Optimizacija mrežnog prometa u decentraliziranom okruženju

Prpić, Emil

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 1513

OPTIMIZACIJA MREŽNOG PROMETA U DECENTRALIZIRANOM OKRUŽENJU

Emil Prpić

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 1513

OPTIMIZACIJA MREŽNOG PROMETA U DECENTRALIZIRANOM OKRUŽENJU

Emil Prpić

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 4. ožujka 2024.

ZAVRŠNI ZADATAK br. 1513

Pristupnik:	**Emil Prpić (0036544153)**
Studij:	Elektrotehnika i informacijska tehnologija i Računarstvo
Modul:	Računarstvo
Mentor:	prof. dr. sc. Gordan Ježić

Zadatak: **Optimizacija mrežnog prometa u decentraliziranom okruženju**

Opis zadatka:

Ograničenje potpuno decentralizirane grafovske neuronske mreže (engl. Graph Neural Network, GNN) je količina generiranog mrežnog prometa. Kako se graf povećava, količina generiranog mrežnog prometa proporcionalno raste, što dovodi do zagušenja i kašnjenja. Vaš je zadatak proučiti grafovsku neuronsku mrežu i njenu primjenu u različitim područjima te istražiti koje se vrste poruka generiraju u decentraliziranoj grafovskoj neuronskoj mreži. Posebnu pozornost posvetite implementaciji i usporedbi algoritama za smanjenje i optimizaciju mrežnog prometa generiranog između čvorova. Dobivene rezultate prikažite za odabrani slučaj uporabe. Svu potrebnu literaturu i uvjete za rad osigurat će Vam Zavod za telekomunikacije.

Rok za predaju rada: 14. lipnja 2024.

SADRŽAJ

1. Uvod

U današnjem digitalnom svijetu, mrežni promet predstavlja ključnu komponentu u funkcioniranju mnogih sustava. Kako se sve više podataka prenosi putem mreža, optimizacija mrežnog prometa postaje ključan izazov za postizanje efikasnosti, smanjenje troškova i poboljšanje performansi sustava. Ovaj rad istražuje metode za optimizaciju mrežnog prometa, s posebnim naglaskom na decentralizirana okruženja, koja sve više dobivaju na važnosti zbog svojih prednosti u pogledu skalabilnosti, sigurnosti i robusnosti.

Decentralizirana mrežna arhitektura omogućava distribuciju podataka i procesa na više čvorova, što smanjuje opterećenje pojedinih točaka i povećava otpornost sustava na kvarove. Međutim, decentralizacija također donosi izazove u pogledu efikasnog upravljanja mrežnim prometom i smanjenja redundancije podataka.

U ovom radu, prvo ćemo se upoznati s osnovama grafovskih neuronskih mreža (engl. Graph Neural Network, skr. GNN) koje se koriste za analizu i optimizaciju podataka u obliku grafova. Nakon toga, istražit ćemo algoritme za smanjenje i optimizaciju mrežnog prometa, uključujući kompresijske tehnike koje igraju ključnu ulogu u smanjenju količine podataka koji se prenose kroz mrežu. Na kraju, predstavit ćemo testno okruženje i metodologiju evaluacije koja će nam omogućiti procjenu učinkovitosti predloženih rješenja.

Cilj ovog rada je pružiti sveobuhvatan pregled tehnika za optimizaciju mrežnog prometa u decentraliziranim okruženjima, te identificirati najbolje prakse koje se mogu primijeniti u različitim scenarijima.

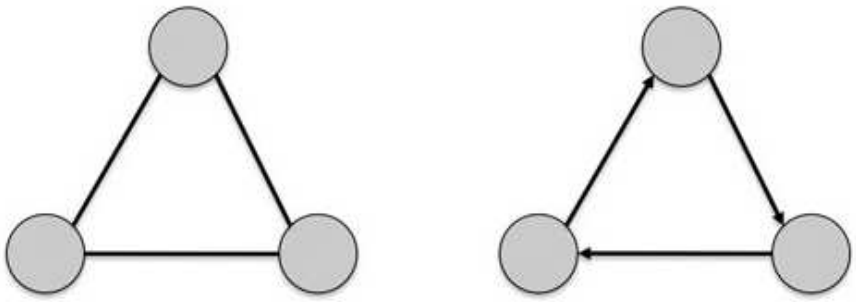
2. Grafovska neuronska mreža

2.1. Općenito o grafovskim neuronskim mrežama

2.1.1. Definicija i osnovni principi

Kako bi došli do definicije grafovske neuronske mreže, prvo trebamo definirati pojam grafa. Graf G je matematička struktura koja se koristi za opisivanje relacija između dvaju objekata iz određene kolekcije. U ovom kontekstu, graf se odnosi na neprazni skup čvorova i kolekciju bridova koji povezuju par čvorova. Skup čvorova obično se označava s $V(G)$, a skup bridova s $E(G)$. Bridovi mogu biti usmjereni ili neusmjereni, ovisno o primjeru. Graf kojem su svi bridovi usmjereni zovemo usmjereni graf; u suprotnom, zovemo ga neusmjereni graf.

U pravom grafu, za koji inicijalno pretpostavljamo da je neusmjeren, linija od točke u do točke v identificira se s linijom od točke v do točke u. U digrafu (skraćeno za usmjereni graf), ta dva smjera smatraju se različitim lukovima, odnosno bridovima. Ako graf G nije usmjeren, tada su dva vrha pridružena jednom bridu međusobno ravnopravna. U usmjerenom grafu, brid može biti usmjeren od jednog vrha prema drugome [9].

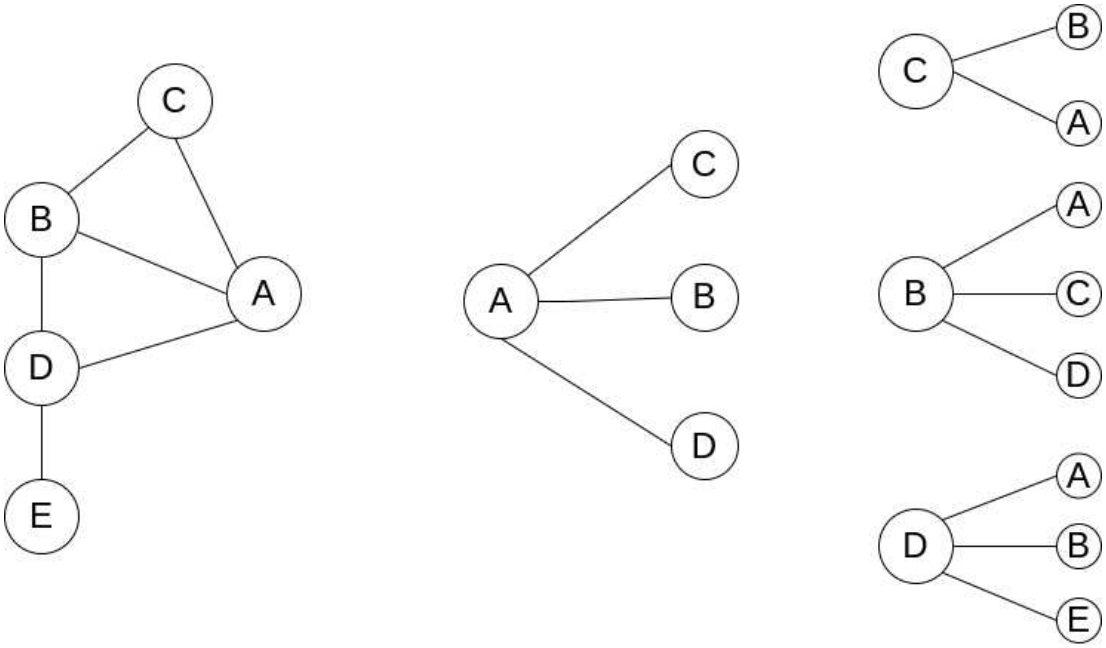


Slika 2.1: Usmjeren i neusmjeren graf. Izvor: [9]

Grafovska neuronska mreža, ili eng. *Graph Neural Network* (skraćeno GNN), predstavlja specifičan oblik neuronske mreže prilagođen za analizu podataka organiziranih u obliku grafova. Koristeći tehnike dubokog učenja, GNN analizira čvorove i njihove veze kako bi se koristio za predviđanje i zaključivanje o osobinama, ponašanju ili svojstvima čvorova i njihovih međusobnih veza u grafu.

2.1.2. Slanje poruka unutar GNN-a

Jedan od osnovnih koncepata u radu grafovske neuronske mreže je mehanizam slanja poruka. GNN ne mijenja povezanost ulaznog grafa, što znači da se neće promijeniti povezanost izlaznog grafa, ali će sadržavati nove ugrađene vrijednosti u čvorovima, bridovima i globalnom kontekstu. Mehanizam slanja poruka djeluje na razini čvorova. Uzmimo za primjer graf prikazan na slici 2.2. Ako je čvor "A" naš ciljni čvor, on će agregirati svoje i susjedne osobine. Svaki čvor integrira novo znanje temeljeno na informacijama dobivenim iz primljenih poruka, koje su obično u vektorskom obliku. Važno je napomenuti da ovo nije uvijek slučaj kod svih tipova GNN-a. Na primjer, u 1-slojnom GNN-u, svaki sloj neuronske mreže predstavlja l-hop (gdje je l broj slojeva) od početnog čvora. Drugim riječima, što koristimo više slojeva, čvor ima pristup više informacija, ali to znači i potrebu za kompleksnijim računanjem, što može produžiti vrijeme treniranja modela. U primjeru slike 2.2, ako se radi o 1-slojnom GNN-u i promatramo početni čvor A, čvoru A su potrebne informacije o osobinama čvorova B, C i D (uključujući i svoje). Ako se radi o 2-slojnom GNN-u, čvoru A su potrebne informacije o osobinama čvorova B, C, D i E (uključujući i svoje).



Slika 2.2: Agregacija susjednih poruka

Proces slanja poruka odvija se u tri koraka:

1. Za svaki čvor u grafu, sakupljaju se sve ugrađene osobnosti susjednih čvorova (ili poruke)
2. Sve poruke se agregiraju putem funkcije za agregaciju (poput zbrajanja).
3. Sve agregirane poruke se proslijeđuju kroz funkciju ažuriranja, obično naučenu neuronsku mrežu.

Kroz slojeve GNN-a, čvor može na kraju integrirati informacije iz cijelog grafa: nakon tri sloja, čvor ima informacije o čvorovima udaljenim tri koraka od njega. Ovaj proces je vrlo sličan konvoluciji, gdje se informacije o susjednim elementima agregiraju i obrađuju kako bi se ažurirala vrijednost elementa. Broj susjednih čvorova u grafu može varirati, što čini GNN fleksibilnijim u analizi različitih struktura podataka. Slaganjem slojeva GNN-a, čvor postepeno uključuje sve više informacija iz cijelog grafa, što rezultira modelom sposobnim za dublje razumijevanje kompleksnih relacija unutar grafa. [3]

2.1.3. Učenje prikaza grafova (GRL)

Cilj učenja prikaza grafova (engl. Graph Representation Learning, skr. GRL) je konstruirati skup ugrađenih vektora koji predstavljaju strukturu grafa i podatke na njemu. Možemo razlikovati tri osnovne vrste takvih vektora: čvorovna ugrađivanja(engl. node embeddings) reprezentiraju svaki čvor u grafu, bridovna ugrađivanja (engl. edge embeddings) reprezentiraju svaki brid, dok grafovna ugrađivanja (engl. graph embeddings) predstavljaju čitav graf kao cjelinu. [6] Zatim možemo svrstati zadatke GRL-a u tri glavne kategorije:

1. **Klasifikacija čvorova**: Cilj je dodijeliti kategorije pojedinim čvorovima u grafu. Na primjer, ako imate socijalnu mrežu, možete koristiti GNN za klasifikaciju korisnika kao "student", "zaposlenik", "umirovljenik" itd. Ovo je korisno za zadatke poput preporuka sadržaja ili pronalaženja zajednica unutar grafa.

2. **Klasifikacija grafova**: Cilj je dodijeliti klasifikacijske oznake ili kategorije nad cijelim grafom. To može biti korisno kada želite klasificirati grafove prema nekom svojstvu. Na primjer, možete koristiti GNN za klasifikaciju socijalnih mreža prema njihovoj političkoj orijentaciji.

3. **Predviđanje veza**: Cilj je odrediti postojanja veza između čvorova u grafu. Na primjer, u socijalnoj mreži, možete koristiti GNN za predviđanje prijateljskih veza između korisnika koji se još nisu povezali. Ovo također može biti korisno za predviđanje mogućih suradnji.

2.1.4. Primjene GNN-a u različitim područjima

Grafovske neuronske mreže postale su ključni alat u analizi podataka koji se mogu prikazivati u obliku grafova. Njihova sposobnost efikasne obrade kompleksnih veza između entiteta čini ih nezamjenjivima u različitim domenama. Primjeri problema koje GNN rješava:

1. **Obrada prirodnog jezika**: Grafičke neuronske mreže su korisne za analizu teksta, strojno prevođenje, generiranje teksta i druge zadatke u obradi prirodnog jezika.

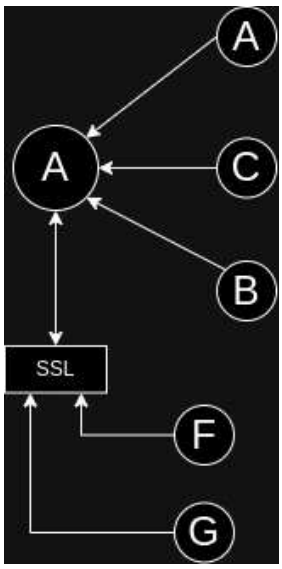
2. **Internetska sigurnost**: Primjenjuju se u detekciji prijevara, identifikaciji anomalija i analizi sigurnosnih prijetnji na internetu, koristeći grafičku strukturu za modeliranje složenih odnosa između entiteta.

3. **Računalni vid**: Koriste se za segmentaciju slike, prepoznavanje objekata i druge zadatke koji zahtijevaju razumijevanje prostornih odnosa i konteksta u slikama.

2.2. Decentralizirani GNN

2.2.1. Osnovni principi i pregled arhitekture

Zbog rasta područja Interneta stvari (engl. Internet of Things, skr. IoT) i generiranja velike količine podataka, tradicionalni centralizirani sustavi postaju sve manje prikladni za rješavanje ovog problema, posebice u smislu skalabilnosti. Stoga se pojavila potreba za decentraliziranim sustavom grafovskih neuronskih mreža. Cilj potpuno decentraliziranog sustava je omogućiti distribuiranu obradu podataka i učenje iz tih podataka. Ovaj pristup obećava poboljšanu skalabilnost, otpornost na kvarove, te veću privatnost i sigurnost, ali također uvodi nove izazove.



Slika 2.3: Prikaz jednog sloja u decentraliziranom sustavu

Decentralizaciju smo postigli korištenjem više metoda, od kojih je jedna učenje putem tračanja (engl. gossip learning). Učenje putem tračanja je metoda za decentralizirano treniranje modela bez potrebe za centralnim agregatorom ili koordinatorom. Ova metoda omogućuje svim čvorovima da konvergiraju prema istim parametrima modela. Svaki čvor sadrži FIFO (engl. First In First Out) međuspremnik s posljednja dva modela primljena od susjeda. U svakoj iteraciji, čvor spaja ta dva modela uzimanjem

prosjeka njihovih težina, izvodi jedan korak lokalnog treniranja i zatim rezultat prosljeđuje nasumično odabranom čvoru.

Prilikom odabira nasumičnog čvora suočavamo se s nekoliko izazova. Prvo, ne želimo da jedan čvor sadrži informacije o svim čvorovima u mreži. Drugo, moramo omogućiti jednostavno dodavanje ili uklanjanje novih čvorova (uređaja). Zato koristimo uzorak ravnopravnih sudionika(engl. peer sampling). To je protokol s ravnopravnim sudionicima (engl. peer-to-peer) koji omogućuje svakom uređaju da izgradi mali, ravnomjerno distribuirani slučajni uzorak cijele mreže. Ovi uzorci se često razmjenjuju između sudionika, obično kao dodatak postojećim komunikacijama, što svakom uređaju daje priliku da sazna za i kontaktira bilo kojeg drugog sudionika. Nadalje, uzorci su obično vremenski označeni kako bi se zastarjeli unosi, koji predstavljaju čvorove koji su napustili mrežu, mogli brzo identificirati i ukloniti. [7]

2.2.2. Prednosti i izazovi decentraliziranih pristupa u GNN-ima

Decentralizirani pristup u grafovskim neuronskim mrežama donosi niz prednosti zbog kojih ga vrijedi razmotriti:

1. **Skalabilnost**: Decentralizacija omogućuje horizontalno skaliranje sustava dodavanjem novih čvorova ili resursa bez potrebe za preinakama u centralnom sustavu. Ovaj pristup omogućuje rast sustava kako se povećava broj čvorova ili obujam podataka, čime se olakšava suočavanje s rastućim zahtjevima.

2. **Otpornost na kvarove**: Centralizirani sustavi su osjetljivi na kvarove ili prekide veze s centralnim posrednikom. Decentralizirani pristupi omogućuju da sustav nastavi funkcionirati čak i ako neki čvorovi ili veze izgube funkcionalnost. Ova sposobnost čini ih pouzdanijima u dinamičkim i nepouzdanim okruženjima, poput IoT uređaja.

3. **Povećana privatnost i sigurnost**: Eliminacija potrebe za centralnim posrednikom smanjuje rizik od krađe podataka ili napada na centralnu točku. Podaci se distribuiraju i obrađuju na lokalnoj razini, čime se smanjuje izloženost osjetljivih informacija vanjskim napadima.

4. **Potpuna asinkronost**: Decentralizirani pristup omogućuje potpunu asinkronost u učenju. Svaki čvor lokalno čuva sve potrebne ulaze za treniranje modela na određenom sloju, uključujući podatke susjeda i podatke slučajno odabranih čvorova. To omogućuje svakom čvoru obradu podataka bez čekanja na najnovije in-

formacije, no istovremeno gubi se svježina podataka zbog korištenja djelomično zastarjelih informacija.

5. **Fleksibilnost u upravljanju čvorovima**: Decentralizirani pristup omogućuje laku integraciju i isključivanje novih čvorova u mrežu. Budući da svaki čvor djeluje neovisno, dodavanje ili uklanjanje čvorova iz mreže ne zahtijeva složene promjene u centralnom sustavu ili prekide u radu. Ova fleksibilnost posebno je korisna u kontekstu Interneta stvari, gdje se uređaji mogu dinamično pridruživati i napuštati mrežu bez narušavanja funkcionalnosti sustava

Uz brojne prednosti postoje i neki izazovi koje uvodi decentralizirani GNN kao što su:

1. **Memorijska kompleksnost**: Korištenje gossip learning algoritma zahtijeva spremanje tri kopije podataka na svakom čvoru, što može biti problematično za uređaje s ograničenim resursima, kao što su IoT uređaji. Ova potreba za dodatnom memorijom može dovesti do povećanog opterećenja memorije na takvim uređajima, ograničavajući njihovu sposobnost učinkovitog sudjelovanja u decentraliziranim grafovskim neuronskim mrežama.

2. **Mrežno opterećenje**: Decentralizirano treniranje GNN-a zahtijeva intenzivnu komunikaciju između čvorova kako bi se osiguralo da svi sudionici imaju ažurirane modele i podatke. Ovaj zahtjev za čestom razmjenom informacija može rezultirati značajnim povećanjem mrežnog opterećenja. U mrežama s ograničenom propusnošću ili visokim kašnjenjem, ovaj povećani promet može dovesti do zagušenja i degradacije performansi.

3. **Neistraženost pristupa**: Decentralizirani pristup grafovskim neuronskim mrežama još uvijek predstavlja nedovoljno istraženo područje. Budući da postizanje potpuno decentraliziranog treniranja GNN-a zahtijeva kombinaciju različitih tehnika, mnogi aspekti ovog pristupa mogu se dalje istražiti i analizirati. Sigurno postoje još nedostaci za koje nismo svjesni.

2.2.3. Poruke unutar decentraliziranog GNN-a

U decentraliziranoj grafovskoj neuronskoj mreži se generiraju dvije glavne vrste poruka. Prvo, gossip poruke se koriste kako bi se eliminirala potreba za centralnim agregatorom. Svaki čvor nakon što obradi podatke na temelju svojih susjeda, te informacije

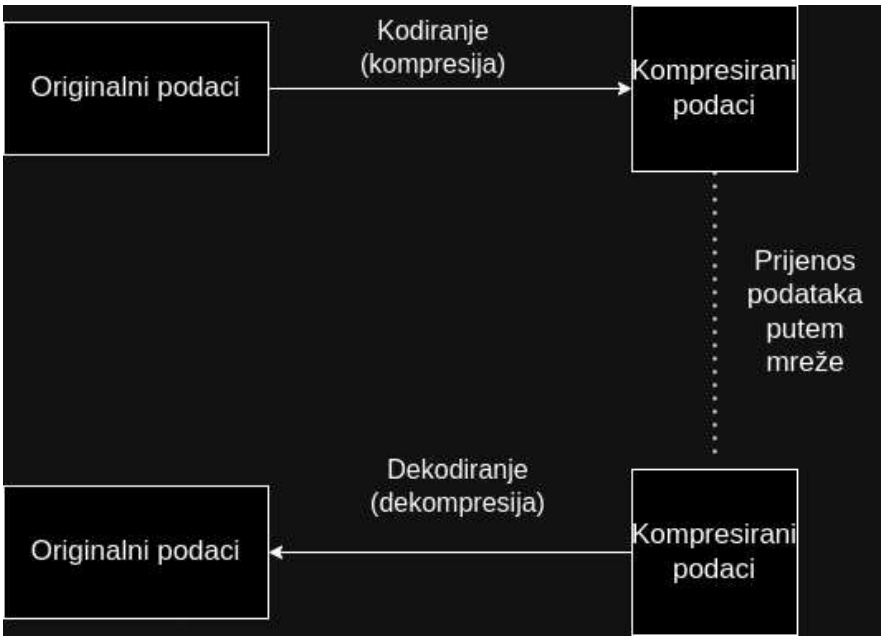
šalje nasumično odabranim čvorovima u mreži. Time se omogućuje da svaki čvor, korak po korak, uči ne samo o svojim direktnim susjedima, već i o cjelokupnoj mreži.

Druga vrsta poruka su izlazne poruke. Nakon obrade podataka, svaki čvor šalje te informacije svojim susjedima kako bi ih mogli koristiti u budućnosti. Ova razmjena informacija omogućuje suradnju između čvorova i doprinosi poboljšanju učinkovitosti učenja i prilagodljivosti sustava

3. Algoritmi za smanjenje i optimizaciju mrežnog prometa

3.1. Kompresija podataka

Potreba za kompresijom podataka proizlazi iz različitih zahtjeva i potreba. Prvo, bitno je smanjiti prostor potreban za pohranu podataka. Nadalje, kompresija je ključna prilikom slanja podataka kako bi se smanjilo opterećenje mreže. U kontekstu usluga poput video prijenosa, brzina prijenosa je vrlo važna za kvalitetu korisničkog iskustva, što kompresiju čini neophodnom. Također, za uređaje s ograničenom memorijom, kompresija je ključna za smanjenje potrošnje prostora za pohranu podataka i poboljšanje učinkovitosti njihovog rada.



Slika 3.1: Proces kompresije

Kompresija podataka se najčešće izražava kao stupanj ili omjer kompresije. Omjer kompresije nam govori koliko je puta veličina podatka smanjena primjenom kompresijskog algoritma. Stupanj kompresije je ključan za razumijevanje učinkovitosti algoritma jer nam pruža informaciju o procijeni koliko se prostora može uštedjeti pri pohranjivanju ili prijenosu podataka.

Stupanj kompresije računamo na sljedeći način:

$$\text{Stupanj kompresije} = \frac{\text{Veličina originalnog podatka}}{\text{Veličina komprimiranog podatka}}$$

Osim stupnja kompresije, možemo prikazati i postotkom uštede. Postotak uštede dodatno ilustrira učinkovitost kompresije i računa se na sljedeći način:

$$\text{Postotak uštede} = \left(1 - \frac{\text{Veličina komprimiranog podatka}}{\text{Veličina originalnog podatka}}\right) \times 100\%$$

Međutim, postoje ograničenja koliko neki podatak možemo smanjiti, kako bi to razumjeli bitno je definirati entropiju. Entropija je mjera prosječne nesigurnosti u nasumičnoj varijabli. Ona predstavlja broj bitova koji su u prosjeku potrebni za opisivanje nasumične varijable.[10] Definirana je izrazom:

$$H(X) = -\sum_{i=1}^{n} p(x_i) \log_2 p(x_i)$$

Shannonov teorem o izvornom kodiranju: Za kodiranje N uzastopnih ishoda istog eksperimenta opisanog slučajnom varijablom $X \sim B(p)$ postoji izvorno kodiranje kod kojega će prosječna duljina kodne riječi biti jednaka $NH(X)$ bita po simbolu, a ne N bita po simbolu. [4] Shannonov teorem o kodiranju izvora postavlja osnovnu granicu za kompresiju podataka, definirajući koliko se podaci mogu efikasno komprimirati bez gubitka informacije. Pokazuje da je entropija $H(X)$ donja granica za prosječan broj bitova potrebnih za kodiranje svakog simbola izvora bez gubitka informacije.

3.1.1. Kompresija bez gubitaka

Kod kompresije bez gubitaka, cilj je zadržati potpunu informaciju izvornih podataka dok se istovremeno smanjuje veličina datoteke.

Osnovne tehnike kompresije bez gubitaka:

1. **Uklanjanje redudancije**: Otklanjanje redundancije u podacima, kao što su ponavljajući obrasci ili nepotrebne informacije, bez promjene ili gubitka korisnih informacija.

2. **Preslagivanje podataka**: Kod složenih podataka, poput slika ili zvuka, podaci se mogu reorganizirati na način koji omogućuje bolje iskorištavanje prirodnih svojstava podataka radi veće kompresije.

3. **Predviđanje**: Predviđanje budućih vrijednosti na temelju prethodnih i kodiranje samo razlike između predviđenih i stvarnih vrijednosti, što rezultira smanjenjem potrebnog prostora za pohranu.

4. **Kodiranje podataka**: Korištenje algoritama poput Huffmanovog kodiranja ili aritmetičkog kodiranja za učinkovito kodiranje podataka uz minimalan dodatni prostor.

Huffmanovo kodiranje

Huffmanovo kodiranje temelji se na ideji dodjeljivanja kraćih kodova često korištenim simbolima i dužih kodova rijetko korištenim simbolima.

Algoritam stvaranja Huffmanovog koda:

1. Sortiraj simbole po padajućim vjerojatnostima

2. Pronađi dva simbola s najmanjim vjerojatnostima

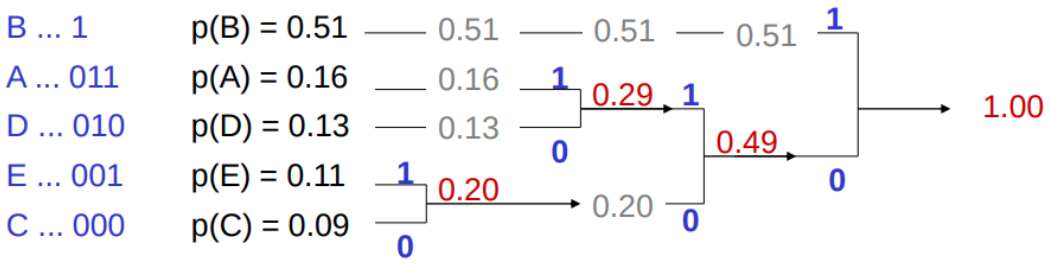
3. Jednom od njih dodijeli simbol “0”, drugom “1”

4. Kombiniraj ta dva simbola u jedan nadsimbol (nadsimbol je novi simbol čija je vjerojatnost pojavljivanja jednaka zbroju vjerojatnosti pojavljivanja dvaju simbola od kojih je nastao) i zapiši ih kao dvije grane binarnog stabla, a nadsimbol kao račvanje iznad njih

5. Ponavljaj 1-4 dok ne dobiješ samo jedan nadsimbol

6. Povratkom kroz stablo očitaj kodove

[11]



Slika 3.2: Primjer kodiranja skupa znakova {A, B, C, D, E} Izvor: [11]

Primjene:

Kompresija bez gubitaka koristi se za pohranu tekstualnih datoteka, tablica, baza podataka i arhiviranje podataka s ciljem dugoročnog očuvanja informacija. Neki od popularnih algoritama koji se koriste za kompresiju bez gubitaka su:

Zlib

Zlib je jedan od najčešće korištenih algoritama za kompresiju podataka. Razvijen je kao dio zlib biblioteke i koristi Deflate metodu za kompresiju podataka te Inflate metodu za dekompresiju. Deflate je kombinacija LZ77 algoritma i Huffmanovog kodiranja. LZ77 algoritam pretražuje prethodne nizove podataka i zamjenjuje ih kratkim simbolima ili referencama, dok Huffmanovo kodiranje dodatno komprimira podatke dodjeljivanjem kraćih kodova čestim simbolima i duljih kodova rjeđim simbolima. [2]

LZMA

LZMA (Lempel-Ziv-Markov chain algorithm) je algoritam za kompresiju podataka koji koristi sličnu metodu kao Deflate, ali umjesto Huffmanovog kodiranja koristi kodiranje raspona (engl. range encoding). LZMA je osmišljen za postizanje visoke stope kompresije, brze dekompresije i niske potrošnje memorije pri dekompresiji. Osim toga, omogućava paralelizam prilikom kompresije i dekompresije [5].

Bzip2

Bzip2 je algoritam za kompresiju podataka koji se temelji na Burrows-Wheeler transformaciji (BWT) i Huffmanovom kodiranju. Često se koristi za kompresiju teksta i

arhiviranje datoteka. Karakteriziraju ga visoka stopa kompresije, ali je brzina kompresije i dekompresije obično sporija u usporedbi s drugim algoritmima poput Zliba [1].

3.1.2. Kompresija sa gubitcima

Kod kompresije sa gubicima, cilj je smanjiti veličinu podataka, pri čemu su manji značajni gubici informacija prihvatljivi. Ova tehnika se često koristi u području multimedije kako bi se uštedjelo na prostoru pohrane ili brzini prijenosa, pri čemu se istovremeno smanjuje kvaliteta.

Osnovne tehnike kompresije sa gubicima:

1. **Uklanjanje redundancije**: Otklanjanje redundancije u podacima, kao što su ponavljajući obrasci ili nepotrebne informacije, bez promjene ili gubitka korisnih informacija.

2. **Uklanjanje irelevantnosti**: Fokusira se samo na bitne dijelove podataka, dok se irelevantni ili manje bitni dijelovi podataka odbacuju. Ova tehnika osigurava da se zadrži samo najvažnija informacija, što omogućuje veću kompresiju bez značajnog gubitka informacija.

3. **Kvantizacija**: Smanjuje se preciznost podataka tako da se mogu predstaviti manjim brojem bitova. Ova tehnika može rezultirati gubitkom informacija, ali je prihvatljiva u mnogim situacijama gdje je manji gubitak kvalitete prihvatljiv u zamjenu za veću kompresiju.

4. Testno okruženje i metodologija evaluacije

4.1. DecGNN simulator

DECGNN simulator je napredni višedretveni C++ program razvijen za rad na Linux operacijskim sustavima, koristeći LibTorch, C++ distribuciju PyTorcha, za efikasno duboko učenje. Simulator je sposoban simulirati više od 100 uređaja po procesorskoj jezgri, kada je svaki uređaj konfiguriran za izvršavanje jednog koraka treniranja po sekundi. Skalabilnost na veći broj uređaja može se postići povećanjem broja korištenih jezgri ili smanjenjem frekvencije treniranja.

Simulator detaljno simulira memoriju, računske procese i mrežnu komunikaciju svakog uređaja. Glavno ograničenje je odsustvo simulacije IP protokola, što rezultira pouzdanim isporukama poruka primatelju unutar jedne iteracije. [8]

4.2. Primjena kompresijskih algoritama

U decentraliziranom sustavu potrebno je sažeti poruke. U simulatoru je to izvedeno na sljedeći način. Svaka poruka sadrži ugrađene vektore koji zauzimaju puno memorije, pa je taj dio potrebno sažeti. Simulator je napisan u C++ jeziku i koristimo libTorch, pa su tako ugrađeni vektori podatkovnog tipa torch::Tensor. Taj se tip podatka ne može direktno sažeti, pa ga moramo serijalizirati i prebaciti u String oblik kako bismo ga mogli sažeti nekim od algoritama.

Serijalizacija je proces pretvaranja podataka u format koji se može lako pohraniti ili prenijeti. U slučaju naših torch::Tensor objekata, to znači pretvaranje Tensor objekta u string koji predstavlja binarne podatke Tenzora.

Nakon serijalizacije podataka, kompresiramo ih koristeći neki od algoritama iz Boost biblioteke. Algoritmi podržavaju parametar ‘level kompresije‘ koji određuje stu-

panj kompresije; što je veći, veći je stupanj kompresije, ali je sporije vrijeme kompresije i dekompresije. Korištena je Boost biblioteka zato što je široko prihvaćena u industriji zbog svoje visoke kvalitete i pouzdanosti.

4.3. Proces prikupljanja i analize podataka

Podaci koje prikupljamo su sljedeći: vrijeme kompresije, veličina podataka prije kompresije, veličina podataka poslije kompresije i vrijeme dekompresije. Vrijeme kompresije, veličinu podataka prije i poslije kompresije možemo vezati zajedno, dok je vrijeme dekompresije pohranjeno u odvojenoj datoteci. Budući da izlazna poruka ide na više čvorova, imamo puno više podataka o vremenu dekompresije. Budući da radimo analizu prosjeka na velikom broju podataka i ulazni podaci su uvijek iste veličine, nije nam bitna veza između vremena dekompresije i originalnog podatka.

Rezultate na kraju obrađujemo pomoću Python skripti jer Python nudi brojne alate za rad s velikim podacima i izradu grafova, kao što su Pandas za manipulaciju podacima, NumPy za numeričke operacije, Matplotlib i Seaborn za vizualizaciju podataka te SciPy za napredne statističke analize. Uz to, koristimo GraphX, koji je dio Apache Spark ekosustava, za obradu i analizu velikih grafova. Ovi alati omogućuju nam učinkovitu analizu i prezentaciju rezultata, čime dobivamo korisne uvide iz prikupljenih podataka.

5. Analiza rezultata

5.1. Usporedba kompresijskih algoritama

Parametar	Razina 1	Razina 3	Razina 9
Prosječna veličina originalne datoteke (B)	3187.0	3187.0	3187.0
Prosječna veličina komprimirane datoteke (B)	1459.56	1445.41	1438.01
Prosječni omjer kompresije	0.4580	0.4535	0.4512
Postotak uštede (%)	54.2028	54.6468	54.8790
Prosječno vrijeme kompresije (s)	0.02542	0.02688	0.05047
Prosječno vrijeme dekompresije (s)	0.00069	0.00139	0.00612

Tablica 5.1: Rezultati kompresije korištenjem LZMA algoritma

Parametar	Razina 1	Razina 9
Prosječna veličina originalne datoteke (B)	3187.0	3187.0
Prosječna veličina komprimirane datoteke (B)	1680.17	1668.31
Prosječni omjer kompresije	0.5272	0.5235
Postotak uštede (%)	47.2804	47.6527
Prosječno vrijeme kompresije (s)	0.00127	0.00305
Prosječno vrijeme dekompresije (s)	0.00061	0.00164

Tablica 5.2: Rezultati kompresije korištenjem Bzip2 algoritma

Parametar	Razina 5
Prosječna veličina originalne datoteke (B)	3187.0
Prosječna veličina komprimirane datoteke (B)	1464.11
Prosječni omjer kompresije	0.4594
Postotak uštede (%)	54.0598
Prosječno vrijeme kompresije (s)	0.00206
Prosječno vrijeme dekompresije (s)	0.00026

Tablica 5.3: Rezultati kompresije korištenjem Zlib algoritma

LZMA algoritam pokazuje najbolje rezultate u pogledu omjera kompresije i uštede prostora, posebno na višim razinama kompresije (Razina 9). Međutim, potrebno je više vremena za kompresiju i dekompresiju u usporedbi s drugim algoritmima. Bzip2 pruža bržu kompresiju i dekompresiju, ali s nešto manje uštede prostora. Zlib pruža uravnotežen rezultat s dobrim omjerom kompresije i relativno brzim vremenom kompresije/dekompresije.

Opis	bzip2 R1-9	LZMA R3-9	LZMA R1-3
Promjena u uštedi (%)	0.37	0.23	0.45
Vrijeme kompresije (s)	0.00178	0.02359	0.00146
Vrijeme dekompresije (s)	0.00103	0.00473	0.0007

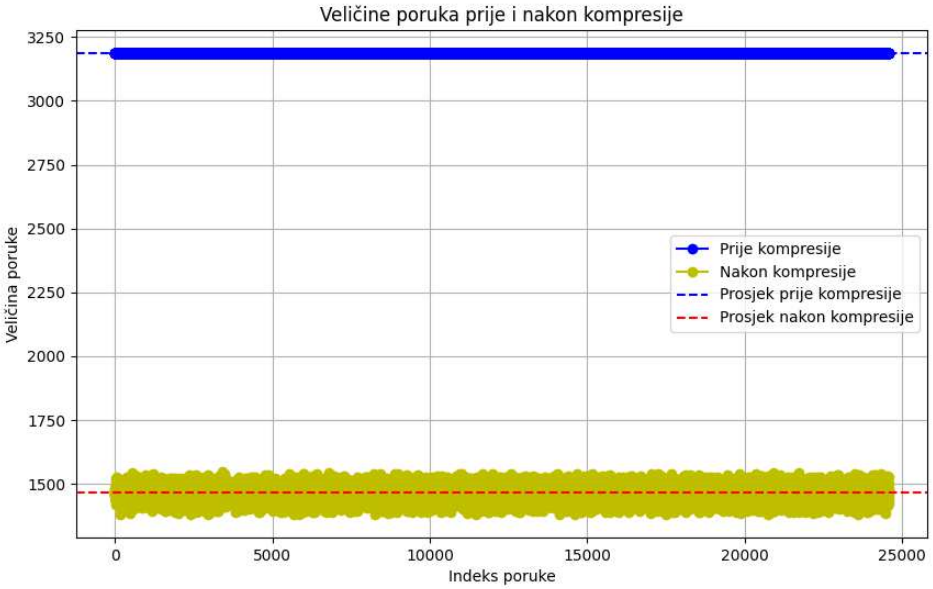
Tablica 5.4: Optimizirane promjene u uštedi prostora i vremenu pri kompresiji/dekompresiji

Povećanje razine kompresije kod LZMA i bzip2 algoritama rezultira relativno malim dobitkom u uštedi prostora. Međutim, to dolazi uz značajno povećanje vremena potrebnog za kompresiju, posebno za LZMA s razine 3 na razinu 9. Unatoč manjim dobitcima u uštedi prostora, dodatno vrijeme kompresije i dekompresije može biti znatno, što bi moglo negativno utjecati na performanse u realnom vremenu ili u scenarijima gdje je brzina obrade kritična.

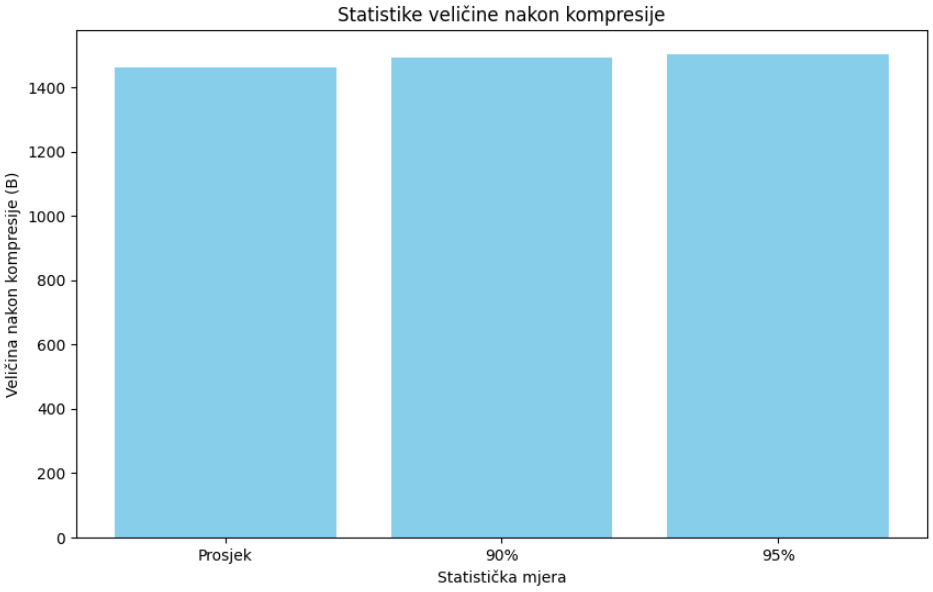
Stoga, u scenarijima gdje je brzina bitnija od maksimalne uštede prostora, niže razine kompresije mogu biti prikladniji izbor. U situacijama gdje prostor za pohranu dominira kao faktor, više razine mogu biti isplative unatoč većim vremenima obrade. Izbor razine kompresije treba temeljito razmotriti u skladu s konkretnim zahtjevima i okolnostima uporabe.

5.2. Diskusija rezultata i zaključci

Za detaljniju analizu je uzet zlib algoritam jer se pokazao kao najbolji izbor za naš slučaj. Nudi veliku stopu kompresije, a vremena kompresije i dekompresije su mala.



Slika 5.1: Veličina poruke prije i nakon kompresije

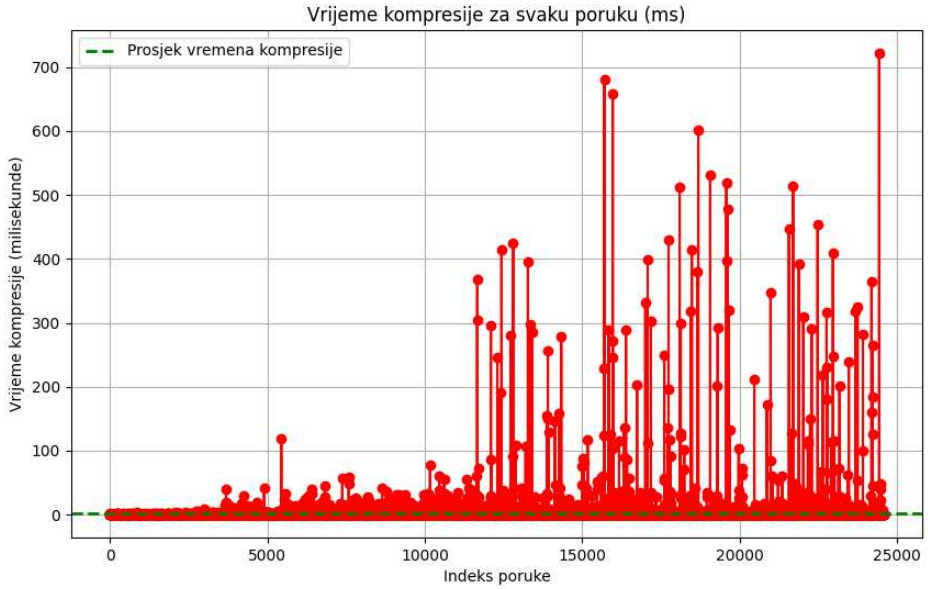


Slika 5.2: Percenitli veličine poruke nakon kompresije

Analiza Veličine nakon kompresije

- **Prosjek:** Prosečna veličina nakon kompresije je 1464.11 bajta. To predstavlja tipičnu veličinu komprimirane poruke unutar vašeg skupa podataka.
- **90. Percentil:** Veličina nakon kompresije za 90% poruka je do 1495 bajta. To znači da samo 10% poruka ima veću veličinu od 1495 bajta.
- **95. Percentil:** Na 95. percentilu, veličina nakon kompresije doseže do 1504 bajta. Ovo ukazuje da je većina poruka (95%) komprimirana na veličinu do 1504 bajta, s tek malim brojem poruka koje zauzimaju više prostora.

Iz ove analize veličine nakon kompresije, možemo zaključiti da je distribucija veličine nakon kompresije prilično uska, što ukazuje na konzistentnost u procesu kompresije. Mala razlika između prosječne veličine i 95. percentila (svega oko 40 bajta) pokazuje da većina poruka zauzima sličnu količinu prostora nakon kompresije.



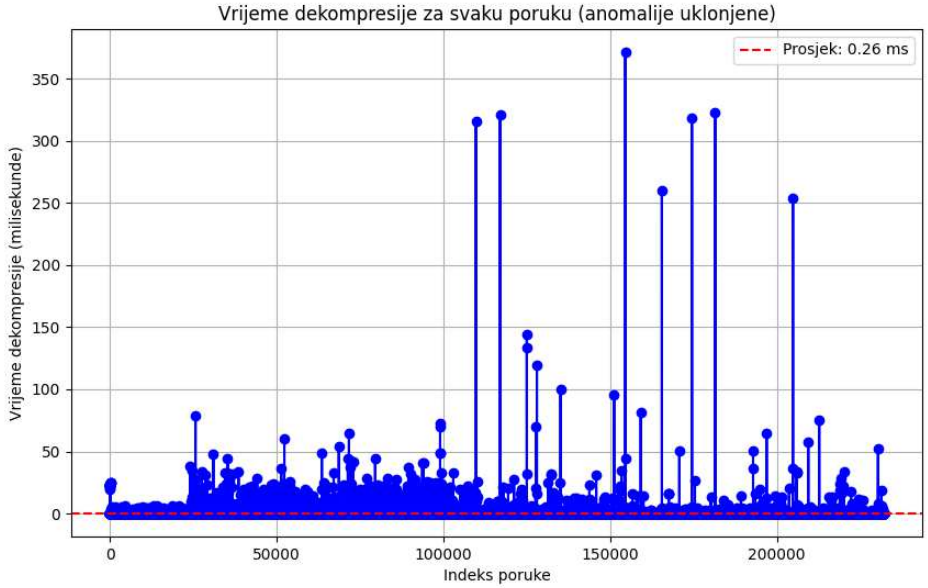
Slika 5.3: Vrijeme komprezsije

Slika 5.4: Prosjek i percentili vremena kompresije

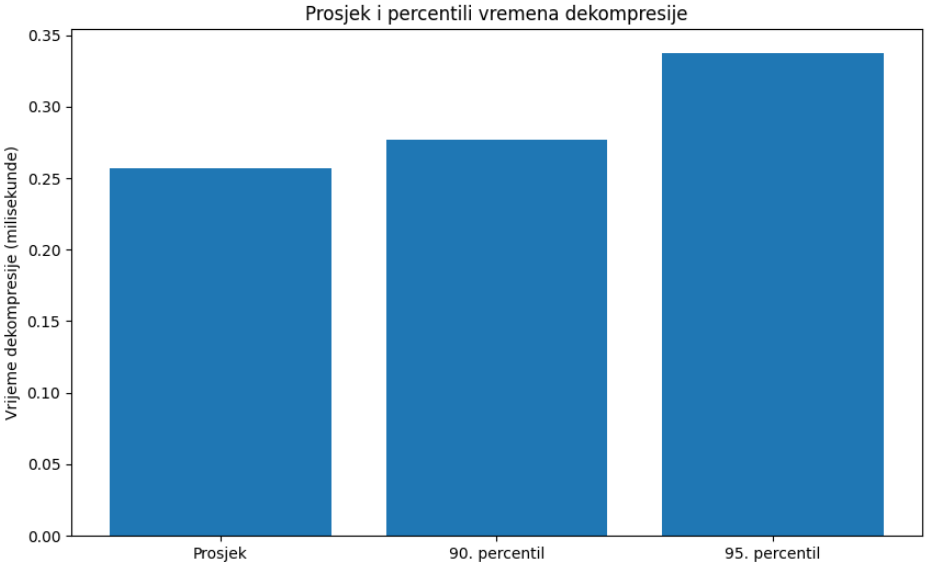
Analiza vremena kompresije

- **Prosjek:** Prosečno vrijeme kompresije je 2.06 ms. To predstavlja tipično vrijeme potrebno za kompresiju poruke unutar vašeg skupa podataka.
- **90. Percentil:** Vrijeme kompresije za 90% poruka je 1.66 milisekundi ili manje. To sugerira da većina kompresijskih operacija završava brže od prosječnog vremena, što ukazuje na visoku efikasnost kompresije za većinu poruka.
- **95. Percentil:** Na 95. percentilu, vrijeme kompresije doseže do 3.73 milisekundi. To ukazuje na to da za 5% poruka proces kompresije traje značajno dulje nego što je uobičajeno, što može biti posljedica specifičnih karakteristika tih poruka koje otežavaju brzu kompresiju.

Iz ove analize vremena kompresije možemo zaključiti da je proces kompresije generalno vrlo brz, s većinom operacija koje se završavaju unutar 1.66 milisekundi. Međutim, postoji mali postotak poruka (5%) za koje kompresija traje znatno dulje, što može biti pokazatelj postojanja izazovnih uvjeta za kompresiju u tih poruka. Razumijevanje ovih slučajeva može pružiti uvide za daljnju optimizaciju kompresijskog algoritma. S obzirom na to da je veličina ulaznih podataka uvijek ista, ove varijacije u vremenu kompresije mogu biti rezultat različitih vrsta sadržaja unutar poruka, što utječe na složenost kompresijskog procesa.



Slika 5.5: Vrijeme dekomprezsije



Slika 5.6: Prosjek i percentili vremena dekompresije

Analiza vremena dekompresije

– **Prosjek:** Prosečno vrijeme dekompresije iznosi 0.26 milisekundi. To ukazuje na vrlo brz proces dekompresije za standardnu poruku unutar vašeg skupa podataka.

– **90. Percentil:** Vrijeme dekompresije za 90% poruka je 0.28 milisekundi ili manje. Ovo pokazuje da se gotovo sve operacije dekompresije odvijaju brzo, s minimalnim odstupanjima od prosjeka.

– **95. Percentil:** Na 95. percentilu, vrijeme dekompresije raste do 0.34 milisekundi. Iako je ovo malo više u usporedbi s 90. percentilom, još uvijek je to vrlo kratko vrijeme, ali ukazuje na to da postoje neke poruke čija dekompresija zahtijeva malo više vremena, što bi moglo biti povezano s određenim osobinama tih poruka.

Analiza vremena dekompresije ukazuje na iznimnu brzinu i efikasnost dekompresijskog procesa za većinu poruka, s gotovo svim operacijama koje završavaju unutar 0.28 milisekundi. Međutim, postoji mali broj poruka (5%), za koje dekompresija traje do 0.34 milisekundi, što je i dalje vrlo brzo, ali relativno sporije u usporedbi s većinom. Ove varijacije mogu biti rezultat različitih karakteristika podataka unutar poruka, kao što je sadržaj koji je manje optimalno strukturiran za dekompresiju.

6. Zaključak

U ovom radu istražili smo metode za optimizaciju mrežnog prometa u decentraliziranim grafovskim neuronskim mrežama (GNN). Kroz detaljnu analizu, usporedili smo nekoliko algoritama za kompresiju podataka, uključujući LZMA, Bzip2 i zlib.

Naši eksperimenti su pokazali da zlib, kao jedan od najefikasnijih algoritama za kompresiju, nudi optimalnu ravnotežu između brzine kompresije i učinkovitosti smanjenja veličine podataka. Zbog svoje visoke učinkovitosti i relativno niske računalne zahtjevnosti, zlib je izabran za daljnju analizu u kontekstu decentraliziranih GNN sustava.

Korištenjem DecGNN simulatora, došli smo do prosječne kompresije od oko 54

Zaključno, optimizacija mrežnog prometa putem decentraliziranih GNN-a uz primjenu zlib algoritma predstavlja obećavajuće rješenje koje može značajno unaprijediti efikasnost i pouzdanost mrežnih sustava.

LITERATURA

[1] Bzip2. Poveznica: https://en.wikipedia.org/wiki/Bzip2. Stranica posjećena: lipanj 2024.

[2] Zlib. Poveznica: https://en.wikipedia.org/wiki/Zlib. Stranica posjećena: lipanj 2024.

[3] Sanchez-Lengeling B., Reif E., Pearce A., i Wiltschko A.B. A gentle introduction to graph neural networks. *Distill*, 14:1, 2021.

[4] Bojanjac D. i Kolar P. *Kako mjeriti i kodirati informaciju*. FER, 2023.

[5] Salomon D. *Data Compression The Complete Reference*. Springer London, 2007.

[6] Frasca F., Rossi E., Eynard D., Chamberlain B., Bronstein M., i Monti F. Scalable inception graph neural networks. *arxiv*, 2004.11198:2, 2020.

[7] Giaretta L. i Girdzijauskas Š. Fully-decentralized training of gnns using layer-wise self-supervision. stranica 7, 2023.

[8] Giaretta L. i Girdzijauskas Š. Fully-decentralized training of gnns using layer-wise self-supervision. stranica 9, 2023.

[9] Fošner M. i Kramberger T. Math e article. *Teorija grafova i logistika*, 14:1, 2023.

[10] Cover T.M. i Thomas J.A. *Elements of Information Theory*. Wiley-Interscience, 2006.

[11] FER zavod za telekomunikacije. Osnove kodiranja i kompresije, 2018. Poveznica: https://www.fer.unizg.hr/_download/repository/VMK_2018_02_Osnove_kompresije.pdf.

Optimizacija mrežnog prometa u decentraliziranom okruženju

Sažetak

Ovaj rad istražuje optimizaciju mrežnog prometa u decentraliziranim grafovskim neuronskim mrežama (GNN). Decentralizirane GNN-e su ključne za skalabilne i učinkovite distribucije podataka, no s njima dolazi i izazov smanjenja redundancije podataka i mrežnog opterećenja. U ovom istraživanju usporedili smo nekoliko algoritama za kompresiju podataka, uključujući LZMA, Bzip2 i zlib, kako bismo utvrdili najefikasniji pristup za optimizaciju mrežnog prometa u ovom kontekstu.

Rezultati eksperimenata pokazali su da zlib algoritam nudi najbolju ravnotežu između brzine kompresije i učinkovitosti smanjenja veličine podataka. Zbog svoje visoke učinkovitosti i niske računalne zahtjevnosti, zlib je izabran za daljnju analizu. Korištenjem DecGNN simulatora, utvrdili smo da primjena zlib algoritma omogućava prosječnu kompresiju od oko 54%, što značajno smanjuje mrežno opterećenje.

Ova optimizacija ne samo da smanjuje količinu podataka koji se prenose kroz mrežu, već i poboljšava ukupne performanse mrežnog sustava, smanjujući operativne troškove i povećavajući otpornost na zagušenja i kvarove. Na temelju dobivenih rezultata, preporučuje se daljnje istraživanje i razvoj ovih tehnika kako bi se dodatno unaprijedila njihova primjena u stvarnim uvjetima, te iskoristile sve prednosti koje nude u pogledu skalabilnosti, sigurnosti i robusnosti.

Ključne riječi: GNN, decentraliziracija, mrežni promet, algoritmi sažimanja

Optimizing Network Traffic in a Decentralized Environment

Abstract

This final thesis investigates the optimization of network traffic in decentralized graph neural networks. Decentralized GNNs are crucial for scalable and efficient data distribution, but they also pose the challenge of reducing data redundancy and network load. In this study, we compared several data compression algorithms, including LZMA, Bzip2, and zlib, to determine the most efficient approach for optimizing network traffic in this context.

The experimental results showed that the zlib algorithm offers the best balance between compression speed and data size reduction efficiency. Due to its high efficiency and low computational requirements, zlib was selected for further analysis. Using the DecGNN simulator, we found that applying the zlib algorithm achieved an average compression rate of approximately 54%, significantly reducing network load.

This optimization not only reduces the amount of data transmitted over the network but also improves the overall performance of the network system, lowering operational costs and increasing resilience to congestion and failures. Based on the obtained results, further research and development of these techniques are recommended to enhance their application in real-world conditions and to leverage all the advantages they offer in terms of scalability, security, and robustness.

Keywords: GNN, decentralization, network traffic, compression algorithms