ponavljajućih veza [61].
    - **Nestajući gradijent** (eng. *Vanishing Gradient*) - gradijenti parametara modela postaju vrlo mali tijekom povratne propagacije. Kada gradijenti postanu vrlo mali, algoritam optimizacije može raditi samo male korake tijekom ažuriranja težinskih faktora, što može dovesti do spore konvergencije, poteškoća u učenju smislenih reprezentacija u nižim slojevima, poteškoćama u hvatanju dugoročnih ovisnosti u nizovima podataka za RNN-ove ili čak potpuno zaustaviti proces učenja. Ovaj problem također se javlja u dubokim mrežama jer se gradijenti umnožavaju kroz više slojeva tijekom širenja unazad. Kao rezultat toga, gradijenti eksponencijalno padaju dok prolaze kroz mnoge slojeve, uzrokujući da raniji slojevi vrlo malo ili uopće ne ažuriraju svoje težinske faktore [61].

4.2.4. Mreže dugog kratkoročnog pamćenja

Kako bi se riješio problem nestajućeg gradijenta u tradicionalnim RNN-ovima i postiglo učinkovito hvatanje dugoročnih ovisnosti u sekvencijalnim podacima, razvijene su neke specijalizirane varijante povratnih mreža. Jedna od najpoznatijih je neuronska mreža dugog kratkoročnog pamćenja (eng. *Long Short Term Memory*/LSTM) [62,63].

Smisao LSTM mreže je postići da mreža može pamtiti mnogo podataka, određivati koji su podaci relevantni i njih zadržati, a zaboraviti one koji nisu. Na taj način smanjuje se broj izračuna koje mreža mora raditi i njezina zatrpanost nevažnim podacima, a povećava moć dugoročnog pamćenja i detektiranja obrazaca [62,63].

Usporedbom arhitekture RNN i LSTM mreže (Slika 22.), vidimo kako LSTM mreža također ima oblik lanca ponavljajućih modula. U standardnim RNN-ovima, ovaj ponavljajući modul će imati vrlo jednostavnu strukturu, kao što je jedan TanH sloj, dok kod LSTM mreže, umjesto jednog sloja, postoje četiri koja međusobno djeluju na poseban način. Glavna inovacija LSTM mreža je ugradnja posebnih memorijskih ćelija i mehanizama usmjeravanja koji omogućuju mreži da zadrži i kontrolira protok informacija tijekom vremena [62,63].



Slika 22. Usporedba arhitektura RNN (a) i LSTM (b) [62]

U usporedbi s RNN-om, LSTM kao dodatni ulaz u ćeliju, uz ulazni podatak u trenutnom vremenskom koraku (x_t) i skriveno stanje iz prethodnog koraka (h_{t-1}), imaju i stanje ćelije (C_{t-1}). Stanje ćelije je ono što djeluje kao dugoročna memorija ove mreže, koja može selektivno zapamtiti ili zaboraviti informacije na temelju ulaznih podataka i

prošlih stanja, što ga čini učinkovitim u hvatanju dugoročnih ovisnosti u nizovima. Ono je vektor koji prolazi kroz sve ćelije i kroz njih se mijenja, jer neke podatke zadržava, neke gubi, a neke nove podatke pohranjuje.

Kontrola protoka informacija kroz mrežu postiže se implementacijom struktura koje se nazivaju sklopovi (eng. *Gate*) te djeluju kao svojevrsni filteri. LSTM ćelija uključuje tri glavna sklopa (Slika 23.) [62,63]:

1. Forget gate

Ovaj dio mreže određuje koji podaci su manje ili više važni, odnosno koje podatke ili koliki dio podataka stanje ćelije zadržava, a što zaboravlja. Ulazni podatak (x_t) i skriveno stanje iz prethodnog vremenskog koraka (x_t), uz pripadajuće parametre (w_f, b_f) prolaze kroz sigmoidalnu aktivacijsku funkciju:

$$f_t = \sigma(w_f \cdot [h_{t-1}, x_t] + b_f) \tag{27}$$

Sigmoidalna funkcija brojeve preslikava u interval od 0 do 1, pa je rezultat navedene funkcije matrica (f_t) kojom množimo stanje ćelije iz prethodnog vremenskog koraka (C_{t-1}) i time filtriramo podatke:

$$C_t^f = C_{t-1} \cdot f_t \tag{28}$$

2. Input gate

Ovaj dio mreže određuje koji podaci, dobiveni od ulaznog podatka u trenutnom vremenskom koraku (x_t) i skrivenog stanja iz prethodnog koraka (h_{t-1}), će se pohraniti u stanje ćelije u trenutnom vremenskom koraku. Sastoji se od dva sloja. Prvi sloj sadrži sigmoidalnu funkciju i tvori matricu-filter sličnu vratima zaborava:

$$i_t = \sigma(w_i \cdot [h_{t-1}, x_t] + b_i) \tag{29}$$

U drugom sloju, primjenom pripadajućih parametara (w_C, b_C) i aktivacijske funkcije ($TanH$), dobivamo novo stanje ćelije (C_t'), odnosno dio koji će se prenijeti u stanje ćelije iz prethodnog koraka:

$$C_t' = TanH(w_C \cdot [h_{t-1}, x_t] + b_C) \tag{30}$$

Množenjem ova dva izraza, mreža odlučuje koje informacije je važno prenijeti u novo stanje ćelije (C_t^i):

$$C_t^i = C_t^{'} \cdot i_t \tag{31}$$

Tada se stanje ćelije u trenutnom vremenskom koraku (C_t^i) dobiva zbrajanjem:

$$C_t = C_t^{'} + C_t^i \tag{32}$$

3. Output gate

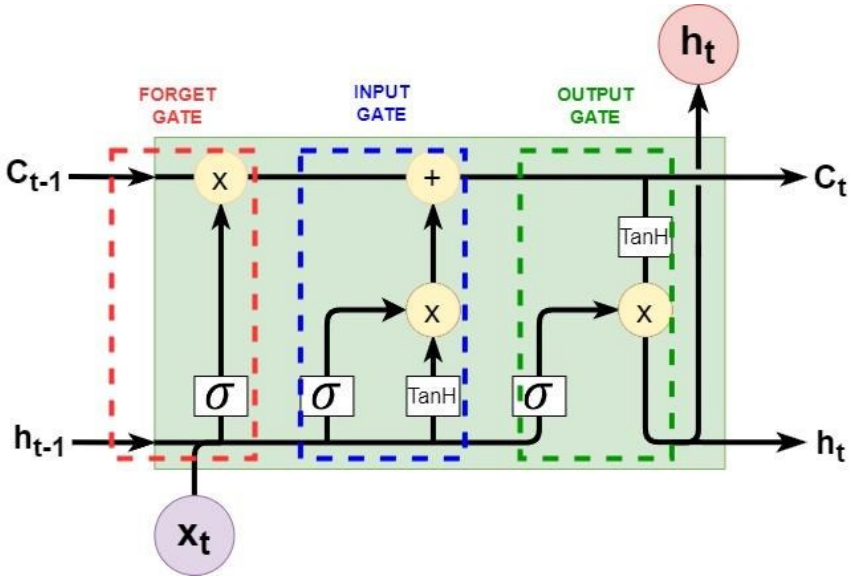
U ovom dijelu mreže računa se skriveno stanje u trenutnom vremenskom koraku (h_{t-1}). Primjenom pripadajućih parametara (w_o, b_C) i aktivacijske funkcije (σ) na ulazne podatke dobiva se izlazni filter (o_t):

$$o_t = \sigma(w_o \cdot [h_{t-1}, x_t] + b_o) \tag{33}$$

Na stanje ćelije primjenjuje se TanH funkcija, kako bi se vrijednosti postavile u interval od -1 do 1 te se ona množi filterom, kako bi se dobilo novo skriveno stanje:

$$h_t = o_t \cdot TanH(C_t) \tag{34}$$

Novo skriveno stanje koristi se kao ulaz u sljedeći vremenski korak, ali i kao izlaz iz mreže za trenutni vremenski korak.



Slika 23. Primjer ćelije LSTM mreže (prema: [62])

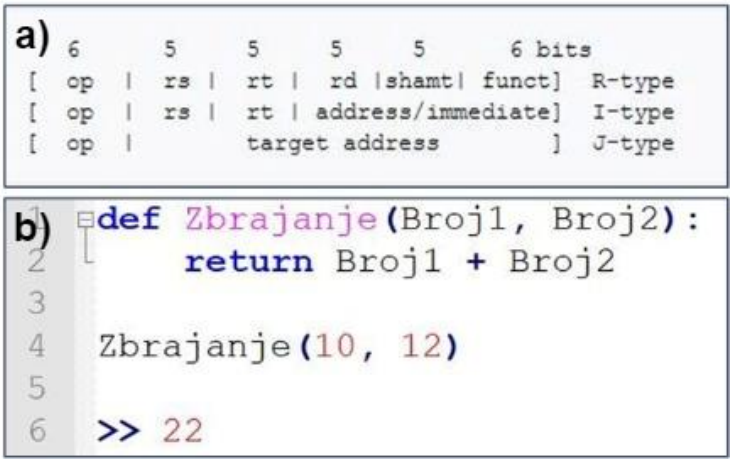
Ovdje je opisana standardna arhitektura LSTM mreže, no postoje različite varijante i modifikacije koje omogućavaju poboljšanja performansa mreže, ovisno o specifičnom zadatku i skupu podataka. Istraživači često eksperimentiraju s različitim arhitekturama i njihovim kombinacijama kako bi pronašli najprikladniji model za svoje primjene, pa se u mnogim radovima mogu pronaći primjeri različitih modificiranih verzija LSTM mreža [62].

4.2.5. Programski jezik Python

Programski jezik je formalni skup pravila i sintakse koji se koristi za komunikaciju s računalom i davanje uputa za obavljanje određenih zadataka. Služi kao posrednik između ljudi i strojeva, omogućujući programerima pisanje koda na jeziku koji računala mogu razumjeti i izvršiti. Programski jezici ključni su alati za razvoj softvera i programerima omogućuju stvaranje širokog spektra aplikacija, od jednostavnih skripti do složenih softverskih sustava. Programerima pružaju način da izraze svoje algoritme, logiku i manipulacije podacima na strukturiran i razumljiv način.

Programski jezici mogu se kategorizirati u dvije glavne vrste [64]:

- **Programski jezici niske razine** - bliži su hardveru i specifični su za arhitekturu određenog računala ili procesora. Sličniji strojnom kodu, nisu bliski ljudskom jeziku i ne pružaju apstrakciju (Slika 24., a)). Omogućuju izravniju kontrolu nad hardverom i često se koriste za programiranje na razini sustava, upravljačke programe uređaja i razvoj ugrađenih sustava. Primjer ovakvog jezik su strojni kod i Asembler.
- **Programski jezici visoke razine** - dizajnirani su da budu čitljiviji ljudima i bliži prirodnim jezicima, što programerima olakšava pisanje koda (Slika 24., b)). Jezici visoke razine obično apstrahiraju detalje specifične za hardver, čineći ih prenosivima na različite platforme i arhitekture. Primjeri jezika visoke razine uključuju Python, Java, PHP, C++, JavaScript, Ruby i mnogi drugi.



Slika 24. Primjer koda programskog jezika niske razine (a) i visoke razine (b) [64]

Svaki programski jezik ima svoje snage i slabosti, a izbor jezika ovisi o čimbenicima kao što su priroda projekta, ciljna platforma, zahtjevi za performansama i programerovo poznavanje i stručnost jezika.

Programiranje ima veliku ulogu u strojnom učenju jer je temelj za razvoj, obuku i implementaciju modela strojnog učenja. Algoritmi i modeli strojnog učenja implementiraju se pomoću programskih jezika, a razne biblioteke i okviri pružaju alate za rad s podacima, izradu modela i procjenu njihove izvedbe.

Jedan od najraširenijih programskih jezika korištenih za izradu modela pomoću neuralnih mreža je Python. Stvorio ga je Guido van Rossum i prvi put je objavljen 1991. godine [65]. Prema TIOBE Indeksu, Python se od 2003. godine nalazi među prvih deset najpopularnijih programskih jezika na svijetu, a od 2021. godine je na prvom mjestu, ispred C i C++, a poznat po svojoj jednostavnosti, čitljivosti i svestranosti [66].

Najvažnije karakteristike Pythona su [65]:

- **Jednostavan je za učenje i čitanje** – sintaksa je jasna i jednostavna, a često se uspoređuje s engleskim jezikom
- **Jezik opće namjene** - može koristiti za širok raspon aplikacija, kao što su web razvoj, analiza podataka, znanstveno računalstvo, automatizacija, umjetna inteligencija i drugo
- **Interpretirani jezik** - kod se izvodi redak po redak pomoću Python interpretera, što omogućava brzi razvoj i otklanjanje pogrešaka. Međutim, u usporedbi s prevedenim jezicima, može imati malo sporije vrijeme izvršenja.

- **Dinamičko tipkanje** - tipovi varijabli određuju se tijekom izvođenja, na temelju vrijednosti koja im je dodijeljena. To pruža fleksibilnost, ali može dovesti do mogućih pogrešaka povezanih s vrstom ako se s njima ne postupa pažljivo.
- **Opsežna standardna biblioteka** - pruža širok raspon modula i paketa za razne zadatke, što olakšava izvođenje uobičajenih operacija bez potrebe za dodatnim vanjskim bibliotekama.
- **Paketi i biblioteke trećih strana** - Python ima golemi sustav biblioteka i paketa trećih strana, koje proširuju mogućnosti Pythona za određene domene. Neke od njih, posebno one korisne kod modela s neuronskim mrežama, su NumPy, Pandas, Matplotlib, TensorFlow, Pytorch, itd.
- **Struktura s indentacijama i blokovima** - Python koristi indentacije za definiranje blokova koda, što poboljšava čitljivost koda. Ova je značajka jedinstvena za Python i često se doživljava kao prednost u odnosu na druge programske jezike.
- **Velika zajednica i dostupna podrška** - Python ima veliku i aktivnu zajednicu programera, što znači da je moguće pronaći opsežne resurse, dokumentaciju i pomoć na internetu.

Najnovija glavna verzija Python-a koja se preporuča za izradu novih projekata je Python 3. Navedeni su opći zahtjevi sustava za Python 3 [67]:

- Operativni sustav: Windows 7,10 ili 11, Mac OS X 10.11 ili više, 64-bit ili Linux: RHEL 6/7, 64-bit (gotovo sve biblioteke rade u Ubuntu)
- Procesor: x86 64-bit CPU (Intel / AMD arhitektura)
- Radna memorija: 4 GB RAM
- Slobodan prostor: 5 GB

5. IZRADA MODELA

Za izradu modela korišten je programski jezik Python 3, verzija 3.9.16. Modeli su kodirani u IDE-u (Integrated Development Environment) Spyder, verzija 5.4.3., te pokretani na računalu s AMD Ryzen 5 5500U CPU-om, 8 Gb RAM-a i integriranom AMD Radeon Graphics grafičkom karticom.

5.1. Korištene biblioteke

Velika prednost korištenja programskog jezika Python u izradi modela baziranih na neuronskim mrežama je dostupnost različitih biblioteka. Biblioteke su zbirke unaprijed napisanih kodnih modula koji pružaju različite funkcionalnosti i alate za pojednostavljenje procesa razvoja modela. One sadrže funkcije, klase i metode koje se mogu uvesti u kod, što omogućava korištenje određenih mogućnosti bez implementacije svega od nule. Biblioteke su osmišljene kako bi pomogle uštedjeti vrijeme, smanjiti složenost koda te izgraditi učinkovitije i robusnije modele, a pokrivaju širok raspon zadataka, od osnovnih operacija do složenih algoritama. U Pythonu je dostupno i nekoliko biblioteka dizajniranih specifično za izradu neuronskih mreža.

U izradi modela korištene su sljedeće biblioteke:

- **Numpy (Numerical Python)** - temeljna biblioteka u Python-u koja implementacijom širokog raspona matematičkih funkcija omogućava rad s višedimenzionalnim nizovima i matricama numeričkih podataka. Osnovna struktura podataka u NumPyju je ndarray (n-dimenzionalni niz) - homogen i višedimenzionalni niz koji može sadržavati elemente različitih tipova numeričkih podataka (integer, float, itd.). Nizovi u NumPy-ju mogu imati bilo koji broj dimenzija, što omogućava rad s podacima različitih oblika i veličina. Neke od njegovih mogućnosti su: matematičke funkcije (aritmetičke, trigonometrijske, logaritamske, eksponencijalne, množenje i transponiranje matrica itd.), statističke funkcije, „element-wise“ operacije, stvaranje, inicijalizacija, indeksiranje, rezanje, preoblikovanje nizova (https://numpy.org).
- **Pandas** - biblioteka otvorenog koda koja sadrži alate za manipulaciju podacima i analizu strukturiranih podataka. Izrađen je na temelju biblioteke NumPy i dizajniran za rukovanje dvjema primarnim strukturama podataka: Series i DataFrame. Pandas nije izravno povezan s neuronskim mrežama, ali igra ključnu ulogu u pretprocesiranju podataka i fazama upravljanja podacima u izgradnji i obuci neuronskih mreža. Neuronske mreže obično zahtijevaju da ulazni podaci budu dobro prethodno obrađeni i organizirani prije nego što se mogu učinkovito učiti. Neke od upotreba su: učitavanje i istraživanje podataka, predobrada podataka, stvaranje i izmjena postojećih značajki, razdvajanje podataka, formatiranje podataka, čišćenje podataka i kodiranje oznaka (https://pandas.pydata.org/).

- **PyTorch** - okvir za strojno učenje otvorenog koda koji pruža fleksibilan i dinamičan pristup izgradnji i obuci neuronskih mreža. PyTorch uvodi koncept tenzora, koji su višedimenzionalni nizovi slični NumPy nizovima, ali s dodatnim funkcijama optimiziranim za operacije dubokog učenja. Pruža klasu modula koja olakšava definiranje i organiziranje složenih arhitektura neuronskih mreža te uključuje opsežnu biblioteku unaprijed izgrađenih slojeva neuronske mreže, funkcija pogreške i optimizacijskih algoritama (kao što je gradijentno spuštanje). Također, Pytorch je interoperabilan s NumPy-em (https://pytorch.org/.
- **Scikit-learn** - biblioteka za strojno učenje otvorenog koja pruža sveobuhvatan skup alata za razne zadatke strojnog učenja, uključujući klasifikaciju, regresiju, grupiranje, smanjenje dimenzionalnosti i dr.
- **Matplotlib** - biblioteka za vizualizaciju podataka otvorenog koda za stvaranje statičnih, interaktivnih i animiranih vizualizacija u Pythonu. Omogućuje širok raspon funkcija za izradu različitih vrsta dijagrama, dijagrama, grafikona i drugih vizualnih prikaza podataka (https://scikit-learn.org/stable/).
- **Hydroeval** - biblioteka napravljena za evaluaciju hidroloških i hidrauličkih modela. Pruža zbirku metričkih podataka i alata za procjenu izvedbe modela koji se najčešće koriste u hidrologiji (https://pypi.org/project/hydroeval/).

Sve u nastavku opisane funkcije dostupne su na stranicama pripadajućih modula/biblioteka.

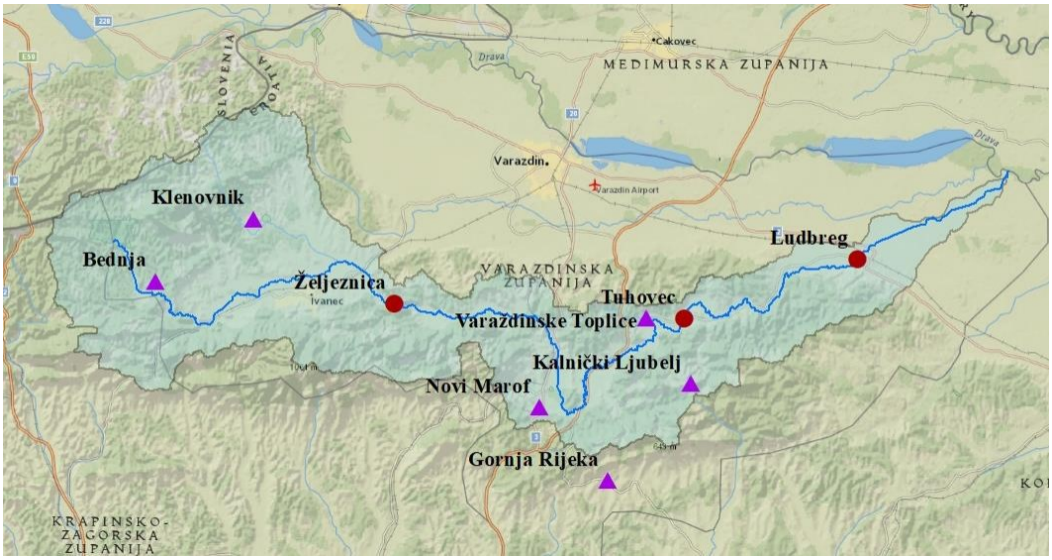
5.2. Ulazni podaci

Za izradu modela korišteni su podaci o količinama oborina [mm] izmjereni na šest meteoroloških postaja (tri kišomjerne i tri klimatološke) i podaci o dnevnim protocima [m^3/s] s tri hidrološke postaje. Sve meteorološke postaje, osim Gornje Rijeke, nalaze se na području sliva Bednje, dok se hidrološke postaje nalaze na vodotoku Bednja. Korišteni su podaci za razdoblje od 1.1.2010. do 31.12.2020. Svi podaci dobiveni su od Državnog hidrometeorološkog zavoda (DHMZ) u obliku Excel dokumenta za potrebe izrade diplomskog rada.

Podaci o postajama prikazani su u Tablici 3., a njihove lokacije označene su na karti na Slici 25.

Tablica 3. Podaci o meteorološkim i hidrološkim postajama

Postaja	Nadmorska visina [m]	Geografska širina	Geografska dužina	Godina početka rada	Tip postaje
Bednja	240	46° 14' 0"	15° 59' 0"	2006.	KMP
Gornja Rijeka	206	46° 06' 33"	16° 23' 34"	1954.	KŠP
Klenovnik	230	46° 16’ 23”	16° 04’ 15”	1962.	KŠP
Kalnički Ljubelj	320	46° 10’ 05”	16° 28’ 13”	1954	KMP
Novi Marof	200	46° 9' 17"	16° 19' 51"	1941.	KMP
Varaždinske Toplice	200	46° 12’ 41”	16° 25’ 39”	1952.	KŠP
Šifra postaje	**Postaja**	**Nadmorska visina [m]**	**Geografska širina**	**Geografska dužina**	**Tip postaje**
5089	Ludbreg	147,35	46°24'77.6"	16°61'96.06"	AD
5075	Željeznica	196,828	46°21'99.2"	16°20'02.4"	AD
5065	Tuhovec	162,85	46°21'04.8"	16°46'30.5"	AD



Slika 25. Lokacije meteoroloških i hidroloških postaja

5.3. Razvoj modela

Napravljen je hidrološki model baziran na neuronskoj mreži dugog kratkoročnog pamćenja (LSTM), kojemu je ulaz niz podataka o oborinama i protocima određene dužine (broj dana), a izlaz vrijednost protoka na određenoj hidrološkoj postaji za sljedeći dan. U nastavku je opisan proces izrade modela.

Predobrada podataka

Podaci su iz .csv dokumenta učitani u Pandas Dataframe. Iz Dataframe-a su izvađeni samo relevantni podaci za promatrano razdoblje i kronološki poredani (Slika 26.).

Datum	LUDBREG	Bednja	Gornja Rijeka	Klenovnik	Kalnički Ljubelj	Novi Marof	Vž Toplice	TUHOVEC	ŽELJEZNICA
2010-01-01	5.69	3.9	12	3.7	9	17.8	12.4	5.56	3.53
2010-01-02	5.77	0	0	0	0	0	0.9	5.18	3.9
2010-01-03	4.46	0.8	2	0.6	0	2.4	1.2	3.98	2.48
2010-01-04	3.57	0	0	0	0	0	0	3.35	2.1
2010-01-05	3.16	4.1	4.8	4.8	3.9	2	3.9	3.06	2.4

Slika 26. Prvih pet redaka Dataframe-a s ulaznim podacima

Skup podataka podijeljen je na ulazne podatke, X (eng. *Predictors*) i izlazne podatke, y (eng. *Targets*). Ulazne podatke čine oborine sa svih meteoroloških postaja i protoci s hidroloških postaja Tuhovec i Željeznica, a izlazne protoci na hidrološkoj postaji Ludbreg.

Sljedeći korak bio je skaliranje cijelog skupa podataka pomoću funkcije **StandardSkaler**. StandardScaler je jedna od metoda koje se koriste u predobradi podataka koju pruža sklearn.preprocessing modul biblioteke scikit-learn. Koristi se za standardizaciju ili normalizaciju značajki skupa podataka njihovim skaliranjem na način da imaju srednju vrijednost 0 i standardnu devijaciju 1. Ovo je osobito korisno kad u skupu podataka postoje značajke s različitim rasponima veličina. Takve skupove podataka poželjno je dovesti na jednu zajedničku skalu, kako bi se izbjegla pristranost algoritma prema značajkama s vrijednostima većih magnituda prilikom obuke mreže. Također, skaliranje značajki može poboljšati proces optimizacije čineći tijek spuštanja gradijenta glatkijim i pomažući algoritmima da brže dosegnu minimum funkcije pogreške.

Za obuku LSTM mreže nizove podataka potrebno je prevesti u odgovarajući oblik, tj. podijeliti u ulazne sekvence i odgovarajuće ciljne sekvence. Za to je definirana funkcija **split_sequences**(input_sequences, output_sequence, n_steps_in, n_steps_out). Parametri funkcije su ulazni (X) i izlazni (y) podaci te dužina niza ulaznih podataka i broj izlaznih podataka koji želimo da model predviđa.

Skup podataka zatim je podijeljen na tri podskupa podataka, važnih za obuku, podešavanje i procjenu izvedbe modela [52]:

- **Skup za obuku (trening)** - podskup skupa podataka koji se koristi za obuku parametara modela (težine i pristranosti). Tijekom procesa obuke, model uči davati predviđanja prilagođavanjem svojih parametara na temelju obrazaca i odnosa prisutnih u podacima obuke. Podaci o obuci obično čine većinu skupa podataka.
- **Skup za validaciju** - zasebni su podskup skupa podataka koji se ne koristi tijekom obuke, već se koristi se za podešavanje hiperparametara i praćenje performansi modela tijekom obuke. Procjenom izvedbe modela na podacima za validaciju mogu se donositi informirane odluke o prilagodbi hiperparametara kako bi se poboljšala sposobnost generalizacije modela. To pomaže u sprječavanju prekomjernog prilagođavanja i usmjerava proces obuke modela.
- **Skup za testiranje** - podskup skupa podataka koji se ne koristi ni tijekom obuke ni prilikom podešavanja hiperparametara. Koristi se za procjenu izvedbe modela nakon završetka obuke. Procjenom koliko dobro model radi na ovim podacima, može se dobiti uvid u njegovu sposobnost generalizacije na neviđene primjere i predviđanja u praktičnom kontekstu.

Omjeri podjele mogu varirati ovisno o veličini skupa podataka i količini hiperparametara, a tipično se radi o omjerima 60:20:20 ili 70:15:15 [52].

Za izradu ovog modela podaci su podijeljeni na:

- Obuka: 1.1.2010. – 31.12.2014.
- Validacija: 1.1.2015. – 31.12.2017.
- Testiranje: 1.1.2018. – 31.12.2020.

Temeljna struktura podataka u PyTorch-u su **tenzori**. U kontekstu strojnog učenja i dubokog učenja, tenzori su generalizacija skalara, vektora i matrica na više dimenzije. Oni su osnovni građevni blokovi za predstavljanje i manipuliranje podacima u numeričkim proračunima, a u PyTorchu se tenzori koriste za kodiranje ulaza i izlaza modela, kao i parametara modela. Stoga je zadnji korak predobrade podataka za model njihova pretvorba u **torch.Tensor**.

Definiranje klasa

U PyTorchu se neuronske mreže konstruiraju pomoću modula. Modul može predstavljati jedan sloj, grupu slojeva ili čak cijeli model. Osnovna klasa za sve module je **torch.nn.Module**. To je modul PyTorch-a koji sadrži alate i građevne blokove za stvaranje i obuku neuronskih mreža te uključuje širok raspon klasa i funkcija koje omogućavaju definiranje različitih vrsta arhitektura neuronskih mreža, slojeva, funkcija pogreške, optimizacijskih algoritama itd. Izvorni kod modula i njegovih komponenti dostupan je na: https://pytorch.org/

Kako bi se definirala arhitektura neuronske mreže kreirana je klasa **Bednja_LSTM**, koja nasljeđuje od nn.Module-a. Struktura mreže definirana je pomoću (Slika 27.):

- **__init__** metoda – konstruktorska metoda čija je svrha inicijalizacija atributa i svojstava objekta unutar klase
- **forward** metoda – metoda specifična za klase koje se koriste kao modeli na bazi neuronskih mreža, u njoj se definira algoritam unaprijedne propagacije, odnosno kako podaci prolaze kroz slojeve

Arhitektura modela sastoji se od sljedećih slojeva:

- nn.LSTM (input_size, hidden_size, num_layers) – klasa koja nasljeđuje od nn.Module-a te u sebi sadrži algoritme LSTM ćelije, a može sadržavati jedan ili više LSTM slojeva vezanih jedan na drugi (ovisno o parametru num_layers)
- nn.Linear (input_size, output_size) – potpuno povezani (eng. *fully stacked*) ili linearni sloj povezuje svaki neuron u trenutnom sloju sa svakim neuronom u sljedećem sloju. U ovakvom sloju ulazni podaci prolaze linearnu transformaciju, primjenom množenja matrica, nakon čega slijedi zbrajanje (pristranosti).
- nn.Linear – drugi linearni sloj, čiji su izlazi ujedno i rezultati modela

Između navedenih slojeva primijenjuje se aktivacijska funkcija ReLU, kako bi se u podatke uvela nelinearnost.

```python
class LSTM(nn.Module):

    def __init__(self, num_classes, input_size, hidden_size, num_layers):
        super().__init__()
        pl.seed_everything(42)
        self.num_classes = num_classes
        self.num_layers = num_layers
        self.input_size = input_size
        self.hidden_size = hidden_size
        # LSTM model
        self.lstm = nn.LSTM(input_size=input_size, hidden_size=hidden_size,
                            num_layers=num_layers, batch_first=True, dropout=0)
        self.fc_1 = nn.Linear(hidden_size, 64)
        self.fc_2 = nn.Linear(64, num_classes)
        self.relu = nn.ReLU()

    def forward(self,x):
        h_0 = Variable(torch.zeros(self.num_layers, x.size(0), self.hidden_size))
        c_0 = Variable(torch.zeros(self.num_layers, x.size(0), self.hidden_size))
        output, (hn, cn) = self.lstm(x, (h_0, c_0))
        output =  output[:, -1, :]
        out = self.relu(output)
        out = self.fc_1(out)
        out = self.relu(out)
        out = self.fc_2(out)
        return out
```

Slika 27. Arhitektura izrađenog modela u Python kodu

Za provođenje obuke i validacije modela definirana je pomoćna klasa **Optimization**, koja sadrži sljedeće funkcije:

- **train_val** – definira prolaz ulaznih podataka za obuku kroz unaprijednu propagaciju te primjenu metoda za izračun optimizacijskog algoritma (.optimizer.zero_grad()), funkcije pogreške (.loss_fn()), povratne propagacije (.backward()) te ažuriranje parametara modela (.optimizer.step()) te prolaz podataka za validaciju kroz model
- **evaluate** – omogućava prolaz skupa podataka kroz model te kao izlaz daje skup izmjerenih podatak i skup predviđenih (modeliranih) podataka. Služi za evaluaciju modela na skupu podataka za testiranje.

Dodatno je definirano nekoliko pomoćnih funkcija za: inverzno skaliranje izlaznih podataka, izradu grafova i izračun mjera procjene učinkovitosti modela.

Obuka mreže

Hiperparametri su parametri koji se postavljaju prije nego što se model počne učiti i ne mijenjaju se tijekom samog procesa obuke. Oni kontroliraju različite aspekte procesa obuke i utječu na to kako model uči iz podataka. Ispravno podešavanje hiperparametara

ključno je za postizanje optimalne izvedbe modela. U Tablici 4. prikazani su hiperparametri korišteni u ovom modelu te njihove vrijednosti nakon prilagođavanja.

Tablica 4. Hiperparametri modela

Hiperparametar	Opis	Model_1	Model_2
n_steps_in	broj prethodnih dana koji se koristi za predviđanje	30	30
n_steps_out	broj dana u budućnosti za koje model predviđa protok	1	1
num_classes	broj izlaznih klasa (protok na jednoj postaji)	1	1
input_size	broj ulaznih značajki	8	6
hidden_size	broj značajki (neurona) u skrivenim slojevima	14	12
num_layers	broj međusobno povezanih slojeva u nn.LSTM	2	2
n_epochs	broj epoha - iteracija kroz skup podataka tijekom obuke modela	600	600
learning_rate	veličina koraka tijekom svake iteracije procesa optimizacije	0,0005	0,0004

U procesu učenja mreže cilj je pronaći optimalan skup vrijednosti parametara koji dovodi do najboljeg mogućeg učinka modela, pomoću skupa podataka za obuku. **Optimizer** je algoritam ili metoda koja se koristi za ažuriranje parametara neuronske mreže tijekom procesa obuke kako bi se minimizirala funkcija pogreške [68].

Za izradu modela u ovom radu korišten je optimizer Adam (Adaptive Moment Estimation), koji je dostupan kao dio modula torch.optim u PyTorch-u. Adam kombinira prednosti dva druga popularna optimizacijska algoritma: AdaGrad (Adaptive Gradient Algorithm) i RMSProp (Root Mean Square Propagation), istovremeno uvodeći elemente koji pomažu u kontroli vrijednosti pristranosti u ranim fazama učenja. Glavna prednost Adam-a je prilagođavanje stopa učenja (eng. *learning rate*) za svaki parametar, uključivanjem prošlih gradijenata i kliznih prosjeka kvadrata gradijenata. Ova prilagodba omogućuje Adamu postizanje dobrih svojstava konvergencije na širokom rasponu funkcija [68,69].

Za izračun funkcije pogreške korištena je MSELoss (Mean Squared Error Loss), odnosno srednja kvadratna pogreška, koja je dostupna kao dio torch.nn modula. To je često korištena funkcija pogreške kod modela kod kojih je cilj predvidjeti kontinuirane

numeričke vrijednosti. MSE računa prosjek kvadrata razlika između predviđenih vrijednosti i ciljanih stvarnih vrijednosti.

Nakon definiranja modela, optimizera i funkcije pogreške model je prošao obuku primjenom funkcije **train_val** iz pomoćne klase **Optimization**:

- optimization = Optimization(model, loss_fn, optimizer)
- optimization.train_val(n_epochs, X_train_t, y_train_t, X_val_t, y_val_t)

Obuka modela završava kad nakon podešavanja hiperparametara, pri čemu se oslanjamo na iznose funkcije pogreške na skupu podataka za validaciju, on daje zadovoljavajuće rezultate. Rezultati modela primijenjena je **inverse.transform()** metoda, kako bi ih postavili na prvotnu skalu.

Nakon izrade Modela_1 s osam ulaznih značajki (oborine sa šest postaja i protoci s dvije postaje), za usporedbu je napravljen Model_2 koji za ulazne podatke uzima samo oborine s istih postaja. Kako bi se pokušalo postići bolje rezultate s manjim brojem ulaznih podataka, tijekom učenja mreže neki hiperparametri su izmijenjeni (Tablica 4.), dok je arhitektura mreže ostala ista.

Unutar klase **Bednja_LSTM** korištena je **seed_everything()** metoda, koja je dostupna iz pytorch_lightning.utilities.seed modula. Pomoću ove funkcije postavlja se početna vrijednost (eng. *seed)* za RNG-ove (Random Number Generator) u Pytorch-u, Numpy-u i Python-u, čime se postiže da oni pri svakom pokretanju koda generiraju iste „nasumične“ brojeve. Na taj način osigurava se da mreža svaki puta inicijalizira iste početne vrijednosti parametara i daje rezultate koji se uvijek mogu reproducirati.

Kompletan Python kod za implementaciju modela dostupan je u Prilogu.

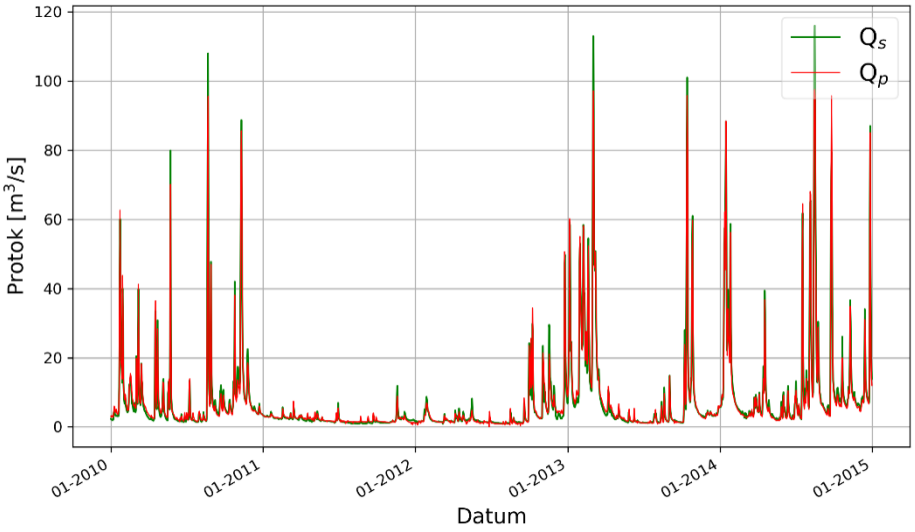
6. REZULTATI I RASPRAVA

6.1. Grafički prikaz rezultata modela

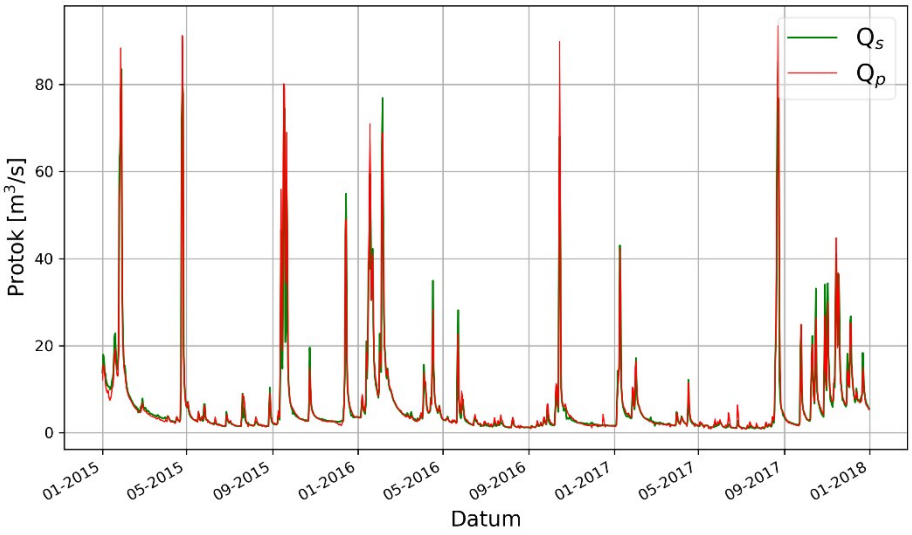
Nakon što je model završio s učenjem, njegova učinkovitost provjerena je na sva tri skupa podataka (obuka, validacija, testiranje) pomoću funkcije **evaluate**, te su rezultati grafički prikazani korištenjem funkcija iz **matplotlib.pyplot** modula.

Na Slikama 28.-30. prikazani su hidrogrami s predviđenim (Q_p) i izmjerenim stvarnim (Q_s) protocima na hidrološkoj postaji Ludbreg za Model_1. Na Slikama 31.-33. isti podaci

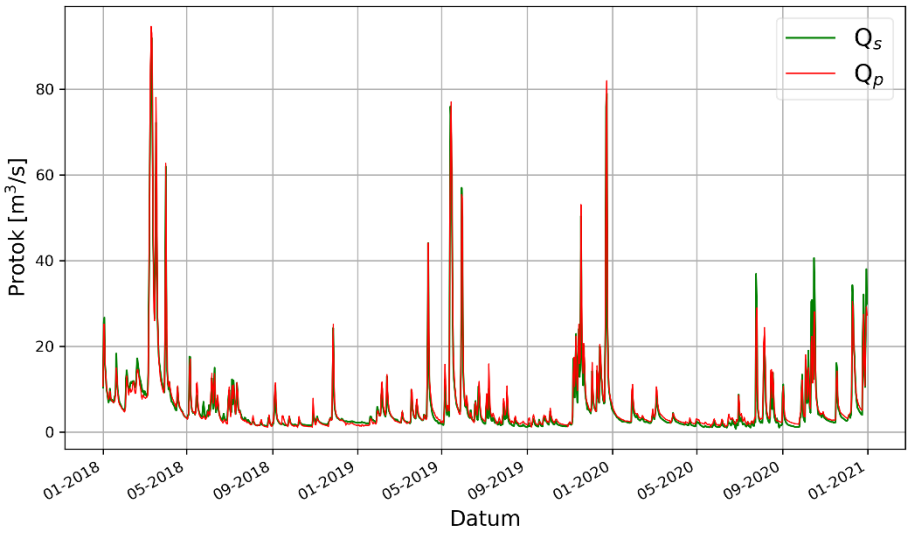
prikazani su za Model_2, koji u izračun ne uzima podatke s hidroloških postaja Tuhovec i Željeznica.



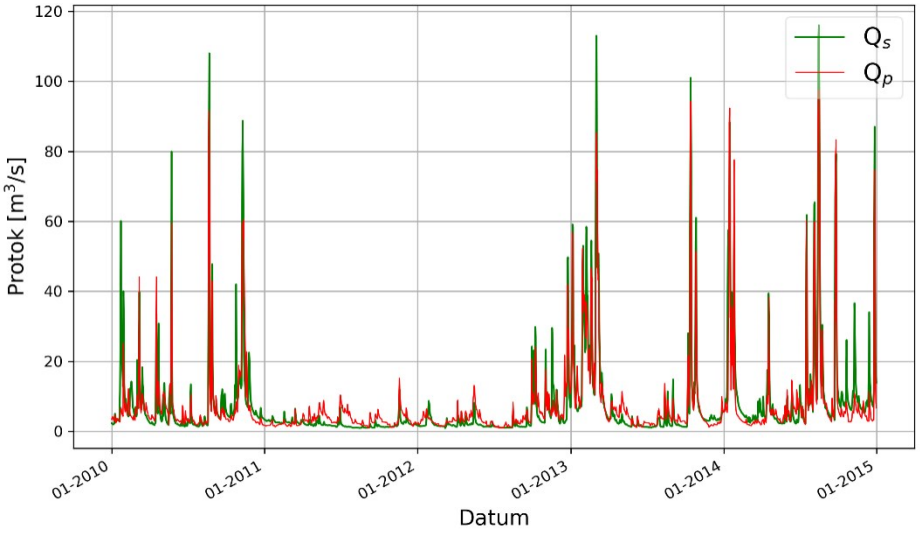
Slika 28. Model_1 – podaci za obuku



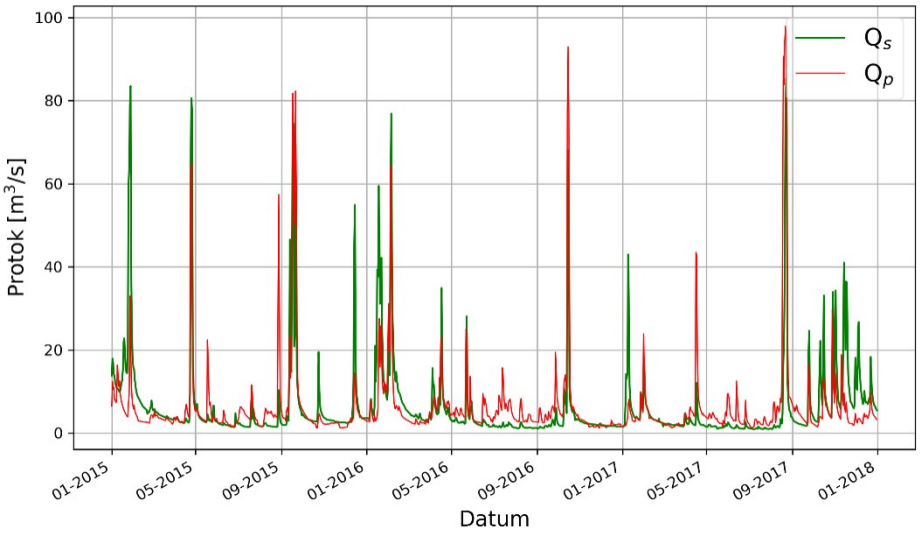
Slika 29. Model_1 – podaci za validaciju



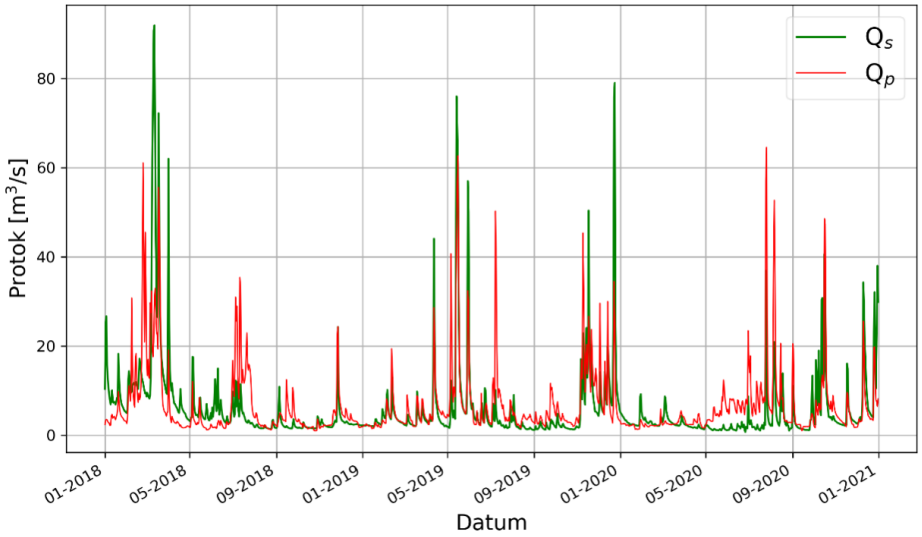
Slika 30. Model_1 – podaci za testiranje



Slika 31. Model_2 – podaci za obuku



Slika 32. Model_2 – podaci za validaciju



Slika 33. Model_2 – podaci za obuku

6.2. Mjere za procjenu učinkovitosti modela

Za ocjenu izvedbe i kvalitete modela koriste se različite kvantitativne mjere koje pomažu u procjeni učinkovitosti i prikladnosti modela za određeni zadatak. Za procjenu učinkovitosti izrađenih hidroloških modela izračunate su sljedeće mjere:

1) **Nash-Sutcliffe Efficiency (NSE)** – daje vrijednosti u rasponu od -∞ do 1, a vrijednost bliže 1 ukazuje na bolji model. Njime se kvantificira usklađenost između promatranih i modeliranih vrijednosti, uspoređujući varijabilnost modeliranih podataka s onom promatranih podataka. NSE pruža uvid u to koliko dobro hidrološki model reproducira vremenske varijacije i cjelokupno ponašanje promatranog toka. [70]

$$NSE = 1 - \frac{\sum_{i=1}^{n}(Q_{p,i} - Q_{s,i})^2}{\sum_{i=1}^{n}(Q_{s,i} - \bar{Q}_s)^2} \quad (35)$$

Pri čemu je:

Q_p - predviđene vrijednosti

Q_s - izmjerene vrijednosti

$\bar{Q}_s$ - aritmetička sredina izmjerenih vrijednosti

2) **Kling-Gupta Efficiency (KGE)** – daje vrijednosti u rasponu od -∞ do 1, a vrijednost bliže 1 ukazuje na bolji model. KGE pruža integriranu procjenu izvedbe hidrološkog modela, uzimajući u obzir različite aspekte modeliranog i

promatranog protoka. Smatra se da je model: vrlo dobar za KGE>0,9, umjereno dobar za 0,7<KGE<0,9 i ne zadovoljava za KGE<0,7 [71].

$$KGE = 1 - \sqrt{(r-1)^2 + \left(\frac{\sigma_p}{\sigma_s} - 1\right)^2 + \left(\frac{Q_p}{Q_s} - 1\right)^2} \quad (36)$$

Pri čemu je:

r - linearna korelacija između predviđenih i izmjerenih vrijednosti

σ_p- standardna devijacija predviđenih vrijednosti

σ_s- standardna devijacija izmjerenih vrijednosti

3) **Percent Bias (PBIAS) –** daje vrijednosti u rasponu od – ∞ do ∞, a vrijednost bliže 0 ukazuje na bolji model. Iznos PBIAS-s ukazuje na eventualno precjenjivanje (+ vrijednosti) ili podcjenjivanje (- vrijednosti) modela [72]

$$PBIAS = \frac{\sum_{i=1}^{n} Q_{s,i} - Q_{p,i}}{\sum_{i=1}^{n} Q_{s,i}} \cdot 100 \quad (37)$$

Rezultati proračuna za obje verzije modela navedeni su u Tablica 5 5.

Tablica 5. Vrijednosti mjera za procjenu modela za Model_1 i Model_2

Model_1	**Train**	**Val**	**Test**
KGE	0,975	0,950	0,938
PBIAS	-0,248	2,609	5,930
NSE	0,984	0,968	0,976
Model_2	**Train**	**Val**	**Test**
KGE	0,875	0,706	0,600
PBIAS	-0,621	-1,137	8,403
NSE	0,824	0,406	0,143

6.3. Interpretacija rezultata

MODEL_1

Ocjena izvedbe Modela_1 napravljena je na podacima za obuku, validaciju i testiranje, na način da su ulazni podaci provedeni kroz gotov model, kako bi se dobila predviđanja. Modelirani protoci, Q_p (na hidrološkoj postaji Ludbreg) prikazani su na grafovima,

zajedno sa stvarnim izmjerenim protocima, Q_s (Slika 28.-30.). Za kvantitativnu procjenu izvedbe modela izračunate su mjere za procjenu učinkovitosti modela i prikazani u Tablici 5.

Slika 28. prikazuje rezultate izvedbe modela na podacima za obuku. Vidljivo je odlično podudaranje modeliranih i izmjerenih protoka, što potvrđuju i izračunate mjere procjene, od kojih sve tri pokazuju gotovo potpuno podudaranje. Takvi rezultati su očekivani s obzirom da je model učio na ovom skupu podataka i već ih je „vidio“, te oni sami za sebe još ne govore dovoljno o učinkovitosti modela.

Kod modela baziranih na neuronskim mrežama postoji mogućnost **prekomjernog prilagođavanj**a (eng. *overfitting*). To je čest problem ovakvih mreža, kod kojeg model ima izuzetne rezultate na podacima za obuku, ali ne uspijeva generalizirati na nove, neviđene podatke. Drugim riječima, prekomjerno prilagođen model uči pamtiti podatke za obuku umjesto da hvata temeljne obrasce i odnose u njima. Kao rezultat toga, loše radi na podacima s kojima se prije nije susreo. Ovaj problem prepoznaje se po previsokom omjeru između (niske) prosječne pogreške na podacima za obuku i (visoke) pogreške na podacima za validaciju i/ili testiranje. Dva su pristupa rješavanju problema: obuka mreže na većem skupu podataka ili promjena kompleksnosti mreže (izmjenom strukture ili parametara modela) [73].

Suprotan problem koji se također ponekad javlja je **nedovoljno prilagođavanje** (eng. *underfitting*). To je situacija u kojoj je model previše jednostavan za hvatanje temeljnih obrazaca u podacima za obuku. Nedovoljno prilagođen model ima loše rezultate ne samo na podacima za obuku, već i na novim, neviđenim podacima. Ne uspijeva uhvatiti složenost odnosa između ulaznih značajki i ciljanih ishoda, što rezultira neoptimalnim predviđanjima. Ovaj problem relativno je lako riješiti na način da se, promjenom strukture modela, kao što je dodavanje više slojeva i/ili više neurona slojevima, poveća kompleksnost i kapacitet modela [73].

Slika 29. prikazuje rezultate izvedbe modela na podacima za validaciju. Podudaranje modeliranih i izmjerenih protoka i ovdje je odlično, što potvrđuju i mjere procjene. PBIAS iznosi 2,609 što ukazuje na blago precjenjivanje od strane modela. To je najviše uočljivo kod viših vršnih protoka, dok je kod baznog toka model pouzdaniji. Iako model nije učio na podacima za validaciju svejedno ih je već na neki način vidio. Iz tog razloga

najvažnija ocjena izvedbe modela je na skupu podataka za testiranje. Ona daje realnu mjeru koliko će model dobro funkcionirati na novim, dosad nepoznatim podacima.

Rezultati izvedbe modela na podacima za testiranje prikazani su na Slici 30. Pregledom grafičkog prikaza i izračunatih mjera procjene modela vidljivo je vrlo dobro podudaranje modeliranih i izmjerenih protoka. Iznos PBIAS-a od 5,930 pokazuje da model generalno blago precjenjuje iznose protoka, a odstupanja su najviša pri višim vršnim protocima. Pregledom podataka o oborinama za cijelo razdoblje od 1.1.2010. – 31.12. 2020., možemo pretpostaviti da je razlog tome općenito nešto niža godišnja količina oborina u razdoblju koje obuhvaćaju podaci za obuku, od one kod podataka na kojima je model učio. Iznos KGE od 0,938 ukazuje na dobru izvedbu modela (>0,9) i da model učinkovito bilježi vremenske varijacije, korelaciju i varijabilnost promatranog protoka. Iznos NSE od 0,976 potvrđuje vrlo dobru izvedbu modela.

MODEL_2

Model_2 izrađen je modifikacijom Modela_1. Iz ulaznih značajki uklonjeni su protoci s hidroloških postaja Željeznica i Tuhovec, te su kao ulazni podaci korištene samo oborine s šest meteoroloških postaja. Kako bi se izvedba modela pokušala približiti onoj Modela_1, određeni hiperparametri su izmijenjeni (Tablica 4.).

Na podacima za obuku i prema izračunatim NSE i KGE, Model_2 pokazuje relativno dobru izvedbu. PBIAS pokazuje minimalna odstupanja, no na grafičkom prikazu (Slika 31.) vidljiva su primjetna odstupanja, uglavnom pozitivna kod baznog toka i negativna pri vršnim protocima.

Na podacima za validaciju (Slika 32.) i testiranje (Slika 33.) model daje nezadovoljavajuće rezultate. Izmjenama hiperparametra modela i povećanjem njegovog kapaciteta (veći broj LSTM slojeva, veći broj neurona u slojevima,...) nije primijećeno veliko poboljšanje njegove izvedbe. S povećanjem složenosti dolazilo je i do prekomjernog prilagođavanja na podacima za testiranje, dok model na podacima za validaciju (i testiranje) nije postigao zadovoljavajuću učinkovitost.

Možemo zaključiti kako s ulaznim podacima koje čine samo oborine ne možemo postići dovoljno dobru izvedbu modela. Kod Modela_1 protoci s uzvodnih postaja omogućili su da model dobro uhvati bazni tok rijeke, a u kombinaciji s oborinama dobro predviđa i bazni tok i vršne protoke. Sami podaci o oborinama nedostatni su, zbog činjenice da je veći broj dana u godini bez oborina, te su iznosi ulaznih podataka za te

dane 0. U kombinaciji s bujičnim režimom Bednje, model teško te nule „pretvara“ u smislena predviđanja.

6.4. Rasprava

Neuronske mreže široko su prihvaćene zbog svojeg principa rada, koji u određenim aspektima nadilazi ograničenja tradicionalnih modela. Tradicionalni hidrološki modeli često se oslanjaju na dobro definirane fizikalne jednadžbe koje opisuju odnose između hidroloških varijabli na temelju fundamentalnih načela, dok su neuronske mreže modeli vođeni podacima, koji uče obrasce i odnose izravno iz parova ulaznih i izlaznih vrijednosti, pa ne zahtijevaju eksplicitne jednadžbe za predstavljanje temeljnih procesa. Također, tradicionalni modeli često u jednadžbama pretpostavljaju linearne odnose ili pojednostavljene linearne aproksimacije, koje možda neće adekvatno predstavljati složene nelinearne interakcije, dok su neuronske mreže sposobne uhvatiti složene nelinearne odnose unutar podataka, što ih čini prikladnima za modeliranje zamršenih hidroloških procesa. Tradicionalni modeli imaju unaprijed definiranu strukturu i pretpostavke, dok se neuronske mreže po tom pitanju fleksibilne što ih čini primjenjivim na različite skupove podataka.

Široku primjenu u modeliranju pronašle su povratne neuronske mreže, baš zbog činjenice da mogu učinkovito modelirati obrasce u vremenski ovisnim nizovima podataka, kao što su odnosi oborina i protoka, uzimajući u obzir prošla (skrivena) stanja. U hidrologiji se danas uglavnom koristi LSTM tip povratne mreže, koji rješava određene probleme RNN-ova kod nizova podataka s dugoročnim vremenskim ovisnostima.

U radu je predstavljen hidrološki model baziran na LSTM povratnoj mreži, relativno jednostavne arhitekture, koji je pokazao vrlo dobru izvedbu, odnosno hvatanje vremenskih ovisnosti, obrazaca i varijabilnosti, na neviđenom skupu podataka. Kombinacija oborina i protoka s uzvodnih postaja pokazala se kao dobar odabir ulaznih parametara za model. Protoci su „pomogli“ modelu da lakše uhvati bazni tok rijeke, dok su oborine utjecale na učinkovito hvatanje vršnih protoka. To je posebno vidljivo kod verzije modela koja je u izračun uzimala samo oborine, a kod kojeg ni kod prilagodbe hiperparametara nije postignuta dobra izvedba modela. Možemo pretpostaviti da bi se puno bolje rezultate moglo postići korištenjem, uz oborine, i dodatnih značajki koje imaju kontinuirano mjerene vrijednosti, primjerice evapotranspiracije ili razine podzemne vode.

Modeli bazirani na neuronskim mrežama po pitanju vrste ulaznih podataka prilično su fleksibilni, no važno je da između ulazni podataka i ciljanih vrijednosti postoje značajne veze i obrasci koje model može uhvatiti te da je dostupna količina podataka dovoljna za učenje mreže.

Dobro je napomenuti da kod neuronskih mreža postoji mogućnost primjene **prijenosnog učenja** (eng. *transfer learning*). To je metoda strojnog učenja kod koje se znanje stečeno obukom modela na jednom zadatku koristi za poboljšanje izvedbe modela s drugim zadatkom. Prethodno obučeni model, koji je naučio obrasce na velikom skupu podataka, koristi se kao početna točka u učenju drugog modela. Ova metoda korisna je kad se u novom modelu radi na malom skupu podataka, jer su podaci skupi ili je iz nekog drugog razloga do njih teško doći. [74]

Unatoč svojim prednostima, neuronske mreže također imaju i neka ograničenja. One su fleksibilne oko tipa ulaznih podataka, ali zato zahtijevaju robustan skup podataka. Dobivanje dugih i visokokvalitetnih skupova podataka u hidrologiji može biti vrlo teško, osobito za manje slivove ili područja s malim brojem mjernih postaja. Kod ograničenih skupova podataka mreže su sklone već spomenutom prekomjernom prilagođavanju. To su modeli s mnogo parametara, što ih može učiniti računalno skupima i zahtijevati značajne računalne resurse za obuku. Iako su LSTM mreže dizajnirane za hvatanje dugoročnih ovisnosti, još uvijek postoje ograničenja kod iznimno dugoročnih ovisnosti koje obuhvaćaju više godina. One, poput mnogih modela strojnog učenja, pretpostavljaju da je temeljna distribucija podataka stacionarna. U hidrologiji, gdje se klimatski obrasci i hidrološki procesi mogu mijenjati tijekom vremena, ova pretpostavka nije uvijek istinita.

Neuronske mreže, posebno LSTM, u svakom su slučaju moćan alati u hidrološkom modeliranju, ali hidrolozi koji ih koriste moraju biti svjesni njihovih ograničenja i kritički procijeniti je li ovakav pristup prikladan za određeni zadatak modela.

Pristup hidrološkom modeliranju prikazan u ovom radu pokazao se učinkovitim za rješavanje hidroloških zadataka, kao što je modeliranje protoka. Na temelju rezultata dobivenih modela, možemo pretpostaviti da bi se dodatan napredak mogao postići integracijom drugih relevantnih varijabli, izmjenom arhitekture mreže ili izradom hibridnih modela, koji kombiniraju povratne neuronske mreže s drugim metodama modeliranja.

7. ZAKLJUČAK

Hidrologija je znanost koja proučava pojavu, distribuciju i kretanje vode na Zemlji. Informacije dobivene analizom ponašanja i kretanja vode kroz hidrološki ciklus važne su za dobrobit ljudske populacije i okoliša. Utjecaj hidroloških mjerenja i analiza nastavlja rasti s utjecajem klimatskih promjena i sve većim izazovima u upravljanju vodnim resursima. Bez sumnje, jedan od glavnih izazova u vodnom gospodarstvu je predvidjeti buduće ekstremne hidrološke događaje i na vrijeme se pripremiti na njih.

Hidrološko modeliranje omogućava simulaciju i predviđanje složenih procesa u koji se odvijaju u hidrološkim sustavima. Stoga hidrološki modeli predstavljaju moćan alat koji pomaže u donošenju pravovremenih informiranih odluka o upravljanju vodnim resursima. Odabir vrste modela ovisi o ciljevima istraživanja, dostupnim podacima, računalnim resursima i karakteristikama hidrološkog sustava koji se modelira. U ovom radu predstavljen je hidrološki model sliva Bednje oborina-protok baziran na mreži dugog kratkoročnog pamćenja (LSTM), koja implementira specijalni tip arhitekture povratne neuronske mreže (RNN).

Ocjena izvedbe modela grafički i mjerama kvantitativne procjene pokazala je vrlo dobru izvedbu modela na skupovima podataka za obuku, validaciju i testiranje. Pravilnom prethodnom obradom podataka, dizajnom arhitekture modela i optimalnom obukom napravljena je neuronska mreža koja učinkovito hvata vremenske varijacije i obrasce, korelaciju i varijabilnost promatranog protoka.

U ovom radu je pokazano da LSTM mreže imaju znatan potencijal u hvatanju vremenskih ovisnosti i kompleksnih obrazaca svojstvenih hidrološkim procesima. Svojom sposobnošću učenja iz vremenski ovisnih nizova podataka i prilagođavanja promjenjivim uvjetima, ovakvi modeli nude vrijedan alat za povećanje točnosti i mogućnosti predviđanja hidroloških modela. Unatoč tome, važno je naglasiti da ovaj pristup nije bez ograničenja. Kvaliteta i kvantiteta ulaznih podataka, izbor hiperparametara i generalizacija modela u različitim hidrološkim kontekstima ostaju područja koja zahtijevaju stalna istraživanja i usavršavanja.

Kako klimatske promjene nastavljaju utjecati na obrasce oborina i hidrološke sustave, robusni i prilagodljivi okviri modeliranja postaju sve bitniji, stoga se u sve većem broju istraživanja nastoji iskoristiti potencijal umjetne inteligencije. Daljnja istraživanja mogla bi uključiti hibridne modele koji kombiniraju neuronske mreže s drugim tehnikama

strojnog učenja, kao i integraciju dodatnih varijabli okoline kako bi se poboljšala točnost predviđanja modela.

8. IZVORI

[1] Shaw, E. M. (1994): Hydrology in practice, 3rd edition, Chapman & Hall, London, 613 str.

[2] Davie, T. (2019): Fundamentals of Hydrology, London, Routledge, 306 str.

[3] Singh, A. (2018): A Concise Review on Introduction to Hydrological Models, *GRD Journal for Engineering*, 10 (1), str. 14–9,

[4] Devia, G. K., Ganasri, B. P., Dwarakish, G. S. (2015): A Review on Hydrological Models, *Aquatic Procedia*, 4, str. 1001–7, doi: 10.1016/j.aqpro.2015.02.126

[5] Chang, F. J., Chang, L. C., Chen, J. F.: Artificial Intelligence Techniques in Hydrology and Water Resources Management, *Water (Switzerland)*, 1510, str. 1–6, doi: 10.3390/w15101846

[6] IBM Data and AI Team: AI vs. Machine Learning vs. Deep Learning vs. Neural Networks: What's the difference?, URL:https://www.ibm.com/blog/ai-vs-machine-learning-vs-deep-learning-vs-neural-networks/ (1.7.2023.)

[7] Dawson, C. W., Wilby, R. L. (2001): Hydrological modelling using artificial neural networks, *Progress in Physical Geography: Earth and Environment*, 25 (1), str. 80–108, doi: 10.1177/030913330102500104

[8] Chow, Ven Te, Maidment, D. R., Mays, L.W. (1988): Applied Hydrology, New York, McGraw-Hill, 572 str.

[9] Allaby, M., Allaby, A. (1999): Dictionary of Earth Sciences, Oxford, Oxford University Press, 619 str.

[10] Sorooshian i sur. (2008): Hydrological Modelling and the Water Cycle: Coupling the Atmospheric and Hydrological Models, Springer Science & Business Media, 281. str

[11] Todini, E. (2007): Hydrological catchment modelling: Past, present and future, *Hydrology and Earth System Sciences*, 11 (1), str. 468–82, doi: 10.5194/hess-11-468-2007

[12] Todini, E. (2011): History and perspectives of hydrological catchment modelling, *Hydrology Research*, 42 (2–3), str. 73–85, doi: 10.2166/nh.2011.096

[13] Sidle, R. C. (2021): Strategies for smarter catchment hydrology models: incorporating scaling and better process representation, *Geoscience Letters*, 8 (1), doi: 10.1186/s40562-021-00193-9

[14] Cunderllk, J. M. (2003): Hydrologic model selection for the CFCAS project : Assessment of Water Resources Risk and Vulnerability to Changing Climatic Conditions

[15] Godara, N., Bruland O. (2019): Choosing an Appropriate Hydrologic Model, Department of Civil and Environmental Engineering NTNU

[16] McCarthy, J. (1985): What Is Artificial Intelligence Anyway, *American Scientist*, 73 (3), str. 258-64,

[17] Bolf, N. (2021): Strojno učenje, *Kemija u industriji*, 70 (9-10), str. 591–3,

[18] Ongsulee, P. (2018): Artificial intelligence, machine learning and deep learning, 15th International Conference on ICT and Knowledge Engineering, str. 1–6, doi: 10.1109/ICTKE.2017.8259629

[19] Dalbelo Bašić, B., Čupić, M., Šnajder, J. (2008): Umjetne neuronske mreže, Zavod za elektorniku, mikroelektroniku i inteligentne sustave, Fakultet elektrotehnike i računarstva

[20] ASCE Task Committee on Application of Artificial Neural Network in Hydrology. (2000): Artificial Neural Network in Hydrology. I:Priliminary Concepts, *Journal of Hydrologic Engineering*, 5 (2,) str. 115–23, doi: 10.1061/(ASCE)1084-0699(2000)5:2(

[21] Oyebode, O., Stretch, D. (2019): Neural network modeling of hydrological systems: A review of implementation techniques, *Natural Resource Modeling*, 32 (1) doi: 10.1111/nrm.12189

[22] Tanty, R., Desmukh, T. S. (2015): Application of Artificial Neural Network in Hydrology- A Review, *International Journal of Engineering Research & Technology*, 4 (6), str. 184–8, doi: 10.17577/ijertv4is060247

[23] French, M. N., Krajewski, W. F., Cuykendall, R. R. (1992): Rainfall forecasting in space and time using a neural network, *Journal of Hydrology*, 137 (1–4), str. 1–31, doi: 10.1016/0022-1694(92)90046-X

[24] Halff, A, Halff, H. M., Azmoodeh, M. (1993): Predicting Runoff from Rainfall Using Neural Networks, *Engineering Hydrology*, str. 760–5,

[25] Raman, H., Sunilkumar N. (1995): Multivariate modelling of water resources time series using artificial neural networks, *Hydrological Sciences Journal*, 40 (2), str. 145–63, doi: 10.1080/02626669509491401

[26] Barrati, R. i sur. (2003): River flow forecast for reservoir management through neural networks, *Neurocomputing*, 55 (3–4), str. 421–37,

[27] Londhe S., Charhate S. (2010): Comparison of data-driven modelling techniques for river flow forecasting, *Hydrological Sciences Journal*, 55 (7), str. 1163–74, doi: 10.1080/02626667.2010.512867

[28] Jain, A., Avadhnam, K. (2007): Hybrid neural network models for hydrologic time series forecasting, *Applied Soft Computing*, 7 (2), str. 585–92,

[29] Sivakumar, B., Jayawardena, A. W., Fernando, T. M. K. G. (2002): River flow forecasting: Use of phase-space reconstruction and artificial neural networks approaches, *Journal of Hydrology*, 265 (1-4), str. 225–245

[30] Petrić, H. (2011): O nekim nasljeima u porječju rijeke Bednje tijekom srednjeg i početkom ranog novog vijeka, *Kaj*, 94 (5), str. 57–68,

[31] Kos, Ž., Đurin, B., Dogančić, D., Kranjčić, N. (2021): Hydro-energy suitability of rivers regarding their hydrological and hydrogeological characteristics, *Water*, 13 (13), 1777, doi: 10.3390/w13131777

[32] Šimunić, A. (1986): Geološka građa gkolice Lepoglave i osvrt na pojave mineralnih sirovina, Radovi Zavoda za znanstveni rad Jugoslavenske akademije znanosti i umjetnosti, 1, str. 19–32

[33] Šimunić, A., Pikija, M., Hećimović, I. (1978): Osnovna geološka karta SFRJ 1:100.000 s tumačem - list Varaždin, Geološki zavod Zagreb, Zagreb

[34] Leskovar, K., Mrakužić, P., Meaški, H. (2021): Evaluation of remotely sensed precipitation product in a hydrological model of the Bednja watershed, *Građevinar*, 73 (4), str. 335–48, doi: 10.14256/JCE.3055.2020

[35] Počakal, M. (1982): Hidrografske veličine porječja Bednje, *Hrvatski geografski glasnik*, 44 (1), str. 85-101

[36] Čanjevac, I. (2013): Tipologija protočnih režima rijeka u Hrvatskoj, *Hrvatski geografski glasnik*, 75 (1), str. 23–42, doi: 10.21861/HGG.2013.75.01.02

[37] Zeman, S., Srpak, M. (2020): Exploitation of mineral resources and protection of drinking water reservoirs Varaždin county, IMR 2020 Interdisciplinary management research XVI, Opatija, Hrvatska

[38] Državni Hidrometeorološki Zavod - Mreža postaja, URL:https://meteo.hr/infrastruktura.php?section=mreze_postaja&param=pmm&el=glavne (14.7.2023.)

[39] Ivandić, N., Kunst, I., Telišman-Košuta, N., Marković, I.: Strategija razvoja turizma Varaždinske županije 2015.-2025., Zagreb, Institut za turizam

[40] Državni hidrometeorološki zavod (2021): Popis osnove mreže meteoroloških postaja

[41] Wood, T.: How similar are Neural Networks to our Brains?, URL:https://fastdatascience.com/how-similar-are-neural-networks-to-our-brains/ (20.7.2023.)

[42] Nielson, M. A. (2018): Neural Networks and Deep Learning, Determination Press

[43] Wu, Y. C., Feng, J. W. (2018): Development and Application of Artificial Neural Network, *Wireless Personal Communications*, 102 (2), str. 1645–56, doi: 10.1007/s11277-017-5224-x

[44] Werbos, P., John, P. (1975): Beyond regression: new tools for prediction and analysis in the behavioral sciences, Harvard University

[45] Šnajder, J. (2021): Materijali iz kolegija "Strojno učenje" za akad. god. 2021./2022., Fakultet elektrotehnike i računarstva

[46] Norvig, P., Russle, S. (2010): Artificial Intelligence: A Modern Approach, Pearson Education Limited, 1154 str.

[47] Sharma, S., Sharma, S., Anidhya, A. (2020): Activation Functions in Neural Networks, *International Journal of Applied Science and Technology*, 4 (12), str. 310–6,

[48] Es-sabery, F. i sur. (2021): Sentence-Level Classification Using Parallel Fuzzy Deep Learning Classifier, *IEEE Access*, 9 (Siječanj) , str. 17943–85, doi:

10.1109/ACCESS.2021.3053917

[49] Jadav, S. (2018): Weight Initialization Techniques in Neural Networks, URL:https://towardsdatascience.com/weight-initialization-techniques-in-neural-networks-26c649eb3b78 (23.7.2023.),

[50] Yin, L. (2017): A Walk-through of Cost Functions, URL:https://medium.com/machine-learning-for-li/a-walk-through-of-cost-functions-4767dff78f7 (25.7.2023.),

[51] Amari, S. (1993): Backpropagation and stochastic gradient descent method, *Neurocomputing*, 5 (4–5), str. 185–96, doi: 10.1016/0925-2312(93)90006-O

[52] Shah, T. (2017): About Train, Validation and Test Sets in Machine Learning, URL:https://towardsdatascience.com/train-validation-and-test-sets-72cb40cba9e7 (25.7.2023.)

[53] Li, Z., Liu, F., Yang, W., Peng, S., Zhou, J. (2022): A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects, *IEEE Transactions on Neural Networks and Learning Systems*, 33 (12), str. 6999–7019, doi: 10.1109/TNNLS.2021.3084827

[54] Medsker, L., Jain. L. C. (1999): Recurrent Neural Networks Design and Applications, Boca Raton, CRC Press LLC, 416 str.

[55] Kostadinov, S. (2017): How Recurrent Neural Networks work, URL:https://towardsdatascience.com/learn-how-recurrent-neural-networks-work-84e975feaaf7 (28.7.2023.)

[56] Debasis, K. (2022): A Brief Overview of Recurrent Neural Networks (RNN), URL:https://www.analyticsvidhya.com/blog/2022/03/a-brief-overview-of-recurrent-neural-networks-rnn/ (30.7.2023.)

[57] Recurrent Neural Networks Explanation, URL:https://www.geeksforgeeks.org/recurrent-neural-networks-explanation/ (1.8.2023.)

[58] Lillicrap, T. P., Santoro, A. (2019): Backpropagation through time and the brain, Curr. Opin. Neurobiol., 55, str. 82–9, doi: 10.1016/j.conb.2019.01.011

[59] Karpathy, A. (2015): The Unreasonable Effectiveness of Recurrent Neural

Networks, URL: http://karpathy.github.io/2015/05/21/rnn-effectiveness/ (2.8.2023.)

[60] Pascanu, R., Mikolov, T., Bengio, Y. (2012): On the difficulty of training recurrent neural networks, *Proceedings of Machine Learning Research*, 28 (3), str. 1310-1318, doi: 10.1007/978-3-319-93145-6_3

[61] Bohra, Y. (2021): The Challenge of Vanishing/Exploding Gradients in Deep Neural Networks, URL:https://www.analyticsvidhya.com/blog/2021/06/the-challenge-of-vanishing-exploding-gradients-in-deep-neural-networks/ (1.8.2023.)

[62] Olah, C. (2015): Understanding LSTM Networks, http://colah.github.io/posts/2015-08-Understanding-LSTMs/ (1.8.2023.)

[63] Yu, Y., Si, X., Hu, C., Zhang, J. (2019): A Review of Recurrent Neural Networks: LSTM Cells and Network Architectures, *Neural Computation*, 31 (7), str. 1235–70, doi: 10.1162/neco_a_01199

[64] PCChip (2020): Programski jezici visoke i niske razine, URL:https://pcchip.hr/helpdesk/programski-jezici-visoke-i-niske-razine/ (1.8.2023.)

[65] Tulchak, L.V., Marchuk, A. O. (2016): History of Python, Python Course

[66] TIOBE Index, URL:https://www.tiobe.com/tiobe-index/ (1.8.2023.)

[67] March, J. (2023): Enthought Python Minimum Hardware Requirements, URL:https://support.enthought.com/hc/en-us/articles/204273874-Enthought-Python-Minimum-Hardware-Requirements (1.8.2023.)

[68] Gupta A. (2021): A Comprehensive Guide on Optimizers in Deep Learning, URL:https://www.analyticsvidhya.com/blog/2021/10/a-comprehensive-guide-on-deep-learning-optimizers/ (4.8.2023.)

[69] Kingma, D., Ba, J. (2015): Adam: A method for stochastic optimization, Proceedings of the 3rd International Conference on Learning Representations (ICLR 2015)

[70] McCuen, R.H., Knight, Z., Cutter, A.G. (2006): Evaluation of the Nash–Sutcliffe Efficiency Index, *Journal of Hydrologic Engineering*, 11 (6), str. 597–602, doi: 10.1061/(asce)1084-0699(2006)11:6(597)

[71] Knoben, W.J.M., Freer, J.E., Woods, R.A. (2019): Technical note: Inherent benchmark or not? Comparing Nash-Sutcliffe and Kling-Gupta efficiency scores, *Hydrology and Earth System Sciences*, 23 (10), str. 4323–31, doi: 10.5194/hess-23-4323-2019

[72] Moriasi, D. N., Gitau, M. W., Pai, N., Daggupati, P. (2015): Hydrologic and water quality models: Performance measures and evaluation criteria, *Transactions of the ASABE*, 58 (6), str. 1763–85, doi: 10.13031/trans.58.10715

[73] Jabbar, H. K., Khan, R.Z. (2015): Methods to Avoid Over-Fitting and Under-Fitting in Supervised Machine Learning (Comparative Study), Computer Science, Communication & Instrumentation Devices, doi: 10.3850/978-981-09-5247-1_017

[74] Weiss, K., Khoshgoftaar, T. M., Wang, D. D. (2016): A survey of transfer learning, *Journal of Big Data*, 3 (9)

9. POPIS SLIKA

10. POPIS TABLICA

11. PRILOG

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
import numpy as np
import torch
import torch.nn as nn
from torch.autograd import Variable
import pytorch_lightning as pl
import time
from matplotlib.dates import DateFormatter
import hydroeval as he

# HYPERPARAMETERS
n_steps_in = 30
n_steps_out = 1
num_classes = 1
input_size = 8
hidden_size = 14
num_layers = 2
n_epochs = 600
learning_rate = 0.0005

# INPUT DATA
df = pd.read_csv("Bednja_podaci_csv")
df = df.drop(columns=['Unnamed: 0'])
df = df.iloc[1826:]

df_Datumi = pd.to_datetime(df["Datum"])

df.set_index('Datum', inplace=True)

  #MODEL_1
X, y = df.drop(columns=['LUDBREG']), df.LUDBREG.values

  #MODEL_2
#X, y = df.drop(['LUDBREG','TUHOVEC','ŽELJEZNICA'], axis=1),
df.LUDBREG.values

# SCALING
ss = StandardScaler()

X_trans = ss.fit_transform(X)
y_trans = ss.fit_transform(y.reshape(-1, 1))

# SPLIT SEQUENCES
def split_sequences(input_sequences, output_sequence, n_steps_in,
n_steps_out):
    X, y = list(), list()
    for i in range(len(input_sequences)):
        end_ix = i + n_steps_in
        out_end_ix = end_ix + n_steps_out - 1
        if out_end_ix > len(input_sequences): break
        seq_x, seq_y = input_sequences[i:end_ix],
 output_sequence[end_ix-1:out_end_ix, -1]
        X.append(seq_x), y.append(seq_y)
    return np.array(X), np.array(y)
```

```
X_ss, y_ss = split_sequences(X_trans, y_trans, n_steps_in,
n_steps_out)

# TRAIN, TEST, VAL -> TORCH TENSOR
X_train_t = Variable(torch.Tensor(X_ss[:1826]))
y_train_t = Variable(torch.Tensor(y_ss[:1826]))

X_val_t = Variable(torch.Tensor(X_ss[1826:2922]))
y_val_t = Variable(torch.Tensor(y_ss[1826:2922]))

X_test_t = Variable(torch.Tensor(X_ss[-1096:]))
y_test_t = Variable(torch.Tensor(y_ss[-1096:]))

# CLASSES AND FUNCTIONS
class Bednja_LSTM(nn.Module):

    def __init__(self, num_classes, input_size, hidden_size,
num_layers):
        super().__init__()
        pl.seed_everything(42)
        self.num_classes = num_classes
        self.num_layers = num_layers
        self.input_size = input_size
        self.hidden_size = hidden_size
        # LSTM model
        self.lstm = nn.LSTM(input_size=input_size,
hidden_size=hidden_size,
                            num_layers=num_layers, batch_first=True,
dropout=0)
        self.fc_1 = nn.Linear(hidden_size, 64)
        self.fc_2 = nn.Linear(64, num_classes)
        self.relu = nn.ReLU()

    def forward(self,x):
        h_0 = Variable(torch.zeros(self.num_layers, x.size(0),
self.hidden_size))
        c_0 = Variable(torch.zeros(self.num_layers, x.size(0),
self.hidden_size))
        output, (hn, cn) = self.lstm(x, (h_0, c_0))
        output =  output[:, -1, :]
        out = self.relu(output)
        out = self.fc_1(out)
        out = self.relu(out)
        out = self.fc_2(out)
        return out

class Optimization:
    def __init__(self, model, loss_fn, optimizer, ):
        self.model = model
        self.loss_fn = loss_fn
        self.optimizer = optimizer

        self.train_losses = []
        self.val_losses = []

    def train_val(self, n_epochs, X_train, y_train, X_val, y_val):
        for epoch in range(n_epochs):
            start_time = time.time()

            train_preds = self.model.forward(X_train)
            self.optimizer.zero_grad()
```

```
            train_loss = self.loss_fn(train_preds, y_train)
            train_loss.backward()
            self.optimizer.step()
            train_loss += train_loss.item()
            self.train_losses.append(train_loss)

            with torch.no_grad():
                model.eval()
                val_preds = self.model(X_val)
                val_loss = self.loss_fn(val_preds, y_val)
                val_loss += val_loss.item()
                self.val_losses.append(val_loss)

                elapsed = time.time() - start_time

                if epoch % 100 == 0:
                    print("Epoch: %d, train loss: %1.5f, val loss:
%1.5f, elapsed time: %1.5f" % (epoch,

train_loss.item(),

val_loss.item(),elapsed))

    def evaluate(self, X_test, y_test):
        with torch.no_grad():
            actual, predicted = [], []
            test_preds = model(X_test)
            test_loss = loss_fn(test_preds, y_test)
            actual += torch.squeeze(y_test[:, -
1]).data.cpu().numpy().tolist()
            predicted += torch.squeeze(test_preds[:, -
1]).data.cpu().numpy().tolist()
            return actual, predicted, test_loss

def inverse(X_t, y_t):
    actual, predicted, loss = optimization.evaluate(X_t, y_t)  actual
    = ss.inverse_transform(np.asarray(actual).reshape(-1,1))
    predicted = ss.inverse_transform(np.asarray(predicted).reshape(-
1,1))
    return actual, predicted, loss

def actual_DF(actual, Datum):
    actual_DF = pd.DataFrame(actual)
    actual_DF.columns = ["Actual"]
    actual_DF = actual_DF.set_index(Datum)
    return actual_DF

def predicted_DF(predicted, Datum):
    predicted_DF = pd.DataFrame(predicted)
    predicted_DF.columns = ["Predicted"]
    predicted_DF = predicted_DF.set_index(Datum)
    return predicted_DF

def plot(actual_DF, predicted_DF, name):
    fig, ax = plt.subplots(figsize=(10,6), dpi=600)  plt.plot(actual_DF,
    label='y_stv', color="green", linewidth=1.2) plt.plot(predicted_DF,
    label='y_pred', color="red", linewidth=0.8)
    plt.gcf().autofmt_xdate()
    date_form = DateFormatter("%m-%Y")
    ax.xaxis.set_major_formatter(date_form)
    plt.grid()
```

```
    plt.xlabel("Datum", fontsize=14)
    plt.ylabel("Protok [m$^3$/s]", rotation=90, fontsize=14)
    plt.legend(["Q$_s$", "Q$_p$"], fontsize="16", loc ="upper right",
framealpha=0.4)
    plt.title(name)
    plt.savefig(name + ".png")
    plt.show()
    return fig

def eval_metric(actual, predicted):
    kge, _, _, _ = he.evaluator(he.kge, actual, predicted)
    pbias = he.evaluator(he.pbias, actual, predicted)
    nse = he.evaluator(he.nse, actual, predicted)
    eval_metrics = {"KGE": kge, "PBIAS":pbias, "NSE":nse}
    eval_metrics = pd.Series(eval_metrics).astype(float)
    print (eval_metrics)
    return eval_metrics

def metrics(actual_train, predicted_train, actual_val, predicted_val,
actual_test, predicted_test):
    eval_metrics_train = eval_metric(actual_train, predicted_train)
    eval_metrics_val = eval_metric(actual_val, predicted_val)
    eval_metrics_test = eval_metric(actual_test, predicted_test)
    metrics = pd.concat([eval_metrics_train, eval_metrics_val,
eval_metrics_test], axis=1)
    metrics.columns = ["Train", "Val", "Test"]
    metrics.to_excel("Metrics.xlsx")
    return metrics

# TRAINING
model = Bednja_LSTM(num_classes, input_size, hidden_size, num_layers)
optimizer = torch.optim.Adam(model.parameters(), lr=learning_rate)
loss_fn = torch.nn.MSELoss()
optimization = Optimization(model, loss_fn, optimizer)
optimization.train_val(n_epochs, X_train_t, y_train_t, X_val_t,
y_val_t)

# TRAIN_GRAPH
actual_train, predicted_train, train_loss = inverse(X_train_t,
y_train_t)

Datum_train = df_Datumi[:1826]
actual_train_DF = actual_DF(actual_train, Datum_train)
predicted_train_DF = predicted_DF(predicted_train, Datum_train)

name = "Train"
fig = plot(actual_train_DF, predicted_train_DF, name)

# VAL_GRAPH
actual_val, predicted_val, val_loss = inverse(X_val_t, y_val_t)

Datum_val = df_Datumi[1826:2922]
actual_val_DF = actual_DF(actual_val, Datum_val)
predicted_val_DF = predicted_DF(predicted_val, Datum_val)

name = "Val"
fig = plot(actual_val_DF, predicted_val_DF, name)

# TEST_GRAPH
actual_test, predicted_test, test_loss = inverse(X_test_t, y_test_t)
```

```
Datum_test = df_Datumi[-1096:]
actual_test_DF = actual_DF(actual_test, Datum_test)
predicted_test_DF = predicted_DF(predicted_test, Datum_test)

name = "Test"
fig = plot(actual_test_DF, predicted_test_DF, name)

# METRICS
metrics = metrics(actual_train, predicted_train, actual_val,
predicted_val, actual_test, predicted_test)
```