# Predviđanje proizvodnje električne energije solarnih elektrana pomoću dubokog učenja

**Kinder, Irena**

SVEUČILIŠTE U ZAGREBU
**FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA**

DIPLOMSKI RAD br. 331

# PREDVIĐANJE PROIZVODNJE ELEKTRIČNE ENERGIJE SOLARNIH ELEKTRANA POMOĆU DUBOKOG UČENJA

Irena Kinder

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
**FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA**

DIPLOMSKI RAD br. 331

# PREDVIĐANJE PROIZVODNJE ELEKTRIČNE ENERGIJE SOLARNIH ELEKTRANA POMOĆU DUBOKOG UČENJA

Irena Kinder

Zagreb, lipanj 2024.

**SVEUČILIŠTE U ZAGREBU**
**FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA**

Zagreb, 4. ožujka 2024.

# DIPLOMSKI ZADATAK br. 331

| | |
|---|---|
| Pristupnica: | **Irena Kinder (0130320698)** |
| Studij: | Računarstvo |
| Profil: | Programsko inženjerstvo i informacijski sustavi |
| Mentor: | izv. prof. dr. sc. Marin Šilić |

Zadatak: **Predviđanje proizvodnje električne energije solarnih elektrana pomoću dubokog učenja**

Opis zadatka:

Proučiti i opisati trenutno stanje tehnologije dubokog učenja s posebnim naglaskom na najuspješnije metode za predviđanje tokova podataka i vremenske serije. Pronaći i prikupiti javno dostupne skupove podataka o proizvodnji električne energije solarnih elektrana i meteorološkim uvjetima za neko područje. Oblikovati i programski ostvariti sustav zasnovan na dubokom učenju za predviđanje proizvodnje električne energije solarnih elektrana iz povijesnih podataka o proizvodnji energije te podataka o meteorološkim uvjetima. Ispitati uspješnost ostvarenog sustava primjenom prikladno odabranih mjera te prikazati i opisati rezultate ispitivanja. Uz rad je potrebno predati i dokumentirati izvorni kod ostvarenog sustava, korištene skupove podataka te navesti korištenu literaturu.

Rok za predaju rada: 28. lipnja 2024.

**Sadržaj**

# Uvod

Posljednjih desetak godina, duboko učenje postalo je najbrže rastuće područje unutar strojnog učenja i šireg, sveobuhvatnijeg istraživačkog područja umjetne inteligencije. Sposobnost modela dubokog učenja da prepoznaje kompleksne uzorke prilikom obrade velike količine podataka primjenjiva je na brojne industrije, uključujući zdravstvo, energetiku i financije.

U prvom poglavlju ovog rada, Uvod u duboko učenje, područje dubokog učenja smješteno je unutar širih konteksta istraživanja. Nadalje, kroz povijesni kontekst pojašnjeni su ključni koncepti koji su utjecali na razvoj tehnologije dubokog učenja do stanja kakvo je danas. Zaključno, razmotreni su tehnološki napredci koji su omogućili značajan napredak ovog područja u posljednja dva desetljeća.

Drugo poglavlje, Oblikovanje algoritama dubokog učenja, uvod je u oblikovanje algoritama strojnog učenja, s naglaskom na algoritme dubokog učenja. U drugom dijelu poglavlja opisuje se glavni algoritam učenja dubokih mreža, poznat kao gradijentni spust.

U trećem poglavlju definirana je vremenska serija, opisane su njene komponenti i dana je njena matematička formulacija. Također, objašnjena je razlika između kratkoročnog i dugoročnog previđanja vremenskih serija.

Četvrto i peto poglavlje posvećeni su metodama dubokog učenja za predviđanje vremenskih serija. Metode su podijeljene na one prilagođene za kratkoročno predviđanje vremenskih serija, obrađene u poglavlju Kratkoročno predviđanje vremenskih serija, te na one koje mogu predviđati vremenske serije velikih duljina, opisane u poglavlju Dugoročno predviđanje vremenskih serija.

Tema posljednja dva poglavlja je predviđanje vremenske serije podataka o proizvodnji energije solarne elektrane. U poglavlju Predviđanje proizvodnje solarne energije iznesena je motivacija za predviđanje proizvodnje solarne energije, te su nabrojani izvori podataka potrebnih za predviđanje proizvodnje. U posljednjem poglavlju, Sustav za predviđanje proizvodnje solarne energije, opisan je postupak kojim je ostvareno predviđanje proizvodnje energije solarne elektrane.

# 1. Uvod u duboko učenje

Ovo poglavlje sadrži uvod u područje dubokog učenja. Prvo se razmatra koncept umjetne inteligencije kao najšireg okvira, nakon čega se fokusira na strojno učenje te na specifičnije područje učenja reprezentacija. Drugi dio poglavlja analizira razvoj dubokog učenja kroz tri ključne faze, pri čemu je posljednja faza još uvijek u tijeku. Također se razmatraju relevantni koncepti koji su se pojavili tijekom tih faza. Nadalje, opisuju se uvjeti koji su omogućili nedavni porast popularnosti dubokog učenja. U završnom dijelu poglavlja istražuju se različite primjene dubokog učenja.

## 1.1. Duboko učenje u kontekstu umjetne inteligencije

Početkom razvoja umjetne inteligencije brzo su se nalazila rješenja za probleme koji su ljudima intelektualno zahtjevni, ali računalima relativno jednostavni za rješavanje. Primjer takvog problema je šah, igra koja se može predstaviti skupom formalnih, odnosno matematičkih pravila. Pravi izazov za umjetnu inteligenciju je rješavanje druge vrste zadataka – onih koje ljudi obavljaju automatski, kao što je razumijevanje govora ili prepoznavanje lica. Takve probleme ljudi rješavaju intuitivno i teško ih je opisati formalno, odnosno pretočiti u skup pravila po kojima se izvode [1].

Svakodnevni život zahtijeva ogromnu količinu znanja o svijetu, od kojeg većina nije formalno opisiva i subjektivna je. Jedan od ključnih izazova umjetne inteligencije je kako neformalno znanje predati računalu [1].

Drugačiji pristup potreban je za rješavanje zadataka intuitivnih za obavljanje čovjeku. Umjesto definiranja znanja, računalu se može omogućiti da samo gradi znanje iz skupa podataka. Prednost ovog pristupa je što više nije potreban čovjek koji bi formalno definirao znanje o nekoj pojavi. Umjesto direktnog usađivanja znanja, računalo ga skuplja kroz iskustvo, učenjem o podacima. Sposobnost računala da samo gradi znanje učenjem uzoraka u podacima naziva se **strojno učenje** (engl. *machine learning*, ML), koje je ujedno naziv širokog područja umjetne inteligencije [1].

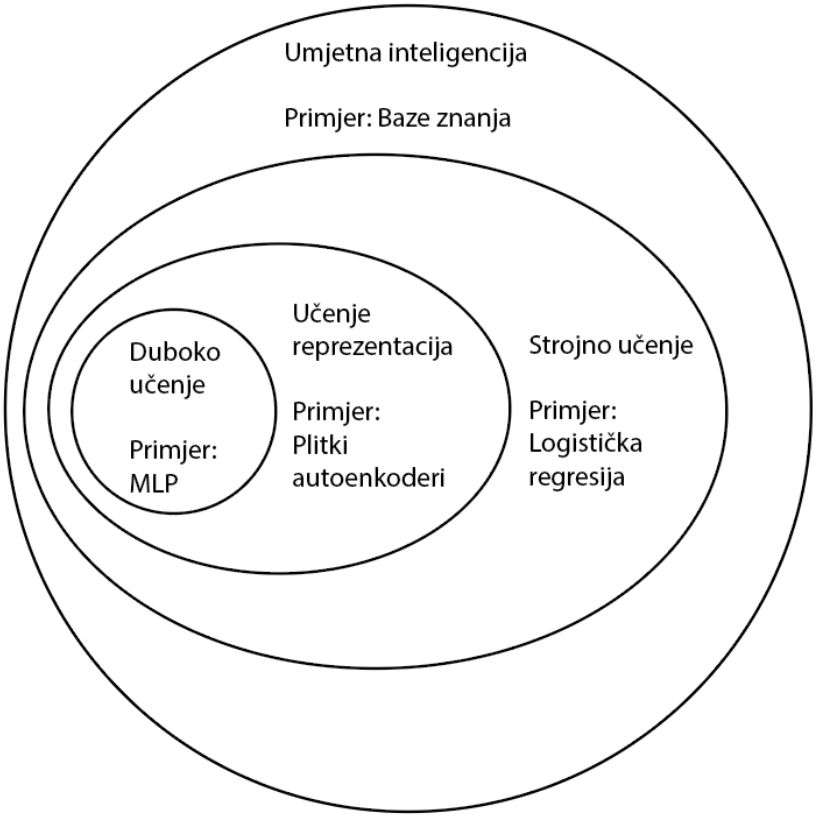
Kao u svakodnevnom životu, problemima se lakše pristupa kada su informacije koje imamo o problemu strukturirane i inteligentno organizirane. Uspješnost jednostavnih algoritama strojnog učenja uvelike ovisi o načinu reprezentiranja podataka za učenje. Primjerice, želi li se uz pomoć računala odrediti je li pacijent zaražen gripom, računalo ne pregledava pacijenta direktno, nego dobiva informacije o pacijentu kao što su: ima li pacijent temperaturu ili prisustvo kašlja. Svaka informacija koja je dio reprezentacije pacijenta naziva se **značajka**. Mnogi zadaci umjetne inteligencije rješivi su jednostavnim algoritmima strojnog učenja ako su podaci za učenje pažljivo pripremljeni i značajke odabrane za predstavljanje pojave korisne za učenje računala [1].

Ipak, definiranje niza značajki koje se trebaju ekstrahirati o nekoj pojavi nije uvijek jednostavno kao u opisanom slučaju detekcije gripe. U slučaju detekcije lica na slikama, kada bi se trebalo predstaviti nos kao značajku, bilo bi zahtjevno definirati njegovu reprezentaciju pomoću piksela. Učenje reprezentacije uz učenje preslikavanja reprezentacije u izlazni podatak, jedan je od načina rješavanja navedenog problema definiranja apstraktnih reprezentacija. Naziv ovog pristupa upravo je **učenje reprezentacija** (engl. *representation learning*). Definiranje apstraktnih reprezentacija može biti jednako zahtjevno kao početni problem koji se rješava i predstavlja centralni problem učenja reprezentacija [1].

Uži pristup strojnog učenja suočava se s glavnim problemom učenja reprezentacija – kako naučiti kompleksne reprezentacije podataka. Snažna ideja ključna za ovaj pristup je graditi složene reprezentacije podataka kombiniranjem onih jednostavnijih. Ovim pristupom računalo može izgraditi razumijevanje svijeta u obliku hijerarhije koncepata, gdje je svaki koncept definiran odnosom s manje kompleksnim konceptima nižima u hijerarhiji. Kada bi se grafom prikazalo naučene koncepte o nekom problemu, bio bi slojevit i dubok, odakle dolazi naziv ovog pristupa strojnog učenja, **duboko učenje** (engl. *deep learning*, DL) [1].

Zaključno, posebna vrsta strojnog učenja koja omogućuje računalima da uče iz podataka i postaju bolji s iskustvom je duboko učenje. Smatra se da je strojno učenje jedini način za oblikovanje sustava umjetne inteligencije primjenjivih na stvarna, složena okruženja. Snaga dubokog učenja je u gradnji slojevite hijerarhije koncepata koja omogućuje definiranje apstraktnih ideja pomoću onih manje apstraktnih. Ono što duboko učenje razlikuje od klasičnog strojnog učenja je količina

kompozicije naučenih koncepata o nekoj pojavi, odnosno naučenih funkcija za preslikavanje ulaznih u izlazni podatak. Odnos dubokog učenja s drugim područjima umjetne inteligencije prikazan je na slici dolje (Slika 1.1). Pokazalo se kao uspješan alat u mnogim područjima razvoja softvera kao što su računalni vid, obrada zvuka i govora, obrada prirodnog jezika, robotika, bioinformatika i kemija, video igre, pretraživači, internetsko oglašavanje i financije [1].




Slika 1.1 Vennov dijagram odnosa dubokog učenja s drugim disciplinama umjetne inteligencije [1]

## 1.2. Povijesni trendovi u dubokom učenju

Povijesni kontekst pomaže u razumijevanju dubokog učenja i omogućuje dobivanje potpunije i sveobuhvatnije slike o ovom području. Zbog nedavnog porasta popularnosti, podatak o godinama u kojima se javlja duboko učenje može biti iznenađujući. Duboko učenje ima dugu i bogatu povijest tijekom koje je bilo poznato pod različitim imenima koja odražavaju različite filozofske perspektive te je prolazilo kroz fluktuacije u popularnosti. S porastom računalne moći i dostupnosti podataka za učenje, postalo je sve korisnije u rješavanju sve složenijih problema uz porast

točnosti. Razvoj područja počeo je 1940-ih godina. Grubo definirano, prošlo je kroz tri značajne faze razvoja:

- Kibernetika (engl. *cybernetics*) 40-ih godina,
- Konekcionizam (engl. *connectionism*) 80-ih i 90-ih godina,
- Trenutni val razvoja pod imenom duboko učenje s početkom 2006. godine [1].

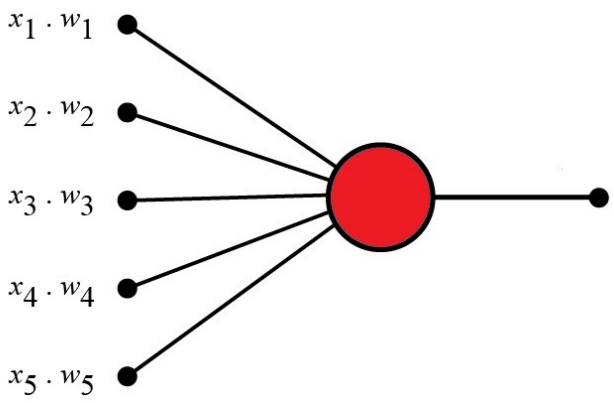
### 1.2.1. Prvi val razvoja

Neki od algoritama dubokog učenja koji su poznati od ranih dana razvoja područja, razvijali su se s namjerom računalnog modeliranja biološkog učenja. Nastojalo se modelirati kako se odvija, odnosno kako se može odvijati učenje u mozgu. Zbog toga je umjetna neuronska mreža (engl. *artificial neural network*, ANN) jedan od naziva pod kojim je duboko učenje poznato. Perspektiva na duboko učenje koju ima ova vrsta modela inspirirana je biološkim mozgom te iako se ponekad koristila za stjecanje razumijevanja o radu mozga, umjetne neuronske mreže nisu osmišljene kao njegova realna preslika. Dvije su motivacije za neuronsku perspektivu dubokog učenja. Prva motivacija je da je mozak dokaz inteligentnog ponašanja i njegov način rada može poslužiti kao inspiracija za modeliranje inteligencije. Ideja je reverznim inženjerstvom definirati računalnu pozadinu mozga i replicirati njegov rad. Druga motivacija je mogućnost da algoritmi strojnog učenja osim inženjerske primjene daju uvid u principe koji omogućuju inteligentno razmišljanje [1].

Međutim, moderno duboko učenje ide dalje od neuronske perspektive. Nije nužno inspirirano mozgom nego se oslanja na generalniju ideju učenja slojevite kompozicije koncepata. [1]

Preteča modernih modela dubokog učenja su jednostavni linearni modeli nastali pod utjecajem neuroznanstvene perspektive. Dizajn ovih modela je kao ulaz primati niz od $n$ ulaznih podataka $x_1, \ldots, x_n$ povezanih s nekih izlazom $y$. Izlaz modela računa se kao funkcija ulaza i naučenog skupa parametara, odnosno težina $w_1, \ldots, w_n$, $f(x, w) = x_1 w_1 + \cdots + x_n w_n$. Opisani modeli dio su prvog vala razvoja neuronskih mreža pod nazivom **kibernetika**. Prvi model koji je mogao naučiti težine

za model koji raspoznaje kategorije ulaznih podataka je model **perceptrona** (Slika 1.2). Osmislio ga je Rosenblatt 1950-ih godina.




Slika 1.2 Prvi model perceptrona [14]

U istom razdoblju Widrow i Hoff osmislili su tzv. adaptivni linearni element (ADALINE). Ovi jednostavni algoritmi usmjerili su razvoj modernog dubokog učenja. Algoritam učenja korišten za ažuriranje težina ADALINE-a posebna je vrsta **stohastičkog gradijentnog spusta** (engl. *stochastic gradient descent*, SGD), algoritma koji u neznatno modificiranoj verziji dominira među algoritmima učenja modernih dubokih modela [1].

Modeli ustanovljeni na preslikavanju $f(x,w)$ koje koriste perceptron i ADALINE nazivaju se **linearni modeli** (engl. *linear models*). Njihova primjena je dalje relevantna, uz izmjene u načinu treniranja u odnosu na originalne algoritme. Ipak, brojni su nedostaci linearnih modela od kojih je najpoznatiji nemogućnost modeliranja funkcije XOR, ujedno i izvor kritika prema biološki inspiriranom učenju modela te uzrok prvog pada popularnosti neuronskih mreža [1].

Neuroznanost nastavlja imati inspirativnu ulogu u razvoju područja dubokog učenja, iako se više ne smatra dominantnom smjernicom za istraživanje. Glavni razlog je nedovoljno razumijevanje rada mozga koje bi služilo kao vodič za daljnji razvoj algoritama. Osnovni koncept koji je inspirirala neuroznanost je da velika skupina računalnih jedinica može postati inteligentna kao rezultat njihovih međusobnih interakcija. Međutim, bez detaljnijeg uvida u principe biološkog učenja, neuroznanost ne može ponuditi inspiraciju van one prethodno navedene [1].

### 1.2.2. Drugi val razvoja

80-ih godina prošlog stoljeća počinje drugi val istraživanja u području neuronskih mreža naziva **konekcionizam**, poznat i pod nazivom **paralelno raspodijeljeno procesiranje**. Kontekst iz kojeg se javio ovaj pokret je kognitivna znanost, interdisciplinaran pristup istraživanja rada uma. Okosnica konekcionizma je ideja da velika količina jednostavnih računalnih jedinica može postići inteligentno ponašanje kada su umrežene, koncept koji se jednako odnosi na biološki živčani sustav i sustav sastavljen od umjetnih računalnih jedinica [1].

Nekoliko je koncepata koji su se pojavili 80-ih godina koji su i dalje relevantni za duboko učenje. Jedan od tih koncepata je **distribuirano reprezentiranje** (engl. *distributed representation*). Ideja distribuiranog reprezentiranja je ulaze u model reprezentirati velikim skupom značajki uz uvjet da svaka od značajki bude dio reprezentacije više različitih mogućih ulaza. Primjer koji ilustrira ovaj koncept je prepoznavanje primjerice stola, kuće i lopte koji mogu biti bijele, žute ili plave boje. Umjesto da postoje računalne jedinice od kojih svaka detektira neku od devet kombinacija predmeta i boja, distribuirano reprezentiranje predlaže model koji sadrži neurone zadužene za detekciju svake od tri navedene boje i neurone zadužene za prepoznavanje svakog od tri navedena predmeta. Jedna prednost ovog pristupa je manji broj neurona, u ovom slučaju smanjio se s devet na šest neurona. Druga prednost je što neuroni imaju više primjera za učenje jer nisu zaduženi samo za specifičnu boju ili specifični predmet kao što je npr. bijela kuća. Neuroni zaduženi za detekciju bijele boje uče na slikama bijelih predmeta, a neuroni zaduženi za detekciju kuća uče na slikama kuća svih različitih boja [1].

Drugi značajan koncept razvijen u ovom razdoblju je **učenje propagacijom unatrag** (engl. *back-propagation*), algoritam koji je unatoč usponima i padovima u popularnosti postao dominantan pristup treniranju dubokih modela [1].

Važni napredci napravljeni su tijekom 90-ih godina u modeliranju tokova podataka pomoću neuronskih mreža. Nakon što su početkom 90-ih prepoznati ključni matematički izazovi u modeliranju dugačkih tokova podataka, krajem desetljeća osmišljena je tzv. ćelija s dugoročnom memorijom (engl. *long short-term memory*, LSTM). LSTM mreže riješile su neke od tih problema i danas imaju široku primjenu u rješavanju problema tokova podataka.

Sredinom 90-ih godina područje ponovno dolazi do pada u popularnosti. S ciljem prikupljanja investicija, projekti koji su koristili neuronske mreže kao baznu tehnologiju davali su nerealna obećanja. Investitori razočarani nedostatkom rezultata prestaju ulagati u neuronske mreže. Uz to, druga područja umjetne inteligencije doživljavaju rast u razvoju poput jezgrenih strojeva i grafičkih modela. Stanka u razvoju neuronskih mreža trajala je do 2006. godine kada su se ponovno popularizirale [1].

### 1.2.3. Treći val razvoja

Do početka zadnjeg vala, veliki doprinos istraživanju neuronskih mreža dao je Kanadski institut za napredna istraživanja (engl. *The Canadian Institute for Advanced Research*, CIFAR). Multidisciplinarnom istraživačkom inicijativom NCAP (engl. *Neural Computation and Adaptive Perceptron*) okupilo se skupine istraživača u području strojnog učenja sa Sveučilišta u Torontu, Sveučilišta u Montrealu i Sveučilišta New York. U to vrijeme, treniranje dubokih mreža smatralo se vrlo računalno zahtjevnim zadatkom što je kočilo istraživanja. Tek 2006. godine postaje moguće koristiti algoritme poznate od 80-ih godina kada počinje posljednji val razvoja istraživanja o neuronskim mrežama [1].

Geoffrey Hinton, voditelj istraživačke skupine sa Sveučilišta u Torontu, 2006. godine osmislio je način učenja zvano pohlepno treniranje sloj po sloj (engl. *greedy layer-wise pretraining*) za treniranje posebne vrste neuronske mreže naziva **duboka probabilistička mreža** (engl. *deep belief network*, DBN). Druge istraživačke skupine povezane s CIFAR-om brzo pokazuju da se ova nova metoda može primijeniti i na druge algoritme dubokog učenja. Tijekom ovog razdoblja postalo je moguće trenirati modele sve većih dubina zbog čega se počinje koristiti naziv duboko učenje. Također, duboke mreže tada su počele pokazivati jače performanse od mnogih drugih algoritama strojnog učenja. Ovaj val razvoja nastavlja se do danas, uz promjenu u fokusu istraživanja u odnosu na početak vala. Početak je obilježilo istraživanje novih metoda nenadziranog učenja uz uspješno generaliziranje na malim skupovima podataka. Trenutno je fokus na puno starijim metodama nadziranog učenja i učenju na velikim skupovima podataka [1].

### 1.2.4. Uvjeti napretka

Nekoliko je ključnih napredaka u tehnologiji bilo ključno za uspon dubokog učenja. Bitan napredak je povećana dostupnost velikih skupova podataka za treniranje, potaknuta digitalizacijom društva i skladištenjem sve veće količine informacija u računala. Uz veću dostupnost, postalo je lakše prikupiti potrebne podatke za velike skupove prikladne za zadatke strojnog učenja [1].

Drugi ključan razlog uspješnosti neuronskih mreža danas je postojanje hardverskih resursa potrebnih za treniranje jako velikih mreža. Brži procesori, grafički procesori opće namjene (engl. *general-purpose graphics processing unit*, GPGPU), brži Internet i bolja infrastruktura za raspodijeljeno računanje omogućili su povećanje modela te se očekuje daljnji rast računalne moći u budućnosti [1].

# 2. Oblikovanje algoritama dubokog učenja

Ovo poglavlje bavi se oblikovanjem algoritama dubokog učenja. Prvi dio poglavlja služi kao podloga za razumijevanje postupka učenja dubokih modela. Navode se komponente algoritama strojnog učenja, koje su ujedno komponente algoritama dubokog učenja. Opisani su koncepti ključni za oblikovanje algoritama koji uspješno uče na ulaznim podacima, odnosno pospješuju sposobnost modela koja se zove generalizacija. U drugom dijelu poglavlja opisan je optimizacijski postupak korišten u algoritmima dubokog učenja.

## 2.1. Oblikovanje algoritama strojnog učenja

Da bi se steklo dobro razumijevanje dubokog učenja, potrebno je prvo objasniti temeljne principe strojnog učenja. Razlog tome je što su određene metode tradicionalnog strojnog učenja snažno utjecale na razvoj algoritama dubokog učenja [1].

Na ulazu algoritma strojnog učenja su **primjeri za učenje**. Kao što je spomenuto u prvom poglavlju, primjeri su reprezentirani skupom značajki. Formalni zapis primjera dan je izrazom (1).

$$\mathrm{x} = (\mathrm{x}_1, \mathrm{x}_2, \ldots, \mathrm{x}_\mathrm{n}) \tag{1}$$

Strojno učenje generalno se može podijeliti na **nadzirano** (engl. *supervised machine learning*) i **nenadzirano** (engl. *unsupervised machine learning*) učenje. Ako podaci za učenje imaju **oznaku** (engl. *label*) $\mathrm{y}$, učenje je nadzirano. Ako nemaju oznaku, učenje je nenadzirano. Svaki algoritam strojnog učenja čine tri komponente:

- Model,
- Funkcija pogreške,
- Optimizacijski postupak.

**Model** je skup funkcija definiranih velikih brojem parametara. Učenje je zapravo postupak pretraživanja tog skupa funkcija, odnosno traženje parametara funkcije koja najbolje modelira podatke za učenje. Model se može formalno definirati izrazom (2), gdje su $\theta$ parametri pojedine funkcije $\mathrm{h}$ koja se zove **hipoteza**.

$$H = \{h(x;\theta)\}_\theta \tag{2}$$

**Funkcija pogreške** je funkcija kojom se evaluira model. Može se općenito prikazati izrazom (3), gdje je $L$ tzv. **funkcija gubitka**. Funkcija gubitka za svaki od primjera za učenje računa pogrešku između očekivane vrijednosti primjera u odnosu na vrijednost koju procjenjuje model. Prosjek gubitaka svih primjera za učenje čini ukupnu pogrešku.

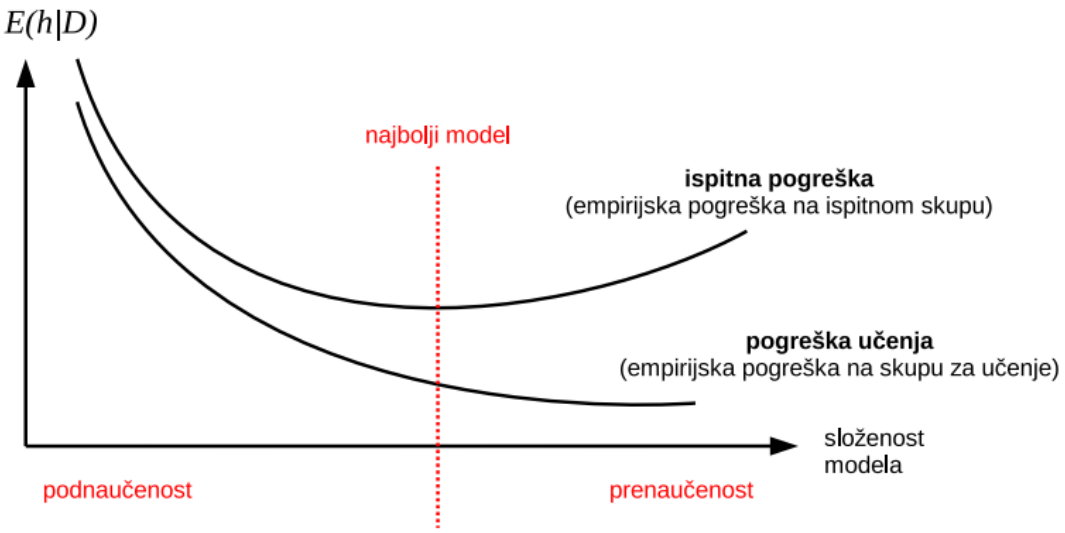
$$E(\theta|D) = \frac{1}{N}\sum_{i=1}^{N} L\left(y^{(i)}, h\left(x^{(i)};\theta\right)\right) \tag{3}$$

**Optimizacijski postupak** je postupak kojim se nalaze parametri funkcije koja najbolje aproksimira podatke, odnosno parametri funkcije za koju je ukupna pogreška modela najmanja. Optimizaciju se može predstaviti izrazom (4).

$$\theta^* = \arg\min_\theta E(\theta|D) \tag{4}$$

Modeli mogu biti proizvoljne kompleksnosti, ali je optimalna složenost ona za koju model dobro **generalizira**. Generalizacija je ključna sposobnost modela da radi dobro na primjerima koje nije vidio tijekom treniranja. Problem prejednostavnih modela je da ne mogu obuhvatiti kompleksnost podataka koji se žele modelirati. Opisana situacija naziva se **podnaučenost** (engl. *underfitting*). U slučaju presloženih modela, model se previše prilagođava primjerima za učenje i zbog toga puno griješi na podacima koji nisu bili uključeni u treniranje. Naziv ove situacije je **prenaučenost** (engl. *overfitting*) [8].

**Unakrsna provjera** metoda je kojom se može procijeniti koliko dobro model generalizira. Spomenuto je da je uvjet za dobru generalizaciju, dobar rad modela na neviđenim primjerima. Unakrsna provjera radi se podjelom skupa za učenje na skup za učenje, odnosno treniranje, i skup za provjeru, odnosno testiranje. Podjela se radi tako da nema preklapanja u skupovima, a česti omjeri podjele su 70:30 ili 60:40. Grafovi pogrešaka modela na skupu za učenje i na skupu za ispitivanje kao funkcija složenosti modela prikazana je na slici (Slika 2.1). Grafovi prikazuju idealizirani slučaj te u praksi kada se radi slučajnim uzorcima podataka je normalno da ispitna pogreška nakon padanja raste zatim ponovno pada, ili da je pogreška učenja za neke složenosti modela veća od ispitne pogreške [8].




Slika 2.1 Graf pogreške modela na skupu za učenje i na skupu za provjeru

Dizajniranje i treniranje neuronske mreže vrlo je slično treniranju bilo kojeg drugog modela strojnog učenja pomoću optimizacijskog postupka koji se zove **gradijentni spust** (engl. *gradient descent*) [1].

## 2.2. Gradijentni spust

Recept za oblikovanje algoritma strojnog učenja je kombiniranje modela, funkcije pogreške i optimizacijskog postupka. Optimizacija je postupak minimiziranja ili maksimiziranja funkcije $\mathrm{f}(\mathrm{x})$ promjenom varijable $\mathrm{x}$. U većini slučajeva optimizacijski problem se postavlja kao problem minimiziranja funkcije $\mathrm{f}(\mathrm{x})$. Maksimiziranje funkcije $\mathrm{f}(\mathrm{x})$ može se pretvoriti u minimiziranje ako se problem postavi kao minimiziranje funkcije $-\mathrm{f}(\mathrm{x})$. Generalno, funkcija koja se minimizira ili maksimizira naziva se **kriterijska funkcija** (engl. *criterion*).

U slučaju minimizacije, naziva se funkcija cijene (engl. *loss function*). Drugi naziv za funkciju cijene je funkcija pogreške, što je također naziv za funkciju koja se koristi za evaluaciju modela strojnog učenja. Vrijednost za koju je funkcija minimalna može se definirati izrazom (5) [1].

$$\mathrm{x}^* = \arg\min \mathrm{f}(\mathrm{x}) \tag{5}$$

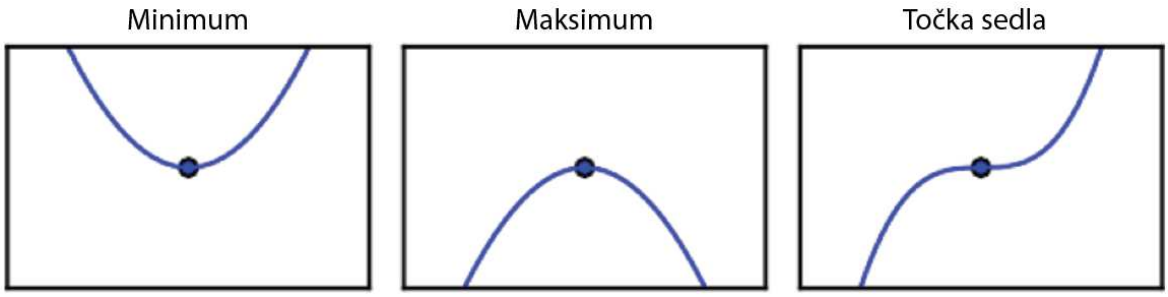
Za funkciju $\mathrm{y} = \mathrm{f}(\mathrm{x})$ derivacija se označava s $\mathrm{dy}/\mathrm{dx}$ i njena vrijednost odgovara nagibu funkcije $\mathrm{f}(\mathrm{x})$ u točki $\mathrm{x}$. Drugim riječima, derivacija specificira kako skalirati

malu promjenu ulaza da bi se dobila odgovarajuća promjena izlaza. Ova ideja može se matematički zapisati izrazom (6).

$$f(x + \epsilon) \approx f(x) + \epsilon f'(x) \tag{6}$$

Derivacija funkcije korisna je informacija za minimiziranje funkcije jer sadrži informaciju o tome kako treba promijeniti $x$ da bi se napravio pomak u $y$ prema minimumu funkcije. Dakle, $f(x)$ može se smanjiti pomacima u $x$ u smjeru suprotnom od smjera gradijenta jer njegova vrijednost odgovara faktoru kojim raste funkcija. Ova tehnika zbog toga se naziva **gradijentni spust** [1].

Točke u kojima derivacija ne daje nikakvu informaciju o smjeru pomaka nazivaju se **kritične točke** (engl. *critical points*) ili **stacionarne točke** (engl. *stationary points*). Tri su tipa stacionarne točke: lokalni minimum u kojoj je vrijednosti funkcije minimalna u odnosu na susjedne, lokalni maksimum u kojoj je vrijednost funkcije maksimalna u odnosu na susjede, i točka sedla koja ima susjede čije su vrijednosti s jedne strane manje, a s druge veće od njene. Primjeri karakterističnih točaka prikazani su slikom (Slika 2.2). Točka u kojoj funkcija ima apsolutno najnižu vrijednost naziva se **globalni minimum** [1].




Slika 2.2 Stacionarne točke – točke u kojima je derivacija funkcije jednaka nuli [1]

Za funkcije s više ulaza, potrebno je računati derivacije funkcije po svakom od ulaza, tzv. parcijalne derivacije. Kombinacija svih parcijalnih derivacija naziva se **gradijent**, od čega dolazi naziv gradijentni spust. Algoritam koji efikasno računa gradijente složenih funkcija zove se **algoritam propagacije greške unazad** (engl. *back-propagation algorithm*) [1].

Za učenje dubokih neuronskih mreža potrebno je računati gradijente vrlo kompleksnih funkcija, koje mogu imati više od jednog minimuma, odnosno maksimuma. Optimizacijski postupak može se zaustaviti na nekom od lokalnih minimuma, stoga nije uvijek garantirano da će optimizacijski algoritam pronaći

globalni minimum. U kontekstu dubokog učenja ovo generalno nije problem sve dok je rezultat optimizacije dovoljno mala vrijednost funkcije pogreške. Primjer funkcije s nekoliko lokalnih minimuma prikazana je na slici dolje (Slika 2.3) [1].


Slika 2.3 Funkcija s nekoliko lokalnih minimuma [1]

# 3. Vremenska serija

Tema ovog poglavlja su vremenske serije. Prvi odjeljak sadrži definiciju vremenske serije te opis njezinih komponenti. U drugom odjeljku dana je matematička formulacija vremenske serije. U posljednjem odjeljku navedene su primjene predviđanja vremenskih serija, s posebnim naglaskom na predviđanje pomoću tehnologije dubokog učenja. Posebno je naglašena razlika između kratkoročnog i dugoročnog predviđanja kao ključnog faktora pri odabiru odgovarajućeg modela za predviđanje.

## 3.1. Definicija vremenske serije

**Vremenska serija** (engl. *time series*) je niz podataka s vremenskim poretkom. Vremenske serije koriste se za promatranje kako se određene pojave mijenjaju kroz vrijeme. Primjeri vremenskih serija su podaci o potrošnji ili proizvodnji energije, zagađenju zraka, koncentraciji ozona, cijenama dionica, meteorološki podaci, itd. [4].

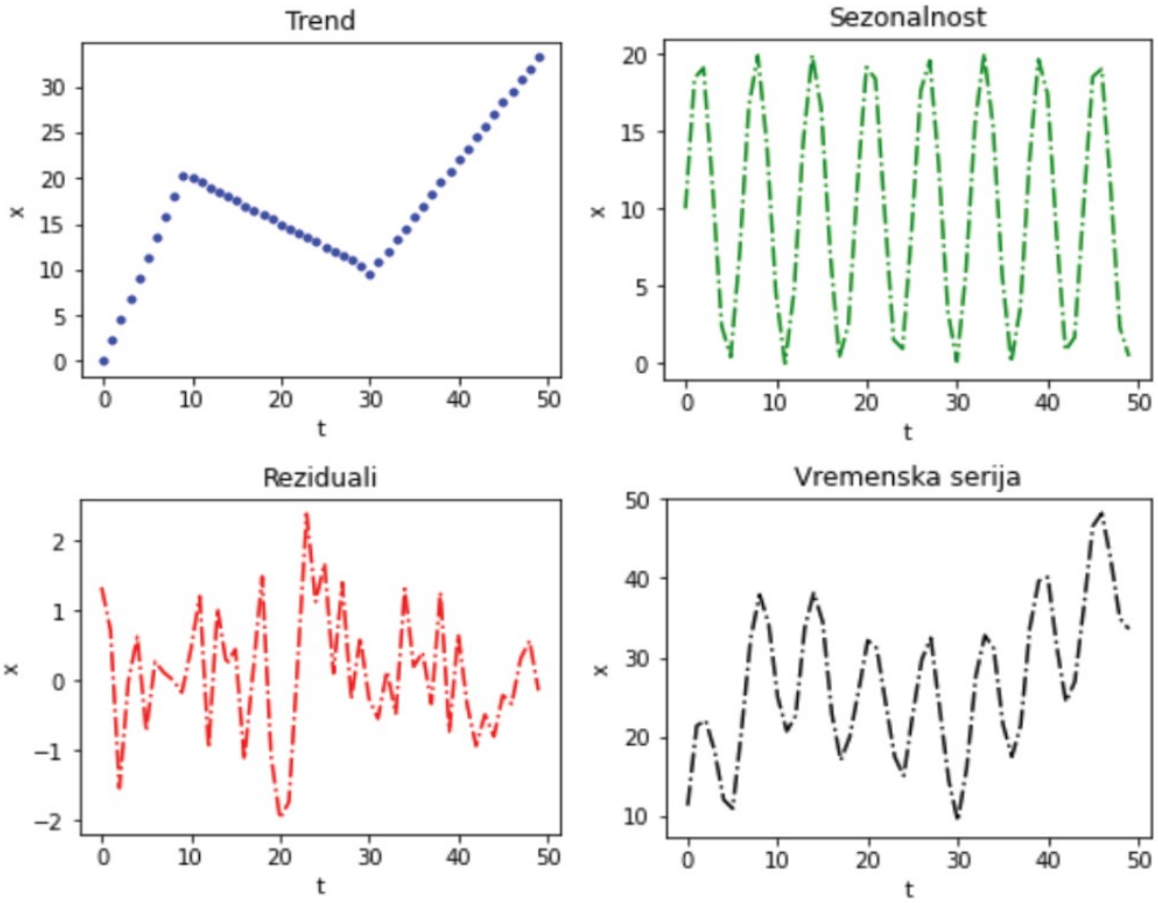
Zapisi vremenske serije prikupljaju se kroz period proizvoljnog trajanja. Svaki od zapisa zabilježen je u specifičnom trenutku i posloženi su kronološki, s uvjetom da je interval između mjerenja njihovih vrijednosti konstantan. Ovaj uvjet primjenjuje se u većini slučajeva modeliranja vremenske serije. Neki od razloga zašto nije uvijek moguće modelirati vremensku seriju s konstantnim vremenskim intervalima između susjednih vrijednosti su:

- Nedostajući podaci,
- Česti izuzetci u podacima (engl. *outliers*),
- Prikupljanje podataka u nasumičnim intervalima.

Postoji puno pristupa rješavanju problema nedostajućih podataka. Najčešće se zapisi kojima nedostaju vrijednosti izostavljaju iz serije ili nadomještaju naknadno izračunatim vrijednostima [2].

Vremenska serija generalno se može opisati kroz tri komponente: trend, sezonalnost, i nepravilne komponente, odnosno reziduali [2]. Primjer vremenske serije i njene dekompozicije na navedene komponente prikazan je u nastavku (Slika

3.1). Grafovi komponenata i vremenske serije na horizontalnoj osi sadrže vrijeme te na vertikalnoj osi vrijednosti zabilježene u nekom vremenskom trenutku.




Slika 3.1 Vremenska serija i njene komponente

**Trend** je generalan smjer promjene u podacima kroz promatrani period, izoliran od sezonalnosti i rezidualnih vrijednosti. Drugo ime pod kojim je poznata ova komponenta je dugoročna varijabilnost. Najčešći trendovi među vremenskim serijama su linearna, eksponencijalna i parabolična [2].

**Sezonalnost** je komponenta koja obuhvaća promjene u podacima koje se pojavljuju u jednakim intervalima. Analizom sezonalnosti mogu se saznati bitne informacije o uzorcima koji postoje u podacima. Primjeri uzroka sezonalnosti su klima i ciklusi u ekonomiji [2].

**Reziduali** su vrijednosti koje preostaju kada se iz vremenske serije uklone trend i sezonalnost. Vrijednosti reziduala mogu biti tolikih amplituda da se trend i sezonalnost ne mogu uočiti u vremenskoj seriji. U tim slučajevima reziduali se smatraju izuzetcima u podacima. Čest način suočavanja s ovim problemom je primjena robusne statistike. Pojava reziduala može imati različite uzroke što otežava

njihovo modeliranje. U slučajevima kada ih je moguće modelirati, reziduali se mogu promatrati kao pokazatelj, odnosno indikator nadolazećih promjena u trendu vremenske serije [2].

## 3.2. Matematička formulacija vremenske serije

Predviđanje vremenske serije je aproksimiranje budućih vrijednosti vremenske serije [4]. Modeli koji predviđaju buduće vrijednosti vremenske serije mogu biti univarijatni ili multivarijatni. Univarijatni modeli podrazumijevaju jednu izlaznu varijablu, a multivarijatni više izlaznih varijabli. Univarijatni i multivarijatni modeli mogu biti drastično različiti, ali većina dubokih modela podržava predviđanje jedne, odnosno više izlaznih varijabli [2].

Neka je univarijatna vremenska serija definirana izrazom (7).

$$\mathrm{y} = \mathrm{y}(\mathrm{t}-\mathrm{L}), \dots, \mathrm{y}(\mathrm{t}-1), \mathrm{y}(\mathrm{t}), \mathrm{y}(\mathrm{t}+1), \dots, \mathrm{y}(\mathrm{t}+\mathrm{h}) \tag{7}$$

Sastoji se od $\mathrm{L}$ povijesnih podataka gdje svaki $\mathrm{y}(\mathrm{t}-\mathrm{i}), \mathrm{i} = 0, \dots, \mathrm{L}$ predstavlja vrijednost varijable $\mathrm{y}$ u trenutku $\mathrm{t}-\mathrm{i}$. Predviđanje se radi za vrijednost u trenutku $\mathrm{t}+1$, odnosno vrijednost $\mathrm{y}(\mathrm{t}+1)$, s nastojanjem da razlika između stvarne i previđene vrijednosti bude što manja. Predviđena vrijednost označava se s $\hat{\mathrm{y}}(\mathrm{t}+1)$, a tipična formulacija funkcije za minimiziranje je $\mathrm{y}(\mathrm{t}+1) - \hat{\mathrm{y}}(\mathrm{t}+1)$. Međutim, predikcija se ne mora nužno raditi za jednu vrijednost. Moguće je predviđati $\mathrm{h}$ vrijednosti nakon $\mathrm{y}(\mathrm{t})$, odnosno $\mathrm{y}(\mathrm{t}+\mathrm{i}), \mathrm{i} = \mathrm{i}, \dots, \mathrm{h}$. Tada je funkcija greške koju treba minimizirati oblika (8).

$$\sum_{\mathrm{i}=1}^{\mathrm{h}} ((\mathrm{y}+\mathrm{i}) - \hat{\mathrm{y}}(\mathrm{t}+\mathrm{i})) \tag{8}$$

Multivarijatna vremenska serija može se prikazati tablično izrazom (9) [2].

$$\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \\ y_T \end{pmatrix} = \left(\begin{array}{cccc|ccc} y_1(t-L) & \dots & y_1(t-1) & y_1(t) & y_1(t+1) & \dots & y_1(t+h) \\ y_2(t-L) & \dots & y_2(t-1) & y_2(t) & y_2(t+1) & \dots & y_2(t+h) \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ y_n(t-L) & \dots & y_n(t-1) & y_n(t) & y_n(t+1) & \dots & y_n(t+h) \end{array}\right) \tag{9}$$

## 3.3. Primjena dubokog učenja u predviđanju vremenskih serija

Zadnjih nekoliko godina porastao je interes za primjenu dubokog učenja u predviđanju vremenskih serija. Algoritmi dubokog učenja, vodećeg područja unutar strojnog učenja, pokazali su se superiornijima u odnosu na druge tehnike strojnog učenja. Razlog zašto su algoritmi dubokog učenja toliko efikasni za ovu primjenu je sposobnost dubokih modela da modeliraju nelinearnosti u podacima [4].

Kada se razmatra predviđanje vremenskih serija pomoću dubokog učenja, ključan aspekt je vremenski horizont, što označava duljinu vremenskog perioda za koji se žele predviđati vrijednosti. Predviđanje vremenskih serija se po duljini horizonta dijeli na kratkoročno i dugoročno predviđanje. **Kratkoročno predviđanje** (engl. *short-term forecasting*) usredotočeno je na predviđanje događaja u bliskoj budućnosti, što uključuje predviđanje vrijednosti unutar nekoliko minuta do nekoliko sati. **Dugoročno predviđanje** (engl. *long-term forecasting*) odnosi se na predviđanje trendova ili ciklusa koji se protežu kroz dulje vremenske intervale, kao što su dani, tjedni, mjesečni ili godišnji periodi [9].

# 4. Kratkoročno predviđanje vremenskih serija

Ovo poglavlje temeljito istražuje različite pristupe dubokog učenja koji su se pokazali efikasnim za kratkoročno predviđanje vremenskih serija. Među istaknutim pristupima su:

- Povratne neuronske mreže,
- Konvolucijske neuronske mreže,
- Graf neuronske mreže,
- Generativni modeli.

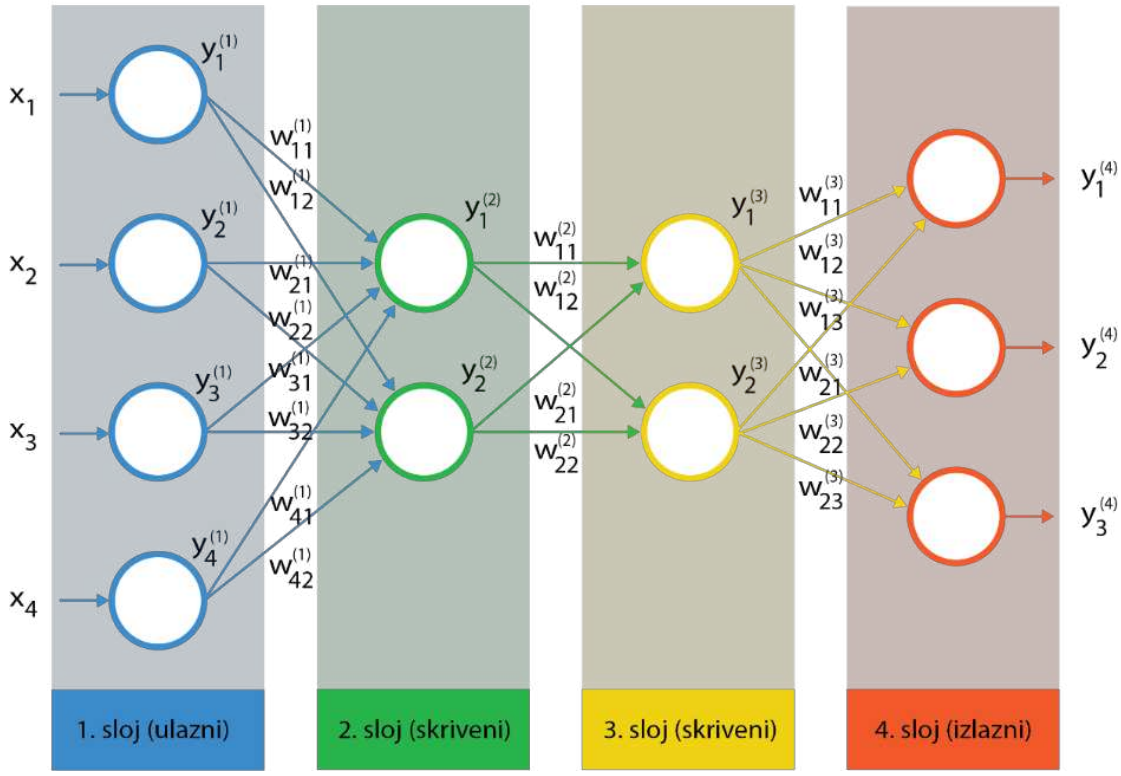
Za svaki od ovih pristupa opisani su aspekti koji ih čine prikladnima za kratkoročno predviđanje, kao i nedostaci u njihovoj primjeni u kontekstu predviđanja vremenskih serija. Kao uvod u duboke modele prvo je opisan temeljni model dubokog učenja – unaprijedna neuronska mreža.

## 4.1. Unaprijedne neuronske mreže (FNN)

Unaprijedne neuronske mreže poznate su pod nekoliko naziva: duboke neuronske mreže, duboke unaprijedne mreže, itd. Temeljni su model dubokog učenja i time ključne za razumijevanje kompleksnijih dubokih modela. Arhitektura dubokih unaprijednih mreža tema je prvog odjeljka, zatim je u drugom odjeljku riječ o postupku treniranja dubokih unaprijednih mreža. U posljednjem odjeljku dotaknulo se nedostataka dubokih unaprijednih mreža u primjeni na predviđanje vremenskih serija.

### 4.1.1. Arhitektura unaprijedne neuronske mreže

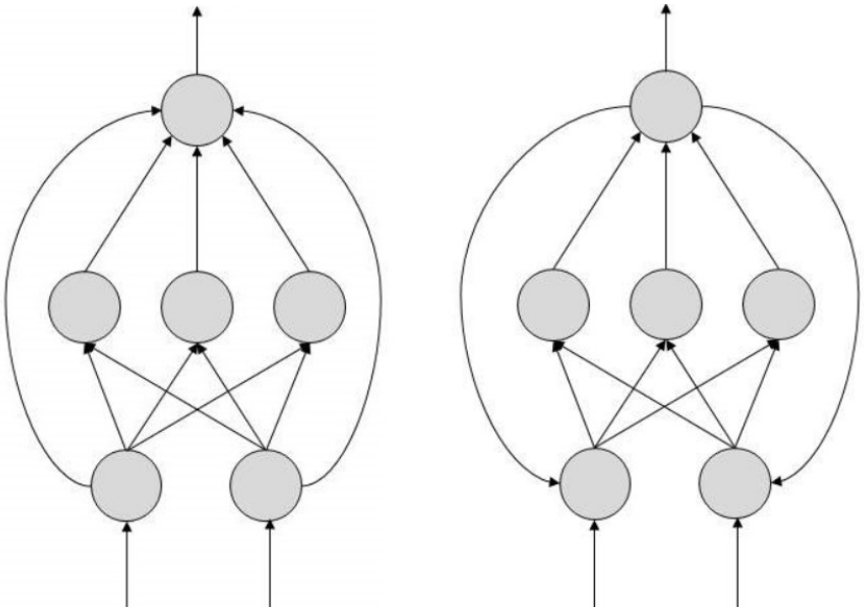
**Duboke unaprijedne mreže** (engl. *feedforward neural networks, FNN*) nastale su iz potrebe za modeliranjem funkcija koje se nisu mogle modelirati pomoću perceptrona spomenutog u prvom poglavlju. Sastoje se od minimalno tri sloja neurona, zbog čega je višeslojni perceptron (engl. *multi-layer perceptron, MLP*) drugi naziv pod kojim su poznate duboke unaprijedne mreže [2]. Osnovnu arhitekturu mreže čine ulazni i izlazni sloj te tzv. skriveni slojevi između njih, sukladno prikazu na slici ispod (Slika 3.1).




Slika 4.1 Arhitektura duboke unaprijedne mreže

Unaprijedna neuronska mreža je esencijalni duboki model. Cilj unaprijedne neuronske mreže je aproksimirati neku funkciju $f^*$. Primjerice, u slučaju klasifikacije, $y = f^*(x)$ mapira ulaz $x$ u neku kategoriju $y$. Unaprijedna neuronska mreža definira mapiranje $y = f(x;\theta)$, gdje su $\theta$ parametri funkcije, a postupak učenja je računanje parametara koji daju najbolju aproksimaciju funkcije koja mapira ulaz u izlaz [1].

Nazivaju se unaprijednima zbog smjera u kojem informacije teku kroz funkciju koja se aproksimira. Funkcija se evaluira od ulaza $x$, kroz skrivene slojeve, do izlaza $y$. Sve veze u mreži usmjerene su od ulaza prema izlazu, odnosno nema povratnih veza među slojevima mreže. Povratna veza podrazumijeva da neuron šalje svoj izlaz neuronu u nekom od slojeva iza njega, odnosno sloju bližem ulazu mreže od njegovog [1]. Primjer unaprijednih i povratnih veza prikazani su na slici ispod (Slika 4.2). Lijevi isječak mreže sadrži samo unaprijedne veze, a desni uz unaprijedne sadrži par povratnih veza.



Slika 4.2 Unaprijedne i povratne veze u neuronskim mrežama [7]

Dodavanjem povratnih veza u unaprijednu neuronsku mrežu, mreža postaje **povratna neuronska mreža** (engl. *recurrent neural network*). Povratne neuronske mreže generalizirane su unaprijedne neuronske mreže sa značajnom primjenom u obradi prirodnog jezika i procesiranju sekvenci podataka. Osnovni model unaprijedne neuronske mreže također se može specijalizirati za rješavanje specifičnih problema [1].

Nadogradnjom unaprijednih neuronskih mreža širi se skup problema na koje se može primijeniti duboko učenje [1]. Zbog toga su one jako bitan koncept u strojnom učenju, a razumijevanje njihovog rada nužno je za razumijevanje mreža kao što su povratne neuronske mreže i konvolucijske neuronske mreže, čiji će se rad i primjena detaljnije pojasniti u nastavku.

### 4.1.2. Treniranje unaprijedne neuronske mreže

Unaprijedne neuronske mreže tipično se mogu reprezentirati kompozicijom velikog broja funkcija. Zbog toga se nazivaju mrežama. Primjerice, za neke tri funkcije $f^{(1)}$, $f^{(2)}$ i $f^{(3)}$, njihova kompozicija u obliku lanca formira funkciju (10).

$$f(x) = f^{(3)}(f^{(2)}(f^{(1)}(x))) \tag{10}$$

Ovakve ulančane strukture najčešće su korištene strukture neuronskih mreža. U funkciji definiranoj izrazom (10), funkcija $f^{(1)}$ odgovara prvom sloju mreže, $f^{(2)}$

drugom sloju, itd. Posljednji sloj naziva se izlazni sloj i računa izlaz funkcije. Pritom duljina lanca određuje dubinu modela. Postupkom treniranja nastoji se pronaći funkciju $f(x)$ koja najbolje aproksimira preslikavanje $f^*(x)$. Svaki od podataka za učenje daje primjer aproksimativne evaluacije funkcije $f^*(x)$ u nekoj točki. Za svaki $x$ poznata je njegova oznaka $y \approx f^*(x)$. Primjeri specificiraju kakav je izlaz posljednjeg, izlaznog sloja za svaki $x$. Ostali slojevi nemaju takvo direktno definirano ponašanje te se njihovo ponašanje definira učenjem. Jer primjeri za učenje ne specificiraju izlaze unutarnjih slojeva, oni se nazivaju skrivenim slojevima (engl. *hidden layers*) [1].

Linearni modeli mreža efikasno se i pouzdano koriste za aproksimiranje linearnih funkcija, bilo u zatvorenoj formi ili konveksnom optimizacijom. Njihov očiti nedostatak je što su ograničeni na aproksimiranje samo linearnih funkcija. Proširenje linearnih modela kojim se postiže nelinearno mapiranje ulaza u izlaz je primjena linearnog modela umjesto na ulaz $x$, na transformirani ulaz $\phi(x)$, gdje je $\phi$ nelinearna transformacija. $\phi$ se može smatrati skupom značajki koje opisuju ulaz $x$, odnosno novom reprezentacijom ulaza $x$ [1].

Pitanje je kako odrediti $\phi(x)$. Strategija dubokog učenja je naučiti transformaciju ulaza. Model tada preuzima izraz (11).

$$y = f(x; \theta, w) = \phi(x; \theta)^T w \tag{11}$$

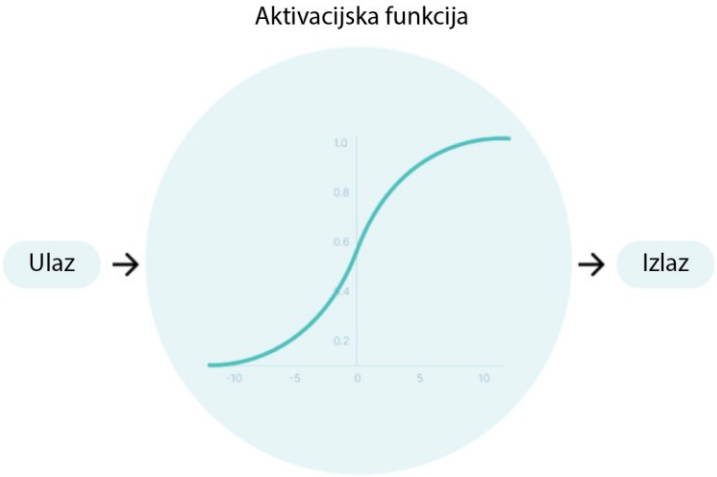
Model sada sadrži parametre $\theta$ koji se koriste za učenje funkcije $\phi$ te parametara $w$ koji rade mapiranje $\phi(x)$ u željeni izlaz $y$. U ovoj formulaciji $\phi$ definira skrivene slojeve i cilj optimizacijskog postupka je pronaći parametre $\theta$ koji daju najbolju reprezentaciju ulaza [1].

Ovaj generalni princip unaprjeđenja modela učenjem značajki nije isključivo vezan za unaprijedne neuronske mreže nego i za druge vrste dubokih modela. Primjena u drugim vrstama modela bit će spomenuta u nastavku rada [1].

Postupak optimizacije modela provodi se algoritmom opisanim u poglavlju Gradijentni spust.

### 4.1.3. Skriveni slojevi

Skriveni slojevi su dodatak neuronskim mrežama koji je specifičan za duboko učenje. Bez skrivenih slojeva, mreža je obična neuronska mreža. Posebnost skrivenih slojeva je uvođenje nelinearnosti u model mreže. Način na koji se uvodi nelinearnost je korištenje tzv. **aktivacijske funkcije**. Drugi naziv za aktivacijsku funkciju je prijenosna funkcija. Ukratko, ideja je da neuron svoj ulaz prvo lineamo transformira zatim primijeni aktivacijsku funkciju i tako dobije nelinearnu vrijednost na izlazu. Izlazna vrijednost postaje ulaz neurona u idućem sloju, kao što je prikazano na slici (Slika 4.3). Prilikom dizajniranja dubokih neuronskih mreža potrebno je odabrati vrstu neurona, odnosno vrstu aktivacijske funkcije koja će se koristiti u skrivenim slojevima [1].




Slika 4.3 Aktivacijska funkcija [10]

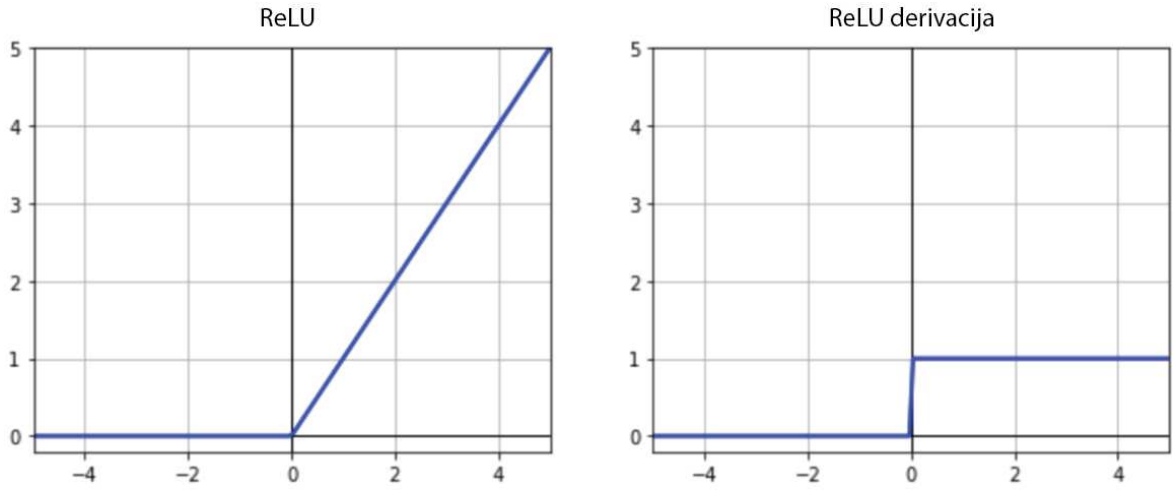
Transformirani ulaz $\phi(x)$ opisan u prethodnom dijelu poglavlja postiže se primjenom aktivacijskih funkcija u skrivenim slojevima. Cjelokupna operacija neurona u skrivenim slojevima dana je izrazom (12), gdje su $W$ težine lineame transformacije, $c$ pomaci (engl. *biases*), i $g$ aktivacijska funkcija [1].

$$h = g(W^T x + c) \tag{12}$$

Poznate aktivacijske funkcije su:

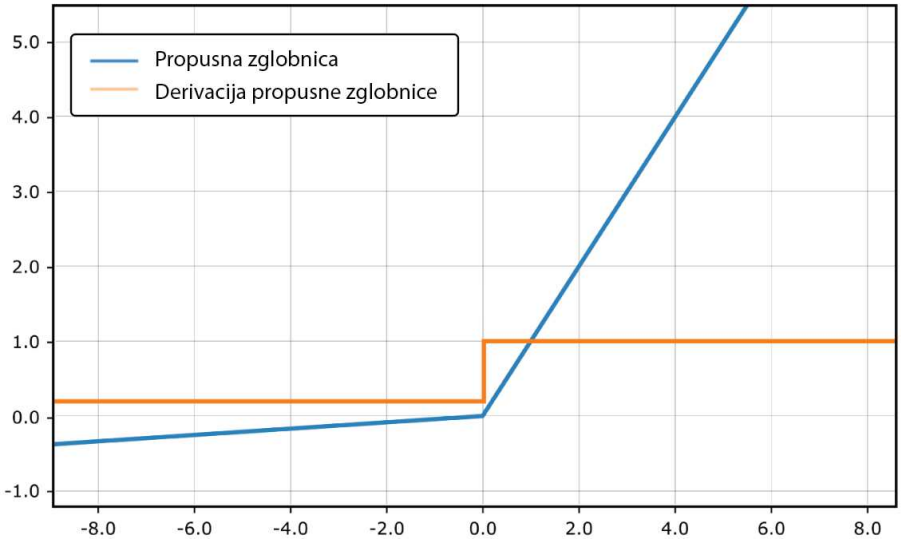
- Zglobnica,
- Propusna zglobnica,
- Sigmoidalna funkcija,
- Tangens hiperbolni.

**Zglobnica** je funkcija $g(z) = \max\{0, z\}$. Neuroni koji koriste zglobnicu kao aktivacijsku funkciju imaju kraticu ReLU (engl. *rectified linear unit*). Graf zglobnice i njene derivacije prikazan je na donjoj slici (Slika 4.4). Iz grafa zglobnice vidi se da za vrijednosti manje od nule daje izlaz 1 te za vrijednosti veće od nule vraća istu tu vrijednost. Derivacija je nula za vrijednosti manje od nule i jednaka 1 za vrijednosti veće od nule. Preporuka modernog pristupa treniranju neuronskih mreža je koristiti ReLU neurone.



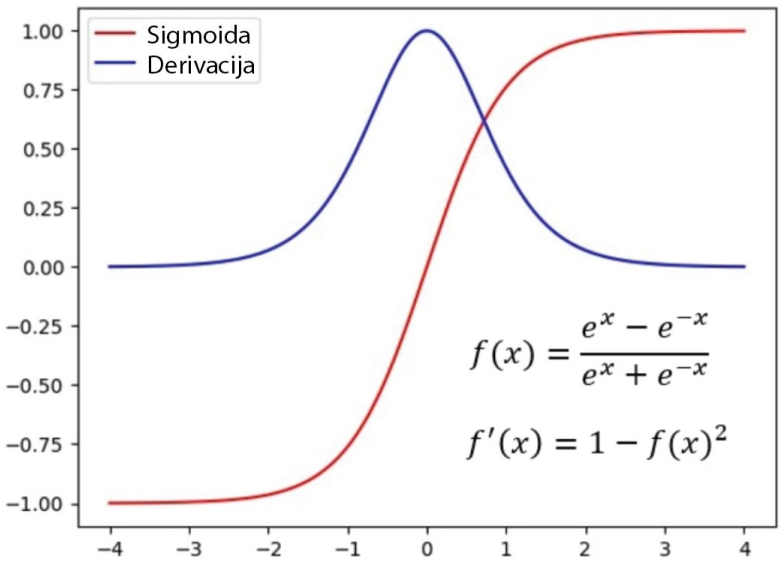

Slika 4.4 Zglobnica i njena derivacija [11]

**Propusna zglobnica** je modificirana zglobnica za vrijednosti manje od nule. Umjesto postavljanja svih vrijednosti na nulu, množi ih s faktorom $0.01$ kako derivacija ne bi bila jednaka nuli. Neuroni kojima je propusna zglobnica aktivacijska funkcija imaju kraticu LReLU (engl. *leaky rectified unit*).



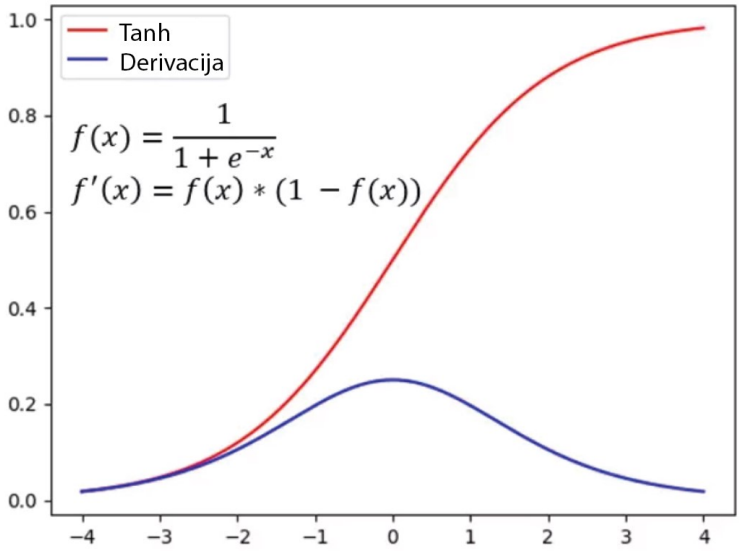

Slika 4.5 Propusna zglobnica i njena derivacija [12]

Aktivacijska funkcija koja se koristila prije pojave ReLU neurona, je **sigmoidalna funkcija**. Njen graf i graf njene derivacije prikazani su na donjoj slici (Slika 4.6).




Slika 4.6 Sigmoidalna aktivacijska funkcija i njena derivacija [13]

Uz sigmoidalnu funkciju, koristio se **tangens hiperbolni**. Na donjoj slici prikazani su grafovi tangensa hiperbolnog i njegove derivacije (Slika 4.7).




Slika 4.7 Tangens hiperbolni kao aktivacijska funkcija i njegova derivacija [13]

U kontekstu dubokog učenja, sigmoida i tangens hiperbolni uglavnom se koriste za aktivacijske funkcije neurona u izlaznim slojevima [1].

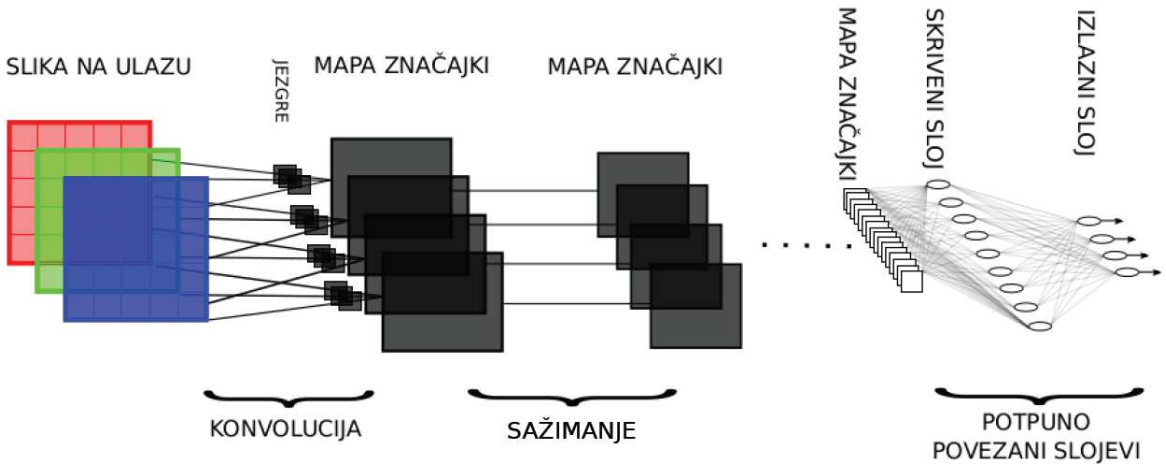
## 4.2. Konvolucijski modeli

Tema ovog poglavlja su konvolucijski modeli. Slojevi karakteristični za konvolucijsku neuronsku mrežu su konvolucijski sloj, po kojem je mreža dobila ime, sloj sažimanja te potpuno povezani sloj. Nove vrste slojeva istražene su u prvom dijelu poglavlja. U drugom dijelu poglavlja ukratko je opisana varijanta konvolucijske mreže nazvana temporalna konvolucijska mreža.

### 4.2.1. Konvolucijske neuronske mreže (CNN)

**Konvolucijske neuronske mreže** (engl. *convolutional neural networks*, CNN) jedna su od najčešće korištenih arhitektura mreža za procesiranje slika i računalni vid (engl. *computer vision*) [2]. Specijalizirane su za rad s podacima s topologijom rešetke, odnosno podacima u kojima postoje relacije između susjeda. Primjeri takvih podataka za različite dimenzije su vremenski slijed u slučaju 1D podataka, slike u slučaju 2D podataka i volumen u slučaju 3D podataka [17]. CNN mreže dobile su naziv po matematičkoj operaciji koju koriste, imena **konvolucija** (engl. *convolution*). Konvolucija je specijalizirana vrsta linearne operacije. Razlika između konvolucijskih i običnih neuronskih mreža je u tome što konvolucijske mreže koriste operaciju konvolucije umjesto matričnog množenja u barem jednom od svojih slojeva [1]. CNN mreže sastoje se od tri vrste karakterističnih slojeva:

- Konvolucijski sloj (engl. *convolution layer*),
- Sloj sažimanja (engl. *pooling layer*),
- Potpuno povezani sloj (engl. *fully connected layer*) [2].




Slika 4.8 Klasična struktura konvolucijskog modela [17]

Klasična struktura konvolucijskog modela prikazana je na slici (Slika 4.8).

**Konvolucijski slojevi** naziv su dobili po operaciji konvolucije koja se primjenjuje nad ulazima u sloj. Svrha operacije konvolucije u neuronskoj mreži je modelirati **lokalne interakcije** u podacima i preslikati ih u tzv. mapu značajki. Općenito, operacija se provodi nad dvije funkcije i označava se zvjezdicom, kao u izrazu (13) [17].

$$h(t) = (x * w)(t) \tag{13}$$

Za kontinuirane vrijednosti konvolucija se računa kao integral umnožaka ulaza definiranog s $x$ i funkcije $w$. Matematički zapis konvolucije dan je izrazom (14).

$$h(t) = (x * w)(t) = \int x(t + a)\, w(a)\, da \tag{14}$$

U diskretnom slučaju se umjesto integrala koristi suma po izrazu (15).

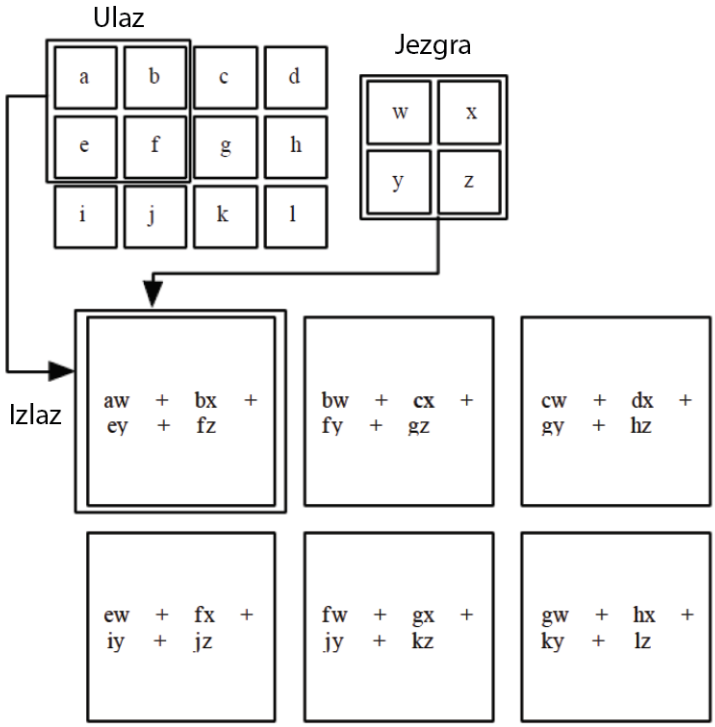
$$h(t) = (x * w)(t) = \sum_{a=-\infty}^{\infty} x(t + a)\, w(a) \tag{15}$$

U kontekstu konvolucijskih mreža, prvi argument konvolucije, u ovom slučaju funkcija $x$, poznata je kao **ulaz**, a drugi argument, u ovom slučaju funkcija $w$, poznata je kao **jezgra** (engl. *kernel*). Za izlaz konvolucije $h(t)$, nekad se koristi naziv već spomenuta **mapa značajki** (engl. *feature map*) [1].

Za strojno učenje je čest slučaj da su ulazni podaci višedimenzionalni te da je jezgra višedimenzionalna lista parametara koji se uče algoritmom. Višedimenzionalni vektori se u stručnoj literaturi zovu **tenzori** (engl. *tensors*). Također, konvolucije se često koriste nad više od jedne dimenzije odjednom, stoga se operacija u slučaju dvodimenzionalnih podataka i jezgre računa izrazom (16).

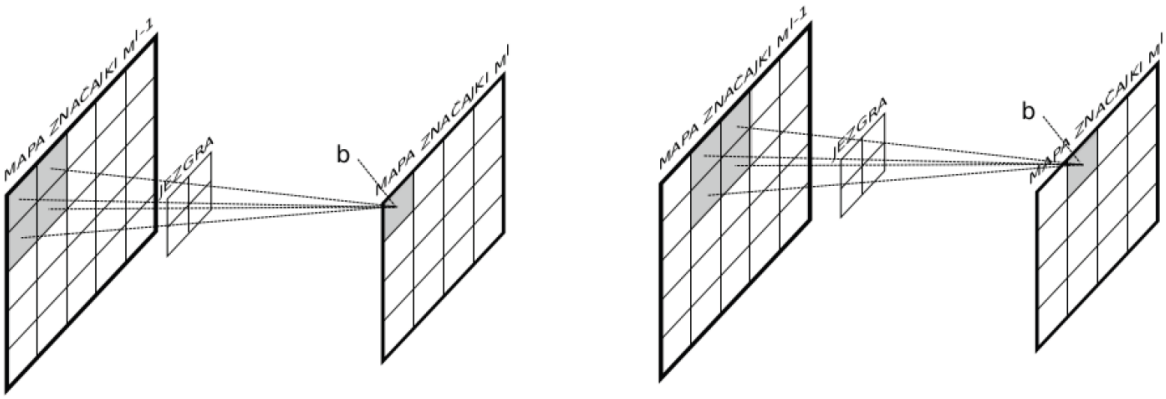
$$H(i,j) = (I * K)(i,j) = \sum_{m} \sum_{n} I(i + m, j + n)\, K(m,n) \tag{16}$$

Primjer dvodimenzionalnog ulaznog podatka je slika, te je u gornjem izrazu označena s $I$. Jezgra je označena s $K$ te su njene dimenzije $[m, n]$. Opisana intuicija operacije konvolucije ilustrirana je donjom slikom (Slika 4.9) [1].




Slika 4.9 Primjena jezgre nad ulaznim podacima

Konvolucija se primjenjuje na ulaznim podacima s pomakom kao što je prikazano na slici (Slika 4.10) [1]. Iz ove slike da se uočiti da se svi elementi mape značajki računaju uz pomoć istog skupa parametara. Ovo svojstvo konvolucijskih mreža naziva se **dijeljenje parametara**. Dijeljenje parametara odnosi se na korištenje istog parametra u više od jedne funkcije modela. U klasičnoj neuronskoj mreži, svaki parametar koristi se točno jednom za izračun nekog od izlaza neurona u sloju. U CNN mreži se isti skup parametara jezgre koristi za svaku poziciju ulaza. Drugim riječima, umjesto da se uči zaseban skup parametara za svaki dio ulaza, uči se samo jedan skup. Još jedna prednost dijeljenja parametara je smanjeni broj parametara u odnosu na klasičnu neuronsku mrežu [1].

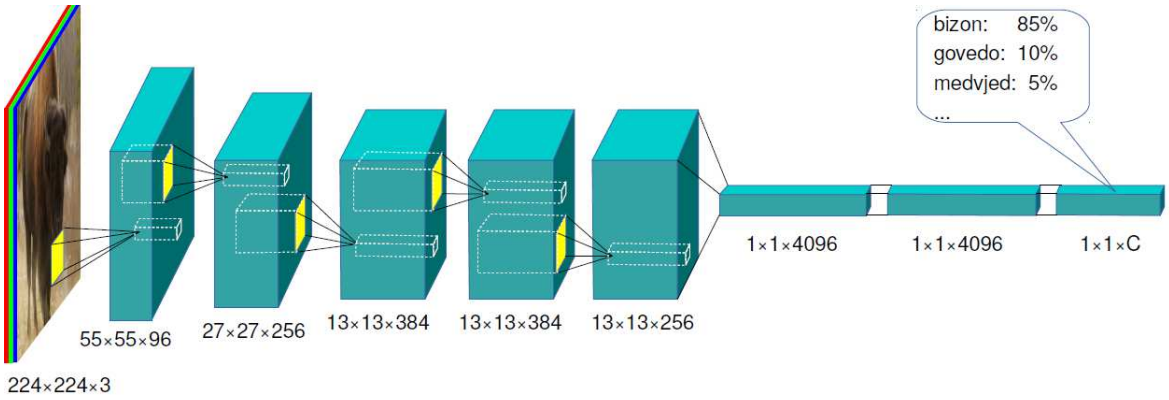



Slika 4.10 Mape značajki dobivene konvolucijom [17]

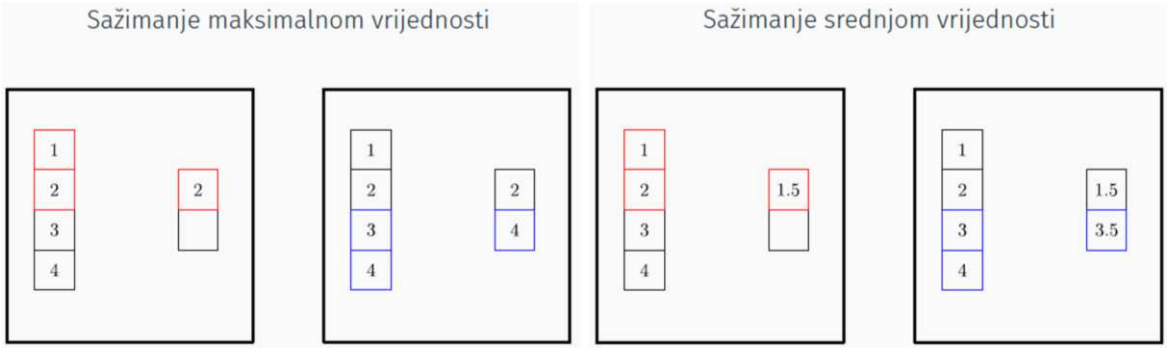
Navedeni način dijeljenja parametara u slučaju operacije konvolucije rezultira svojstvom koje se naziva **ekvivarijantnost s obzirom na pomak** (engl. *equivariance to translation*). Općenito, funkcija $f(x)$ je ekvivarijantna s obzirom na funkciju $g$ ako vrijedi $f(g(x)) = g(f(x))$. Ovo svojstvo znači da će specifični ulazni podaci imati istu reprezentaciju nakon konvolucije, neovisno o svojoj poziciji ili pomaku u određenom smjeru. Primjer koji jasno ilustrira ovo svojstvo je slika na kojoj će neki predmet, bez obzira na to gdje se nalazi, i dalje će biti prepoznat kao isti predmet. Prilikom obrade podataka vremenske serije, rezultat konvolucije je zbog ovog svojstva neka vrsta vremenske crte koja prikazuje kada se različite značajke pojavljuju u ulazu. Također, ako neki događaj pomaknemo u budućnost za neki broj vremenskih koraka i dalje će se ista ta reprezentacija pojaviti u izlazu, samo kasnije u vremenu [1].

Bitno je napomenuti da konvolucija nije prirodno ekvivarijantna s obzirom na neke druge transformacije, poput skaliranja ili rotacija slika, te je za postizanje ekvivarijantnosti s obzirom na te transformacije potrebno uvesti dodatne mehanizme [1].

**Funkcija sažimanja** (engl. *pooling function*) koristi se u sloju sažimanja za mapiranje prostorno bliskih značajki ulaza u jednu značajku na izlazu. Najčešće se računa kao maksimalna vrijednost značajki (engl. *max pooling*) ili srednja vrijednost značajki (engl. *average pooling*). Primjena obje spomenute vrste sažimanja prikazana je na slici u nastavku (Slika 4.12). Sažimanje se koristi za prevođenje nekog tenzora u simboličku kategoriju [17]. Primjer koji ilustrira rezultat sažimanja vidi se na slici ispod (Slika 4.11).




Slika 4.11 Mreža sa slojevima konvolucije i sažimanja [17]




Slika 4.12 Vrste sažimanja [17]

### 4.2.2. Temporalne konvolucijske mreže (TCN)

Područje konvolucijskih modela koje je manje istraženo je uporaba temporalnih konvolucijskih mreža (engl. *temporal convolutional networks*, TCN), varijante CNN-a koja koristi kauzalne i dilatirane konvolucije. Ove konvolucije omogućuju širi pregled povijesnih podataka i imaju sposobnost preskakanja veza, čime se omogućava korištenje podataka iz prethodnih slojeva i očuvanje gradijenta. U usporedbi s LSTM mrežom koja je obrađena u nastavku poglavlja, TCN mreža ima neke prednosti kada je u pitanju obrada sekvenci podataka. Međutim, još nije jasno vrijedi li isto za obradu vremenskih serija [3].

## 4.3. Povratni modeli

Povratni modeli neuronskih mreža tema su ovog poglavlja. U prvom odjeljku opisana je osnovna verzija povratne neuronske mreže. Zatim su u iduća dva odjeljka opisane specifične varijante povratne neuronske mreže te njihove karakteristike. U posljednjem odjeljku ukratko su navedeni nedostaci povratnih modela koji se protežu unatoč izmjenama osnovnog modela.

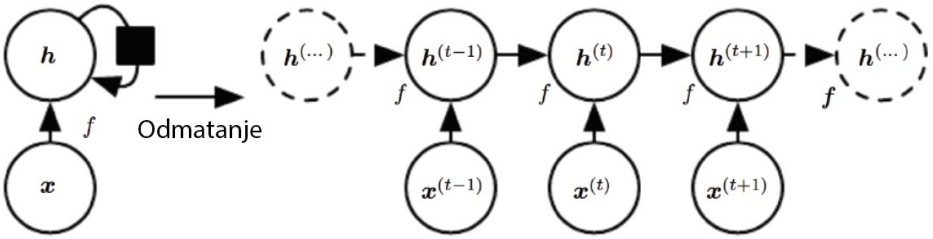
### 4.3.1. Povratna neuronska mreža (RNN)

**Povratne neuronske mreže** (engl. *recurrent neural networks*, RNN) posebno su dizajnirane za obradu sekvencijalnih podataka. Primjeri primjene RNN mreža uključuju prevođenje teksta, prepoznavanje govora iz audio zapisa, predviđanje vremenskih serija i druge slične zadatke. Zajedničko obilježje sekvencijalnih podataka u navedenim primjerima primjene su vremenske ovisnosti.

Osnovni duboki model, unaprijedna neuronska mreža, ne uzima u obzir vremenske ovisnosti među podacima prilikom treniranja, zbog čega su povratne neuronske mreže prikladnije za takve zadatke [2].

Arhitektura povratne neuronske mreže inspirirana je **dinamičkim sustavima** (engl. *dynamical systems*). Polazna ideja je da se stanje modela $\mathrm{h}^{(\mathrm{t})}$ ažurira za svaki ulazni podatak $\mathrm{x}^{(\mathrm{t})}$, prema izrazu (17) [1]. Opisano ažuriranje stanja može se vidjeti na slici ispod (Slika 4.13).

$$\mathrm{h}(\mathrm{t}) = \mathrm{f}\left(\mathrm{x}^{(\mathrm{t})}, \mathrm{h}^{(\mathrm{t}-1)}\right) \tag{17}$$




Slika 4.13 Dinamički sustav (lijevo) i njegovo odmatanje (desno) [1]

Funkcija po kojoj se u osnovnoj verziji povratne neuronske mreže računa novo stanje $\mathrm{h}(\mathrm{t})$ definirana je izrazom (18).

$$\mathrm{h}(\mathrm{t}) = \mathrm{g}\left(\mathrm{W}_{\mathrm{hh}}\, \mathrm{h}^{(\mathrm{t}-1)} + \mathrm{W}_{\mathrm{xh}}\, \mathrm{x}^{(\mathrm{t})} + \mathrm{b}_{\mathrm{h}}\right) \tag{18}$$

Funkcija $\mathrm{g}$ je funkcija nelinearnosti, koja može biti neka od spomenutih aktivacijskih funkcija u poglavlju Skriveni slojevi, ali se najčešće koristi tangens hiperbolni. $\mathrm{W}_{\mathrm{hh}}$ i $\mathrm{W}_{\mathrm{xh}}$ su matrice parametara, a $\mathrm{b}_{\mathrm{h}}$ vektor pristranosti [18].

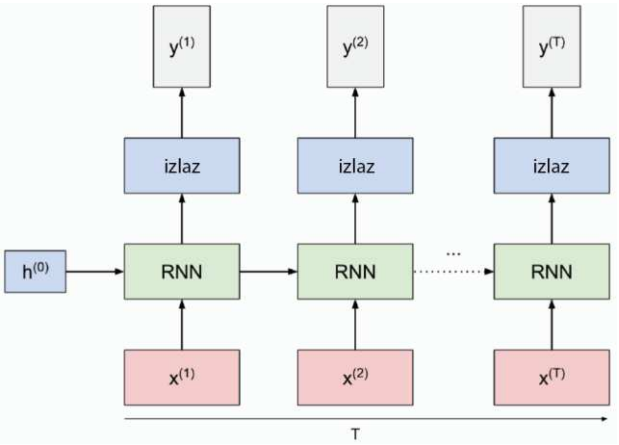
Stanje $\mathrm{h}(\mathrm{t})$ je reprezentacija dosad neviđenog teksta. Za dobivanje te reprezentacije primjenjuje se nelinearna funkcija nad transformiranim ulazom i transformiranim stanje u trenutku $\mathrm{t}-1$. Razlog zašto se transformiraju ulaz i prethodno stanje je potencijalno uklanjanje nebitnih informacija [18].

Prije određivanja predikcije $\mathrm{y}^{\mathrm{t}}$, skriveno stanje $\mathrm{h}(\mathrm{t})$ projicira se iz skrivenog prostora u izlazni prostor, množenjem s matricom $\mathrm{W}_{\mathrm{hy}}$ prema izrazu (19).

$$\mathrm{o}^{(\mathrm{t})} = \mathrm{W}_{\mathrm{hy}}\mathrm{h}^{(\mathrm{t})} + \mathrm{b}_{\mathrm{o}} \tag{19}$$

Vektor $\mathrm{o}^{(\mathrm{t})}$ sadrži vrijednost koje nisu još normalizirane, za predikciju u vremenskom koraku $\mathrm{t}$. Svrha projekcije u izlazni prostor je, kao i u prethodnim množenjima s

matricama težina, filtriranje nebitnih informacija. Opisani tijek dobivanje izlaza iz povratne neuronske mreže prikazan je na donjoj slici (Slika 4.14).




Slika 4.14 Vremenski koraci RNN mreže [19]

Akumulacija gradijenta po parametrima povratne neuronske mreže čini optimizaciju temeljenu na gradijentu jako nestabilnom. Zbog toga se osnovna verzija povratne neuronske mreže rijetko koristi u praksi [18]. Problemi koji mogu nastati akumulacijom gradijenta su **nestajući** i **eksplodirajući gradijent** (engl. *vanishing and exploding gradients*). Uzrok ovih problema s gradijentom je dijeljenje parametara između velikog broja uzastopnih slojeva. Konkretnije, problem je u višestrukom množenju matricom $W_{hh}$ [19]. Preporučena metoda za suočavanje s problemima s gradijentom je tzv. **podrezivanje gradijenta** (engl. *gradient clipping*) [18]. Zbog spomenutog problema nestabilnog treniranja, obične RNN mreže nisu uspješne u učenju dugoročnih ovisnosti u podacima [4].

RNN mreže koje imaju najširu primjenu u predviđanju vremenskih serija su ćelije s dugoročnom memorijom i propusne povratne ćelije [2].

### 4.3.2. Ćelija s dugoročnom memorijom (LSTM)

Ćelija s dugoročnom memorijom (engl. *long short-term memory*, LSTM) specijalizirana je RNN mreža originalno predložena za obradu prirodnog jezika [4]. Motivacija za izgradnju arhitekture LSTM mreže je rasterećenje parametara obične RNN mreže, čiji su problemi u treniranju spomenuti u prethodnom odjeljku [19].

Uvodi se vektor stanja ćelije $c^{(t)}$ koji samo pamti dosadašnje informacije. Drugi vektor koji se uvodi u LSTM mrežama je $\hat{c}^{(t)}$, u kojem se čuva vrijednost za

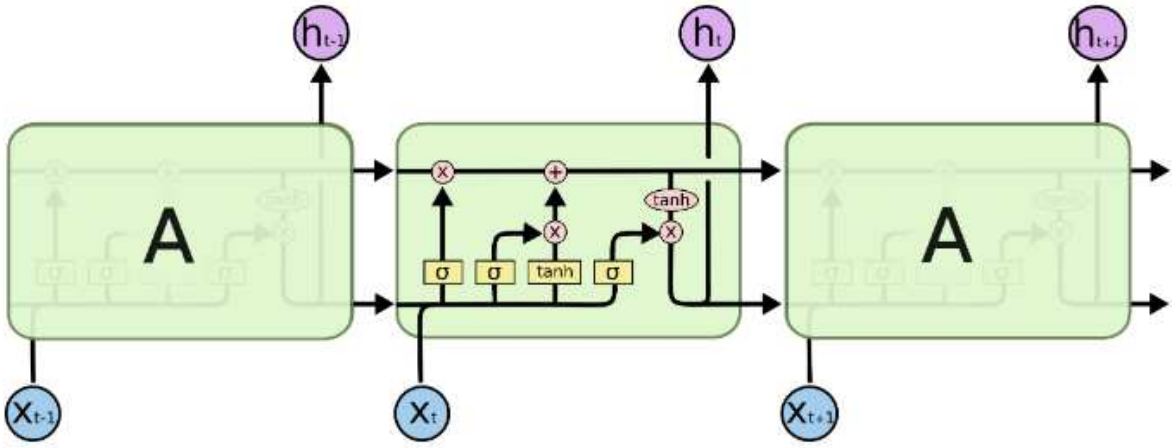
ažuriranje stanja ćelije. Uz vektore $c^{(t)}$ i $\hat{c}^{(t)}$, uvode se tzv. propusnice $f^{(t)}$ i $i^{(t)}$. Izraz prema kojem se računa stanje ćelije $c^{(t)}$ je izraz (20).

$$c^{(t)} = f^{(t)} \odot c^{(t-1)} + i^{(t)} \odot \hat{c}^{(t)} \tag{20}$$

Vektor $h^{(t)}$ prenamijenjen je tako da samo računa izlaz prema izrazu (21).

$$h^{(t)} = o^{(t)} \odot \tanh\left(c^{(t)}\right) \tag{21}$$

Operacija $\odot$ računa tzv. Hadamardov umnožak, odnosno umnožak vektora po elementima. Na slici ispod ilustrirano je kako se računaju stanja u LSTM mreži (Slika 4.15).




Slika 4.15 Propusnice u LSTM ćelijama [16]

Propusnice $f^{(t)}$ i $i^{(t)}$ uvode se s ciljem filtriranja informacija i izbjegavanja problema procesiranja dugih sljedova podataka koji ima obična RNN mreža [19]. Propusnica $f^{(t)}$ naziva se **propusnica zaboravljanja** (engl. *forget gate*), jer služi za zaboravljanje dijela informacija iz prošlog stanja. Računa se prema izrazu (22).

$$f^{(t)} = \sigma\left(W_{fhh}\, h^{(t-1)} + W_{fxh}\, x^{(t)} + b_{fh}\right) = \sigma\left(a_f^{(t)}\right) \tag{22}$$

Druga propusnica, $i^{(t)}$, naziva se **propusnica novog ulaza** (engl. *input gate*) jer služi za propuštanje podskupa informacija iz ulaza, i primjenjuje se prema izrazu (23).

$$i^{(t)} = \sigma\left(W_{ihh}\, h^{(t-1)} + W_{ixh}\, x^{(t)} + b_{ih}\right) = \sigma\left(a_i^{(t)}\right) \tag{23}$$

Međurezultat ažuriranja stanja ćelije, $\hat{c}^{(t)}$, računa se prema izrazu (24).

$$\hat{c}^{(t)} = \tanh\left(W_{chh}\, h^{(t-1)} + W_{cxh}\, x^{(t)} + b_{ch}\right) = \tanh\left(a_c^{(t)}\right) \tag{24}$$

Posljednja vrsta propusnica je **izlazna propusnica** (engl. *output gate*). Označava se s $o^{(t)}$ i računa se prema izrazu (25).

$$o^{(t)} = \sigma\left(W_{ohh}\, h^{(t-1)} + W_{oxh}\, x^{(t)} + b_{oh}\right) = \sigma\left(a_o^{(t)}\right) \tag{25}$$

Nakon primjene svih navedenih propusnica, može se računati novo stanje ćelije $c^{(t)}$ i skrivena reprezentacija $h^{(t)}$ [19].

Ono što se postiglo uvođenjem propusnica je prenošenje uloge pamćenja na stanje ćelije $c^{(t)}$. Zbog propusnice zaboravljanja i propusnice novog ulaza, stanje ćelije $c^{(t)}$ teško je promijeniti. Time se riješio problem preopterećenja skrivenog stanja $h^{(t)}$ u običnoj RNN mreži, ali je broj parametara narastao četiri puta [19].

### 4.3.3. Propusna povratna ćelija (GRU)

**Propusna povratna ćelija** (engl. *gated recurrent unit*, GRU) je varijacija RNN mreže prvotno osmišljena za statističke sustave za strojno prevođenje (engl. *statistical machine translation*) [4]. Karakteristike njene arhitekture su vraćanje odgovornosti ažuriranja skrivenog stanja mreže i pamćenje već viđenih podataka na stanje $h^{(t)}$, i nove vrste propusnica $r^{(t)}$ i $u^{(t)}$ [19].

**Propusnica ažuriranja** (engl. *update gate*) označava se s $u^{(t)}$ i provodi operaciju zadanu izrazom (26).

$$u^{(t)} = \sigma\left(W_{uhh}\, h^{(t-1)} + W_{uxh}\, x^{(t)} + b_{uh}\right) \tag{26}$$

**Propusnica resetiranja** (engl. *reset gate*), $r^{(t)}$, primjenjuje se prema izrazu (27).

$$r^{(t)} = \sigma\left(W_{rhh}\, h^{(t-1)} + W_{rxh}\, x^{(t)} + b_{rh}\right) \tag{27}$$

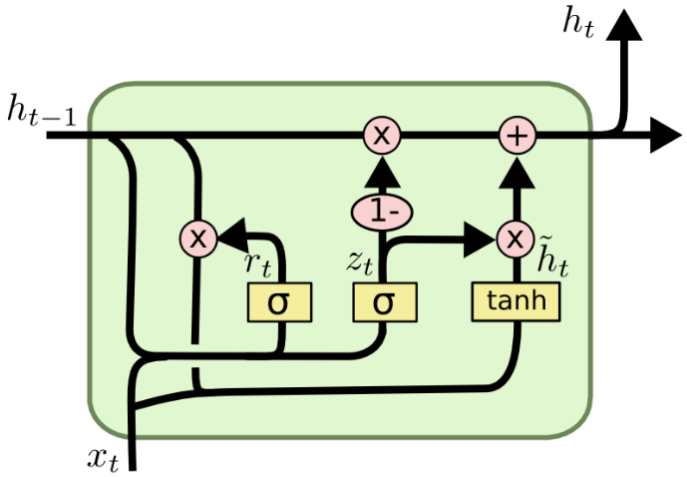
Također, uvodi se privremeno stanje $\hat{h}^{(t)}$ koje se računa pomoću izraza (28).

$$\hat{h}^{(t)} = \sigma\left(W_{hh}\left(r^{(t)} \odot h^{(t-1)}\right) + W_{xh}\, x^{(t)} + b_h\right) \tag{28}$$

Konačno se stanje $h^{(t)}$ računa prema izrazu (29) [19].

$$h^{(t)} = u^{(t)}\, h^{(t-1)} + \left(1 - u^{(t)}\right) \hat{h}^{(t)} \tag{29}$$

Tijek primjene propusnica i računanja stanja prikazan je na slici ispod (Slika 4.16).




Slika 4.16 Propusna povratna ćelija [16]

### 4.3.4. Nedostaci povratnih modela

Bitno je za napomenuti da čak i kada se uklone problemi eksplodirajućeg i nestajućeg gradijenta prilagođavanjem osnovne RNN mreže u LSTM ili GRU mrežu, treniranje i dalje ima nestabilnu prirodu te je učenje ovisnosti između jako udaljenih podataka teško [4].

## 4.4. Generativni modeli

Među najnovijim trendovima istraživanja dubokog učenja su tzv. generativni modeli koji uključuju generativne suparničke mreže i difuzijske modele, familije modela koje su se popularizirale zbog efikasnosti u generiranju sintetičkih slika. Porast popularnosti potaknuo je istraživanja primjene na sekvence podataka, uključujući vremenske serije [4].

### 4.4.1. Generativne suparničke mreže (GAN)

**Generativne suparničke mreže** (engl. *generative adversarial network*, GAN) sastoje se od dvije neuronske mreže: **generatora** (engl. *generator*) i **diskriminatora** (engl. *discriminator*). Generator i diskriminator se treniraju suparničkim pristupom. Uloga generatora je sintetiziranje podataka iz distribucije stvarnih podataka, a uloga diskriminatora je klasifikacija ulaznih podataka u stvarni ili sintetički podatak. Generator na ulaz dobiva šum koji treba transformirati u

sintetički podatak s istom distribucijom kao stvarni podaci. Diskriminator na ulaz dobiva stvarne i sintetičke podatke i računa vjerojatnost da je neki ulazni podatak stvaran, odnosno da nije sintetiziran. Treniranje generatora i diskriminatora obavlja se simultano u obliku tzv. **minimax igre s dva igrača** (engl. *minimax two-player game*). Cilj diskriminatora je maksimizirati točnost klasifikacija stvarnih i sintetičkih primjera, a generator nastoji prevariti diskriminatora i navesti ga na netočne klasifikacije, odnosno minimizirati broj točnih klasifikacija. Ovaj proces može se predstaviti funkcijom iz izraza (30).

$$\min_{G} \max_{D} V(D,G) = \mathbb{E}_{\vec{x} \sim p_x(\vec{x})}[\log D(\vec{x})] + \mathbb{E}_{\vec{z} \sim p_z(\vec{z})}[\log (1 - D(G(\vec{z})))] \quad (30)$$

U izrazu (30) je $\vec{x}$ stvarni podatak iz distribucije $p_x(\vec{x})$, $\vec{z}$ je vektor šuma iz distribucije $p_z(\vec{z})$, $D(\vec{x})$ je distribucija koju predviđa diskriminator, $G(\vec{z})$ je primjer koji je generator sintetizirao iz šuma $\vec{z}$ [4].

GAN mreže mogu se implementirati odabirom bilo koje neuronske mreže za generatora i diskriminatora, primjerice s MLP-om, CNN-om ili LSTM-om [4].

Dva su glavna pristupa kada je u pitanju primjena na predviđanje vremenskih serija. GAN mreže mogu se koristiti za sintetiziranje podataka u svrhu nadopunjavanja malih baza primjera, koje se potom koriste za predviđanje vremenskih serija nekim drugim pristupom, primjerice pomoću LSTM modela. Druga primjena je korištenje GAN mreže za predviđanje vremenske serije, tako da buduće podatke vremenske serije generira pomoću povijesnih podataka, i potencijalno pomoću egzogenih varijabli. Egzogene varijable su varijable koje nisu dio vremenske serije, ali mogu utjecati na nju [4].

### 4.4.2. Difuzijski modeli

Grupa modela koja se nedavno počela razvijati su **difuzijski modeli** (engl. *diffusion models*). Difuzijskim modelima je cilj aproksimirati proces kojim se generiraju novi primjeri podataka iz neke distribucije, pomoću postojećih podataka iz te iste distribucije [4].

Generalna pretpostavka od koje polaze difuzijski modeli je da nakon dodavanja šuma konačan broj puta na podatak iz neke distribucije, originalna distribucija toga podatka mijenja se u normalnu (Gaussovu) distribuciju. Stoga je

zadatak difuzijskog modela modelirati probabilistički proces koji aproksimira originalnu distribuciju iz normalne distribucije. Po toj pretpostavci, bilo koji uzorak iz normalne distribucije može se pomoću modeliranog probabilističkoj procesa, transformirati u novi uzorak iz distribucije ulaznog podatka [4].

Princip rada difuzijskog modela je progresivno dodavati slučajan šum na ulazni podatak **unaprijednim difuzijskim procesom** (engl. *forward diffusion process*), zatim **unatražnim reverznim difuzijskim procesom** (engl. *backward reverse diffusion process*) dobiti taj isti ulazni podatak [4].

U kontekstu predviđanja vremenskih serija, zadnjih nekoliko godina predloženo je par pristupa temeljenih na difuziji, koji primjenjuju prethodno opisano načelo generiranja podataka. U članku [4] može se pročitati više o predloženim difuzijskim modelima.

# 5. Dugoročno predviđanje vremenskih serija

## 5.1. Modeli temeljeni na pozornosti

Modeli temeljeni na pozornosti omogućuju neuronskoj mreži da se fokusira samo na dio ulazne sekvence podataka pri generiranju izlaza, i time rješavaju problem modela kojima performanse pate kada rade s jako dugačkim ulaznim podacima. Tzv. mehanizmi pažnje mogu se ugraditi u razne modele kako bi se riješili spomenutog efekta pri radu s dugačkim ulaznim podacima [2].

### 5.1.1. Transformeri

**Transformer** (engl. *transformer*) je duboka neuronska mreža inicijalno razvijena za obradu prirodnog jezika. Za razliku od tradicionalnih povratnih modela poput RNN-a i LSTM-a, koji procesiraju serijske podatke sekvencijalno, transformeri mogu procesirati cijelu sekvencu podataka odjednom pomoću mehanizma koji se zove **pozornost** (engl. *attention*). Ovaj mehanizam omogućuje paralelnu obradu sekvence podataka i ima sposobnost praćenja ovisnosti u serijskim podacima koji su prostorno ili vremenski jako udaljeni [4].

Jedan od nedostataka klasičnih povratnih modela je nemogućnost paralelnog procesiranja serijskih podataka, jer izlazna stanja ovisne o prethodno izračunatim stanjima. Transformeri nemaju ovaj problem jer ne moraju sekvencijalno procesirati podatke. Druga prednost transformera nad klasičnim povratnim modelima je njihova sposobnost praćenja odnosa između serijskih podataka koji su međusobno jako udaljeni. Informacije se ne gube kao u RNN i LSTM mrežama kada se trebaju prenositi kroz velik broj ulaznih primjera. Dakle, uz pomoć istovremene obrade podataka i mehanizma pozornosti, transformeri uspješno rješavaju navedene probleme klasičnih povratnih modela [4].

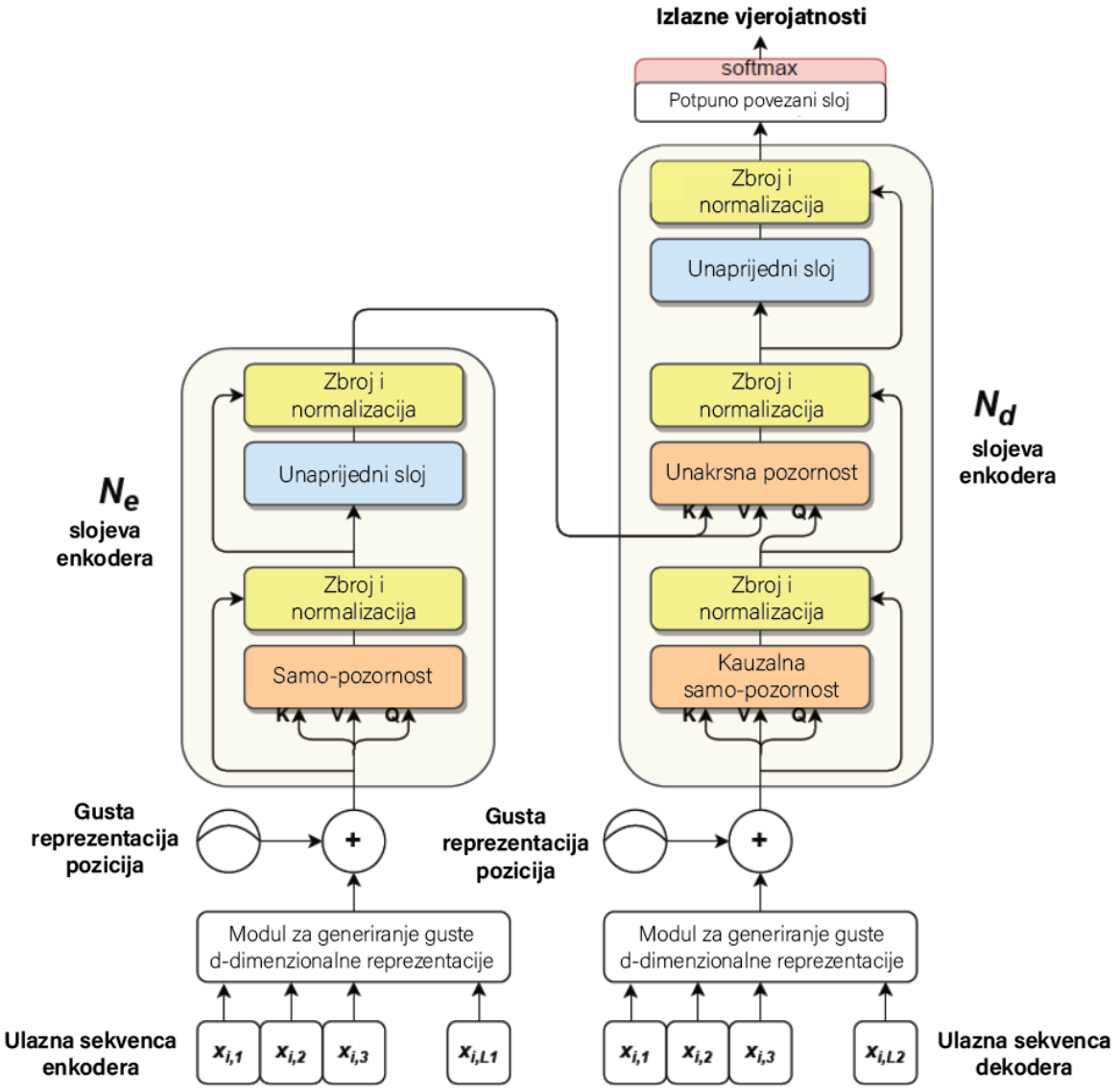
Ulaz u transformer su sekvence povijesnih primjera podataka s vremenskim poretkom duljine $\mathrm{L}$, gdje je svaki primjer reprezentiran $\mathrm{d}$-dimenzionalnim vektorom, što se može zapisati izrazom (31).

$$\mathrm{X} = (\overrightarrow{\mathrm{x}_1}, \overrightarrow{\mathrm{x}_2}, \dots, \overrightarrow{\mathrm{x}_\mathrm{L}}), \overrightarrow{\mathrm{x}_\mathrm{i}} \in \mathbb{R}^\mathrm{d} \tag{31}$$

Bitno je za spomenuti da mehanizmi pozornosti inherentno ne uzimaju u obzir vremenski redoslijed sekvence. Transformer vidi ulaze više kao skup primjera nego kao uređenu sekvencu primjera. Za zadatke u kojima je poredak ulaznih podataka bitan, postoje metode kojima se može kodirati prostorni i/ili vremenski poredak i dodati na gustu reprezentaciju ulaza, kao u izrazu (32) [4].

$$\text{ulaz} = \mathrm{F}(\mathrm{X}) + \mathrm{E}_{\text{pos}} \tag{32}$$

Arhitektura transformera prati generalnu enkoder-dekoder strukturu mreže, prikazanu na donjoj slici (Slika 5.1). Enkoder je dio mreže koji procesira ulaznu sekvencu i generira skrivenu reprezentaciju, a dekoder prima izlaz enkodera i pomoću njega generira izlaznu sekvencu podataka.




Slika 5.1 Arhitektura transformera [4]

Matrica $\mathrm{F}(\mathrm{X}) \in \mathbb{R}^{\mathrm{L}} \otimes \mathbb{R}^{\mathrm{D}}$ je projekcija ulazne sekvence u $\mathrm{D}$-dimenzionalnu reprezentaciju. U slučaju vremenskih serija, koristi se $1\mathrm{D}$ konvolucijski sloj s $\mathrm{D}$ naučenih jezgri kako bi se dobila $\mathrm{D}$-dimenzionalna reprezentacija svakog primjera. Gusta reprezentacija redoslijeda primjera $\mathrm{E}_{\mathrm{pos}}$ može se naučiti ili fiksno zadati [4].

Ulaz enkodera mreže koja radi predviđanje vremenske serije su svi povijesni primjeri $\mathrm{X}$, dok je ulaz dekodera najčešće najnoviji dio primjera, primjerice druga polovica svih primjera, i vektor nula čija duljina odgovara duljini predikcije $\mathrm{P}$. Enkoder se sastoji od $\mathrm{N}_{\mathrm{e}}$ slojeva, a dekoder od $\mathrm{N}_{\mathrm{d}}$ slojeva. Pritom je izlaz jednog sloja, ulaz idućeg sloja [4].

Svaki enkoder sastoji se od 2 dijela. Prvi dio enkodera računa **samo-pozornost** (engl. *self-attention*), odnosno za svaki ulazni primjer računa odnos s ostalima u sekvenci. Drugi dio enkodera radi nelinearnu projekciju svog ulaza. Za efikasnije propagiranje gradijenta i učenje, svaki dio enkodera na svoj izlaz nadodaje ulaz tog dijela enkodera uz rezidualnu vezu, te se provodi normalizacija primjera u normalnu distribuciju s naučenom srednjom vrijednosti i standardnom devijacijom [4].

Struktura svakog od dekodera slična je opisanoj strukturi enkodera, s time da ima jedan dodatan dio. Prvi dio dekodera računa posebnu vrstu samo-pozornosti koja se zove **kauzalna** ili **maskirana samo-pozornost** (engl. *causal or masked self-attention*). Ova vrsta pozornosti osigurava da se procesiranje svakog primjera radi samo u odnosu na primjere koji su prije njega u redoslijedu, i zanemaruje, odnosno stavlja masku na primjere koji su u budućnosti. Nakon toga, u drugom dijelu dekodera primjenjuje se **unakrsna pozornost** (engl. *cross-attention*), koja kao ulaze prima izlaz prošlog dijela dekodera, odnosno rezultat primjene kauzalne pozornosti, i izlaza iz posljednjeg sloja enkodera, odnosno skrivenu reprezentaciju enkodera. Posljednji dio dekodera isto kao enkoder radi nelinearnu projekciju izlaza prethodnog dijela, koji je u slučaju dekodera izlaz unakrsne pozornosti. Povrh toga, svaki od dijelova dekodera omotan je rezidualnim povezivanje (dodavanjem ulaza tog dijela enkodera na izlaz uz rezidualnu vezu) i normalizacijom sloja, kao što je to u oba dijela enkodera. Posljednji dio arhitekture transformera je predaja izlaza iz dekodera sloju koji računa konačnu predikciju [4].

### 5.1.2. Mehanizmi pozornosti

Najbitniji dio arhitekture transformera je spomenuti mehanizam pozornosti, koji daje modelu sposobnost stavljanja fokusa na određene dijelove ulaza. Od brojnih definicija pozornosti, transformeri koriste tzv. **pozornost skaliranjem skalarnog umnoška** (engl. *scaled dot-product attention*). Za računanje pozornosti potrebna su sljedeća tri elementa:

- Skup upita $\mathrm{Q} \in \mathbb{R}^{\mathrm{M}} \otimes \mathbb{R}^{\mathrm{D_k}}$,
- Skup ključeva $\mathrm{K} \in \mathbb{R}^{\mathrm{N}} \otimes \mathbb{R}^{\mathrm{D_k}}$,
- Skup vrijednosti $\mathrm{V} \in \mathbb{R}^{\mathrm{N}} \otimes \mathbb{R}^{\mathrm{D_v}}$.

$\mathrm{D_k}$ i $\mathrm{D_v}$ odnose se na dimenzije prostora u koji su projicirani upiti, ključevi i vrijednosti. $\mathrm{N}$ je kardinalnost, odnosno veličina skupova ključeva i vrijednosti, a $\mathrm{M}$ je kardinalnost skupa upita. Izlaz se za svaki od upita računa prema izrazu (33).

$$\mathrm{Y} = \mathrm{Pozornost}(\mathrm{K}, \mathrm{V}, \mathrm{Q}) = \mathrm{softmax}\left(\frac{\mathrm{QK}^{\mathrm{T}}}{\sqrt{\mathrm{D_k}}}\right)\mathrm{V} \tag{33}$$

Izlaz pozornosti $\mathrm{Y} \in \mathbb{R}^{\mathrm{M}} \otimes \mathbb{R}^{\mathrm{D_v}}$ je matrica koja u $\mathrm{i}$-tom retku sadrži rezultat $\mathrm{i}$-tog upita. Od navedenih skupova, ključevi i vrijednosti najčešće su isti vektori, odnosno vektor vrijednosti odgovara svom ključu. Kao što je u prethodnom dijelu poglavlja opisano, transformer koristi dvije vrste pozornosti, samo-pozornost i unakrsnu pozornost. U slučaju samo-pozornosti, upiti i vrijednosti su isti vektori. U slučaju unakrsne pozornosti, upiti su izlazi prethodnog dijela dekodera, dok su ključevi i vrijednosti izlazi enkodera, odnosno skrivene reprezentacije iz enkodera [4].

**Pozornost s više glava** (engl. *multi-head attention*, MHA) naprednija je verzija prethodno spomenute pozornosti izračunate skaliranjem skalarnog umnoška. Proširenje pozornosti na više glava znači da umjesto da se pozornost računa s obzirom na jedan aspekt sekvence, model može primjenjivati pozornost s obzirom na više reprezentacija, odnosno projekcija podataka [4].

Usporedba obične pozornosti i pozornosti s više glava prikazana je na slici u nastavku (Slika 5.2). Svaka glava pozornosti ima tri naučene matrice koje koristi za projiciranje ključeva, vrijednosti i upita:

- $W_i^K \in \mathbb{R}^{D \times D_k}$ za projiciranje ključeva,
- $W_i^V \in \mathbb{R}^{D \times D_v}$ za projiciranje vrijednosti,
- $W_i^Q \in \mathbb{R}^{D \times D_Q}$ za projiciranje upita.

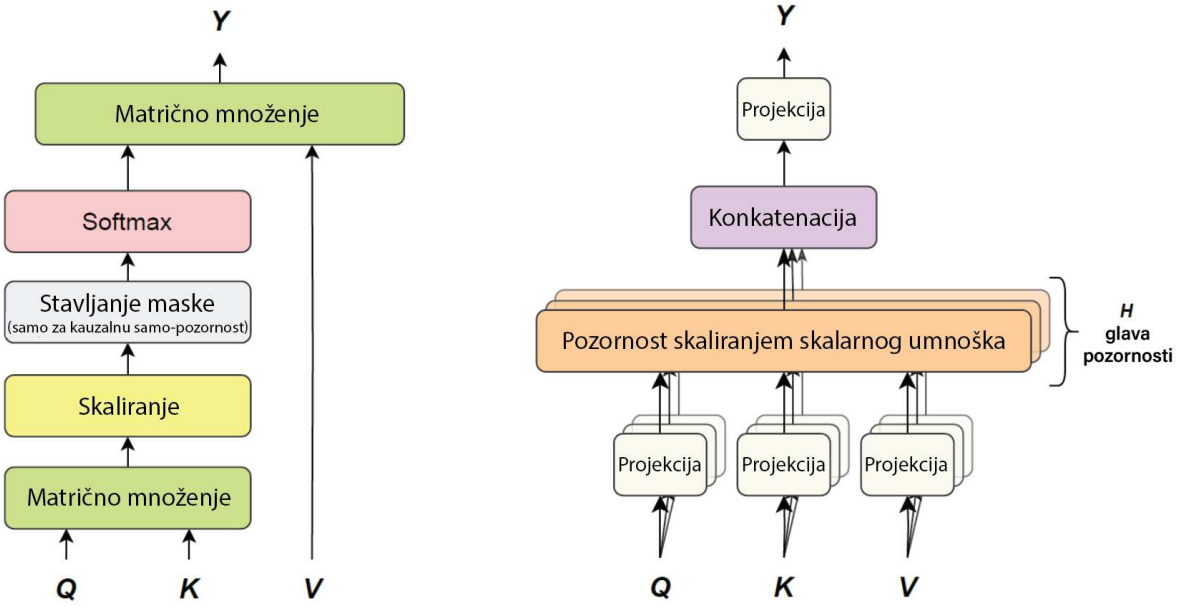
Svaka glava računa pozornost skaliranjem ključeva, vrijednosti i upita prema izrazu (34).

$$h_i = \text{Pozornost}\left(KW_i^K, VW_i^V, QW_i^Q\right) \tag{34}$$

Konačni rezultat dobije se konkatenacijom izlaza svake od glava i projekcijom u prostor dimenzija prethodnog sloja, prema izrazu (35).

$$Y = \text{MHA}(K, V, Q) = \text{Konkatenacija}(h_1, h_2, \ldots, h_H)\, W^O \tag{35}$$

Dakle, svaki od parova projekcija nad ključevima, vrijednostima i upitima, i pozornosti koja se primjenjuje nad njima, naziva se **glava pozornosti** (engl. *attention head*) [4].




Slika 5.2 Pozornost skaliranjem skalarnog umnožaka (lijevo) i pozornost s $H$ glava (desno)

### 5.1.3. Nedostaci transformera

Tri su glavna nedostatka transformera koji su relevantni za primjenu na predviđanje vremenskih serija:

- Manjak osjetljivosti na lokalni kontekst,
- Memorijsko usko grlo,

- Složenost vremena izvođenja [4].

Neosjetljivost na lokalni kontekst podataka izazvana je načinom rada mehanizma pozornosti. Pozornost stavlja fokus na specifične dijelove ulazne sekvence podataka i zbog toga ne zahvaća lokalni kontekst i susjedstvo pojedinih podataka. To je osobito problem za predviđanje vremenskih serija za koje je lokalni kontekst podataka ključan za prepoznavanje trendova i uzoraka među njima. Bez ispravno izgrađenog razumijevanja lokalnih interakcija među podacima, teško je istrenirati model koji dobro generalizira [4].

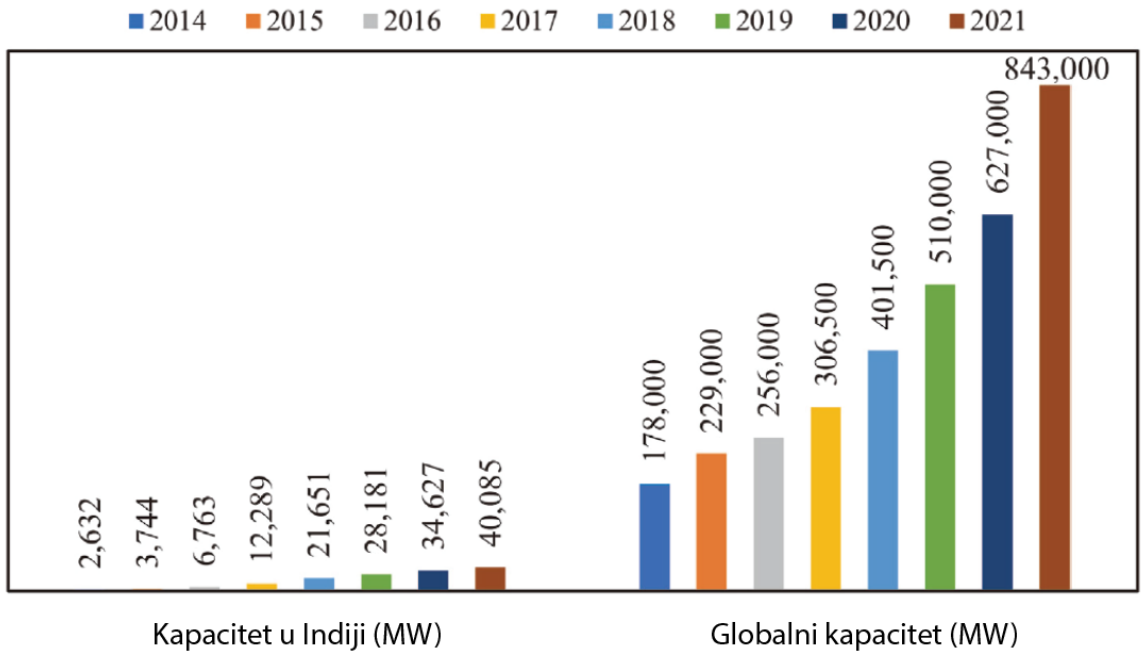
Memorijsko usko grlo uzrokovano je istovremenom obradom cijele sekvence ulaznih podataka. Za sekvencu duljine $L$, prostorna složenost transformera je $O(L^2)$, što znači da memorijski resursi potrebni za treniranje transformera rastu kvadratno s povećanjem duljine sekvence. Kvadratni rast memorijskih resursa potrebnih za obradu dugih sekvenci otežava treniranje transformera, odnosno, duljina sekvenci ograničena je količinom memorije koja je dostupna za treniranja modela. Vremenska složenost treniranja jednaka je prostornoj složenosti pa isti problem vrijedi za vrijeme potrebno za treniranje modela. S rastom sekvenci, raste vrijeme treniranja, stoga se ograničenje na duljinu sekvence također postavlja u slučajevima kada je bitna brzina treniranja i generiranja izlaza modela [4].

# 6. Predviđanje proizvodnje solarne energije

U ovom poglavlju prvo je dana motivacija za predviđanje proizvodnje solarne energije. Zatim su u drugom dijelu poglavlja navedeni izvori podataka o meteorološkim prilikama na određenom području te izvori povijesnih podataka o proizvodnji solarne energije, iz fotonaponskih stanica i solarnih elektrana.

## 6.1. Primjena

Proizvodnja električne energije iz nekonvencionalnih izvora eksponencijalno raste zadnjih desetak godina. Posljedica ovoga je smanjenje potrošnje fosilnih goriva, cijene proizvodnje struje, onečišćenja okoliša i ispušnih plinova [5]. Generiranje energije putem fotonaponskih sustava ističe se zbog svoje sposobnosti smanjenja emisija $CO_2$, što ga čini učinkovitim rješenjem u borbi protiv klimatskih promjena te prijelazu prema čišćoj i održivijoj energetskoj budućnosti [15]. Rast ukupnog kapaciteta instaliranih fotonaponskih sustava u Indiji i globalno od 2014. do 2021. godine prikazan je donjim grafom (Slika 6.1) [5].




Slika 6.1 Rast instalacija fotonaponskih sustava u Indiji i globalno [5]

Priroda proizvodnje fotonaponske energije povremena je i nestabilna zbog svoje direktne ovisnosti o vremenskim prilikama kao što su temperatura zraka,

količina vjetra i solarna radijacija, koje se mijenjaju ovisno o dnevnom ciklusu i izmjeni godišnjih doba. Zbog takve nestabilne prirode, korisno je pokušati predvidjeti proizvodnju fotonaponske energije. To je posebno važno kada je fotonaponski sustav dio energetske mreže, jer manjak proizvodnje može uzrokovati probleme s opskrbom i kvalitetom električne energije. Osim toga, rad elektrana najčešće je uvjetovan dugoročnim ugovorima pa svaka nepredviđena promjena u proizvodnji može dovesti do novčanih gubitaka [15].

## 6.2. Podaci potrebni za predviđanje solarne energije

Za treniranje modela za predviđanje solarne energije potrebna je kombinacija meteoroloških podataka i povijesnih podataka o proizvodnji energije. U idućim odjeljcima prvo će se navesti izvori meteoroloških podataka, nakon čega slijede izvori povijesnih podataka o proizvodnji energije solarnih elektrana i fotonaponskih stanica.

### 6.2.1. Meteorološki podaci

Neki od internetskih izvora meteoroloških podataka su:

1. *Visual Crossing API* s podacima u jednosatnim i jednodnevnim intervalima (https://www.visualcrossing.com/)
2. *Meteomatics API* (https://www.meteomatics.com/en/weather-api/)

### 6.2.2. Povijesni podaci o proizvodnji solarne energije

Neki od internetskih izvora povijesnih podataka o proizvodnji solarne energije:

1. Institut NIST (engl. *National Institute of Technology*), skup meteoroloških podataka i podataka o proizvodnji fotonaponske stanice na kampusu instituta, u intervalima od $1\ \text{min}$ i $1\ \text{s}$ (https://pvdata.nist.gov/)
2. NREL (engl. *National Renewable Energy Laboratory*) katalog podataka (https://data.nrel.gov/submissions)
3. *The Dutch PV Portal* sadrži velik broj prikupljenih skupova podataka o proizvodnji fotonaponske energije (https://www.tudelft.nl/en/ewi/over-de-

faculteit/afdelingen/electrical-sustainable-energy/photovoltaic-materials-and-devices/dutch-pv-portal/pv-power-databases)

4. NASA (https://power.larc.nasa.gov/)
5. Mendeley Data (https://data.mendeley.com/datasets/dbh93b6vp8/1)

# 7. Sustav za predviđanje proizvodnje solarne energije

U ovom poglavlju opisuje se sustav za predviđanje proizvodnje solarne energije. Postupak razvoja sustava sastoji se od nekoliko koraka. Prvi korak je priprema podataka, što je detaljno opisano u prvom dijelu poglavlja. Drugi dio poglavlja posvećen je odabiru značajki relevantnih za predviđanje proizvodnje solarne energije. Treći odjeljak opisuje treniranje modela odabranog za predviđanje proizvodnje energije. Posljednji korak je vrednovanje dobivenih rezultata, što je prikazano u posljednjem odjeljku.

## 7.1. Priprema podataka

Za predviđanje proizvodnje energije neke solarne elektrane potrebni su povijesni podaci o meteorološkim prilikama na području elektrane, te povijesni podaci o proizvodnji energije.

Podaci korišteni u ovom radu preuzeti su s internetske stranice *Kaggle* [20]. Prema opisu podataka na web stranici, solarna elektrana se nalazi u mjestu Irecê u saveznoj državi Bahia na sjeveroistoku Brazila. Naveden je članak [6] u kojem su podaci originalno korišteni. Dozvoljeno je dijeljenje i prilagođavanje podataka u bilo koje svrhe, uključujući komercijalne.

Podaci su organizirani u dvije mape: jednu za meteorološke podatke i drugu za podatke o generiranoj energiji, prikupljene u razdoblju od srpnja 2018. godine do srpnja 2019. godine. Meteorološki podaci su prikupljeni u intervalima od jedne minute, a podaci o generiranoj energiji prikupljeni su u razmacima od 15 minuta.

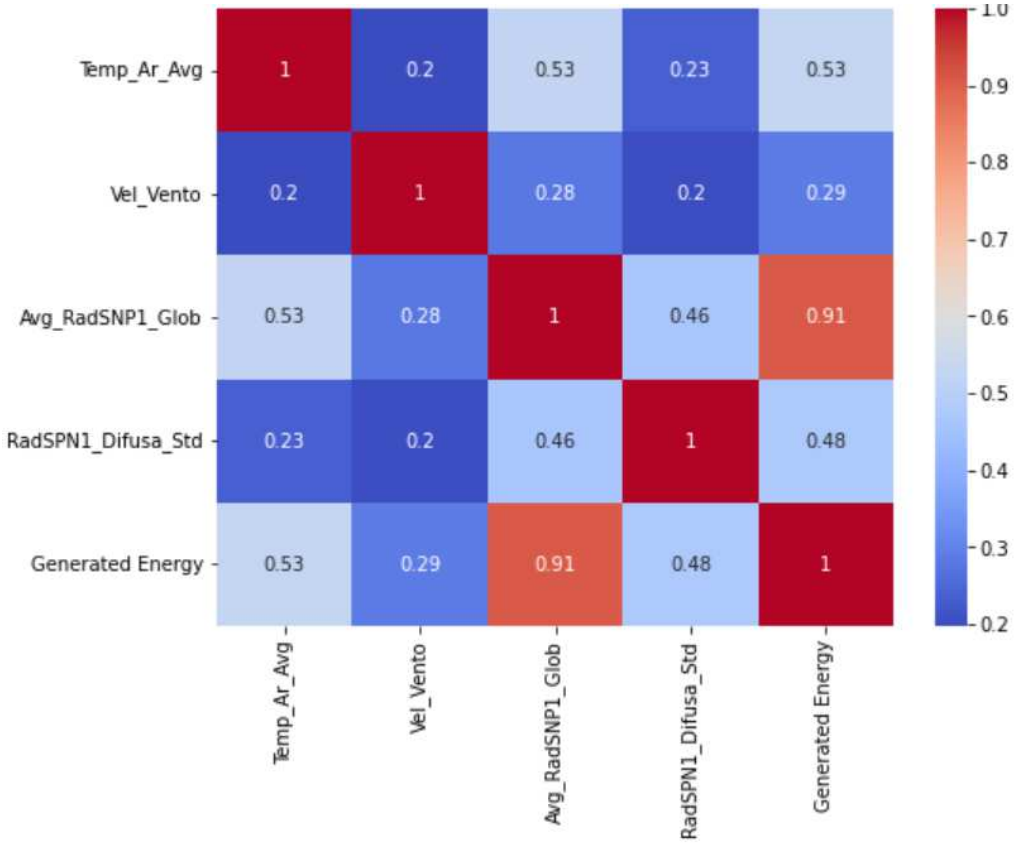
Mapa s meteorološkim podacima sadrži CSV (engl. *comma separated values*) datoteke, od kojih svaka sadrži podatke prikupljene za određeni mjesec iz spomenutog razdoblja od srpnja 2018. do srpnja 2019 godine. Mapa s datotekama koje sadrže podatke o proizvodnji energije organizirana je na isti način. Za učitavanje podataka iz datoteka korištena je biblioteka *pandas* [21]. Meteorološki podaci objedinjeni su u strukturu podataka *Dataframe* iz biblioteke *pandas*. Isto je napravljeno za podatke o proizvedenoj energiji.

Meteorološke podatke čine vrijednosti kao što su temperatura zraka, brzina vjetra, razina vlage u zraku i sl., te vrijeme u kojem su ti podaci prikupljeni. Podaci o proizvodnji energije sadrže razne vrijednosti, primjerice temperatura uređaja, generirana snaga, generirana energija, napon, itd., te vrijeme u kojem su podaci prikupljeni. Ispravljanje grešaka u podacima uključuje dodavanje izostavljenih zareza u nekimod primjera podataka, uklanjanje praznih stupaca i dodavanje imena stupaca koji nemaju naziv.

Nakon uspješnog čišćenja obje skupine podataka, objedinjuju u jedan *Dataframe* tako da se poklapaju u vremenskim trenucima u kojima su prikupljeni.

## 7.2. Odabir značajki

Nakon objedinjavanja podataka potrebno je provjeriti ima li značajki koje su višak, generiranjem tzv. matrice korelacija koja računa koeficijente korelacije između parova značajki. Izbacivanjem viška značajki, matrica korelacije izgleda kao na slici (Slika 7.1).




Slika 7.1 Matrica korelacija između značajki

## 7.3. Treniranje LSTM modela

Podaci su podijeljeni na skup za treniranje, validacijski skup i skup za testiranje, u omjerima 70:15:15. Arhitektura mreže sastoji se od dva LSTM sloja i potpuno povezanog sloja s jednim izlazom. Broj neurona u LSTM slojevima je 32 i 64. Za sastavljanje mreže korištena je biblioteka *keras* [20]. Za optimizacijski algoritam odabran je Adam te je srednja kvadratna pogreška (engl. *mean squared error*, MSE) odabrana za funkciju pogreške. Provedena je unakrsna provjera tijekom koje je zabilježen najbolji model dobiven kroz treniranje u 20 epoha, za veličinu grupe 256. Na slikama ispod vide se pogreške modela na skupu za učenje i skupu za validaciju (Slika 7.2), te graf pogrešaka (Slika 7.2).

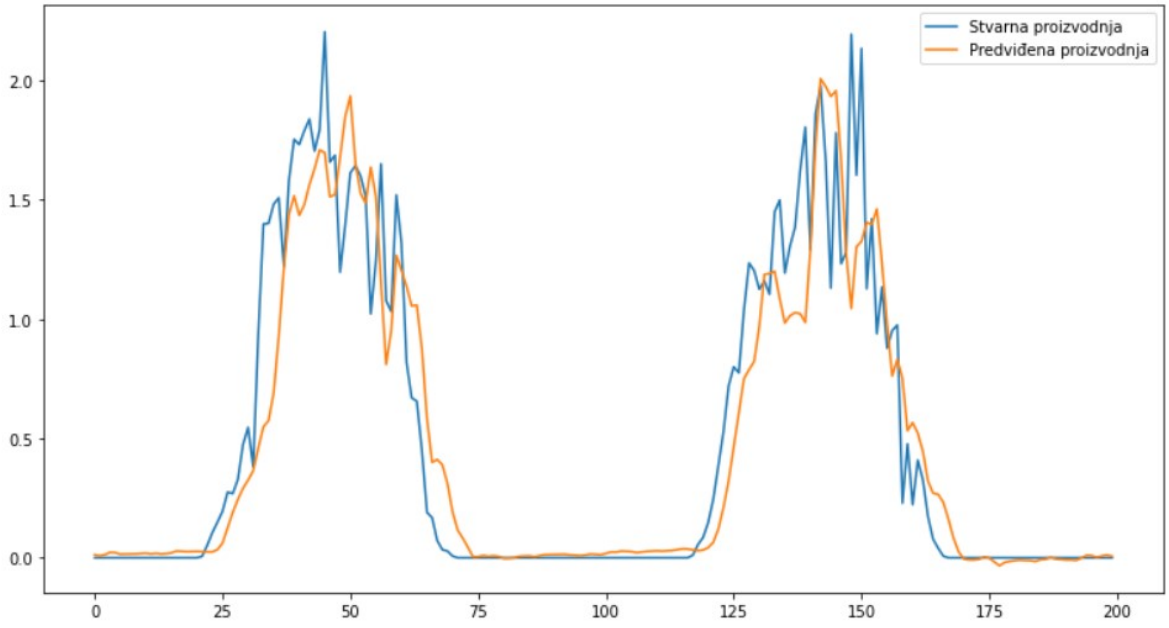
```
Epoch 1/20
77/77 [==============================] - 5s 23ms/step - loss: 0.0271 - val_loss: 0.0122
Epoch 2/20
77/77 [==============================] - 1s 15ms/step - loss: 0.0075 - val_loss: 0.0119
Epoch 3/20
77/77 [==============================] - 1s 15ms/step - loss: 0.0071 - val_loss: 0.0114
Epoch 4/20
77/77 [==============================] - 1s 12ms/step - loss: 0.0067 - val_loss: 0.0114
Epoch 5/20
77/77 [==============================] - 1s 16ms/step - loss: 0.0065 - val_loss: 0.0107
Epoch 6/20
77/77 [==============================] - 1s 16ms/step - loss: 0.0062 - val_loss: 0.0104
Epoch 7/20
77/77 [==============================] - 1s 15ms/step - loss: 0.0060 - val_loss: 0.0103
Epoch 8/20
77/77 [==============================] - 1s 16ms/step - loss: 0.0059 - val_loss: 0.0105
Epoch 9/20
77/77 [==============================] - 1s 15ms/step - loss: 0.0059 - val_loss: 0.0103
Epoch 10/20
77/77 [==============================] - 1s 16ms/step - loss: 0.0059 - val_loss: 0.0101
Epoch 11/20
77/77 [==============================] - 1s 14ms/step - loss: 0.0058 - val_loss: 0.0101
Epoch 12/20
77/77 [==============================] - 1s 13ms/step - loss: 0.0057 - val_loss: 0.0102
Epoch 13/20
77/77 [==============================] - 1s 14ms/step - loss: 0.0058 - val_loss: 0.0102
Epoch 14/20
77/77 [==============================] - 1s 15ms/step - loss: 0.0057 - val_loss: 0.0100
Epoch 15/20
77/77 [==============================] - 1s 14ms/step - loss: 0.0058 - val_loss: 0.0102
Epoch 16/20
77/77 [==============================] - 1s 14ms/step - loss: 0.0058 - val_loss: 0.0101
Epoch 17/20
77/77 [==============================] - 1s 13ms/step - loss: 0.0057 - val_loss: 0.0101
Epoch 18/20
77/77 [==============================] - 1s 16ms/step - loss: 0.0056 - val_loss: 0.0100
Epoch 19/20
77/77 [==============================] - 1s 15ms/step - loss: 0.0057 - val_loss: 0.0102
Epoch 20/20
77/77 [==============================] - 1s 17ms/step - loss: 0.0057 - val_loss: 0.0099
```

Slika 7.2 Greške na skupu za treniranje i skupu za validaciju kroz epohe


Slika 7.3 Graf pogrešaka modela na skupovima za treniranje i validaciju

## 7.4. Vrednovanje modela

Nakon treniranja modela, provjereni su rezultati na skupu za testiranje. Graf s predviđenim vrijednostima kroz dva dana vidi se na slici ispod (Slika 7.4).




Slika 7.4 Predviđanja na skupu za testiranje za dva puna dana

MSE greška na skupu za testiranje je 0.086, korijen srednje kvadratne pogreške (engl. *root mean squared error*, RMSE) iznosi 0.293, te srednja apsolutna pogreška (engl. *mean absolute error*, MAE) iznosi 0.175.

# Zaključak

Metode dubokog učenja za predviđanje vremenskih serija dijele se na one prikladne za kratkoročno predviđanje i one prikladne za dugoročno predviđanje.

Duboki modeli prikladni za kratkoročno predviđanje su konvolucijski modeli, povratni modeli i generativni modeli. Među konvolucijske modele spada obična CNN mreža i specijalizirana TCN mreža. Od povratnih modela efikasne su LSTM i GRU mreže. Generativni modeli koji su prikladni su GAN mreže i posebna vrsta generativnog modela koji se zove difuzijski model. Duboki modeli prikladni za dugoročno predviđanje su modeli temeljeni na pozornosti, konkretnije transformer mreže.

Za problem predviđanja vremenske serije podataka o proizvodnji energije solarne elektrane, LSTM mreža pokazala se uspješnom arhitekturom.

# Literatura


[1] Goodfellow, I., Bengio Y., Courville, A. *Deep Learning*, 2016.

[2] Torres, J. F., Hadjout, D., Sebaa, A., Martínez-Álvarez, F. *Deep Learning for Time Series Forecasting: A Survey*, 2020.

[3] Miller, J. A., Aldosari, M., Saeed, F., Barna, N. H., Rana, S., Arpinar, I. B., Liu, N. *A Survey of Deep Learning and Foundation Models for Time Series Forecasting*, 2024.

[4] Casolaro, A., Capone, V., Iannuzo, G., Camastra, F. *Deep Learning for Time Series Forecasting: Advances and Open Problems*, 2023.

[5] Dhaked, D. K., Dadhich, S., Birla, D. *Power output forecasting of solar photovoltaic plant using LSTM*, 2023.

[6] Ribero, K., Santos, R., Saraiva, E., Rajagopal, R. *A Statistical Methodology to Estimate Soiling Losses on Photovoltaic Solar Plants*. Journal of Solar Energy Engineering, 2021.

[7] *Artificial Intelligence - Neural Networks*. Poveznica: https://www.tutorialspoint.com/artificial_intelligence/artificial_intelligence_neural_networks.htm; pristupljeno 27.6.2024.

[8] Šnajder, J., Bašić, B. D. *Strojno učenje*, 2014.

[9] Hyndman, R. J., Athanasopoulos, G. *Forecasting: Principles and Practice, 3rd ed*, 2021.

[10] Baheti, P. *Activation Functions in Neural Networks [12 Types & Use Cases]*, 2021. Poveznica: https://www.v7labs.com/blog/neural-networks-activation-functions, pristupljeno 27.6.2024.

[11] Chen, B. *Why Rectified Linear Unit (ReLU) in Deep Learning and the best practice to use it with TensorFlow*, 2021. Poveznica: https://towardsdatascience.com/why-rectified-linear-unit-relu-in-deep-learning-and-the-best-practice-to-use-it-with-tensorflow-e9880933b7ef, pristupljeno 27.6.2024.

[12] Ramadhan, L. *Neural Network: The Dead Neuron*, 2021., Poveznica: https://towardsdatascience.com/neural-network-the-dead-neuron-eaa92e575748, pristupljeno 27.6.2024.

[13] *Activation functions in neural networks [Updated 2024]*, 2023., Poveznica: https://www.superannotate.com/blog/activation-functions-in-neural-networks, pristupljeno: 27.6.2024.

[14] Poveznica: https://www.w3schools.com/ai/ai_perceptrons.asp, pristupljeno: 27.6.2024.

[15] Castillo-Rojas, W., Quispe, F. M., Hernández, C. *Photovoltaic Energy Forecast Using Weather Data through a Hybrid Model of Recurrent and Shallow Neural Networks*, 2023.

[16] Olah, C. *Understanding LSTM Networks*, 2015., Poveznica: https://colah.github.io/posts/2015-08-Understanding-LSTMs, pristupljeno 27.6.2024.

[17] Krapac, J., Šegvić, S. *Konvolucijski modeli*, Predavanja iz kolegija Duboko učenje 1, 2020.

[18] Tutek, M. *Povratne neuronske mreže*, Predavanja iz kolegija Duboko učenje 1, 2020.

[19] Tutek, M. *Napredne povratne neuronske mreže*, Predavanja iz kolegija Duboko učenje 1, 2020.

[20] Skup podataka *Solar2-PV-Power-Plant-Irece*, 2021., Poveznica: https://www.kaggle.com/datasets/lscadfacomufms/solar2photovoltaicsolarplantdatairece/data, pristupljeno: 12.6.2024.

[21] Dokumentacija biblioteke *pandas*, poveznica: https://pandas.pydata.org/docs/

[22] Dokumentacija biblioteke *keras*: poveznica: https://keras.io/

# Sažetak

U ovom radu opisan je razvoj dubokog učenja kroz tri glavne faze istraživanja. Kronološki su navedeni napredci tijekom tih faza, kao i relevantni koncepti za današnje stanje tehnologije dubokog učenja. Nakon općeg pregleda, rad se bavi predviđanjem vremenskih serija, definirajući koncept i navodeći njegove različite primjene. Detaljno su opisani modeli dubokog učenja pogodni za predviđanje vremenskih serija. Uvod uključuje temeljno objašnjenje duboke unaprijedne mreže, njezin proces učenja i osnovu za razumijevanje složenijih dubokih modela. Modeli su kategorizirani prema prikladnosti za kratkoročno i dugoročno predviđanje vremenskih serija. Analizirane su karakteristike i nedostaci tih modela u kontekstu predviđanja vremenskih serija. Razvijen je i dokumentiran sustav zasnovan na dubokom učenju za predviđanje proizvodnje električne energije solarne elektrane na temelju povijesnih i meteoroloških podataka. Prikazani su i komentirani rezultati treniranja LSTM modela za predviđanje proizvodnje solarne energije.

Ključne riječi: duboko učenje, vremenska serija, duboka unaprijedna mreža, ćelija s dugoročnom memorijom, solarna energija

# Summary

This thesis provides a comprehensive overview of the development of deep learning through three main phases of research. The chronological progression of these phases is documented, highlighting key concepts relevant to the current state of deep learning technology. Following this general overview, the thesis delves into time series forecasting, defining the concept and listing its various applications. Deep learning models suitable for time series forecasting are introduced and described in detail. The introduction includes a fundamental explanation of the deep feedforward network, its learning process, and the basis for understanding more complex deep models. Models are categorized based on their suitability for short-term and long-term time series forecasting. The characteristics and shortcomings of these models in the context of time series forecasting are analyzed. A deep learning-based system for predicting the power generation of a solar power plant using historical and meteorological data is developed and documented. The performance of an LSTM model in forecasting solar power production is presented and discussed.

Keywords: deep learning, time series, deep feedforward network, long short-term memory, solar power

# Skraćenice

| | | |
|---|---|---|
| MLP | Multilayer perceptron | višeslojni perceptron |
| SGD | Stochastic gradient descent | stohastički gradijentni spust |
| CNN | Convolutional neural network | konvolucijska neuronska mreža |
| RNN | Recurrent neural network | povratna neuronska mreža |
| LSTM | Long short-term memory | ćelija s dugoročnom memorijom |
| GRU | Gated recurrent unit | propusna povratna ćelija |
| GAN | Generative adversarial network | generativna suparnička mreža |