# 1 Wstęp

Ostatnie lata dobitnie pokazują potrzebę demokratyzacji mocy obliczeniowej. Dotychczasowy paradygmat uczenia maszynowego, zwłaszcza w stosunku do dużych modeli, nie przewiduje jakiegokolwiek stanu pomiędzy tymi, którzy posiadają środki na trening na dużą skalę, a tymi, którzy skazani są na korzystanie z zamkniętych API.

Nieliczne promyki nadziei dla 'małych graczy' objawiały się dotychczas w postaci modeli wypuszczanych zgodnie z ideą open - source, z permisywnymi licencjami (LLaMa, DeepSeek, etc.)

Niniejszy dokument ma na celu zwięzłe przedstawienie idei stojących za nadziejami na wytworzeniem systemu rozsianego uczenia, pozwalającemu na trenowanie dużych modeli bez inwestowania w kosztowną infrastrukturę informatyczną.

# 2 Ogólny zarys

Zgodnie z SOTA z roku 2024 ilość danych, koniecznych do wykonania update parametrów modelu wielkości 1.2B wynosi około 74 GB [1]. Zgodnie z [2] możliwa jest redukcja ilości wysyłanych i odbieranych danych o około 857 razy, tj. dla modelu wielkości 1.2B do około 85 MB.

Jest to możliwe za pomocą wymiany optymalizatora na ten przedstawiony w [2]. Z racji, że jest to optymalizacja optymalizatora możliwe jest dalsze zastosowanie algorytmów skupionych na rozsianym treningu dla dalszej redukcji ilości wysyłanych i pobieranych danych w każdym kroku treningu, takich jak np. [3].

W samej rzeczy, naiwne rozdzielenie grafu komputacjonalnego na kilka mniejszych najprawdopodobniej będzie naturalnym wymogiem, z tej racji, że mało która konsumencka karta graficzna posiada dostateczne ilości VRAMu do zmieszczenia całości grafu. Można więc szacunkowo mówić o np. jednym nodzie składającym się z 14 konsumenckich kart graficznych, każda po 6 GB VRAMu - innymi słowy indywidualny fragment danego noda będzie najprawdopodobniej potrzebował wysłać i pobrać dane wielkości 6 MB co każdy krok treningu.

Krytycznie ważnym spostrzeżeniem jest to, że powyższe podejście jest szacunkiem do metody *naiwnej*, tj. niekoniecznie reprezentatywne dla faktycznej implementacji. Niezależnie od tego faktu, mało prawdopodobnym jest oczekiwać znacznych wzrostów ilości pobieranych i wysyłanych danych, ponownie, dzięki temu, że DeMo jest optymalizatorem.

# 3 DeMo

DeMo, Decoupled Momentum Optimization, zostało opracowane przez Nous Research [4], grupę researchową mającą swój oficjalny początek w 2023 i Diederika Kingmę, oryginalnego wynalazcy Adama [5]. Nous jest uznaną organizacją, ich LLMy (np. Hermes [6]) są bardzo dobrej jakości (tj. mają swoje zastosowania nawet w obliczu Claude czy GPT4o, innymi słowy wobec największych graczy)

Algorytm ma potwierdzenie empiryczne, tj. został nim wytrenowany duży model (15B) przez internet, w sposób wcześniej niemożliwy. Kod do reprodukcji jest publicznie dostępny. DeMo nie ma obecnie matematycznie udowodnionych podstaw poza przypuszczeniami.

Proponuje następujący empiryczny test algorytmu na potrzeby średnich i małych modeli:

Na ilość GPU mniej więcej równej tej dostępnej na Politechnice Łódzkiej spróbować wytrenować modele tekstowe 1) małe, tj. poniżej 50M parametrów (z parametrami w bfloat16 poniżej 0.5 GB) 2) średnie 500M (5GB w bfloat16) za pomocą czystego algorytmu DeMo bez algorytmów rozsianego trenowania w 1 przypadku i z najbardziej pesymistyczną, najbardziej naiwną wersja algorytmów rozsianego treningu (tj. chałupnicza implementacja idei a nie SWARM) w przypadku 2.

Należy odnotować fakt, że ilość wysyłanych danych wydaje się, na podstawie empirycznych przesłanek, spadać wraz ze wzrostem wielkości modelu. Co więcej, jeżeli postawi się dedykowane serwery do agregacji danych treningowych i centralnego zarządzania możliwe jest zredukowanie ilości danych *wysyłanych* z każdego noda o około 40 razy, przy takiej samej ilości wysyłanych danych.

Co ważne, wszystkie liczby zakładają *kompletny brak kompresji, tj. tensory przesyłane są as-is.* Dalsza redukcja wielkości wysyłanych i odbieranych danych przez proste algorytmy kompresji wydaje się najoczywistszym kandydatem na dalsze zbicie ilości wysyłanych i odbieranych danych.

Dalej, algorytm DeMo wykazuje *pewną niestabilność* przy małych i średnich modelach, tj. model

wielkości 1.2B jest najmniejszym modelem, który empirycznie converguje bez nieprzewidzianych zachowań. Tym bardziej zasadne jest ustawienie empirycznych testów na *małą* skalę.

Najważniejszą informacją dla naszego przedsięwzięcia jest fakt, że DeMo *jest w dużej mierze odporne na małe ilości nodeów umierających podczas danego kroku.*

# 4 DisTro

Nous ma w planach wypuszczenie swojego własnego oprogramowania od rozsianego treningu [1]. Nous najpewniej jednak nie będzie zajmował się przypadkami modeli małych i średnich. Co więcej, wymagany jest jakiegoś rodzaju centralna administracja; nawet jeżeli nie serwer agregujący dane treningowe dla n nodeów, to na pewno serwer z rejestrem modeli, które są w trakcie treningu albo na trening oczekują - chociażby aby zapewnić poprawne wypełnienie europejskiego AI Act [7].

W samej rzeczy, plany Nous Research z DisTro są w dużej mierze ortogonalne do naszej inicjatywy, jeśli nie marginalnie zbliżone w swoim kierunku - dowolny algorytm oparty na tzw. federated learning [8] można zaadoptować do wypracowanego przez nas frameworku z niewielką ilością pracy.

Co więcej, podobne inicjatywy (tj. community oriented computing power donation) istnieją od wielu lat i są ważnym elementem np. poszukiwania największych znanych liczb pierwszych: [9]. Można wziąć naukę z długich lat doświadczeń podobnych projektów, tym samym przyśpieszając prace administracyjno - organizacyjne.

Co do samego algorytmu DisTro niewiele obecnie jest wiadomo, poza generalnymi założeniami, i faktem, że jako proof of concept publicznie został wytrenowany model językowy o rozmiarze 15B przez internet, z poszczególnymi nodami na różnych kontynentach - coś niemożliwego do wykonania w podobnym czasie bez DeMo (de facto w ogóle nie do wykonania bez DeMo, lub absurdalnej ilości pieniędzy)

# 5 Alternatywne prace i metody prowadzące do redukcji wielkości gradientów

Z przeglądu literatury wynika kilka alternatywnych podejść do redukcji wielkości gradientów. Pierwszym godnym uwagi jest kompresja do niskiej precyzji, gdzie gradienty są reprezentowane przy użyciu formatu 4-bitowego (1 bit znaku, 3 bity wykładnika, 0 bitów mantysy), znanego jako E3M0. Badania empiryczne pokazują, że taka kompresja nie powoduje istotnej degradacji jakości modelu, nawet przy treningu na dużą skalę.

Alternatywną metodą jest wykorzystanie dekompozycji niskiego rzędu (low-rank compression). Techniki te pozwalają na znaczącą redukcję ilości przesyłanych danych poprzez aproksymację macierzy gradientów za pomocą ich reprezentacji niskiego rzędu.

Warto również wspomnieć o metodach łączących różne techniki kompresji (np. zerowanie, wybór top-k wartości, niska precyzja) w jeden spójny algorytm. Takie hybrydowe podejścia mogą osiągać lepsze wyniki niż pojedyncze metody, choć kosztem większej złożoności implementacyjnej.

Na szczególną uwagę zasługuje metoda Streaming DiLoC [10]

Głównym założeniem Streaming DiLoCo jest zmiana sposobu synchronizacji parametrów modelu. Zamiast synchronizować wszystkie parametry jednocześnie podczas rund komunikacyjnych, metoda ta synchronizuje różne podzbiory parametrów (nazywane fragmentami) sekwencyjnie. Na przykład w modelu transformerowym może synchronizować różne bloki transformerowe w różnych momentach. To znacząco redukuje szczytowe wymagania dotyczące przepustowości, ponieważ w danym momencie trzeba przesłać tylko część modelu.

Szczególnie innowacyjnym aspektem jest nakładanie się obliczeń i komunikacji. Podczas gdy jeden fragment jest synchronizowany między węzłami, model kontynuuje trening wykorzystując pozostałe fragmenty. Oznacza to, że system nie musi wstrzymywać treningu w oczekiwaniu na zakończenie komunikacji, co znacząco poprawia ogólną efektywność obliczeniową.

Kolejną kluczową zaletą Streaming DiLoCo jest odporność na warunki sieciowe. W przeciwieństwie do treningu data-parallel, który wymaga

niemal idealnych warunków sieciowych, Streaming DiLoCo może utrzymać jakość treningu nawet przy znacznym opóźnieniu sieci lub sporadycznych awariach węzłów. Sprawia to, że metoda ta jest szczególnie odpowiednia do treningu między centrami danych lub w środowiskach, gdzie warunki sieciowe mogą być mniej niż idealne. Wyniki empiryczne są przekonujące - Streaming DiLoCo osiąga porównywalną jakość modelu do treningu data-parallel, wymagając około 1 Gbits/s przepustowości, w porównaniu do setek Gbits/s potrzebnych w tradycyjnych podejściach. Jest to alternatywa naturalnie zasadniczo gorsza do modelu opartego na DeMo; jest to jednak metoda godna uwagi z racji na kompletnie inną stronę, z której podchodzi do problemu.

# 6 Istniejące prace w tzw. federated learning

Istnieje znaczące ciało prac w kategorii tzw. federated learningu. Głównymi pracami wartymi wspomnienia są SWARM Pararellism [3] i PyTorch Fully Sharded Data Pararell [11].

Innymi słowy, problem napisania algorytmu rozsianego treningu w taki sposób, aby nie limitował treningu modeli najsłabszym urządzeniem w danej siecie (zarówno w ramach mocy obliczeniowej, jak i byc może bardziej znacząca, w ramach dostępnego VRAMu) został rozwiązany.

SWARM nie tylko pozwala na efektywną, asynchroniczą dystrybucje mocy obliczeniowej w taki sposób żeby zminimalizować idle time wszystkich urządzeń, ale pozwala również na najefektywniejsze zarządzanie danymi treningowymi możliwe, tj. w wypadku podjęcia decyzji o braku dedykowanego serwera do agregacji danych treningowych, problem rozdzielanaia danych jest rozwiązany. Co więcej, algorytm SWARM można w stosunkowo prosty sposób rozciągnąc aby w oparciu o prace z zakresu kryptografii, tj, możliwym jest dodatkowe zabezpieczenie danej sieci rozsianego uczenia przed złymi agentami.

PyTorch FSDP rozwiązuje problem 'zmieszczenia' modelu na różnorodnych urządzeniach, tj. jest to algorytm w pełni zintegrowany z PyTorchem pozwalającym na trening modelu na urządzeniu, którego VRAM nie pozwala na zmieszczenie całego modelu na raz. Innymi słowy, PyTorch FSDP rozwiązuje problem tego, że większość konsumenckich kart graficznych nie posiada wystarczających ilości pamięci na tradycyjny trening dużych modeli uczenia maszynowego

# 7 Rola potencjalniej centralnej organizacj

Jak wspomniano wyżej, aby zapewnić poprawne przestrzeganie wymagań nałożonych na trening modeli uczenia maszynowego przez EU AI Act, konieczny jest w minimalnym wypadku rejestr modeli wraz ze spisem danych treningowych wykorzystanych do wytworzenia modelu.

Centralna organizacja jest również konieczna do szybkiej naprawy wszelkiego rodzaju bugów, nadzoru nad trwającymi procesami trenowania modeli, weryfikacją kandydatów na nowe modele do trenowania w sposób rozsiany, promocje przedsięwzięcia i ciągłe udoskonalanie oprogramowania w nadziei na bardziej efektywny trening.

Dodatkowo, przy założeniu dostępu do High Performance Clusters, konieczna jest centralna organizacja aby móc połączyć wysiłki 'tradycyjnego' trenowania modeli z trenowaniem rozsianym.

Ponadto, rolą centralnej organizacji w naturalny sposób byłoby przechowywanie chekpointów w razie awarii danej sieci uczenia rozsianego, jak i również dla powodów udostępniania modeli i archiwalnych. Konieczne jest również wypracowanie holistycznego podejścia do usuwania z danych treningowych materiałów niepożądanych bądź nielegalnych, jak i również dalsza praca w dziedzinie trenowania na dannych potencjalnie wrażliwych z zachowaniem pełnej prywatności.

# 8 Kierunki rozwoju

Jednym z głównych kierunków rozwoju jest opracowanie mechanizmów minimalizujących potencjał na adwersalne działania płynące z jednego czy więcej nodeów, mających na celu zakłócenie przebiegu treningu tj. mechanizmów opartych o kryptografię.

Oczywistym i stałym kierunkiem rozwoju jest, jak wyżej wspomniano, ciągły rozwój i udoskonalanie właściwego oprogramowania odpowiedzialnego za rozsiany trening - uczenie maszynowe jest jedną z szybciej rozwijających sie dziedzin nauk.

Jednym z bardziej odległych kierunków rozwoju jest potencjalna komercjalizacja rozsianego

treningu.

Istnieją duże ilości GPU nie będących wykorzystywanych w pełni, tj działających $> 80\%$ wykorzystania. Wiele z tych GPU należą do podmiotów for - profit. Możliwym byłoby wykorzystanie uczenia rozsianego do zebrania mocy obliczeniowej z ww. GPU, albo sprzedając ją innym zainteresowanym podmiotom, albo udostępniając ją podmiotom posiadającym infrastrukturę, pozwalając im, poniekąd 'za darmo', na trening modeli dostosowanych do ich potrzeb.

W samej rzeczy, wyżej wspomniany model nie wymagał by żadnej dodatkowej pracy ze strony technicznej.

# References


[1] B. Peng, J. Quesnelle, D. Rolnick, A. Lotter, U. H. Adil, and E. L. Rocca, "A preliminary report on distro," 2024. [Online]. Available: https://github.com/NousResearch/DisTrO

[2] B. Peng, J. Quesnelle, and D. P. Kingma, "Demo: Decoupled momentum optimization," 2024. [Online]. Available: https://arxiv.org/abs/2411.19870

[3] M. Ryabinin, T. Dettmers, M. Diskin, and A. Borzunov, "Swarm parallelism: Training large models can be surprisingly communication-efficient," 2023. [Online]. Available: https://arxiv.org/abs/2301.11913

[4] "Nous research." [Online]. Available: https://nousresearch.com/

[5] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2017. [Online]. Available: https://arxiv.org/abs/1412.6980

[6] C. G. teknium, Jeffrey Quesnelle, "Hermes 3 technical report," 2024. [Online]. Available: https://nousresearch.com/wp-content/uploads/2024/08/Hermes-3-Technical-Report.pdf

[7] "Artificial intelligence act," 2024. [Online]. Available: https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:32024R1689

[8] J. Konečný, B. McMahan, and D. Ramage, "Federated optimization:distributed optimization beyond the datacenter," 2015. [Online]. Available: https://arxiv.org/abs/1511.03575

[9] "Great internet mersenne prime search." [Online]. Available: https://www.mersenne.org/

[10] A. Douillard, Y. Donchev, K. Rush, S. Kale, Z. Charles, Z. Garrett, G. Teston, D. Lacey, R. McIlroy, J. Shen, A. Ramé, A. Szlam, M. Ranzato, and P. Barham, "Streaming diloco with overlapping communication: Towards a distributed free lunch," 2025. [Online]. Available: https://arxiv.org/abs/2501.18512

[11] Y. Zhao, A. Gu, R. Varma, L. Luo, C.-C. Huang, M. Xu, L. Wright, H. Shojanazeri, M. Ott, S. Shleifer, A. Desmaison, C. Balioglu, P. Damania, B. Nguyen, G. Chauhan, Y. Hao, A. Mathews, and S. Li, "Pytorch fsdp: Experiences on scaling fully sharded data parallel," 2023. [Online]. Available: https://arxiv.org/abs/2304.11277