Graf neuronske mreže za primjenu u višerobotskim sustavima

Matušin, Mia Sara

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 1673

GRAF NEURONSKE MREŽE ZA PRIMJENU U VIŠEROBOTSKIM SUSTAVIMA

Mia Sara Matušin

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 1673

GRAF NEURONSKE MREŽE ZA PRIMJENU U VIŠEROBOTSKIM SUSTAVIMA

Mia Sara Matušin

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 4. ožujka 2024.

ZAVRŠNI ZADATAK br. 1673

Pristupnica:	**Mia Sara Matušin (0036542853)**
Studij:	Elektrotehnika i informacijska tehnologija i Računarstvo
Modul:	Računarstvo
Mentorica:	izv. prof. dr. sc. Tamara Petrović

Zadatak: **Graf neuronske mreže za primjenu u višerobotskim sustavima**

Opis zadatka:

Mnogi se zadaci višerobotskih sustava mogu opisati primjenom grafova. Primjerice, ako se više mobilnih robota primjenjuje u skladištima, njihova mreža prometnica može se opisati kao graf te se onda kretanje robota temelji na određenim svojstvima grafa i puteva svakog vozila. Graf neuronske mreže razlikuju se od običnih neuronskih mreža jer kao ulazne podatke imaju grafove. Radom na završnom zadatku potrebno je kao prvi korak proučiti područje graf neuronskih mreža te način implementacije. Potom je potrebno osmisliti kako odrediti sigurnost stanja u višerobotskom sustavu primjenom graf neuronskih mreža.

Rok za predaju rada: 14. lipnja 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 1673

GRAF NEURONSKE MREŽE ZA PRIMJENU U VIŠEROBOTSKIM SUSTAVIMA

Mia Sara Matušin

Zagreb, lipanj, 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 4. ožujka 2024.

ZAVRŠNI ZADATAK br. 1673

Pristupnica:	**Mia Sara Matušin (0036542853)**
Studij:	Elektrotehnika i informacijska tehnologija i Računarstvo
Modul:	Računarstvo
Mentorica:	izv. prof. dr. sc. Tamara Petrović

Zadatak: **Graf neuronske mreže za primjenu u višerobotskim sustavima**

Opis zadatka:

Mnogi se zadaci višerobotskih sustava mogu opisati primjenom grafova. Primjerice, ako se više mobilnih robota primjenjuje u skladištima, njihova mreža prometnica može se opisati kao graf te se onda kretanje robota temelji na određenim svojstvima grafa i puteva svakog vozila. Graf neuronske mreže razlikuju se od običnih neuronskih mreža jer kao ulazne podatke imaju grafove. Radom na završnom zadatku potrebno je kao prvi korak proučiti područje graf neuronskih mreža te način implementacije. Potom je potrebno osmisliti kako odrediti sigurnost stanja u višerobotskom sustavu primjenom graf neuronskih mreža.

Rok za predaju rada: 14. lipnja 2024.

Zahvaljujem svojoj mentorici Izv. Prof. Dr. Sc. Tamari Petrović na podršci, stručnom vodstvu i dragocjenim savjetima tijekom izrade ovog završnog rada.

Veliko hvala i mojim prijateljima i obitelji na beskrajnoj ljubavi i podršci tijekom cijelog mog školovanja. S vama je sve bilo lakše.

Sadržaj

1. Uvod

Neuronske mreže, kao moćan alat umjetne inteligencije, našle su široku primjenu u različitim područjima poput prepoznavanja uzoraka, klasifikacije i predikcije. Njihova sposobnost učenja iz podataka i prilagodbe složenim zadacima čini ih idealnim za rješavanje problema koji zahtijevaju visok stupanj apstrakcije i prepoznavanja obrazaca. Posebno su korisne u analizi grafova, struktura koje predstavljaju odnose između entiteta, te se koriste u mnogim primjenama kao što su društvene mreže, biološki sustavi i komunikacijske mreže. Neki primjeri literature koja je korisna za razumijevanje neuronskih mreža su *"Deep Learning"* [1] te *"Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering"* [2].

Grafovi su matematičke strukture koje se sastoje od čvorova povezanih bridovima. Ova struktura omogućuje modeliranje složenih odnosa i interakcija među elementima, što je izuzetno korisno u različitim znanstvenim i inženjerskim disciplinama. U posljednje vrijeme, grafovi su postali ključni alat u razvoju višerobotskih sustava, posebice u skladištima i logističkim centrima, gdje se koriste za optimizaciju kretanja i raspodjelu zadataka među robotima.

Višerobotski sustavi su skupine robota koje surađuju kako bi postigli zajednički cilj. Ovi sustavi imaju brojne prednosti, uključujući povećanu učinkovitost, fleksibilnost i otpornost na kvarove. Postoji nekoliko vrsta višerobotskih sustava, uključujući autonomne mobilne robote u skladištima, koordinirane robotske ruke u proizvodnji i robotske timove za istraživanje nepristupačnih terena. U ovom radu, fokus je na višerobotske sustave u skladištima, gdje je optimizacija kretanja robota ključna za povećanje učinkovitosti i smanjenje zagušenja.

Graf neuronske mreže koriste strukturu grafova kako bi omogućile dubinsko učenje

na podacima koji imaju inherentne odnose između elemenata. Ove mreže posebno su prikladne za probleme gdje je važno uzeti u obzir međusobne odnose između elemenata, kao što su društvene mreže, molekularne strukture i višerobotski sustavi.

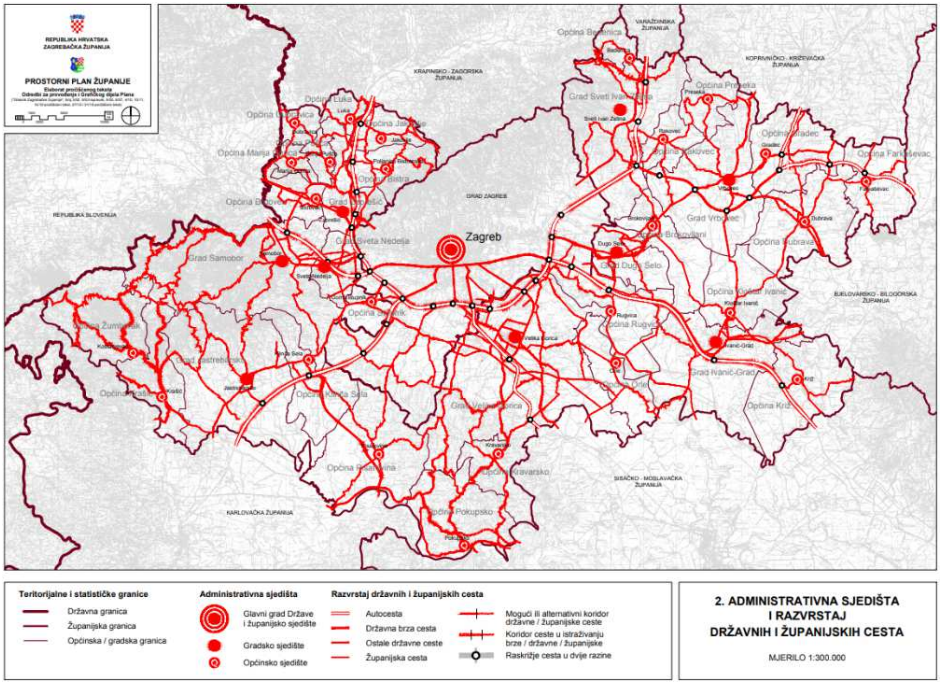
Ovaj rad se fokusira na izgradnju GNN-a koji može: a) Izračunati Fiedlerovu vrijednost grafa, što je korisno za analizu povezanosti mreže. b) Predvidjeti područje zagušenja prometa u višerobotskom sustavu u skladištu, što je ključno za optimizaciju operacija i smanjenje vremena čekanja.

U sljedećem poglavlju, opisani su grafovi, njihova svojstva i primjene. Treće poglavlje pruža detaljan pregled neuronskih mreža i kako one funkcioniraju, s posebnim naglaskom na graf neuronske mreže. Četvrto poglavlje opisuje metodologiju korištenu za izradu mreže za predikciju zagušenja prometa u višerobotskim sustavima i mreže za predikciju Fiedlerove vrijednosti grafa te rezultate i analizu učinkovitosti oba modela. Na kraju, peto poglavlje donosi zaključke i smjernice za budući rad.

2. Grafovi

2.1. Uvod u grafove

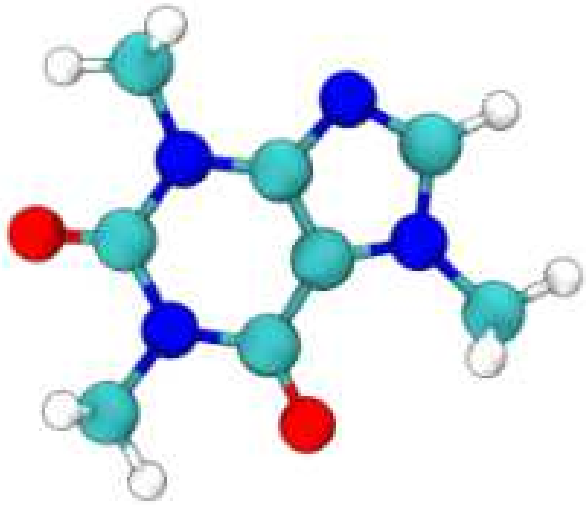
Graf je temeljni matematički pojam. Mnoge pojave na svijetu mogu se opisati grafovima. Primjerice, mreža prometnica u svakom gradu je zapravo graf, kemijske molekule mogu se prikazati kao graf, čak i neke pojave koje se intuitivno ne bi prikazale na taj način, mogu se prikazati kao graf, kao što su na primjer fotografije ili rečenice.



Slika 2.1. Mreža (graf) prometnica u Zagrebačkoj županiji [3]

Na slici 2.1. prikazana je mreža prometnica u Zagrebačkoj županiji, za koju se može primijetiti da ima sva obilježja grafa koja će se objasniti u nastavku.

Isto tako, na slici 2.2. prikazana je molekula kofeina (kemijska formula: $C_8H_{10}N_4O_2$) u obliku grafa gdje svaki čvor predstavlja jedan atom kemijskog elementa



Slika 2.2. Prikaz molekule kofeina kao grafa [4] tamno plava - dušik (N), svijetlo plava - ugljik (C), crvena - kisik (O), bijela - vodik (H)

Graf je nelinearna matematička struktura koja se sastoji od skupa čvorova (engl. *nodes*) i skupa bridova (engl. *edges*). Graf označavamo slovom G, a on je uređeni par skupova (V, E). V označava skup čvorova, dok E označava skup bridova.

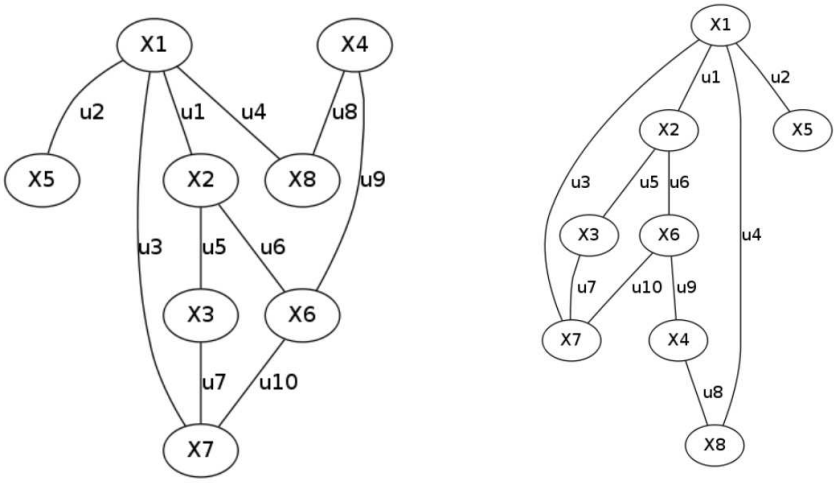
Čvor je osnovni element grafa i može biti povezan s drugim čvorovima u grafu pomoću bridova. *Stupanj čvora* jednak je broju bridova kojima je on krajnja točka, odnosno broj čvorova s kojima je povezan bridovima.

$$\deg(v) = \left|\{u \in V \mid (u, v) \in E \text{ ili } (v, u) \in E\}\right| \tag{2.1}$$

U jednadžbi (2.1) formalno je opisan stupanj čvora. Bitno je naglasiti da je pozicija čvorova i bridova prilikom crtanja grafova u potpunosti nebitna sve dok se poštuju odnosi u grafu.

Na slici 2.3. obje vizualizacije prikazuju isti graf, iako izgledaju drugačije. Ne postoji točna i netočna vizualizacija, prikaz ovisi o tome za što se graf koristi i što se želi istaknuti i bolje prikazati.

Graf u kojemu između svaka dva čvora postoji put je povezani graf. To ne znači da svi čvorovi moraju biti izravno povezani, nego da se prolaženjem kroz graf može doći od



Slika 2.3. Dvije vizualizacije istoga grafa [5]

bilo kojeg čvora do bilo kojeg čvora. Formalno bi se to opisalo na sljedeći način: graf $G = (V, E)$ je povezan ako za svaki par $u, v \in V$ postoji put od u do v. Graf u kojemu između svaka dva čvora postoji brid naziva se potpuni graf. Iznimka je brid koji povezuje čvor sa samim sobom, on može izostati. Suprotno od toga, graf u kojem postoje čvorovi između kojih nema puta, naziva se *nepovezani* graf.

2.2. Svojstva grafova

Put P u grafu G definiran je kao niz čvorova $v_1, v_2, \ldots, v_n$ takav da za svaki i od 1 do $n-1$ vrijedi da je $(v_i, v_{i+1}) \in E$.

Susjedstvo čvorova opisuje koji čvorovi su međusobno povezani. Susjedstvo čvora $v \in V$, označeno s $N(v)$, skup je svih čvorova koji su povezani s v jednim bridom.

$$N(v) = \{u \in V \mid (u, v) \in E\} \tag{2.2}$$

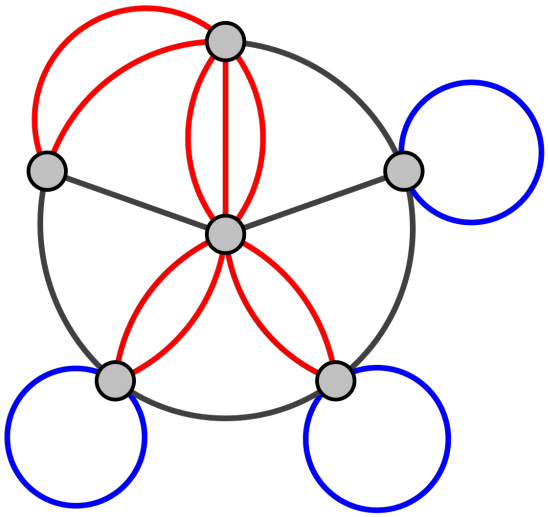
Formalni zapis susjedstva čvorova vidljiv je u jednadžbi (2.2). Najčešće se susjedstvo prikazuje matricom susjedstva. Matrica susjedstva grafa $G = (V, E)$ kvadratna je matrica A dimenzija $n \times n$, gdje je n broj čvorova u grafu.

$$a_{ij} = \begin{cases} 1 & \text{ako postoji brid između čvorova } v_i \text{ i } v_j, \\ 0 & \text{inače.} \end{cases} \tag{2.3}$$

U jednadžbi (2.3) prikazano je kako se definiraju elementi matrice A.

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix}, \tag{2.4}$$

Ako su čvorovi grafa označeni s $v_1, v_2, \ldots, v_n$, matrica susjedstva A je oblika (2.4) gdje je a_{ij} element na presjeku i-tog reda i j-tog stupca matrice. Ako u grafu postoji više bridova koji povezuju dva ista čvora, onda je to *multigraf*. Njegova matrica susjedstva osim 0 i 1, može imati i bilo koji drugi cijeli broj koji određuje broj bridova između 2 čvora.



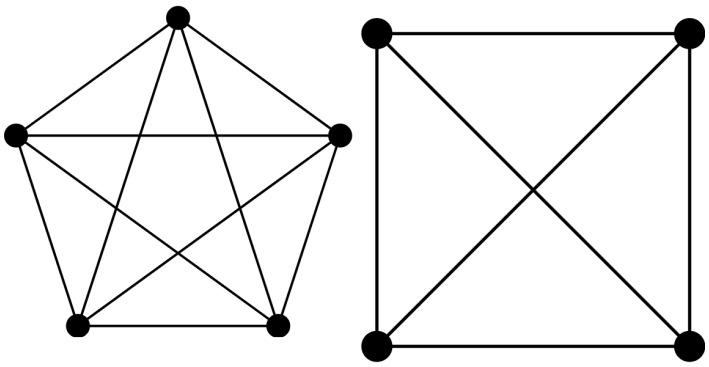
Slika 2.4. Primjer multigrafa [6]

Na slici 2.4. prikazan je primjer multigrafa. Graf se naziva *planarnim* ako se može nacrtati u ravnini bez presijecanja bridova.

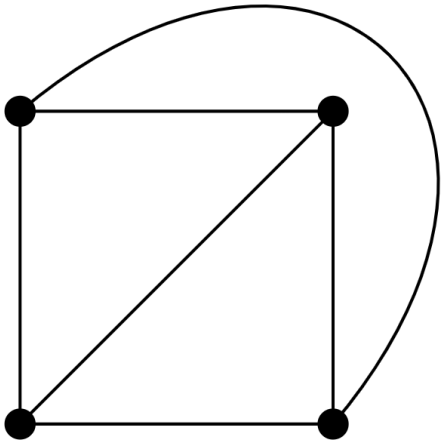
Može se uočiti da na slici 2.5. lijevi graf nije moguće nacrtati na način da se neki od njegovih bridova ne sijeku, dok se desni graf može nacrtati kako je prikazano na slici 2.6.

Graf je *regularan* stupnja R ako i samo ako svi njegovi čvorovi imaju isti stupanj R.

Dva grafa $G_1 = (V_1, E_1)$ i $G_2 = (V_2, E_2)$ nazivaju se *izomorfnim* ako postoji bijektivna



Slika 2.5. Primjer neplanarnog (lijevo) i planarnog (desno) grafa [7]

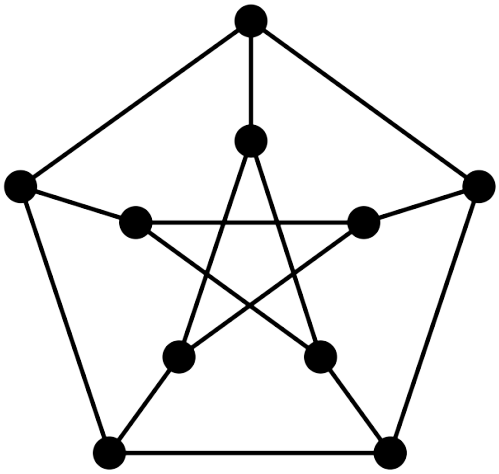


Slika 2.6. Planarna reprezentacija desnog grafa s prošle slike [7]

funkcija $f : V_1 \to V_2$ koja čvorovima grafa G_1 pridružuje čvorove grafa G_2, tako da se očuva struktura grafa.

$$f \text{ je izomorfizam između } G_1 \text{ i } G_2 \iff \forall u, v \in V_1 : (u, v) \in E_1 \iff (f(u), f(v)) \in E_2 \tag{2.5}$$

Formalna definicija izomorfizma između grafova prikazana je jednadžbom (2.5) gdje (u, v) predstavlja brid između čvorova u i v u grafu G_1, a $(f(u), f(v))$ predstavlja odgovarajući brid između čvorova $f(u)$ i $f(v)$ u grafu G_2. Također je bitno spomenuti *reznu točku* (engl. *articulation point*) te *most* (engl. *bridge*). Rezna točka svaki je čvor čije bi brisanje iz grafa rezultiralo razdvajanjem grafa na dva odspojena podgrafa. Slično, most je svaki brid koji bi postigao takav rezultat. *Podgraf grafa* $G = (V, E)$ je graf $G' = (V', E')$ gdje je V' podskup skupa V i E' podskup skupa E. Formalno, podgraf G' se može definirati kao $G' = (V', E')$, gdje je $V' \subseteq V$ i $E' \subseteq E$, pri čemu svaki čvor i svaki brid u E' pripadaju grafu G', odnosno, oba čvora svakog brida u E' nalaze se u skupu V'.

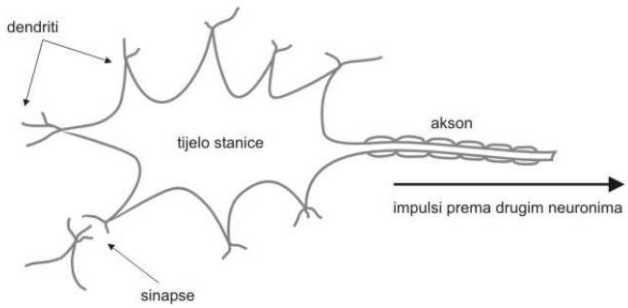


Slika 2.7. Primjer regularnog grafa je Petersenov graf [8]

Usmjereni graf odnosno *digraf* $G = (V, E)$ sastoji se od skupa čvorova V i skupa usmjerenih bridova E. Svaki usmjereni brid e u grafu G definiran je kao uređeni par čvorova (u, v), gdje je u početni čvor i v krajnji čvor. *Ulazni stupanj čvora* (engl. *in-degree*) broj je bridova koji završavaju u njemu. *Izlazni stupanj čvora* (engl. *out-degree*) broj je bridova kojima je on početna točka. *Izvor* je čvor kojemu je ulazni stupanj nula, a čvor kojemu je izlazni stupanj nula nazivamo *ponor*. Usmjereni je graf *čvrsto povezan* ako se iz bilo kojeg čvora može doći u svaki drugi. S druge strane, ako bi graf postao povezan kad bi se bridovi zamijenili neusmjerenim bridovima, onda je on *slabo povezan*. Usmjereni je graf regularan stupnja R ako svaki čvor ima isti ulazni i izlazni stupanj R.

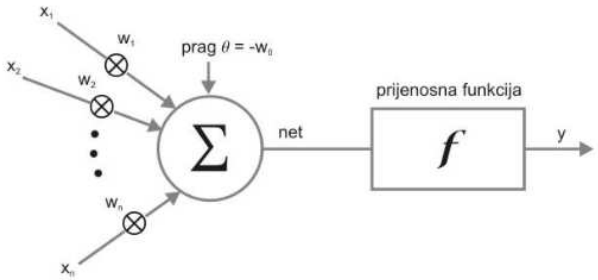
3. Neuronske mreže

Neuronske mreže pokušaj su imitacije rada ljudskog mozga. Funkcioniraju na osnovu neurona koji međusobno komuniciraju i razmjenjuju informacije što im omogućuje učenje i stvaranje novih informacija, baš kao što to radi i ljudski mozak. Biološki neuron u ljudskom mozgu sastoji se od 4 dijela: tijela stanice (soma), skupa dendrita (ogranaka), aksona (dugačkih cjevčica koje prenose električke poruke) i niza završnih članaka. Svaki od tih elemenata ima svoju ulogu.



Slika 3.1. Primjer neurona u ljudskom mozgu [9]

Na slici 3.1. prikazani su i imenovani svi dijelovi neurona u ljuskom mozgu. Po uzoru na njega, nastao je *McCulloch-Pitts umjetni neuron* (engl. *Threshold Logic Unit, TLU*). Sastoji se od: ulaznih signala, težine ulaznih signala, praga(engl. *bias*), prijenosne odnosno aktivacijske funkcije te izlaznog signala.



Slika 3.2. McCulloch-Pitts model umjetnog neurona, TLU [9]

Na slici 3.2. prikazana je građa umjetnog TLU neurona. Ulazni signali, njih ukupno n, označeni su s $x_1, x_2, \ldots, x_n$, i oni čine vektor ulaznih podataka $\mathbf{x}$. *Težine* se označavaju s $\omega_1, \omega_2, \ldots, \omega_n$. Težine ulaza zapravo reprezentiraju jakost sinapse, odnosno važnost ulaznog signala.

$$net = \sum_{i=1}^{n} x_i \cdot \omega_i - \theta \tag{3.1}$$

Jednadžbom (3.1) prikazana je definicija težinske sume. Ona je analogna sumiranju potencijala u tijelu stanice u biološkom neuronu. No često se dogovorno uzima da je vrijednost praga $\theta = -\omega_0$ te se dodaje ulazni signal x_0 s fiksiranom vrijednošću 1.

$$net = \sum_{i=0}^{n} \omega_i x_i = \omega_0 x_0 + \omega_1 x_1 + \omega_2 x_2 + \ldots + \omega_n x_n \tag{3.2}$$

Jednadžba (3.2) jednostavniji je zapis funkcije *net*. Prag (engl. *bias*) dodatni je parametar koji se koristi za uređivanje izlazne vrijednosti neovisno o ulaznim podacima pa tako omogućava pomicanje prijenosne funkcije, veću fleksibilnost i sprječava simetričnost. Neuron proizvodi izlazni signal pomoću prijenosne funkcije. Ona uvodi nelinearnost, normalizira izlaz, omogućava diferencijabilnost i može se prilagoditi specifičnim potrebama različitih vrsta zadataka te je stoga vrlo korisna i potrebna kako bi se mogle konstruirati i učiti složenije neuronske mreže. Izlaz y rezultat je prijenosne funkcije f: $y = f(\text{net})$.

3.1. Učenje neuronske mreže

Prije korištenja mreže za bilo kakve predikcije, potrebno je provesti postupak *učenja* odnosno *treniranja* mreže. Znanje o izlazu kao funkciji ulaza pohranjeno je implicitno u težinama veza između neurona, gdje se tijekom postupka učenja te težine prilagođavaju ovisno o ulazu. Postoji nekoliko vrsta učenja:

- **nadzirano učenje** (engl. *supervised learning*) - na ulaz su proslijeđeni ulazni podaci i očekivani izlaz u obliku $(ulaz, izlaz) = (\mathbf{x}, y)$ te treba pronaći preslikavanje $\hat{y} = f(\mathbf{x})$
- **nenadzirano učenje** (engl. *unsupervised learning*) - na ulazu su samo ulazni po-

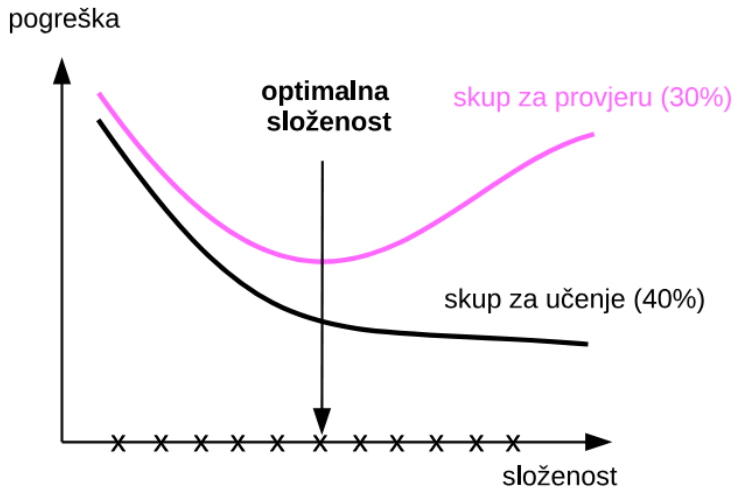
daci, a mreža sama treba pronaći pravilnost u podacima

- **podržano učenje** (engl. *reinforcement learning*) - mreža uči optimalnu strategiju na temelju pokušaja s odgođenom nagradom

Skup primjera za učenje u pravilu se dijeli na tri skupa:

- **skup za učenje** (engl. *training set*) - na njemu mreža pronalazi pravilnosti i korigira težine veza
- **skup za provjeru** (engl. *validation set*) - služi za odabir mreže s optimalnom složenosti kako se model ne bi prenaučio, tijekom učenja računa pogrešku modela i prati kada će funkcija pogreške početi rasti
- **skup za testiranje** (engl. *testing set*) - nakon što se mreža nauči, na ovome se skupu računa točnost modela

Prenaučenost stanje je modela neuronske mreže u kojem savršeno predviđa na skupu za učenje, ali vrlo loše na skupu za testiranje što znači da se taj model previše prilagodio podacima za učenje. Iz tog razloga potrebno je odabrati model koji je što jednostavniji, ali što bolje predviđa nad neviđenim podacima, po uzoru na *Occamovu britvu*.



Slika 3.3. Pogreška i složenost modela na skupu za učenje [10]

3.2. Algoritam propagacije pogreške unatrag

Postupak učenja neuronske mreže temelji se na postupku *propagacije pogreške unatrag* (engl. *error backpropagation*) koji se temelji na izračunu parcijalnih derivacija i njihovoj primjeni u korigiranju težina veza. Parcijalne derivacije govore o tome kako će se promijeniti funkcija pogreške ako se poveća težina veze. S pretpostavkom da je za funkciju pogreške odabrano srednje kvadratno odstupanje koje izgleda ovako:

$$E = \frac{1}{2} \sum_{s=1}^{N} \frac{1}{N} \sum_{i=1}^{No} (ts^{(i)} - os^{(i)})^2 \tag{3.3}$$

- $\frac{1}{2}$: faktor $\frac{1}{2}$ uključen je zbog pojednostavljenja parcijalne derivacije tijekom izračuna gradijenta
- $\sum_{s=1}^{N}$: vanjska suma preko svih uzoraka s u skupu podataka, gdje je N ukupan broj uzoraka u skupu podataka
- $\frac{1}{N}$: faktor normalizacije koji se koristi za izračun prosječne pogreške po uzorku
- $\sum_{i=1}^{No}$: unutarnja suma preko svih izlaznih neurona i, gdje je No ukupan broj izlaznih neurona u mreži
- $ts^{(i)}$: ciljana vrijednost (engl. *target value*) za uzorak s i izlazni neuron i, ovo je stvarna vrijednost
- $os^{(i)}$: izlazna vrijednost (engl. *output value*) mreže za uzorak s i izlazni neuron i, ovo je vrijednost predikcije koju mreža zapravo proizvede

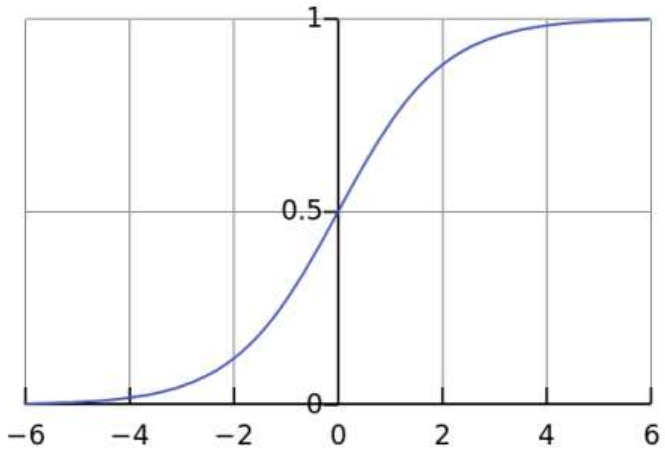
te da je kao prijenosna funkcija korištena sigmoidalna funkcija:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{3.4}$$

postupak izgleda ovako:

Korak 1: Inicijalizacija

- Inicijalizirati sve težine neuronske mreže na slučajne vrijednosti.



Slika 3.4. Grafički prikaz sigmoidalne funkcije [11]

Korak 2: Iterativni postupak

- ponavljati dok nije zadovoljen uvjet zaustavljanja:
  - za svaki uzorak s: $(xs^{(1)}, \ldots, xs^{(Ni)}) \rightarrow (ts^{(1)}, \ldots, ts^{(No)})$ u skupu za učenje, napraviti sljedeće korake:
    1. postaviti podatak $(xs^{(1)}, \ldots, xs^{(Ni)})$ na ulaz mreže
       - $xs^{(i)}$: ulazne vrijednosti uzorka s, gdje je i indeks ulaznog neurona
       - Ni: ukupan broj ulaznih neurona
    2. izračunati izlaze svih neurona u svim slojevima, od prvog prema posljednjem
       - $y_i^{(k)}$: izlaz neurona i u sloju k
    3. označiti izlaze zadnjeg sloja kao $(os^{(1)}, \ldots, os^{(No)})$
       - $os^{(i)}$: izlazna vrijednost neurona i u zadnjem sloju za uzorak s
       - No: ukupan broj izlaznih neurona.
    4. odrediti pogreške neurona izlaznog sloja:

$$\delta_K^{(i)} = os^{(i)} \cdot (1 - os^{(i)}) \cdot (ts^{(i)} - os^{(i)}) \tag{3.5}$$

* $\delta_K^{(i)}$: pogreška izlaznog neurona i u sloju K (zadnjem sloju)

* $ts^{(i)}$: ciljna vrijednost za uzorak s i izlazni neuron i

5. vratiti se sloj po sloj prema početku mreže i izračunati pogreške za svaki neuron i u sloju k:

$$\delta_i^{(k)} = y_i^{(k)} \cdot (1 - y_i^{(k)}) \cdot \sum_{d \in \text{Downstream}} w_{i,d}^{(k)} \cdot \delta_d^{(k+1)} \tag{3.6}$$

   * $\delta_i^{(k)}$: pogreška neurona i u sloju k

   * $y_i^{(k)}$: izlaz neurona i u sloju k

   * Downstream: skup neurona u sloju $k+1$ koji su povezani s neuronom i u sloju k

   * $w_{i,d}^{(k)}$: težina veze između neurona i u sloju k i neurona d u sloju $k+1$

   * $\delta_d^{(k+1)}$: pogreška neurona d u sloju $k+1$

6. ažurirati sve težine $w_{i,j}^{(k)}$ prema izrazu:

$$w_{i,j}^{(k)} \leftarrow w_{i,j}^{(k)} + \eta \cdot y_i^{(k)} \cdot \delta_j^{(k+1)} \tag{3.7}$$

   * η: stopa učenja (engl. *learning rate*), koja određuje koliko brzo se težine prilagođavaju

   * $w_{i,j}^{(k)}$: težina veze između neurona i u sloju k i neurona j u sloju $k+1$

   * $y_i^{(k)}$: izlaz neurona i u sloju k

   * $\delta_j^{(k+1)}$: pogreška neurona j u sloju $k+1$

7. ažurirati pragove $w_{0,j}^{(k)}$ prema izrazu:

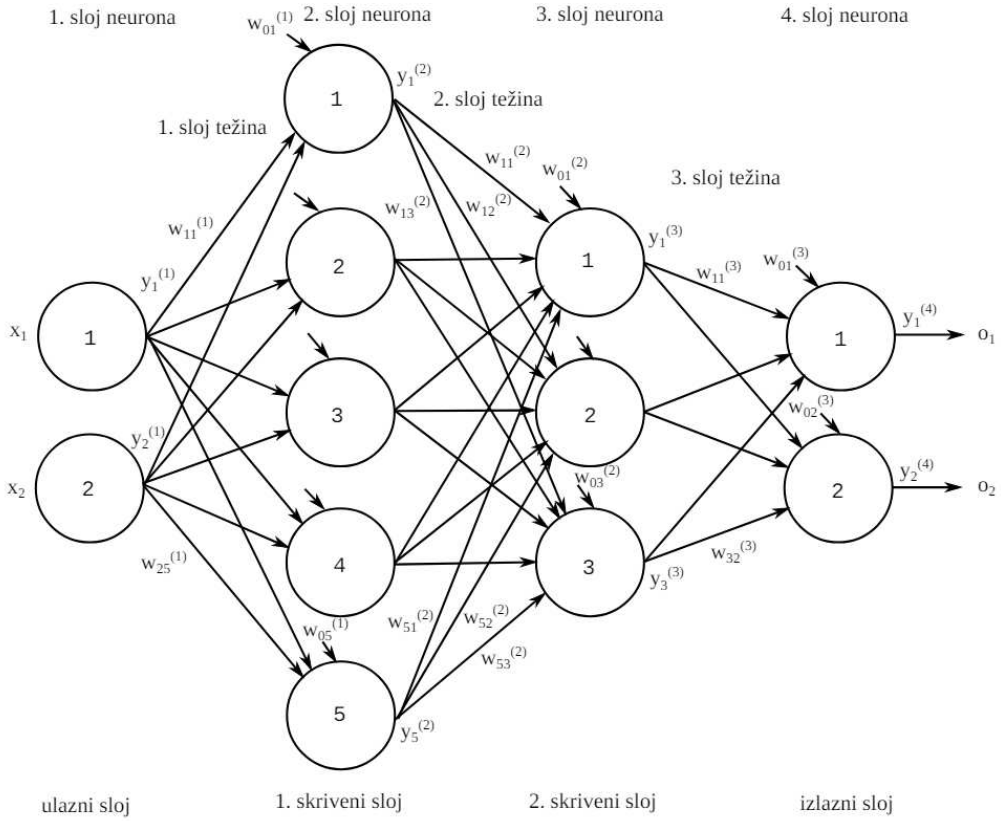
$$w_{0,j}^{(k)} \leftarrow w_{0,j}^{(k)} + \eta \cdot \delta_j^{(k+1)} \tag{3.8}$$

   * $w_{0,j}^{(k)}$: težina praga (engl. *bias weight*) za neuron j u sloju $k+1$

* $\delta_j^{(k+1)}$: pogreška neurona j u sloju $k+1$

3.3. Višeslojne neuronske mreže

Budući da neuroni mogu predočiti samo linearne funkcije, potrebno je koristiti neuronsku mrežu od više povezanih neurona za predočavanje složenijih odnosa. *Arhitektura mreže* predstavlja način na koji su neuroni organizirani i međusobno povezani u mreži te broj neurona u mreži. Mreža je *unaprijedna* kada svi neuroni na ulazu mogu dobiti samo izlaze neurona iz prethodnog sloja. Takva mreža ne sadrži povratne veze. *Povratna veza* je ona u kojoj izlaz jednog neurona može utjecati na njegov vlastiti ulaz ili ulaze prethodnih slojeva. *Potpuno-povezana* mreža je ona u kojoj svaki neuron u sloju n ima vezu sa svim neuronima iz $n-1$ sloja.



Slika 3.5. Primjer unaprijedne slojevite potpuno-povezane neuronske mreže [10]

Na slici 3.5. vidljivo je da se mreža sastoji od ulaznog, izlaznog te n skrivenih slojeva. Ulazni sloj prima ulazne podatke i proslijeđuje ih dalje kroz mrežu. Tako svaki neuron predstavlja jedan atribut ulaznih podataka, odnosno broj neurona u ulaznom sloju odgovara dimenzionalnosti ulaznih podataka. Izlazni sloj na izlazu daje konačne predikcije.

Skriveni slojevi su svi slojevi između ulaznog i izlaznog sloja. Oni zapravo omogućuju učenje neuronske mreže i izvođenje složenih nelinearnih funkcija nad ulaznim podacima.

4. Graf neuronske mreže - GNN

Graf neuronske mreže (engl. *Graph Neural Networks*, GNNs) posebna su vrsta neuronske mreže dizajnirane za rad s podacima koji se mogu predočiti u obliku grafa. Omogućuju predikciju značajki čvorova, bridova i značajki na razini cijelog grafa. Podaci mogu biti sadržani u bridovima ili čvorovima, a nazivaju se značajke ili atributi čvorova odnosno bridova i najčešće su to vektori.[12] Najveća prednost GNN-ova je to što tijekom predikcija mreža uzima u obzir međuodnose čvorova. Osim toga, mogu se prilagođavati dinamičnoj strukturi grafova, osiguravaju heterogenost, a podaci u grafovima mogu predstavljati bilo kakve objekte i njihove karakteristike te su vrlo skalabilni s obzirom da grafovi mogu postati jako kompleksni. [13]

Konvolucijske neuronske mreže (engl. *Convolutional Neural Networks*, CNNs) posebno se koriste s ulaznim podacima kao što su slike, audio signali ili govor. Sastoje se od konvolucijskog sloja (engl. *convolution layer*), sloja uzorkovanja (engl. *pooling layer*) te izlaznog potpuno-povezanog sloja (engl. *FC, fully-connected layer*). Teško ih je primijeniti na grafove zbog proizvoljne i kompliciranе topologije.[14]

Graf konvolucijske mreže (engl. *Graph Convolutional Networks*, GCNs) posebna su vrsta graf neuronskih mreža. Temeljna ideja je ažuriranje reprezentacije čvora agregiranjem i transformacijom značajki njegovih susjednih čvorova i njega samog, što je ideja iz CNN-ova. Sastoje se od konvolucijskog sloja, linearnog sloja i nelinearne funkcije aktivacije.[15]

Koraci učenja graf neuronskih mreža su sljedeći:

1. **prosljeđivanje poruka** (engl. *message passing*) odnosno graf konvolucija, u ovom koraku svaki čvor (neuron) komunicira sa svim susjednim čvorovima na tom sloju kako bi međusobno razmijenili informacije

2. **agregacija informacija od susjeda čvora** preoblikovanje je svih informacija susjeda kako bi se stvorila nova reprezentacija čvora sa svim prikupljenim informacijama

$$h_v^{(k)} = \text{AGGREGATE}\left(\left\{h_u^{(k-1)} : u \in \mathcal{N}(v)\right\}\right) \tag{4.1}$$

gdje je $h_v^{(k)}$ stanje čvora v u k-tom sloju, odnosno sve informacije koje on sadrži u tom sloju, $\mathcal{N}(v)$ je skup susjeda čvora v, a $h_u^{(k-1)}$ je stanje susjeda u u $k-1$ sloju koje se agregira u novu reprezentaciju čvora v. Najčešće korištena funkcija agregacije je suma.

3. **ažuriranje čvora s novom reprezentacijom** iz prošlog koraka

$$h_v^{(k)} = \text{UPDATE}\left(h_v^{(k-1)}, h_v^{(k)}\right) \tag{4.2}$$

gdje UPDATE funkcija kombinira agregirane informacije $h_v^{(k)}$ s trenutnim stanjem čvora $h_v^{(k-1)}$ kako bi formirala novo stanje čvora.

4. **višestruki slojevi** - ovaj postupak ponavlja se za svaki sloj mreže [16]

4.1. Predikcija broja vozila na bridovima grafa pomoću neuronskih mreža

Zadatak je predikcija broja vozila na svakom bridu grafa pomoću neuronskih mreža. Svaki brid predstavlja jednu prometnicu, a svaka se sastoji od dvije trake u oba smjera. Predikcije se provode kako bi se otkrila potencijalna mjesta zagušenja u sustavu prometnica, kao na primjer u sustavu robota u skladištima.

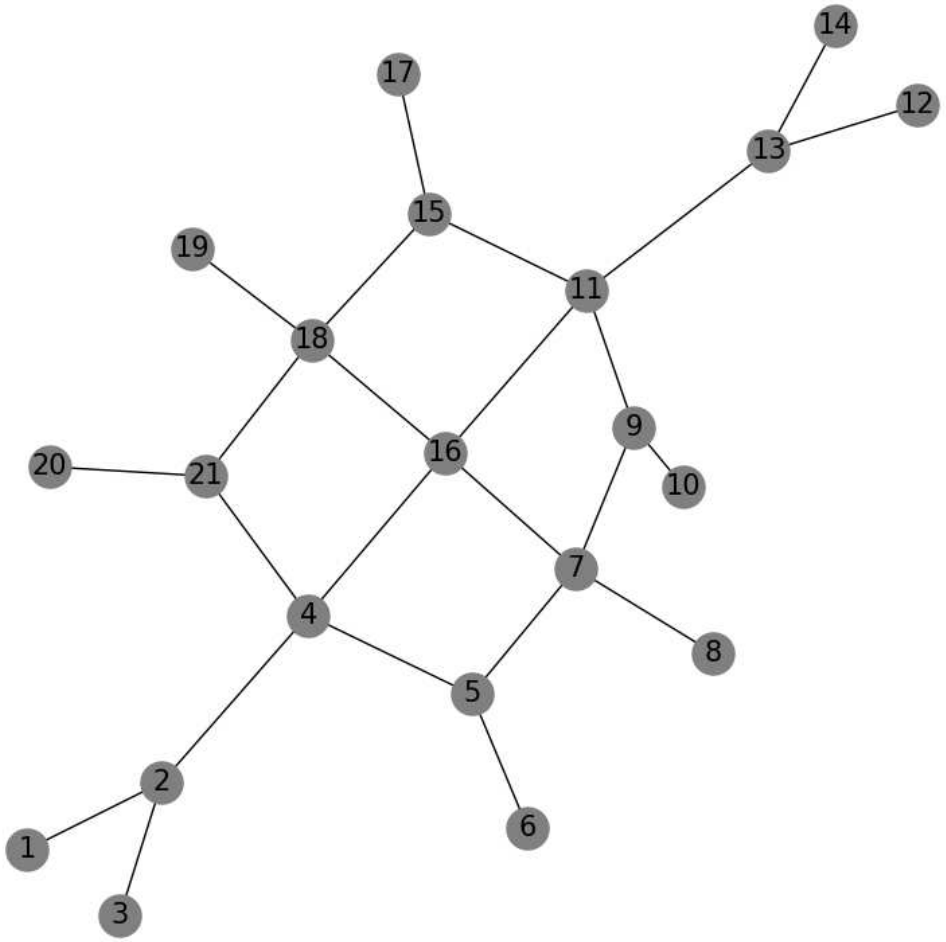
Graf $G = (V, E)$ definiran je sljedećim atributima:

- **nodes** - lista čvorova

$$V = \{1, 2, \ldots, 21\} \tag{4.3}$$

Na primjer za korišteni graf u ovom zadatku koji je prikazan na slici 4.1.:

```
"nodes" : list(range(1, 22)).
```



Slika 4.1. Graf prometnica nad kojim se rade predikcije

- **edges** - lista *tuple* podataka oblika (početni čvor, krajnji čvor), zapisani su bridovi samo u jednom smjeru

$$E = \{(u, v) \mid u, v \in V, u \neq v\} \tag{4.4}$$

Primjer za graf implementiranog sustava u programskom jeziku Python:

```
"edges" : [(1, 2), (2, 3), ... , (18, 21), (11, 16)]
```

- **edge lengths** - duljina svakog brida

$$d(u, v) = \text{težina}((u, v)) \tag{4.5}$$

gdje su u i v susjedni čvorovi. Primjer za graf zadatka je rječnik:

```
"edge_lengths" : {(1, 2): 1, ... , (11, 16): 19 }
```

- **edge indexes** - indexi bridova

$$I = \{1, 2, \dots, |E|\} \tag{4.6}$$

Primjer za graf koji je implementiran je rječnik s definiranim indeksom svakog brida:

```
"edge_indexes" : {(1, 2): 1, ... , (11, 16): 24 }
```

Skup vozila ($\mathcal{V}$) definiran je sljedećim atributima:

- **indeks vozila** (engl. *index*):

$$\text{index} \in \{1, 2, \dots, N\}$$

gdje je N ukupan broj vozila u sustavu. Primjer jednog vozila koje je u skupu za treniranje:

```
"index" : 8
```

- **ruta kojom se vozilo kreće** (engl. *route*):

$$\textbf{čvor}_1 \rightarrow \textbf{čvor}_2 \rightarrow \cdots \rightarrow \textbf{čvor}_n$$

zapisano kao:

$$\text{route} = [\text{čvor}_1, \text{čvor}_2, \dots, \text{čvor}_n]$$

gdje su čvor_i čvorovi na ruti vozila i n je broj čvorova u ruti. Ruta vozila iz ulaznog skupa:

```
"route" : [15, 11, 9, 10, 9, 11, 15]
```

- **brzina vozila** (engl. *speed*):

$$\text{speed} = \frac{\text{broj jedinica duljine brida}}{\text{jedan trenutak vremena}}$$

  Brzina vozila iz primjera:

```
"speed" : 9
```

- **početna pozicija** (engl. *initial position*):

$$\text{initial\_position} = \text{čvor}_i$$

  gdje je čvor_i neki čvor na ruti vozila. Početni čvor ovog vozila:

```
"initial_position" : 11
```

Ulazi u sustav su:

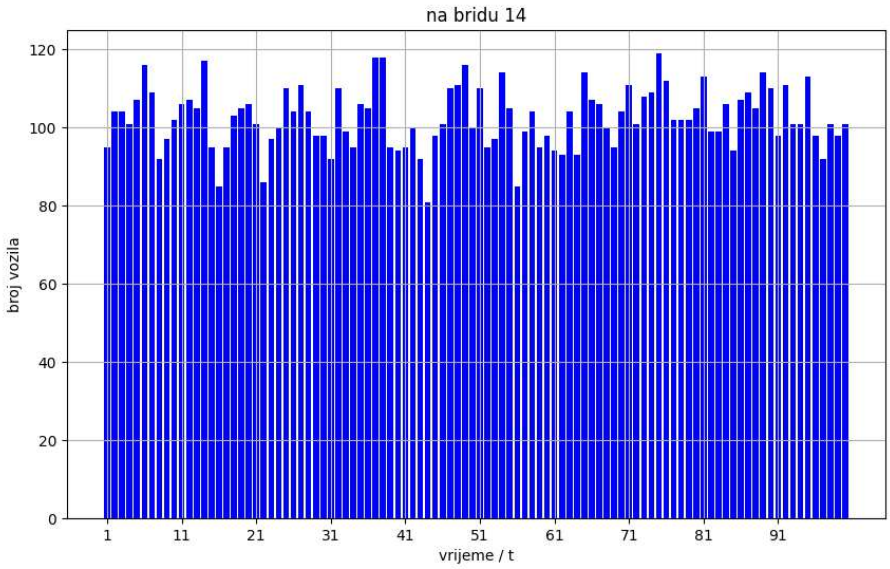
- **graf** $G = (V, E)$ sa svim atributima koji su iznad opisani.
- **skup vozila** ($\mathcal{V}$): skup vozila u sustavu, svako vozilo ima svoje atribute.

Izlaz iz sustava definiran je kao funkcija koja se računa za svaki brid $e \in E$:

$$Alloc(e) = \frac{1}{T} \sum_{t=1}^{T} Alloc(e, t) \tag{4.7}$$

gdje je $Alloc(e)$ prosječan broj vozila na bridu e, T je ukupan broj vremenskih koraka za koje se izračunalo $Alloc(e, t)$, što je broj vozila koja u trenutku t stoje na bridu e.

Za primjer s 1000 vozila s različitim rutama u sustavu koji je prikazan na slici 4.1. prikazan je broj vozila na bridu e u trenutku t za bridove koji su pokazali posebna svojstva. S konkretnim skupom vozila, brid s indeksom 14 je brid s najvećim prometom, dok je brid s indekom 2 brid s najmanjim prometom u tom sustavu.



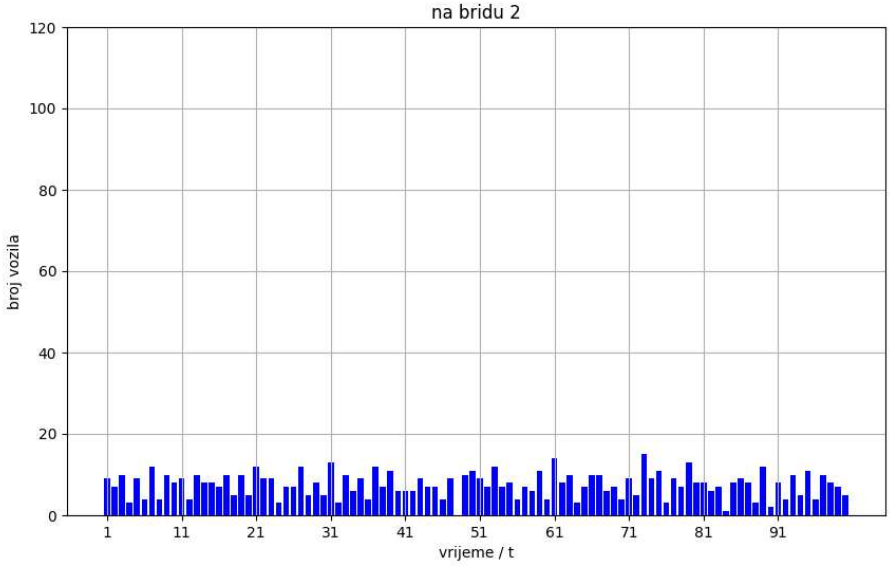
Slika 4.2. Graf koji prikazuje broj vozila u trenutku t na bridu 14

Na slici 4.2. vidljivo je kako je stupčasti graf vrlo gust i stupci su visoki što označava da je na tome bridu u svakom trenutku vremena vrlo velik broj vozila. Isto tako, na slici 4.3. može se uočiti kako je graf vrlo nizak što znači da taj brid nije od velike važnosti u testiranom skupu vozila. Ovakvom analizom mogu se odrediti kritični dijelovi prometnica ili na primjer sustava vozila u skladištu, te na taj način omogućiti optimiranje mreže kretanja vozila kako bi se smanjilo opterećenje najprometnijih bridova i olakšao promet.

4.1.1. Skup za treniranje

Tijekom generiranja skupa podataka vozila za testiranje i treniranje modela, za oblikovanje rute za svako vozilo korišten je BFS algoritam (engl. *Breadth-First Search*). Algoritam koristi red (engl. *queue*) za praćenje posjećenih čvorova. Kada dođe do nekog čvora, označi ga kao posjećenog i stavi u red. Zatim uzima čvorove iz reda, obriše ih iz reda, a sve njihove susjede također stavlja u red. Na taj način briše čvorove iz reda sve dok red nije prazan. Kao početni čvor izabran je *random* odabirom bilo koji čvor iz rute. Brzina je *random* broj iz intervala $[1, 21]$ kako ne bi bila prevelika s obzirom na udaljenosti između čvorova.

Simulacija radi na način da se u svakom trenutku t za svako vozilo *vehicle* izračuna



Slika 4.3. Graf koji prikazuje broj vozila u trenutku t na bridu 2

do sad prijeđeni put formulom:

$$P = t \cdot \text{vehicle[speed]} \tag{4.8}$$

Zatim se prolazi kroz rutu vozila i simulira put kojim je ono do sada prešlo dok se ne dođe do trenutka u kojem je sustav trenutno. Nakon toga se poveća brojač za broj vozila na bridu na kojem je vozilo trenutno te se tako ponavlja za svako vozilo u svakom trenutku.

Treniranje modela izvodi se na skupu za treniranje od 1000 vozila, a podijeljeni su u podskupove od 10 vozila te se nad svakim podskupom posebno izvede simulacija.

4.1.2. Implementacija

Zadatak je implementiran u programskom jeziku Python koji je odabran zbog mnogo biblioteka koje podržavaju i pojednostavljuju rad s grafovima i neuronskim mrežama. Kod se može pronaći na linku [17]. Prilikom implementacije ove mreže korišteno je sljedeće:

- **NumPy**

- **JSON**
- **Torch**
- **Torch.nn**
- **Torch.nn.functional**
- **Torch_geometric.nn**
- **Torch_geometric.data**
- **Torch_geometric.loader**

NumPy je osnovna biblioteka za znanstveno računanje u Pythonu. Ima podršku za računanje matricama, nizove, linearne algebarske operacije i još mnogo toga. U ovome je kodu korištena za stvaranje nizova te inicijalizaciju matrice.

JSON (engl. *JavaScript Object Notation*) je biblioteka koja omogućuje pretvaranje JSON stringova u Python objekte i obrnuto. JSON oblik podataka je vrlo intuitivan i lako čitljiv. Svi skupovi podataka su u JSON obliku u ovoj implementaciji.

Torch je *open source* biblioteka za strojno učenje (engl. *Machine Learning, ML*) koja se koristi za stvaranje dubokih neuronskih mreža. Omogućila je stvaranje i inicijalizaciju tenzor oblika podataka.

Torch.nn je modul koji je dio PyTorch-a, a sadrži definirane slojeve neuronskih mreža te pruža osnovne gradivne blokove za neuronsku mrežu.

Torch.nn.functional je modul koji pruža definiranje mnogo funkcija i slojeva. Koristan je kada se želi definirati prilagođene slojeve i operacije.

Torch_geometric.nn je modul iz *Torch_geometric* biblioteke koji pruža potporu funkcionalnostima specifične za grafove i graf neuronske mreže. Korišten je *GCNConv* je sloj za graf konvolucijske mreže koji se koristi u ovom kodu, te također omogućava *global pooling* pomoću funkcije *global_mean_pool* odnosno agregiranje značajki čvorova pomoću globalnog prosjeka.

Torch_geometric.data je modul koji pruža strukture podataka i funkcionalnosti po-

trebne za rad s grafovima u *Torch_geometric* biblioteci. *Data* je osnovna struktura za pohranu grafova i njegovih značajki koja se koristi i u kodu.

Torch_geometric.loader modul je koji pruža funkcionalnosti za učitavanje grafičkih podataka i njihovu pripremu za upotrebu u neuronskim mrežama, a posebno je korisno to što podržava batch obradu podataka. Konkretno korišteni *DataLoader* specijaliziran je za rad s grafičkim podacima te olakšava iteriranje kroz grafičke podatke, njihovu pripremu za ulaz u model te evaluaciju performansi modela.

4.1.3. Pojedinosti mreže

Implementirana mreža sastoji se od ukupno 4 sloja od čega su tri konvolucijska sloja što ovu mrežu čini **graf konvolucijskom mrežom**, te jedan izlazni linearni sloj koji mapira skrivene značajke u izlazne vrijednosti mreže odnosno predikcije.

Varijabla *hidden_dim* predstavlja dimenziju skrivenih slojeva u mreži. U ovom konkretnom primjeru, ona je 32 što znači da se svaki sloj sastoji od 32 neurona. Veća vrijednost ovog parametra omogućuje mreži da nauči složenije obrasce, no može dovesti i do prenaučenosti ukoliko je prevelika za ulazne podatke.

Za aktivacijsku funkciju korištena je funkcija *elu* (engl. *Exponential Linear Unit*) koja je formalno definirana:

$$\mathrm{ELU}(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha(e^x - 1) & \text{if } x \leq 0 \end{cases} \tag{4.9}$$

Global pooling izvršava se nakon posljednjeg konvolucijskog sloja. Služi tome da se akumuliraju svi podaci iz grafa kako bi se dobila što bolja predikcija.

Dropout je tehnika regulacije kojom se sprječava prenaučenost mreže. Radi na način da se na svaku izlaznu vrijednosti iz sloja primijenjuje *dropout* s vjerojatnošću od 30% u ovom konkretnom primjeru. Time se nasumično isključuje 30% neurona tijekom treninga.

Optimizator je algoritam koji prilagođava težine i pomake u mreži kako bi smanjio funkciju gubitka. U primjeru je korištena metoda *Adam* (engl. *Adaptive Moment Estimation*) koji prilagođava brzinu učenja za svaki parametar posebno. Koristi prve i druge mo-

mente gradijenata (srednje vrijednosti i varijance) za prilagođavanje koraka učenja, što pomaže u stabilizaciji procesa učenja i sprječava prevelike promjene u težinama. Osim toga, često brže konvergira prema globalnom minimumu funkcije gubitka u odnosu na druge optimizatore. Dodatna prednost je niska memorijska potrošnja. U ovom primjeru korišten je s *learning rate-om* od 0.01 koja utječe na to koliko će brzo model konvergirati. Ovo je poprilično niska vrijednost što omogućava stabilnost učenja.

Ovi su parametri odabrani nakon istraživanja i isprobavanja te demonstriraju najbolji učinak u kontekstu predviđanja opterećenja bridova.

4.1.4. Rezultati

Dizajnirana mreža ispisuje funkciju gubitka za svaku epohu učenja čime se može pratiti kako mreža napreduje tijekom treninga. Funkcija gubitka predstavlja razliku između predikcija modela i stvarnih vrijednosti, a smanjenje funkcije gubitka kroz epohe indicira uspješno učenje mreže. Primjera ispisa funkcije pogreške modela tijekom treniranja je sljedeći:

```
...
Epoch 996, Loss 1.851046085357666
Epoch 997, Loss 0.15856091678142548
Epoch 998, Loss 0.970440685749054
Epoch 999, Loss 1.1798771619796753
```

Osim toga, tijekom testiranja mreže ispisuje se stvarna izračunata vrijednost dobivena simulacijom, a pored toga predikcija mreže kao što je prikazano u nastavku.

```
Predikcije za prosječno opterećenje bridova s novim vozilima:
Brid 1: Stvarno: 4.86, Predikcija: 5.24
Brid 2: Stvarno: 10.75, Predikcija: 9.55
Brid 3: Stvarno: 34.63, Predikcija: 37.30
Brid 4: Stvarno: 77.51, Predikcija: 81.82
Brid 5: Stvarno: 22.12, Predikcija: 15.59
...
```

Model je implementiran tako da je moguće odstupanje od stvarne vrijednosti za 20%

za predikcije, što znači da je predikcija p točna ako vrijedi:

$$\left|\frac{p-t}{t}\right| \leq 0.20 \tag{4.10}$$

gdje t predstavlja stvarnu vrijednost parametra. Na primjer, ako je stvarna vrijednost opterećenja brida $t = 100$, predikcija će se smatrati točnom ako je u intervalu $[80, 120]$. Ova fleksibilnost u evaluaciji predikcija omogućuje bolju procjenu modela u stvarnim uvjetima gdje su mala odstupanja prihvatljiva.

Na kraju se ispiše točnost (engl. *accuracy*) mreže koja se računa formulom:

$$\text{točnost} = \frac{\text{broj ispravnih predikcija}}{\text{ukupan broj predikcija}} \times 100 \tag{4.11}$$

Ovdje *broj ispravnih predikcija* predstavlja broj instanci koje je model ispravno predvidio, a *ukupan broj predikcija* je ukupan broj instanci nad kojima je model testiran. Za ovaj primjer to izgleda ovako:

```
Točnost: 95.83%
```

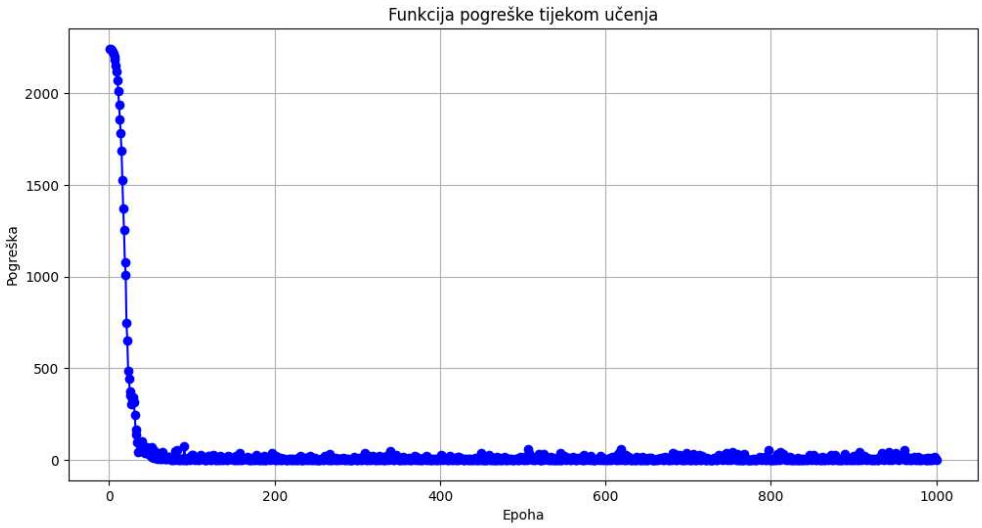
Model je treniran i testiran na 1000 ulaznih primjeraka vozila. Nakon treniranja te testiranja na istom broju vozila, zaključuje se da je točnost ovog modela između 80% i 96%, ovisno o pokretanju i skupu podataka na kojem se izvodi.

Na slici 4.4. prikazana je razdioba funkcije pogreške tijekom epoha učenja ovog modela.

4.2. Implementacija mreže za predikciju Fiedlerove vrijednosti grafa

Zadatak je konstruirati mrežu koja će predviđati Fiedlerovu vrijednost grafa.

Fiedlerova vrijednost koristi se u raznim područjima, pruža uvid u strukturu grafa i njegovu povezanost. Fiedlerov vektor povezan s drugom najmanjom svojstvenom vrijednosti *Laplacijana*, koristi se za analizu povezanosti grafa i detekciju zajednica stoga je korisno znati njegovu vrijednost.



Slika 4.4. Prikaz funkcije pogreške tijekom epoha učenja mreže za prikazani primjer

Ulaz i izlaz iz modela prikazani su na sljedeći način:

$$G(V, A, deg(v)) \rightarrow (Fiedlerova\ vrijednost) \tag{4.12}$$

gdje se na ulaz mreže dovodi graf s brojem čvorova, matricom susjedstva i stupnjem svakog čvora, a mreža kao izlaz predviđa njegovu Fiedlerovu vrijednost.

4.2.1. Skup za treniranje

Skup podataka koji se šalje na ulaz ove mreže sastoji se od mnogo grafova različitih svojstava. Grafovi su generirani na način da se *random* funkcijom odredi broj čvorova iz određenog intervala te se na osnovu broja čvorova napravi matrica susjedstva veličine [*broj_čvorova* x *broj_čvorova*] s 1 i 0 na *random* mjestima u matrici. Nakon toga se dijagonala matrice postavi na 0 kako se ne bi dogodilo da je čvor susjed sam sa sobom.

Graf koji se dodaje u skup podataka za treniranje ili testiranje sadrži:

- **broj čvorova** = $|V|$
- **matricu susjedstva** - opisuje koji čvor je povezan s kojim, a popunjava se po for-

muli (2.3)

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix}, \tag{4.13}$$

- izračunatu **Fiedlerovu vrijednost**
- **značajke čvorova** - stupanj svakog čvora koji se računa po formuli (2.1) u obliku liste

$$[\deg(v_1), \deg(v_2), \ldots, \deg(v_n)] \tag{4.14}$$

Primjer jednog grafa koji ulazi u sustav:

```
{
    "num_nodes" : 6,
    "adjacency_matrix" : [[0, 1, 1, 1, 0, 0],
                          [1, 0, 1, 1, 1, 1],
                          [1, 1, 0, 0, 0, 0],
                          [1, 1, 0, 0, 1, 1],
                          [0, 1, 0, 1, 0, 0],
                          [0, 1, 0, 1, 0, 0]],
    "fiedler_value" : 1.5188056959079848,
    "node_features" : [3, 5, 2, 4, 2, 2]
}
```

4.2.2. Računanje Fiedlerove vrijednosti

Potrebno je nekoliko vrijednosti grafa kako bi se izračunala Fiedlerova vrijednost, a to su Laplaceova matrica te svojstveni vektor.

Laplaceova matrica grafa

Laplaceova matrica, ili *Laplacijan*, ključni je koncept u teoriji grafova i matematičkoj analizi grafova. Koristi se za računanje raznih mreža u području fizike, biologije, soci-

ologije i inženjerstva.

Računa se iz matrice susjedstva ili matrice incidenata, ovisno računamo li ju za neusmjereni ili usmjereni graf. U nastavku je opis računanja Laplacijana za neusmjerene grafove budući da se takvi koriste u ovoj mreži.

Za neusmjereni graf s n čvorova, Laplacijan L definira se kao:

$$L = D - A \tag{4.15}$$

gdje je A matrica susjedstva grafa dimenzija $n \times n$, izračunata po formuli (2.3). D je dijagonalna matrica stupnjeva čvorova, gdje je D_{ii} zbroj stupnjeva povezanosti čvora i, odnosno $D_{ii} = \sum_j A_{ij}$, a izgleda ovako:

$$D = \begin{bmatrix} D_{11} & 0 & \cdots & 0 \\ 0 & D_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & D_{nn} \end{bmatrix} \tag{4.16}$$

Svojstveni vektor

Nakon konstrukcije Laplacijana, sljedeći korak je pronalaženje svojstvenog vektora (engl. *eigenvector*) koji pripada drugoj najmanjoj svojstvenoj vrijednosti (engl. *eigenvalue*). Ovaj svojstveni vektor poznat je kao *Fiedlerov vektor*.

Postupak dekompozicije kvadratne matrice u svojstvene vektore i vrijednosti objašnjen je u nastavku, a još se naziva i *eigendecomposition*. [18] Neka je A kvadratna matrica dimenzije $n \times n$. Svojstveni vektor $\mathbf{v}$ matrice A je ne-nul vektor takav da vrijedi:

$$A\mathbf{v} = \lambda\mathbf{v} \tag{4.17}$$

gdje je λ svojstvena vrijednost pripadna tom vektoru. Da bi se dobio svojstveni vektor, rješava se svojstveni problem. To uključuje traženje vektora $\mathbf{v}$ i pripadne vrijednosti λ koji zadovoljavaju gore navedeni izraz. Svojstveni problem matrice A može se izraziti

kroz karakterističnu jednadžbu:

$$\det(A - \lambda I) = 0 \tag{4.18}$$

gdje je I jedinična matrica. Ova jednadžba daje svojstvene vrijednosti λ matrice A. Nakon što se dobiju svojstvene vrijednosti $\lambda_1, \lambda_2, \ldots, \lambda_n$, treba pronaći odgovarajuće svojstvene vektore $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n$ za svaku svojstvenu vrijednost λ_i. Svaki vektor $\mathbf{v}_i$ rješava izraz $(A - \lambda_i I)\mathbf{v}_i = 0$.

Fiedlerova vrijednost

Fiedlerov vektor svojstveni je vektor koji je povezan s drugom najmanjom svojstvenom vrijednosti Laplaceove matrice grafa. [19]

Kao takav, Fiedlerov vektor zadovoljava sljedeći izraz:

$$L\mathbf{v}_2 = \lambda_2 \mathbf{v}_2 \tag{4.19}$$

gdje je λ_2 Fiedlerova vrijednost, odnosno druga najmanja svojstvena vrijednost Laplacijana grafa.

4.2.3. Implementacija

Ovaj je model također implementiran u porgramskom jeziku *Python*, korišteni su isti paketi i biblioteke kao i u gore navedenoj mreži za predikciju, a to su sljedeći:

- **JSON**
- **Torch**
- **Torch.nn**
- **Torch.nn.functional**
- **Torch_geometric.nn**
- **Torch_geometric.data**

- **Torch_geometric.loader**

4.2.4. Pojedinosti mreže

Ova je mreža također **graf konvolucijska mreža** koja se sastoji od ukupno devet slojeva, od čega je osam konvolucijskih te jedan linearni izlazni sloj. Svaki sloj, osim izlaznog, izgrađen je od 64 neurona. Može se zaključiti da je ova mreža dosta veća i kompleksnija od mreže za predikciju opterećenja bridova što je opravdano budući da druga mreža radi s puno složenijim grafovima i podacima.

Kao i prijašnja mreža, i ova koristi *elu* (engl. *Exponential Linear Unit*) za prijenosnu funkciju. Također koristi i *global pooling*, te *dropout* tehniku s vjerojatnošću od 10%. Optimizator je *Adam* s *learning rate*-om od 0.001 koji omogućava izrazito kontrolirano učenje.

Bitna razlika u dvije implmentirane mreže može se primijetiti u veličini parametara optimizatora i dropouta. *Learning rate* optimizatora razlikuje se čak za cijeli red veličine.

4.2.5. Rezultati

Točnost modela računa se formulom (4.11), te se i u ovom modelu u obzir uzima odstupanje od 20% od stvarne vrijednosti prema formuli (4.10).

Također se za svaku epohu ispisuje pogreška dobivena formulom 3.3, a za svaki ulaz se ispisuje stvarna vrijednost i predikcija modela. Primjer ispisa pogreške za svaku epohu prikazan je u nastavku:

```
...
Graf 496: stvarno: 27.526, predikcija: 28.203
Graf 497: stvarno: 2.789, predikcija: 2.675
Graf 498: stvarno: 5.569, predikcija: 6.085
Graf 499: stvarno: 16.242, predikcija: 18.531
Graf 500: stvarno: 5.997, predikcija: 4.916
```

Slijedi primjer ispisa predikcije i točnosti modela:

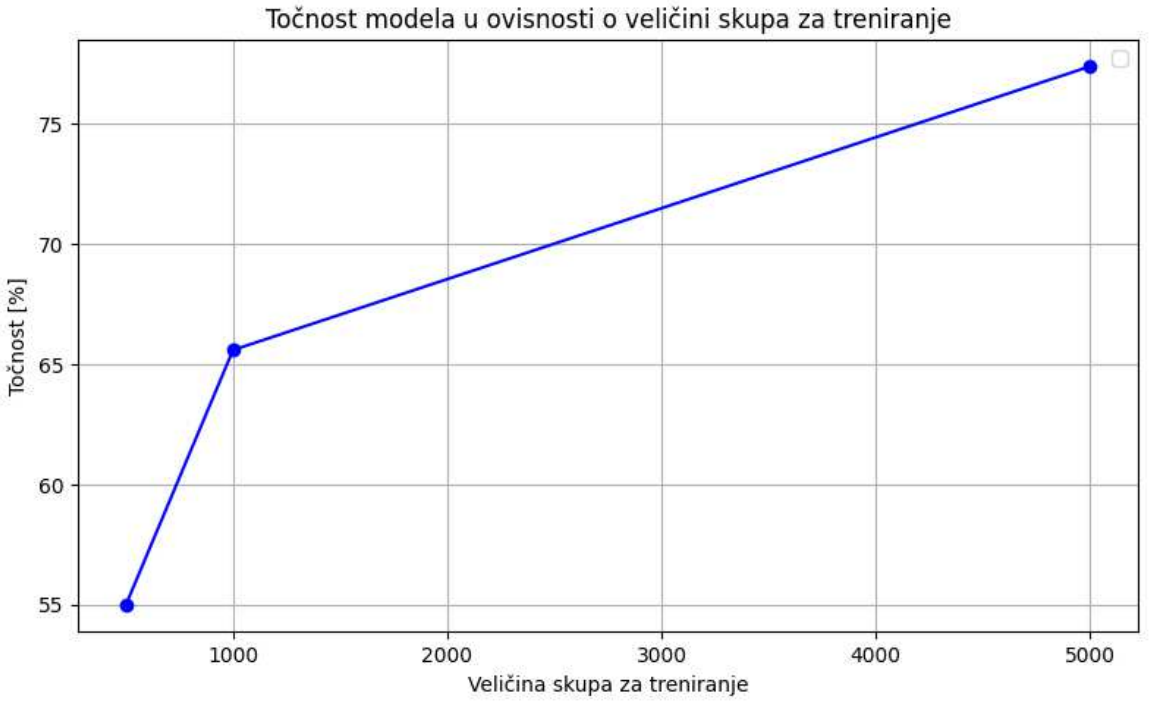
```
...
```

```
Graf 496: stvarno: 27.526, predikcija: 28.203
Graf 497: stvarno: 2.789, predikcija: 2.675
Graf 498: stvarno: 5.569, predikcija: 6.085
Graf 499: stvarno: 16.242, predikcija: 18.531
Graf 500: stvarno: 5.997, predikcija: 4.916
Točnost: 85.80%
```

interval broja čvorova	|skup za treniranje|	|skup za testiranje|	točnost [%]
[5, 15]	500	500	55
[5, 15]	1000	500	65.60
[5, 15]	5000	500	77.40
[5, 100]	500	500	85.80
[5, 100]	1500	1500	88.20

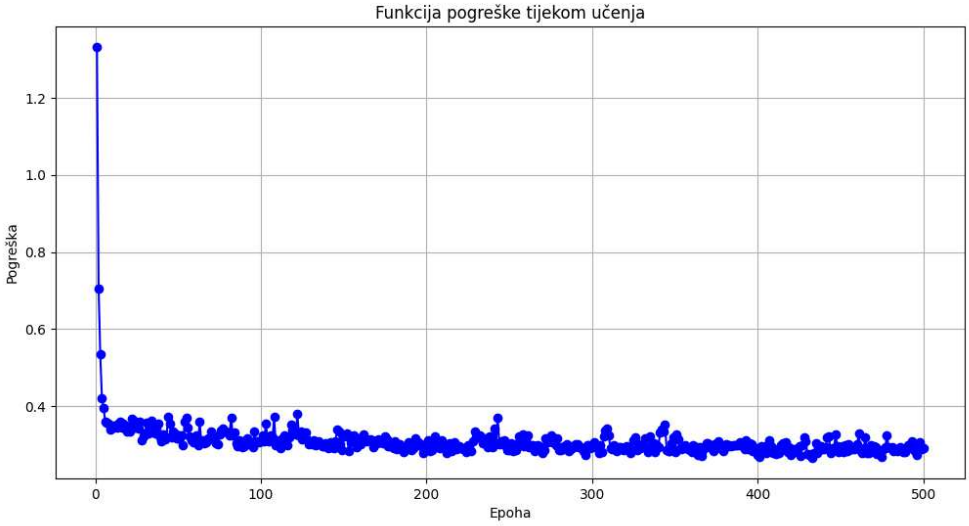
Tablica 4.1. Rezultati modela za predikciju Fiedlerove vrijednosti na različitim skupovima podataka

Iz tablice 4.1. vidljivo je da se točnost modela povećava s povećanjem volumena skupa za treniranje, te se također model puno brže uči na ulaznim grafovima koji su kompleksniji i imaju veći broj čvorova.



Slika 4.5. Prikaz ovisnosti veličine skupa za treniranje i točnosti modela

Prilikom treniranja na skupu od 5000 grafova s brojem čvorova iz intervala [5, 15] te testiranju na 200 grafova čiji se broj čvorova nalazi u intervalu [3, 100], točnost modela pala je na 35%. Ovo pokazuje kako je model potrebno trenirati u istim uvjetima u kojima će se i testirati.



Slika 4.6. Prikaz funkcije pogreške tijekom epoha učenja mreže na 500 uzoraka - prvi stupac u tablici u tablici 4.1

Može se zaključiti da model brže uči kada se koristi skup podataka s grafovima s većim brojem čvorova, u odnosu na skupove s manjim grafovima te da veći volumen skupa za treniranje omogućuje bolje predikcije. Tijekom izgradnje ove neuronske mreže, uočeno je da se točnost modela značajno povećava s povećanjem volumena skupa za treniranje. Tako, važan faktor koji je pridonio porastu točnosti modela bilo je uvođenje značajki čvorova.

5. Zaključak

U ovom radu opisan je osnovni koncept grafova te njihove primjene u matematičkoj teoriji i svakodnevnom životu, kao i detaljan opis neuronskih mreža, njihovih komponenti i procesa učenja.

Neuronske mreže ključni su alat za obradu podataka i učenje iz primjera, s primjenom u različitim područjima kao što su računalni vid, obrada prirodnog jezika, prepoznavanje uzoraka, i druge domene gdje je potrebno rješavati složene probleme temeljene na podacima.

Kao zaključak, proces izgradnje neuronske mreže za različite zadatke zahtijeva temeljito i sistematično pristupanje. Važno je bilježiti sve promjene u parametrima u kodu, kao i detaljno dokumentirati rezultate na različitim skupovima za treniranje i testiranje. Eksperimentiranje s različitim konfiguracijama, poput broja slojeva ili neurona u mreži, ključno je za optimizaciju performansi modela.

Proces treniranja i testiranja neuronskih mreža dugotrajan je i zahtijeva strpljenje i metodološki pristup. Svakako treba pratiti kako se model ponaša na različitim podacima i kako se performanse mijenjaju ovisno o različitim arhitekturama mreže. Upornost u istraživanju i pažljivo vođenje evidencije omogućuju iterativno poboljšanje modela, što je ključno za postizanje visokih performansi u konačnoj primjeni neuronske mreže.

Literatura

[1] A. C. Ian Goodfellow, Yoshua Bengio, “Deep learning”, u *Deep Learning*, 2016.

[2] ——, “Convolutional neural networks on graphs with fast localized spectral filtering”, u *Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering*, 2016.

[3] M. Pavlić, “Analiza cestovne infrastrukture u prostornom planu podruČne razine zagrebaČke Županije”, https://core.ac.uk/download/pdf/197900574.pdf, datum pristupa: lipanj 2024.

[4] A. P. A. B. W. Benjamin Sanchez-Lengeling, Emily Reif, “A gentle introduction to graph neural networks”, https://distill.pub/2021/gnn-intro/, datum pristupa: lipanj 2024.

[5] X.FER, “Osnovni pojmovi teorije grafova”, https://www.fer.unizg.hr/_download/repository/Osnovni_pojmovi-teorija_grafova.pdf, datum pristupa: lipanj 2024.

[6] Wikipedia, “Multigraph”, https://en.wikipedia.org/wiki/Multigraph, datum pristupa: lipanj 2024.

[7] O. Levin, “Planarni grafovi”, https://discrete.openmathbooks.org/dmoi3/sec_planar.html, datum pristupa: lipanj 2024.

[8] Wikipedija, “Petersenov graf”, https://hr.wikipedia.org/wiki/Petersenov_graf, datum pristupa: lipanj 2024.

[9] M. Čupić; Jan Šnajder; Bojana Dalbelo Bašić, “Umjetne neuronske mreže”, https://www.fer.unizg.hr/_download/repository/UmjetneNeuronskeMreze.pdf, datum pristupa: lipanj 2024.

[10] J. Šnajder; Marko Čupić, “Uvod u umjetnu inteligenciju”, https://www.fer.unizg.hr/predmet/uuui, datum pristupa: srpanj 2024.

[11] I. Židov, “Uvod u neuronske mreže”, https://repozitorij.mathos.hr/islandora/object/mathos%3A256/datastream/PDF/view, datum pristupa: srpanj 2024.

[12] P. Sharma, “What are graph neural networks and how do they work?” https://www.analyticsvidhya.com/blog/2022/03/what-are-graph-neural-networks-and-how-do-they-work/, datum pristupa: srpanj 2024.

[13] A. A. Awan, “Comprehensive introduction to graph neural networks (gnns)”, https://www.datacamp.com/tutorial/comprehensive-introduction-graph-neural-networks-gnns-tutorial, datum pristupa: srpanj 2024.

[14] IBM, “Convolutional neural networks”, https://www.ibm.com/topics/convolutional-neural-networks, datum pristupa: srpanj 2024.

[15] A. S. Gillis, “Graph neural networks (gnns)”, https://www.techtarget.com/searchenterpriseai/definition/graph-neural-networks-GNNs, datum pristupa: srpanj 2024.

[16] N. Klinger, “Graph neural networks”, https://viso.ai/deep-learning/graph-neural-networks/, datum pristupa: srpanj 2024.

[17] M. S. Matušin, “Kod zadatka”, https://github.com/miasaramatusin/Zavrsni-rad, datum pristupa: srpanj 2024.

[18] K. Kuttler, “Eigenvalues and eigenvectors of a matrix”, https://math.libretexts.org/Bookshelves/Linear_Algebra/A_First_Course_in_Linear_Algebra_(Kuttler)/07%3A_Spectral_Theory/7.01%3A_Eigenvalues_and_Eigenvectors_of_a_Matrix, datum pristupa: srpanj 2024.

[19] W. Research, “Fiedler vector”, https://mathworld.wolfram.com/FiedlerVector.html, datum pristupa: srpanj 2024.

Sažetak

Graf neuronske mreže za primjenu u višerobotskim sustavima

Mia Sara Matušin

Ovaj rad proučava graf neuronske mreže, posebnu vrstu neuronske mreže za učenje na podacima koji se mogu prikazati grafom. Graf neuronske mreže su implementirane i testirane za dva slučaja uporabe. Prvo se testira primjena graf neuronskih mreža u estimaciji Fiedlerove vrijednosti grafa, a rezultati pokazuju točnost između 50-85%. Drugo, istražuje se primjena na višerobotske sustave. Za takve sustave koji se koriste u skladištima, mapa puta može se prikazati grafom. U ovom radu, za zadan broj vozila i njihove putanje, pomoću graf neuronskih mreža procjenjuje se prosječan broj vozila na svakom bridu. Točnost ostvarena za odabrani skup podataka je 95%. Metoda pokazuje obećavajuće rezultate koji se mogu koristiti za razmatranu ili sličnu primjenu višerobotskih sustava.

Ključne riječi: graf neuronske mreže, Fiedlerova vrijednost, višerobotski sustav

Abstract

Graph Neural Networks for Application in Multi-Robot Systems

Mia Sara Matušin

This work studies graph neural networks, a special type of neural network for learning on data that can be represented by a graph. Graph neural networks are implemented and tested for two use cases. First, the application of graph neural networks in the estimation of Fiedler value of a graph is tested, and the results show an accuracy between 50-85%. Secondly, the application to multi-robot systems is investigated. For such a system used in warehouses, the roadmap can be represented by a graph. In this work, the average number of vehicles on each edge of such a roadmap graph is estimated using graph neural networks, when a certain number of vehicles and their paths are given. The accuracy on the given dataset was 95%. The method shows promising results that could be used in the planning of this or similar multi-robot applications.

Keywords: graph neural networks, Fiedler's value, multi-robot systems