Porównanie formatów danych w grafowych sieciach neuronowych (GNN) oraz ich wydajności obliczeniowej w PyTorch Geometric (PyG) i Deep Graph Library (DGL)

Patryk Graban
Politechnika Gdańska
Wydział Fizyki Technicznej i Matematyki Stosowanej

Błażej Kardynał
Politechnika Gdańska
Wydział Fizyki Technicznej i Matematyki Stosowanej

Bartosz Bycul
Politechnika Gdańska
Wydział Fizyki Technicznej i Matematyki Stosowanej

mgr Robert Benke
Opiekun Projektu
Politechnika Gdańska
Katedra Architektury Systemów Komputerowych

Abstract

Praca koncentruje się na szczegółowej analizie oraz porównaniu formatów danych stosowanych w grafowych sieciach neuronowych (GNNs). W badaniu skupiono się na dwóch popularnych bibliotekach: PyTorch Geometric (PyG) oraz Deep Graph Library (DGL).
Celem jest zrozumienie, które formaty danych (takie jak CSR, COO, CSC) oferują największą efektywność obliczeniową w kontekście różnych modeli GNN (Graph Convolutional Networks, GraphSAGE, Graph Attention Networks) oraz ich zastosowań. Projekt w głównej mierze ma za zadanie odpowiedzieć na pytania dotyczące wpływu formatu danych na czas obliczeń.
Zakładamy, że wybór formatu danych znacząco wpływa na wydajność obliczeniową, a różnice w wynikach będą rosły wraz z wielkością danych wejściowych. Ponadto testy przeprowadzone na GPU wykażą większą wydajność niż na CPU, a modele bazujące na mechanizmach uwagi (np. GAT) będą wykazywać większe obciążenie obliczeniowe w porównaniu do bardziej klasycznych architektur, takich jak GCN.
Wyniki pracy wskazały znaczące różnice w wydajności w zależności od zastosowanego formatu danych oraz użytej biblioteki. Zrealizowano analizę wydajnościową na zestawach danych takich jak Planetoid (Cora, Citeseer, PubMed) oraz OGB (ogbn-arxiv). Zidentyfikowano kluczowe rekomendacje dotyczące wyboru formatów danych i bibliotek w zależności od aplikacji. Szczegółowe wyniki i analizy zostały zaprezentowane w dalszych częściach pracy.

Keywords:

Graph Neural Networks, PyTorch Geometric, Deep Graph Library, Sparse Data Formats, Computational Efficiency, Graph Convolutional Networks, GraphSAGE, Graph Attention Networks, Batching, CPU, GPU, Training Time

1. Cel pracy

Celem niniejszej pracy jest szczegółowa analiza i porównanie formatów danych wykorzystywanych w grafowych sieciach neuronowych (GNN) oraz ich wpływu na wydajność obliczeniową w bibliotekach PyTorch Geometric (PyG) i Deep Graph Library (DGL). Praca ma na celu odpowiedź na pytanie, który format danych (np. COO, CSR, CSC) zapewnia najlepsze wyniki w różnych scenariuszach obliczeniowych. Hipoteza badawcza zakłada, że wybór odpowiedniego formatu może znacznie wpłynąć na czas obliczeń oraz ogólną efektywność modeli GNN.

2. Zarys i teoria

Grafowe sieci neuronowe (GNN) umożliwiają efektywne przetwarzanie danych o strukturze grafowej, co znajduje szerokie zastosowanie w dziedzinach takich jak sieci społeczne, chemia czy systemy rekomendacyjne. Kluczową cechą tych modeli jest zdolność do uchwycenia relacji między węzłami oraz propagacji informacji w grafie, co pozwala na analizę złożonych zależności i struktur danych.

Wraz ze wzrostem rozmiaru grafów, takich jak sieci liczące miliony węzłów i miliardy krawędzi, rosną również wymagania dotyczące mocy obliczeniowej i pamięci. Przetwarzanie dużych grafów wymaga wydajnych formatów danych (np. CSR, COO) oraz zoptymalizowanych implementacji na sprzęcie o wysokiej wydajności, takich jak GPU.

Skalowalność GNN w dużej mierze zależy od optymalizacji sposobu przechowywania i przetwarzania danych, co czyni wybór odpowiedniego formatu danych kluczowym dla osiągnięcia wysokiej efektywności obliczeniowej. W takich kontekstach szczególnie ważna staje się analiza różnych formatów i ich wpływu na wydajność modeli.

2.1. Podstawowe modele GNN

Do najczęściej stosowanych modeli w ramach GNN należą:

- **Graph Convolutional Networks (GCN)**: Model ten wprowadza operacje splotowe na grafach, umożliwiając propagację informacji między sąsiednimi węzłami. Wykorzystuje normalizację macierzy sąsiedztwa i mechanizm samołączeń, co zwiększa stabilność numeryczną i efektywność obliczeń. GCN znajduje szerokie zastosowanie w klasyfikacji węzłów i rozpoznawaniu społeczności.Matematycznie proces ten można zapisać jako:

$$H^{(k)} = \sigma\left(\tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}H^{(k-1)}W^{(k)}\right), \qquad (1)$$

  gdzie $\tilde{A} = A + I$ to macierz sąsiedztwa z dodanymi samołączeniami, $\tilde{D}$ to macierz stopni wierzchołków, $H^{(k)}$ to macierz cech po k-tej iteracji, $W^{(k)}$ to macierz wag, a σ to funkcja aktywacji (np. ReLU). Model ten efektywnie uśrednia cechy sąsiednich węzłów, zachowując ich strukturalne właściwości.
- **GraphSAGE**: GraphSAGE agreguje cechy sąsiednich węzłów w celu generowania dynamicznych reprezentacji węzłów. Dzięki różnorodnym metodom agregacji (np. sumowanie, uśrednianie) model ten jest dobrze skalowalny i wykorzystywany w analizie dużych grafów, a proces można zapisać jako:

$$h_v^{(k)} = \sigma\left(W^{(k)} \cdot \text{AGGREGATE}\left(\{h_u^{(k-1)} : u \in \mathcal{N}(v)\}\right)\right), \qquad (2)$$

  gdzie $\mathcal{N}(v)$ to sąsiedztwo węzła v. GraphSAGE pozwala na dynamiczne budowanie reprezentacji węzłów, co czyni go bardziej skalowalnym w przypadku dużych grafów.
- **Graph Attention Networks (GAT)**: Model GAT wprowadza mechanizm uwagi, który pozwala różnicować ważność sąsiednich węzłów w procesie propagacji informacji. Dzięki temu uzyskuje większą elastyczność w analizie grafów o złożonej strukturze, takich jak molekuły czy sieci transportowe. Dla każdego sąsiedniego węzła u wagę α_{vu} wyznacza się jako:

$$\alpha_{vu} = \frac{\exp\left(\text{LeakyReLU}\left(a^T \cdot [Wh_v \parallel Wh_u]\right)\right)}{\sum_{k \in \mathcal{N}(v)} \exp\left(\text{LeakyReLU}\left(a^T \cdot [Wh_v \parallel Wh_k]\right)\right)}, \qquad (3)$$

  gdzie a to wektor wag mechanizmu uwagi, $\parallel$ oznacza konkatenację, a W to macierz wag. Mechanizm uwagi pozwala modelowi różnicować ważność sąsiednich węzłów przy propagacji informacji.

2.2. Reprezentacje macierzy rzadkich

Grafy są często reprezentowane w formie macierzy sąsiedztwa, które są zwykle rzadkie (tzn. zawierają wiele zerowych wartości). W praktyce stosuje się różne formaty macierzy rzadkich, takie jak:

- **CSR (Compressed Sparse Row)**: Format CSR (Compressed Sparse Row) jest idealny do pracy z macierzami rzadkimi, szczególnie kiedy często operujemy na wierszach. W CSR dane są przechowywane w trzech tablicach: jedna przechowuje wszystkie niezerowe wartości, druga zawiera indeksy kolumn dla tych wartości, a trzecia wskazuje początek każdego

wiersza w tych tablicach. Dzięki temu można szybko uzyskać dostęp do całych wierszy, co jest bardzo efektywne przy mnożeniu macierzy przez wektory. Format CSR oszczędza pamięć, ale dodawanie nowych elementów może być trudne, ponieważ może wymagać przestawienia całej struktury. Format ten jest szeroko stosowany w obliczeniach naukowych.

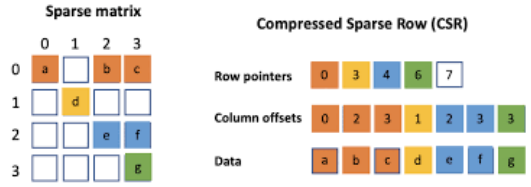

Figure 1: Przykład CSR

- **CSC (Compressed Sparse Column)**: Analogiczny do CSR, ale zoptymalizowany pod kątem operacji na kolumnach.
- **COO (Coordinate Format)**: Format COO (Coordinate Matrix) jest prostym i elastycznym sposobem na przechowywanie macierzy rzadkich. Zamiast jednej zwartej struktury, używa trzech tablic: jednej dla indeksów wierszy, jednej dla indeksów kolumn i jednej dla wartości niezerowych. Ten format jest wygodny do budowania macierzy kawałek po kawałku, ponieważ łatwo można dodawać nowe elementy. Jednakże, ponieważ elementy nie są uporządkowane, operacje arytmetyczne oraz mnożenie przez wektory są mniej efektywne. COO jest często używany jako format przejściowy do tworzenia macierzy przed konwersją na bardziej wydajny format, taki jak CSR lub CSC.

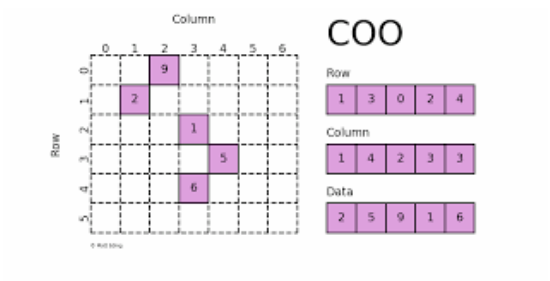

Figure 2: Przykład COO

3. Praktykowane rozwiązania

Grafowe sieci neuronowe (GNN) rozwijają się dynamicznie dzięki wprowadzaniu nowych modeli i technik optymalizacji. Jednym z kluczowych osiągnięć jest wprowadzenie mechanizmu uwagi, który umożliwia dynamiczne różnicowanie ważności informacji propagowanych między węzłami. Mechanizm ten, wdrożony w modelach takich jak Graph Attention Networks (GAT), zwiększa elastyczność i interpretowalność, co pozwala na lepsze modelowanie relacji w złożonych sieciach, takich jak molekuły czy sieci społeczne [2].

W kontekście skalowalności grafowych sieci neuronowych istotnym osiągnięciem jest model GraphSAGE, który wprowadza indukcyjne uczenie reprezentacji. Dzięki próbkowaniu sąsiedztwa oraz agregacji cech węzłów model umożliwia efektywne przetwarzanie dużych grafów, redukując jednocześnie złożoność obliczeniową [3].

Optymalizacja reprezentacji danych jest równie istotna. Format COO, będący prostą i elastyczną formą przechowywania macierzy rzadkich, często pełni rolę formatu przejściowego. Bardziej zaawansowane formaty, takie jak CSR i CSC, pozwalają na znaczną poprawę wydajności obliczeniowej, szczególnie w operacjach algebraicznych odpowiednio na wierszach i kolumnach [1, 3]. Format CSR jest szeroko stosowany w analizie dużych grafów społecznych, natomiast CSC sprawdza się w aplikacjach wymagających intensywnych operacji kolumnowych [1, 2].

Te podejścia wspólnie umożliwiają bardziej efektywne i skalowalne przetwarzanie danych grafowych, co ma kluczowe znaczenie w zastosowaniach, takich jak systemy rekomendacyjne, analiza molekuł czy optymalizacja sieci społecznych. Wybór odpowiednich formatów danych i modeli pozwala na znaczną redukcję kosztów obliczeniowych, szczególnie w kontekście dużych zbiorów danych [1, 2, 3].

4. Opis naszego podejścia

Nasze podejście polegało na wykorzystaniu istniejących metod i technik w celu porównania wydajności obliczeniowej formatów danych oraz bibliotek PyTorch Geometric (PyG) i Deep Graph Library (DGL) w kontekście grafowych sieci neuronowych (GNN). Skupiliśmy się na analizie trzech kluczowych metryk czasowych:

- **Edge-to-edge time:** Całkowity czas od załadowania danych, poprzez obliczenie wag, do forward pass.
- **Training time:** Czas poświęcony na obliczenie gradientów.
- **Inference time:** Czas na generowanie predykcji gotowego modelu.

Eksperymenty przeprowadzono na zestawach danych Planetoid (*Cora*, *Citeseer*, *PubMed*) oraz OGB (*ogbn-arxiv*). W celu zoptymalizowania wykorzystania zasobów obliczeniowych i przyspieszenia przetwarzania dużych grafów zastosowaliśmy **batching**, czyli podział danych wejściowych na mniejsze porcje (*batch-size*).

Batching pozwala na efektywne wykorzystanie

pamięci GPU, szczególnie w przypadku bardzo dużych zbiorów danych, które nie mieszczą się w całości w pamięci jednocześnie. Dzięki temu możliwe jest równoczesne przetwarzanie mniejszych podgrafów, co zmniejsza wymagania pamięciowe i skraca czas trenowania.

5. Wyniki i analiza

5.1. Zbiór PubMed

Na początku skupiliśmy się na stosunkowo małym zbiorze PubMed z rodziny zbiorów Planetoid. Reprezentuje on sieć cytowań artykułów naukowych w dziedzinie biomedycyny. Jest szeroko wykorzystywanych w badaniach nad GNN. Testy przeprowadzone na wcześniej wymienionych modelach, uwzględniając odpowiednie formaty danych teestowane na urządzeniach CPU czyli układ procesora oraz CUDA (Compute Unified Device Architecture) czyli opracowana przez firmę Nvidia uniwersalna architektura procesorów wielordzeniowych, głównie kart graficznych, (w dalszej części nazywana GPU) dały następujące rezultaty.

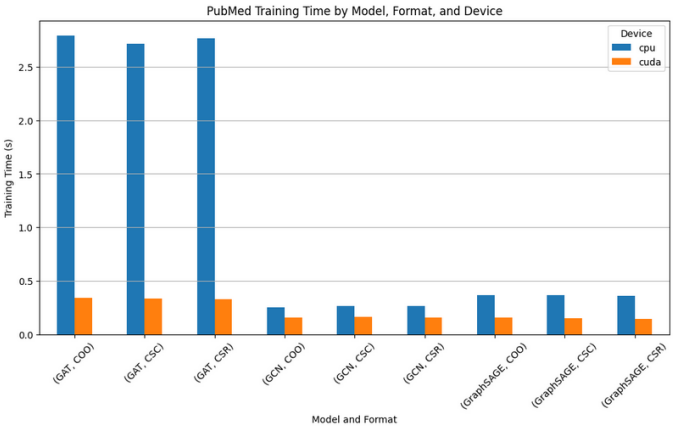

Figure 3: PubMed - Training Time, batch 32

Model GAT charakteryzował się najdłuższym czasem treningu. Różnica ta była szczególnie widoczna w przypadku CPU, gdzie czas treningu był kilkukrotnie dłuższy niż w przypadku GCN czy GraphSAGE. Na GPU model GAT wykazał znaczną poprawę wydajności, zmniejszając czas treningu o ponad 85% w stosunku do CPU. GCN był najbardziej wydajnym modelem, zarówno na CPU, jak i GPU. Czas treningu dla tego modelu był kilkukrotnie krótszy niż w przypadku GAT i około 40% krótszy niż dla GraphSAGE na GPU. GraphSAGE osiągał wyniki pośrednie pomiędzy GCN a GAT, wykazując stabilność w czasach treningu na GPU, gdzie różnice między formatami danych były marginalne.

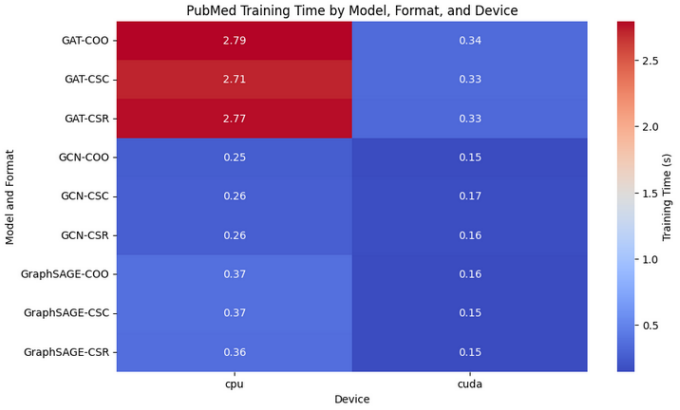

Figure 4: Mapa ciepła zależności w Training Time

Różnice GPU-CPU

Format	Model	CPU E2E Time (s)	CPU Training Time (s)	CPU Inference Time (s)	GPU E2E Time (s)	GPU Training Time (s)	GPU Inference Time (s)
COO	GAT	3.105264	2.768260	0.280342	0.447127	0.352298	0.038054
	GCN	0.326776	0.275998	0.029977	0.255254	0.193430	0.021584
	GraphSAGE	0.472690	0.398942	0.046801	0.252988	0.192258	0.015194
CSC	GAT	3.007289	2.698074	0.258392	0.426476	0.344486	0.033771
	GCN	0.351501	0.294031	0.033776	0.249097	0.186129	0.024745
	GraphSAGE	0.488774	0.414741	0.047038	0.213618	0.168814	0.010439
CSR	GAT	3.002700	2.668755	0.275163	0.430550	0.344484	0.036695
	GCN	0.321865	0.273939	0.029932	0.237027	0.185154	0.019376
	GraphSAGE	0.459006	0.393714	0.044394	0.241117	0.180184	0.015260

Figure 5: Dokładne dane dla batch 32

Format	Model	CPU E2E Time (s)	CPU Training Time (s)	CPU Inference Time (s)	GPU E2E Time (s)	GPU Training Time (s)	GPU Inference Time (s)
COO	GAT	3.111138	2.818688	0.250813	0.432778	0.324389	0.043409
	GCN	0.281645	0.232906	0.029800	0.177470	0.115755	0.022407
	GraphSAGE	0.410397	0.336761	0.046538	0.171694	0.121674	0.011201
CSC	GAT	3.033298	2.731669	0.257454	0.427617	0.324879	0.041849
	GCN	0.282598	0.234318	0.029874	0.205581	0.144891	0.022362
	GraphSAGE	0.385038	0.320707	0.043832	0.199795	0.137466	0.015113
CSR	GAT	3.173167	2.866448	0.259039	0.419628	0.317967	0.043379
	GCN	0.300560	0.252053	0.030122	0.191065	0.133628	0.021275
	GraphSAGE	0.390162	0.325056	0.044214	0.170262	0.114835	0.013767

Figure 6: Dokładne dane dla batch 64

W przypadku modelu GAT różnice w czasach treningu między GPU a CPU były największe, przekraczając 90% dla obu batch size i wszystkich formatów danych. Dla modeli GCN i GraphSAGE różnice GPU-CPU były mniej wyraźne, ale nadal przekraczały 60%, co podkreśla korzyści z akceleracji GPU w przypadku tych modeli.

Wpływ batch size

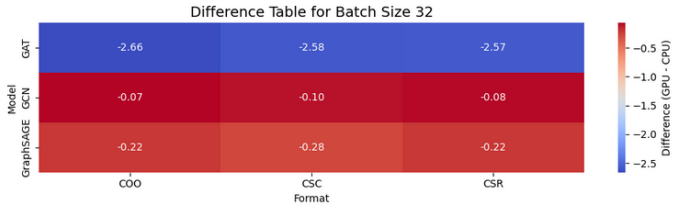

Figure 7: Różnica po zastosowaniu GPU (batch 32)

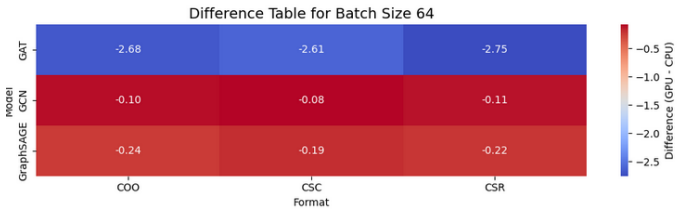

Figure 8: Różnica po zastosowaniu GPU (batch 64)

Wzrost batch size z 32 do 64 wpłynął pozytywnie na czasy treningu na GPU, które uległy redukcji średnio o około 15% dla wszystkich modeli. Na CPU wpływ batch size był mniej widoczny, z niewielkimi różnicami w czasach treningu (poniżej 5%). Największe korzyści z większego batch size zaobserwowano dla modeli GCN i GraphSAGE, gdzie redukcje czasów były najbardziej wyraźne.

Formaty danych

Formaty CSR i CSC wykazywały bardzo zbliżone wyniki w większości przypadków, szczególnie na GPU, gdzie różnice w czasach treningu były poniżej 5% dla wszystkich modeli. (odniesienie do tabelki) Format COO był zauważalnie wolniejszy na CPU, zwłaszcza w przypadku modelu GAT, gdzie różnica względem CSR i CSC wynosiła ponad 10%. Na GPU różnice te były mniej znaczące, ograniczając się do mniej niż 3%.

Poniżej tabele reprezentujące zbiór na różnych wielkościach batchy. Te dane nie biorą udziału w wiarygodności formatu, gdyż przedstawiają uśrednioną wartość traktując urządzenia CPU i GPU jednakowo, co mogło by prowadzić do błędnych wniosków.

PubMed Format Summary for Batch Size 32:

Format	E2E Time (s)	Training Time (s)	Inference Time (s)
COO	0.810016	0.696864	0.071992
CSC	0.789459	0.684379	0.068027
CSR	0.782044	0.674372	0.070137

Figure 9: Różnica po zastosowaniu GPU (batch 32)

PubMed Format Summary for Batch Size 64:

Format	E2E Time (s)	Training Time (s)	Inference Time (s)
COO	0.764187	0.658362	0.067361
CSC	0.755654	0.648988	0.068414
CSR	0.774141	0.668331	0.068633

Figure 10: Różnica po zastosowaniu GPU (batch 64)

Jednakże dobrze pokazuję to wcześniej wspomnianą tendencje do spaku czasu operacyjnego przy zastosowaniu większej próbki.

5.2. Zbiór Cora

Cora to jeden z najpopularniejszych zbiorów danych, również z rodziny Planetoid. Reprezentuje on sieć cytowań artykułów naukowych z dziedziny uczenia maszynowego.

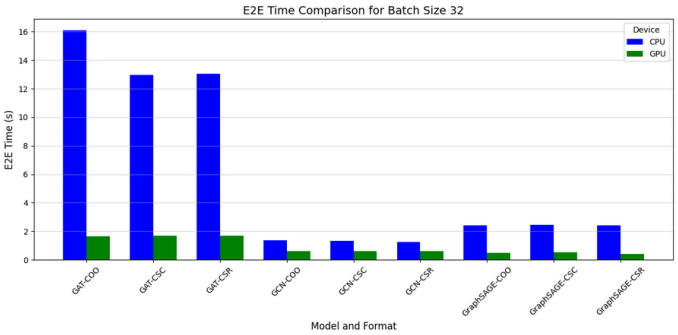

Figure 11: Zbiór Cora batch 32

Zbiory z rodziny Planetoid są stosunkowo podobne, więc ich analiza powinna wykazać podobne obserwacje. A mianowicie model GAT wykazywał najwyższe czasy treningu na CPU w porównaniu do GCN i GraphSAGE dokładnie jak w zbiorze PubMed. Różnice były szczególnie widoczne przy mniejszych batch size, gdzie czas treningu na CPU był kilkunastokrotnie dłuższy w stosunku do GCN.

GCN, jako najwydajniejszy model, osiągał czasy treningu krótsze o ponad 80% w stosunku do GAT na CPU oraz około 50% na GPU. Jest to oczywiście związane ze zwiększoną liczbą potrzebnych operacji w owym modelu.

GraphSAGE plasował się pomiędzy GAT a GCN pod względem wydajności. Czasy treningu na GPU dla GraphSAGE były stabilne i różniły się nieznacznie pomiędzy formatami danych, co wskazuje na jego dobrą optymalizację.

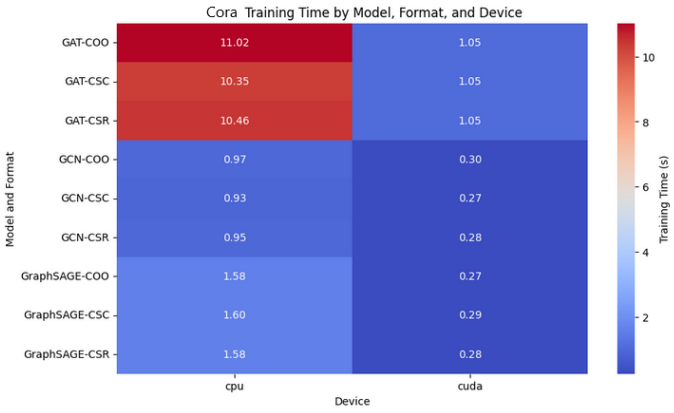

Figure 12: Mapa ciepła zbioru Cora (średnia względem batch)

Zastosowanie GPU

Największe korzyści z wykorzystania GPU zaobserwowano w przypadku modelu GAT, gdzie czas treningu na GPU był krótszy o ponad 90% w porównaniu do CPU.

Dla modeli GCN i GraphSAGE różnice były mniejsze, ale nadal istotne, przekraczając 70% w większości przypadków.

Batching

Wraz ze wzrostem batch size zauważono poprawę wydajności na GPU. Czasy treningu zmniejszyły się średnio o 20% przy przejściu z batch size 32 do 128 dla wszystkich modeli.

Na CPU wpływ batch size był mniej wyraźny, jednak wzrost batch size do 128 prowadził do stabilizacji czasów treningu, szczególnie dla modeli GCN i GraphSAGE.

Cora Dataset - CPU Format and Model E2E Time Comparison by Batch Size:

	E2E Time (s)								
Format	COO			CSC			CSR		
Model	GAT	GCN	GraphSAGE	GAT	GCN	GraphSAGE	GAT	GCN	GraphSAGE
Batch Size									
32	16.083444	1.390213	2.395665	12.987549	1.333699	2.458369	13.061496	1.246826	2.431277
64	11.372219	1.094461	1.739607	11.196409	1.019279	1.709191	11.204181	1.043790	1.726140
128	9.899184	0.890914	1.385452	9.560788	0.840513	1.453852	9.747180	0.907361	1.378708

Figure 13: Cora - zależność batchy CPU

Cora Dataset - GPU Format and Model E2E Time Comparison by Batch Size:

	E2E Time (s)								
Format	COO			CSC			CSR		
Model	GAT	GCN	GraphSAGE	GAT	GCN	GraphSAGE	GAT	GCN	GraphSAGE
Batch Size									
32	1.659554	0.605574	0.470627	1.710260	0.617266	0.524351	1.689765	0.618111	0.421710
64	1.294106	0.324861	0.345566	1.288344	0.313979	0.360696	1.292594	0.311449	0.388084
128	0.921137	0.315045	0.266001	0.908852	0.203561	0.278805	0.903066	0.235095	0.255244

Figure 14: Cora - zależność batchy GPU

Stosowany format

Dla pracy na procesorze model GAT charakteryzował się gorszą wydajnością na macierzy COO w porównaniu do pozostałych formatów. Formaty CSR i CSC zaś wypadają podobnie. Jednakże przy obserwacjach na GPU wyniki te nie są jednoznaczne.

Na pozostałych modelach różnica ta się zaciera, co pozwala wnioskować, że znaczenie to ma przy bardziej wymagających modelach.

5.3. Zbiór ArXiv

Przechodząc teraz do znacznie wiekszego zbioru z Open Graph Benchmark (OGB), gdzie liczba węzłów to juz 169 tys. wraz z liczbą krawędzi 1.1 miliona. Reprezentuje on sieć cytowań artykułów naukowych w wielu dziedzinach, publikowanych na platformie arXiv.

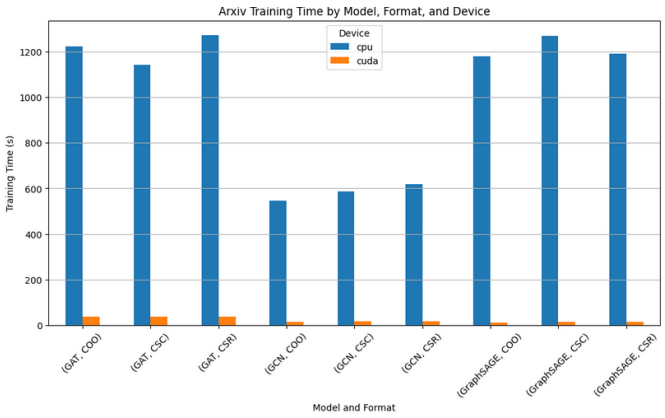

Figure 15: Zbiór Arxiv uśredniony

W niektórych przypadkach czas treningu na CPU był ponad 10-krotnie dłuższy niż na GPU. Najbardziej jest to widoczne przy modelu GraphSAGE.

Tym razem przy pracy z CPU zaobserwować można podobieństwo GraphSage do GAT, przeciwnie jak to miało miejsce przy mniejszych zbiorach.

GCN okazał się najbardziej wydajnym modelem, osiągając czasy treningu krótsze o ponad 80% w stosunku do GAT na CPU oraz o około 50% na GPU.

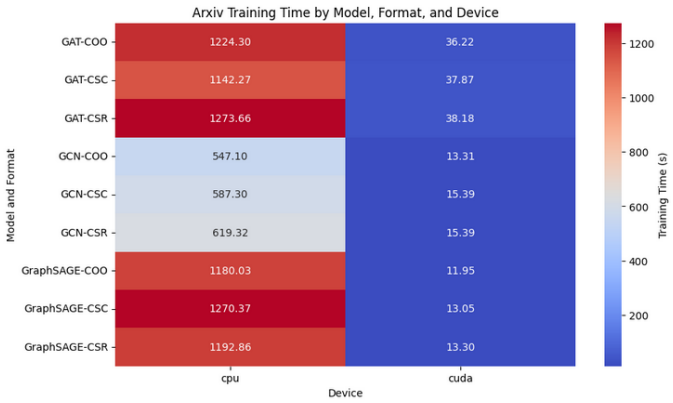

Figure 16: Mapa ciepła uśredniona względem batch

Zwiększenie próbki

Wzrost batch size z 32 do 64 przyczynił się do redukcji czasów treningu na GPU jak i CPU, zachowując tendencje ukazaną przy mniejszych zbiorach. Oczywiście zauważalna jest skala, że ta różnica jest znacznie większa przy tak dużym zbiorze, co jest powodem do stosowania urządzeń CUDA przy pracy z takimi zbiorami danych. Poniżej zamieszczono dwie tabele pokazujące zmiany numeryczne dla obu wielkości próbek.

Stosowany format

Badania przy zbiorze ArXiv nie pozwalają jednoznacznie stwierdzić charakterystyk formatu w porównaniu do innych. Zaobserwowana zależność COO jako najszybszego formatu na urządzeniu GPU, co daje sprzeczność z wcześniej zakładaną hipotezą. Wymagać to będzie ponownego zagłębienia się w strukturę i skupienia się tylko i wyłącznie na tym aspekcie

Format	Model	CPU E2E Time (s)	CPU Training Time (s)	CPU Inference Time (s)	GPU E2E Time (s)	GPU Training Time (s)	GPU Inference Time (s)
COO	GAT	1543.727375	1440.632495	101.113215	41.111250	36.743870	2.609728
	GCN	906.740286	755.795412	148.086487	22.073641	19.906387	0.584578
	GraphSAGE	1854.048564	1541.934928	310.336884	20.620213	17.971669	0.797827
CSC	GAT	1380.391637	1282.035447	96.164650	45.737837	40.639152	2.830617
	GCN	906.121105	742.981165	161.208298	25.936280	22.185282	1.348098
	GraphSAGE	2060.747300	1734.870655	322.617797	21.539405	18.498070	0.807790
CSR	GAT	1484.254823	1381.568479	100.460989	45.538499	40.503433	2.785275
	GCN	917.625242	763.823670	151.944047	25.300709	22.482125	0.969545
	GraphSAGE	1887.303048	1564.935930	320.362062	21.600745	18.567652	0.806736

Figure 17: Tabela dla próbki 32

Format	Model	CPU E2E Time (s)	CPU Training Time (s)	CPU Inference Time (s)	GPU E2E Time (s)	GPU Training Time (s)	GPU Inference Time (s)
COO	GAT	1064.213782	1007.965363	54.314518	40.646023	35.954262	2.825716
	GCN	558.166839	442.757336	113.596878	13.342774	10.012565	1.352334
	GraphSAGE	1112.812263	818.132538	292.746288	11.623813	8.931710	0.809108
CSC	GAT	1066.310538	1002.503645	61.620951	39.907651	36.482784	2.052710
	GCN	552.125175	431.619170	118.544619	15.087394	11.987372	1.110366
	GraphSAGE	1104.130496	805.864011	296.457507	12.807548	10.332416	0.594350
CSR	GAT	1218.985236	1165.748724	51.038721	42.099890	37.021596	2.831964
	GCN	589.390741	474.809863	112.779051	15.391453	11.851060	1.345041
	GraphSAGE	1127.780205	820.779196	304.969949	13.322483	10.658722	0.682729

Figure 18: Tabela dla próbki 64

6. Podsumowanie

Wydajność GPU w porównaniu do CPU

We wszystkich zbiorach danych zastosowanie GPU prowadziło do znaczącej redukcji czasów treningu i inference. Korzyści te były bardziej widoczne dla większych zbiorów, takich jak PubMed i ArXiv, gdzie różnice czasowe między GPU a CPU wynosiły nawet ponad 90%. Dla mniejszych zbiorów, takich jak Cora, GPU również przynosiło korzyści, ale różnice w wydajności były mniej wyraźne, co wynika z niższego zapotrzebowania na zasoby obliczeniowe.

Wpływ batch size

Wzrost batch size pozytywnie wpływał na wydajność obliczeń na GPU, redukując czasy treningu średnio o 15–25% przy przejściu z 32 do 64 i 128. Na CPU efekt wzrostu batch size był mniej znaczący, ale zauważalny w kontekście stabilizacji czasów obliczeń.

Charakterystyka modeli

- **GCN**: Najbardziej wydajny model na CPU i GPU we wszystkich zbiorach danych, co wynika z jego prostoty i mniejszego zapotrzebowania na zasoby obliczeniowe.
- **GraphSAGE**: Stabilny pod względem wydajności, szczególnie na GPU. Model ten wykazywał dobrą skalowalność, zwłaszcza w dużych zbiorach danych.
- **GAT**: Model najbardziej wymagający obliczeniowo, z najwyższymi czasami treningu, szczególnie na CPU. Jednocześnie wykazywał największe korzyści z zastosowania GPU, redukując czasy treningu o ponad 85% w porównaniu do CPU.

Wpływ formatu danych

- CSR i CSC oferowały porównywalną wydajność, szczególnie na GPU, gdzie różnice między tymi formatami były marginalne.
- Format COO wykazywał większą zmienność w wynikach, czasami osiągając wydajność zbliżoną do CSR i CSC, ale w innych przypadkach był mniej efektywny, zwłaszcza na CPU. Przy dużym zbiorze jednak porównanie to dało niejednoznaczne wyniki. Będzię to dla nas motywacja do dalszej pracy i rozwijania projektu w tym temacie.

Wnioski

- **Optymalizacja GPU**: GPU wykazuje znaczne korzyści w zadaniach związanych z GNN, szczególnie dla bardziej złożonych modeli (np. GAT) i dużych zbiorów danych. Rekomenduje się dalsze badania nad wydajnością GPU w kontekście różnorodnych parametrów i architektur.
- **Formaty danych**: CSR i CSC są najbardziej efektywne w analizowanych scenariuszach, ale dalsza optymalizacja COO może zwiększyć jego przydatność w specyficznych przypadkach.
- **Zastosowanie batch size**: Większe batch size poprawiają wydajność GPU, co jest kluczowe dla przetwarzania dużych grafów.

Spojrzenie globalne na zbiory pokazuje, że wybór modelu, formatu danych oraz parametrów takich jak batch size ma kluczowe znaczenie dla efektywności obliczeniowej w zastosowaniach GNN.

Zagłębienie się w struktury algorytmów, w tym identyfikacja faktycznych funkcji i parametrów odpowiedzialnych za zmiany w wydajności, pozwoli nie tylko na ich optymalizację, ale także na opracowanie nowych metod, które mogą znacząco zwiększyć efektywność w aplikacjach grafowych.

References

[1] William L. Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *arXiv preprint arXiv:1706.02216*, 2017.

[2] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2017.

[3] Petar Veličković et al. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.