Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)
Кафедра Штучного інтелекту
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Дослідження методів оптимізації навчання генеративних нейронних мереж
(тема)

Виконав:
здобувач другого року навчання,
групи СШІм-24-2
Владислав ХОЛОДЕНКО
(власне ім'я, прізвище)
Спеціальність 122 Комп'ютерні науки
(код і повна назва спеціальності)
Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)
Освітня програма Системи штучного інтелекту
(повна назва освітньої програми)
Керівник проф. Наталія РЯБОВА
(посада, власне ім'я, прізвище)

Допускається до захисту

Завідувач кафедри ШІ _____ Лариса ЧАЛА
(підпис) (власне ім'я, прізвище)

2026 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук

Кафедра Штучного інтелекту

Рівень вищої освіти другий (магістерський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)

Освітня програма Системи штучного інтелекту
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«_____» _____ 20 ___ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Холоденку Владиславу Сергійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження методів оптимізації навчання генеративних нейронних мереж

затверджена наказом університету від 6 квітня 20 26 р. № 269Ст

2. Термін подання здобувачем роботи до екзаменаційної комісії 26 травня 20 26 р.

3. Вихідні дані до роботи Операційна система: Linux (Ubuntu 20.04+) або Windows 10/11 із встановленим середовищем виконання NVIDIA CUDA та Docker. Середовище розробки – VS Code або PyCharm Professional; мова програмування Python 3.10+; основні бібліотеки та фреймворки – PyTorch, NumPy, Einops та Matplotlib.

4. Перелік питань, що потрібно опрацювати в роботі
1) Аналіз предметної галузі
2) Постановка задачі дослідження
3) Реалізація запропонованих методів
4) Дослідження ефективності цих методів

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Строк / термін виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	06.04.2026	виконано
2	Огляд літературних джерел та аналіз предметної	12.04.2026	виконано
	галузі		
3	Формулювання математичних моделей та	20.04.2026	виконано
	гібридних методів		
4	Програмна реалізація моделей VAE, GAN та	25.04.2026	виконано
	Transformer		
5	Проведення експериментальних досліджень та	28.04.2026	виконано
	збір метрик		
6	Оформлення пояснювальної записки та	02.05.2026	виконано
	висновків		
7	Нормоконтроль	19.05.2026	виконано
8	Антиплагіат	20.05.2026	виконано
9	Захист перед ЕК	26.05.2026	

Дата видачі завдання 6 квітня 2026 р.

Здобувач _____
(підпис)

Керівник роботи _____ проф. Наталія РЯБОВА
(підпис) (посада, власне ім’я, прізвище)

РЕФЕРАТ

Пояснювальна записка: 66 с., 15 рис., 3 табл., 1 дод., 13 джерел.

ГЕНЕРАТИВНІ НЕЙРОННІ МЕРЕЖІ, КОЛАПС МОД, МАТРИЧНЕ ПЕРЕДУМОВЛЕННЯ, НИЗЬКОРАНГОВА ПРОЄКЦІЯ, ОПТИМІЗАЦІЯ, ADAMW, GALORE, MUON, SOPHIA.

Об’єкт дослідження – методи навчання та оптимізації генеративних архітектур нейронних мереж.

Предмет дослідження – алгоритми адаптивної оптимізації, методи другого порядку та методи зниження споживання пам'яті станами оптимізатора.

Мета роботи – підвищення ефективності навчання генеративних моделей шляхом розробки та дослідження гібридних методів оптимізації, що вирішують проблему «стіни пам’яті» та спектрального колапсу.

Методи дослідження – методи стохастичної апроксимації, диференціальної геометрії, сингулярного розкладу матриць та ітераційних методів Ньютона-Шульца. Анотація.

ABSTRACT

Master’s thesis contains: 66 pp., 15 fig., 3 tabl., 1 ann., 13 references.

ADAMW, GALORE, GENERATIVE NEURAL NETWORKS, LOW-RANK PROJECTION, MATRIX PRECONDITIONING, MODE COLLAPSE, MUON, OPTIMIZATION, SOPHIA.

The object of the study is methods for training and optimizing generative neural network architectures.

The subject of the study is adaptive optimization algorithms, second-order methods, and methods for reducing memory consumption by optimizer states.

The aim of the work is to improve the efficiency of training generative models by developing and studying hybrid optimization methods that solve the problem of the “memory wall” and spectral collapse.

Research methods – methods of stochastic approximation, differential geometry, singular value decomposition of matrices, and iterative Newton-Schultz methods. Abstract.

ЗМІСТ

Перелік умовних позначень, символів, одиниць, скорочень і термінів 8
Вступ 9
1 Аналіз предметної галузі 11
1.1 Опис предметної галузі 11
1.1.1 Топологія ландшафту функції втрат 12
1.1.2 Структура пам'яті під час навчання 12
1.2 Актуальність дослідження 13
1.3 Огляд сучасних методів оптимізації 14
1.3.1 Адаптивні скалярні методи першого порядку: AdamW, Lion, AdEMAMix 14
1.3.2 Методи другого порядку та діагональні апроксимації Гессіана: Sophia 16
1.3.3 Методи передумовлення матриць: Shampoo, SOAP та Muon ... 17
1.3.4 Низькорангові та пам'ять-ефективні просторові методи: GaLore та APOLLO 18
1.3.5 Алгоритми без розкладу (Schedule-Free) 19
2 Постановка задачі дослідження 21
2.1 Пропозиція нового методу: динамічна рангово-адаптивна тензорна оптимізація (DRATO) 21
2.2 Пропозиція нового методу: квантований предиктивний імпульс QPM 21
2.3 Гібридний підхід №1: інтеграція низькорангової проєкції GaLore з матрицею Гессе оптимізатора Sophia 22
2.4 Гібридний підхід №2: інтеграція Schedule-Free у простори матричного передумовлення 23
2.5 Пропозиція модифікованого методу Muon для згорткових та багатошарових архітектур 23
3 Реалізація запропонованих методів 26

3.1 Архітектурна реалізація та математичне обґрунтування еталонних генеративних моделей 27
3.1.1 Багатошаровий перцептрон: варіаційний автоенкодер (MLP VAE) 27
3.1.2 Згорткова генеративно-змагальна мережа (CNN DCGAN) 29
3.1.3 Генеративний трансформер-автоенкодер (Generative ViT) 31
3.2 Програмна реалізація та математичне обґрунтування модифікованого алгоритму Muon 33
3.3 Програмна реалізація гібридного проєкційного оптимізатора (GaL-Sophia-SF) 35
4 Дослідження ефективності цих методів 39
4.1 Організація експериментальної бази, апаратне забезпечення та налаштування гіперпараметрів 39
4.2 Аналіз нелінійної динаміки оптимізації варіаційного автоенкодера (MLP VAE) 41
4.3 Дослідження змагальної оптимізації у згорткових генеративних мережах (CNN DCGAN) 44
4.4 Аналіз навчання трансформера-автоенкодера (Generative ViT) та вразливість гібридних просторів 50
4.5 Аналіз алгоритмів без жорсткого розкладу (Schedule-Free) в умовах нестаціонарної кривизни 55
4.6 Порівняльний аналіз експериментальних графіків конвергенції генеративних лосів 56
4.7 Масштабування параметрів моделі та проблема спектральної нормалізації швидкості навчання (RMS Scaling) 60
Висновки 63
Перелік джерел посилання 64
Додаток А Відомість кваліфікаційної роботи 66

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ШІ – штучний інтелект.

EMA – Exponential Moving Average – експоненційне ковзне середнє;

KL-дивергенція – Kullback-Leibler Divergence – дивергенція Кульбака-Лейблера;

LR – Learning Rate – швидкість навчання;

MSE – Mean Squared Error – середньоквадратична помилка;

RMS – Root Mean Square – середньоквадратичне значення;

SVD – Singular Value Decomposition – сингулярний розклад матриці;

VRAM – Video Random Access Memory – відеопам’ять графічного процесора.

ВСТУП

Стрімкий і безпрецедентний розвиток генеративних штучних нейронних мереж, який спостерігається протягом останніх років, спровокував тектонічний зсув у багатьох галузях науки та індустрії. Архітектура Transformer, яка лежить в основі переважної більшості великих мовних моделей та мультимодальних генеративних систем, продемонструвала унікальну здатність до масштабування. Відповідно до емпіричних законів масштабування, продуктивність генеративних моделей передбачувано зростає зі збільшенням трьох ключових факторів: кількості параметрів моделі, обсягу тренувальних даних та загального обсягу обчислень [11].

Однак цей експоненціальний ріст масштабу призвів до виникнення фундаментального технологічного бар'єра, відомого як «стіна пам'яті» та «стіна обчислень». Сучасні моделі досягли розмірів у сотні мільярдів і навіть трильйони параметрів. Для їхнього навчання недостатньо просто розробити архітектуру; критичним елементом стає математичний алгоритм, який оновлює ці параметри на основі градієнтів цільової функції.

Історично еволюція методів оптимізації для глибокого навчання рухалася від базового стохастичного градієнтного спуску (SGD) до методів з імпульсом (Momentum, Nesterov), а згодом – до адаптивних методів, таких як AdaGrad, RMSProp та Adam. Оптимізатор Adam, та його модифікація AdamW [10], залишалися індустріальним стандартом де-факто протягом багатьох років. Проте зі зростанням масштабу моделей AdamW почав демонструвати свої концептуальні обмеження. Найбільшим із них є вимога до збереження станів оптимізатора. Для кожного параметра моделі AdamW вимагає збереження експоненційного ковзного середнього (EMA) першого моменту та EMA другого моменту. У високоточному форматі FP32 це означає додаткові 8 байтів пам'яті на кожен параметр. Таким чином, лише для збереження станів оптимізатора моделі з 100 мільярдами параметрів

потрібно понад 800 гігабайтів відеопам'яті, що вимагає використання величезних кластерів із сотнями спеціалізованих графічних процесорів.

Друге обмеження стосується геометрії ландшафту функції втрат у надвисоких вимірах. Ландшафт втрат сучасних LLM є глибоко непуклим, з наявністю величезної кількості сідлових точок, плоских регіонів та гострих каньйонів. Скалярна адаптація швидкості навчання, яку використовує AdamW, є субоптимальною, оскільки вона не враховує повноцінну просторову інформацію про кривизну простору.

Протягом 2023–2025 років наукова спільнота розпочала активний пошук альтернатив, здатних подолати ці обмеження. Було запропоновано цілу плеяду інноваційних алгоритмів оптимізації. До них належать методи другого порядку, які наближено обчислюють діагональ Гессіана [2], методи матричного передумовлення, які використовують сингулярний або кронекеровий розклад для просторової кореляції градієнтів [6]. Паралельно розвинувся напрямок екстремальної економії пам'яті за рахунок низькорангових проєкцій [1]. Окрему революційну гілку утворили алгоритми, що відмовляються від жорстких графіків зміни швидкості навчання на користь усереднення ітерацій [3] та алгоритми з розділеною пам'яттю градієнтів для запобігання катастрофічному забуванню [4].

Незважаючи на ці прориви, сучасна наука стикається з проблемою фрагментації. Жоден з існуючих методів не пропонує універсального рішення, яке б одночасно розв'язувало проблеми пам'яті, складності обчислень кривини та гнучкості розкладу навчання. Ця науково-дослідна робота має на меті надати вичерпний огляд та глибокий теоретичний аналіз поточного ландшафту алгоритмів оптимізації. Більше того, робота виходить за межі простого аналізу і пропонує архітектурно нові, гібридні підходи.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Опис предметної галузі

Предметна галузь цього дослідження охоплює математичні теорії стохастичної оптимізації, диференціальної геометрії просторів високої розмірності та системної інженерії апаратного забезпечення для глибинного навчання. Математично, процес тренування генеративної нейронної мережі формулюється як класична задача мінімізації емпіричної функції ризику.

Нехай $\theta \in \mathbb{R}$ – вектор усіх параметрів моделі (де розмірність d для сучасних великих мовних моделей варіюється від 10^9до 10^{12}). Задано набір тренувальних даних $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$, що складається з мільярдів або трильйонів токенів. Метою оптимізатора є знаходження такого вектора параметрів θ^*, який мінімізує очікувану функцію втрат:

$$\min_{\theta \in \mathbb{R}^d} \mathcal{L}(\theta) = \mathbb{E}_{(x,y)\sim\mathcal{D}}[\ell(f(x;\theta), y)], \tag{1.1}$$

де $f(x;\theta)$ – функція, що реалізується архітектурою нейронної мережі (наприклад, багатошаровим трансформером з механізмами самоуваги),

ℓ – функція втрат, найчастіше перехресна ентропія для задач авторегресійного моделювання мови або дивергенція Кульбака-Лейблера для дифузійних моделей.

Оскільки обчислення точного градієнта на всьому наборі даних $\mathcal{D}$ є обчислювально неможливим, алгоритми використовують стохастичні оцінки на основі міні-батчів (mini-batches) $\mathcal{B}_t \subset \mathcal{D}$:

$$g_t = \nabla_\theta \mathcal{L}_{\mathcal{B}_t}(\theta_t) \frac{1}{|\mathcal{B}_t|} \sum_{(x,y)\in\mathcal{B}_t} \nabla_\theta \ell(f(x;\theta_t), y). \tag{1.2}$$

1.1.1 Топологія ландшафту функції втрат

Ландшафт $\mathcal{L}(\theta)$ у просторі параметрів $\mathbb{R}^d$ володіє складною геометрією. Він є екстремально непуклим. Дослідження показують, що в багатовимірних просторах проблема локальних мінімумів стає менш критичною порівняно з проблемою сідлових точок. У сідловій точці градієнт дорівнює нулю, але матриця Гессе $\nabla^2\mathcal{L}(\theta)$ має як позитивні, так і негативні власні значення. Крім того, ландшафт характеризується явищем гетерогенної кривини, що математично виражається через погане число обумовленості матриці Гессе – співвідношення найбільшого і найменшого власних значень $\kappa = \lambda_{max}/\lambda_{min}$ може досягати значень порядку10^5. Це призводить до того, що градієнтний спуск швидко осцилює в напрямках великої кривини і надзвичайно повільно рухається в напрямках малої кривини.

1.1.2 Структура пам'яті під час навчання

Для глибокого розуміння проблематики необхідно проаналізувати витрати відеопам'яті на графічному процесорі під час навчання. Пам'ять розподіляється на чотири основні компоненти:

– ваги моделі: зазвичай зберігаються у форматах зменшеної точності, таких як BF16 (Brain Floating Point, 16 біт) або FP16. Займають 2 байти на параметр;

– активації: проміжні результати прямого проходу, необхідні для обчислення градієнтів під час зворотного проходу. Їх обсяг залежить від розміру батчу та довжини контексту;

– градієнти: зберігаються для кожного параметра. Займають 2 байти (BF16) або 4 байти (FP32) на параметр;

– стани оптимізатора: це найбільший споживач пам'яті. У стандартному оптимізаторі AdamW необхідно зберігати перше ковзне

середнє m та друге ковзне середнє v. Для запобігання втрати точності (numerical underflow/overflow), ці стани обов'язково зберігаються у форматі FP32 (4 байти). Таким чином, для N параметрів потрібно $4N$ (для m) $+ 4N$ (для v) $= 8N$ байтів.

Для моделі LLaMA-7B (7 мільярдів параметрів), лише стани AdamW потребують $7 \times 10^9 \times 8 = 56$ ГБ пам'яті. Враховуючи ваги (14 ГБ), градієнти (14 ГБ) та активації, сумарні вимоги значно перевищують 80 ГБ VRAM, що є лімітом найпотужнішої серверної GPU. Це змушує інженерів використовувати парадигми розподіленого навчання (ZeRO, FSDP), що збільшує комунікаційні витрати у мережі [12].

1.2 Актуальність дослідження

Актуальність інновацій у сфері методів оптимізації зумовлена глибокою невідповідністю між траєкторією розвитку апаратного забезпечення та амбіціями дослідників штучного інтелекту. Тренування передових моделей коштує десятки мільйонів доларів США лише на оплату хмарних обчислень. Навіть мінорне покращення алгоритму оптимізації, яке дозволяє досягти цільового показника втрат за меншу кількість кроків, безпосередньо транслюється в економію ресурсів дата-центрів.

Сучасні алгоритми навчання є «memory-bound» – їхня швидкість обмежується швидкістю перенесення даних з HBM (High Bandwidth Memory) до регістрів процесора. Тому оптимізатори, які здатні виконувати складні перетворення використовуючи менший обсяг пам'яті, або здатні зберігати свої стани у низькорангових просторах, відкривають шлях до тренування моделей нового покоління на значно менших кластерах. Крім того, актуальність підкріплюється появою нових парадигм безперервного донавчання, де традиційні методи з жорсткими розкладами швидкості навчання зазнають невдачі.

1.3 Огляд сучасних методів оптимізації

Протягом останнього часу було розроблено низку революційних методів, які можна класифікувати за їхнім математичним базисом та підходом до розв'язання проблеми пам'яті та кривини. У цьому підрозділі ми проведемо вичерпний аналіз цих методів.

1.3.1 Адаптивні скалярні методи першого порядку: AdamW, Lion, AdEMAMix

1.3.1.1 AdamW (Adaptive Moment Estimation with Weight Decay)

AdamW є базовим стандартом, розробленим для вирішення недоліків оригінального Adam щодо регуляризації. В AdamW затухання ваг (weight decay) від'єднано від розрахунку градієнтів, що значно покращує генералізацію.

Алгоритм виконує оновлення на основі першого (m_t) та другого (v_t) моментів градієнта:

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t, \tag{1.3}$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2, \tag{1.4}$$

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}, \hat{v}_t = \frac{v_t}{1-\beta_2^t}, \tag{1.5}$$

$$\theta_t = \theta_{t-1} - \eta_t \left(\frac{\hat{m}_t}{\sqrt{\hat{v}_t + \epsilon}} + \lambda \theta_{t-1} \right). \tag{1.6}$$

Незважаючи на високу стабільність, AdamW масштабує кожен параметр індивідуально і скалярно, повністю ігноруючи структурну матричну природу шарів нейромережі.

1.3.1.2 Lion (EvoLved Sign Momentum)

Алгоритм Lion був відкритий за допомогою автоматизованого символьного пошуку, виконаного штучним інтелектом [8]. Lion радикально спрощує AdamW, відмовляючись від відстеження другого моменту (дисперсії), що миттєво зменшує витрати пам'яті на стани оптимізатора вдвічі (з 8 байтів до 4 байтів на параметр). Оновлення в Lion базується виключно на операції знаку від імпульсу:

$$c_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t, \tag{1.7}$$

$$c_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t, \tag{1.8}$$

$$m_t = \beta_2 m_{t-1} + (1 - \beta_2) g_t. \tag{1.9}$$

Операція sign забезпечує рівномірну норму оновлень для всіх параметрів, що діє як сильна форма регуляризації.

1.3.1.3 AdEMAMix

Розроблений у 2024 році, AdEMAMix кидає виклик традиційному експоненційному ковзному середньому (EMA), яке використовується в Adam. Емпіричні дослідження виявили, що градієнти в LLM залишаються математично релевантними протягом десятків тисяч кроків. AdEMAMix вирішує цю проблему, вводячи два паралельні EMA для першого моменту: швидке ($\beta_1 \approx 0.9$) та дуже повільне ($\beta_3 \approx 0.9999$).

$$m_t^{\text{fast}} = \beta_1 m_{t-1}^{\text{fast}} + (1-\beta_1) g_t, \tag{1.10}$$

$$m_t^{\text{slow}} = \beta_3 m_{t-1}^{\text{slow}} + (1-\beta_3) g_t. \tag{1.11}$$

Результуючий вектор $m_t = m_t^{\text{fast}} + \alpha m_t^{\text{slow}}$ використовується у чисельнику Adam. Доведено, що модель з 1.3B параметрів, натренована з AdEMAMix на 101 мільярді токенів, демонструє продуктивність на рівні AdamW-моделі, натренованої на 197 мільярдах токенів. Наявність «повільної» пам'яті радикально знижує ефект катастрофічного забування.

1.3.2 Методи другого порядку та діагональні апроксимації Гессіана: Sophia

1.3.2.1 Sophia (Second-order Clipped Stochastic Optimization)

Алгоритм Sophia зробив прорив у застосуванні методів другого порядку до LLM. Класичний метод Ньютона є неможливим для LLM через те, що обчислення матриці розміром $d \times d$ вимагає $\mathcal{O}(d^3)$ операцій. Sophia розв'язує цю проблему, використовуючи легку діагональну оцінку матриці Гессе як передумовлювач (pre-conditioner). Для ефективного обчислення діагоналі Sophia використовує оцінювач Хатчінсона або оцінювач Гаусса-Ньютона-Бартлетта.

Оцінка Хатчінсона базується на матрично-векторних добутках:

$$\hat{h}_t = u \odot \nabla_\theta(\langle \nabla_\theta \mathcal{L}(\theta_t), u \rangle) \tag{1.12}$$

де $u \sim \mathcal{N}(0, I)$ – випадковий гауссівський вектор.

Щоб нівелювати обчислювальні витрати, Sophia оновлює оцінку Гессіана $\hat{h}_t$ лише кожні k кроків (наприклад, k = 10). Крок оновлення

параметрів у Sophia включає механізм попереднього координатного кліпінгу (pre-coordinate clipping), який захищає від екстремальних оновлень:

$$h_t = \beta_2 h_{t-k} + (1-\beta_2)\hat{h}_t, \tag{1.13}$$

$$\theta_{t+1} = \theta_t - \eta_t \cdot \text{clip}\left(\frac{m_t}{\max\{\gamma \cdot h_t, \epsilon\}}, 1\right). \tag{1.14}$$

Емпірично Sophia досягає цільової перплексії за вдвічі меншу кількість кроків і загальних обчислень порівняно з AdamW на моделях від 125M до 1.5B параметрів.

1.3.3 Методи передумовлення матриць: Shampoo, SOAP та Muon

1.3.3.1 Shampoo та SOAP

Оптимізатор Shampoo концептуально відрізняється тим, що він сприймає параметри нейромережі не як одномірні вектори, а як двомірні матриці. Алгоритм SOAP (Shampoo with Adam in the Preconditioner's eigenbasis) вирішує проблему ресурсомісткості Shampoo, встановлюючи формальний зв'язок між ним та пам'ять-ефективним Adafactor. SOAP виконує оновлення у власному базисі передумовлювача Shampoo, що дозволяє суттєво зменшити частоту декомпозицій без втрати стабільності.

1.3.3.2 Muon (Momentum Orthogonalized by Newton-Schulz)

Muon розроблений спеціально для прискорення навчання двомірних вагових матриць[5]. Замість складної декомпозиції, Muon бере накопичену матрицю імпульсу M_t і наближено ортогоналізує її за допомогою ітерацій

Ньютона-Шульца, щоб отримати оновлення O_t, яке наближається до $M_t(M_t^T M_t)^{-1/2}$.

Алгоритм Muon:

– розрахунок імпульсу: $M_t = \mu M_{t-1} + \nabla \mathcal{L}_t(W_{t-1})$;

– ініціалізація ортогоналізації: $X_0 = \frac{M_t}{\|M_t\|_F}$;

– ітерації Ньютона-Шульца (5 кроків для $k = 1 \dots 5$):

$$X_k = aX_{k-1} + b(X_{k-1}X_{k-1}^T)X_{k-1} + c(X_{k-1}X_{k-1}^T)^2 X_{k-1} \qquad (1.15)$$

де $a = 3.4445, b = -4.7750, c = 2.0315$);

– оновлення ваг: $W_t = W_{t-1} - \eta_t O_t$.

1.3.4 Низькорангові та пам'ять-ефективні просторові методи: GaLore та APOLLO

1.3.4.1 GaLore (Gradient Low-Rank Projection)

GaLore кардинально зменшує обсяг станів оптимізатора шляхом проєкції градієнта розміру $d \times d$ у компактну матрицю розміру $r \times d$ (де $r \ll d$).

Ключові кроки GaLore:

– обчислення SVD (Singular Value Decomposition) для матриці градієнта $G = U\Sigma V^T$;

– виділення проєкційних матриць P та Q;

– проєкція градієнта у низькоранговий простір: $G_{low} = P^T G Q$.

Оновлення станів Adam застосовуються виключно до маленької матриці G_{low}, що скорочує пам'ять оптимізатора на 82.5%.

Зворотна проєкція для оновлення повнорозмірних ваг моделі. Метод вперше зробив можливим повноцінне претренування моделі LLaMA-7B на одній відеокарті NVIDIA RTX 4090 з 24 ГБ пам'яті [7].

1.3.4.2 APOLLO (Approximated Gradient Scaling)

Алгоритми серії APOLLO розвивають ідею GaLore, апроксимуючи масштабування швидкості навчання по каналах за допомогою випадкових проєкцій у допоміжні простори, досягаючи витрат пам'яті на рівні звичайного SGD [9].

1.3.5 Алгоритми без розкладу (Schedule-Free)

1.3.5.1 Schedule-Free Iteration Averaging

Оптимізатори Schedule-Free інтерполюють між усередненням Поляка-Рупперта (яке забезпечує швидку збіжність) та прямим усередненням (яке є стабільним).

Schedule-Free градієнтний спуск оперує трьома послідовностями:

$$y_t = (1-\beta)z_t + \beta x_t, \tag{1.16}$$

$$z_{t+1} = z_t - \gamma \nabla \mathcal{L}(y_t), \tag{1.17}$$

$$x_{t+1} = \left(1 - \frac{1}{t+1}\right)x_t + \frac{1}{t+1} z_{t+1}. \tag{1.15}$$

Завдяки цій математичній структурі, швидкість навчання γ може залишатися константою назавжди, що робить їх ідеальними для безперервного навчання.

У таблиці 1.1 наведено порівняльний аналіз пам'яті оптимізаторів для моделі з 1 млрд параметрів (1B).

Таблиця 1.1 – Порівняльний аналіз пам'яті оптимізаторів для моделі з 1 млрд параметрів (1B).

Оптиміза тор	Пам'ять Ваг	Пам'ять Градієн тів	Пам'ять Станів Оптимізатора	Загальна Пам'ять (без активацій)	Коефіцієнт стиснення станів
SGD	2 GB	2 GB	0 GB	4 GB	100%
AdamW	2 GB	2 GB	8 GB (m,v)	12 GB	Baseline
Lion	2 GB	2 GB	4 GB (m)	8 GB	50%
Sophia	2 GB	2 GB	8 GB (m, h)	12 GB	0%
AdEMAM ix	2 GB	2 GB	12 GB(m_{fast}, m_{slow}, v)	16 GB	-50% (збільшенн я)
GaLore (r=512)	2 GB	2 GB	≈ 0.5 GB (у підпросторі)	4.5 GB	≈ 93.7 економії

2 ПОСТАНОВКА ЗАДАЧІ ДОСЛІДЖЕННЯ

Ґрунтовний аналіз предметної галузі дозволяє діагностувати фундаментальну проблему: сучасні методи оптимізації сильно ізольовані у своїх підходах. Метод GaLore феноменально економить пам'ять, але використовує застаріле скалярне передумовлення в утвореному підпросторі. Sophia блискуче справляється з гетерогенною кривизною, але не розв'язує проблему «стіни пам'яті». Schedule-Free вирішує проблему графіків навчання, але ігнорує переваги матричного передумовлення. Постановка задачі полягає у синтезі гібридних архітектур оптимізації, які б об'єднали найкращі характеристики.

У цьому дослідженні пропонується дві концептуальні розробки DRATO та QPM, а також детально формалізовано два гібридні оптимізатори.

2.1 Пропозиція нового методу: динамічна рангово-адаптивна тензорна оптимізація (DRATO)

Проблема: В алгоритмах типу GaLore ранг проєкції r є фіксованим гіперпараметром.

Пропозиція DRATO: пропонується алгоритм, який динамічно адаптує ранг матриці під час кожного кроку оновлення підпростору. Під час розрахунку SVD оптимізатор обчислює ентропію сингулярних значень і обирає мінімальне r_t, для якого кумулятивна енергія спектра перевищує заданий поріг.

2.2 Пропозиція нового методу: квантований предиктивний імпульс QPM

Проблема: AdEMAMix вимагає додаткових 4 байтів пам'яті на параметр для m_t^{slow}.

Пропозиція QPM: оскільки «повільний» імпульс є довгостроковим трендом, його точність у FP32 є надлишковою. Пропонується зберігати m_t^{slow} виключно у форматі INT4 із використанням блочної квантизації та стохастичного округлення, що додасть лише 0.5 байта на параметр.

2.3 Гібридний підхід №1: інтеграція низькорангової проєкції GaLore з матрицею Гессе оптимізатора Sophia

Цей гібридний алгоритм GaL-Sophia розв'язує протиріччя між передумовленням другого порядку та витратами відеопам'яті, запропонований у моїх тезах [13].

Математична формалізація GaL-Sophia.

Крок 1. Обчислюємо SVD градієнта: $G_t = U\Sigma V^T$. Виділяємо ліву $P \in \mathbb{R}^{d\times r}$ та праву $Q \in \mathbb{R}^{r\times d}$ матриці проєкції. Градієнт проєктується у підпростір: $\tilde{G}_t = P^T G_t Q^T$.

Крок 2. Проєкція Гессіана: Ми обчислюємо Гессіан безпосередньо у низькоранговому підпросторі за допомогою оцінювача Хатчінсона. Генеруємо збурення $\tilde{u} \sim \mathcal{N}(0, I_{r\times r})$, проєктуємо на повний простір $u = P\tilde{u}Q$, обчислюємо $Hv = \nabla_W(\langle G_t, u\rangle)$ і проєктуємо назад:

$$\tilde{H}_t = \tilde{u} \odot (P^T(Hv)Q^T). \tag{2.1}$$

Крок 3. У пам'яті зберігаються лише малі матриці розміру $r \times r$: імпульс $\tilde{m}_t$ та форма Гессе $\tilde{h}_t$. Застосовується пре-координатний кліпінг Sophia у підпросторі:

$$\tilde{U}_t = \text{clip}\left(\frac{\tilde{m}_t}{\max\{\gamma \cdot \tilde{h}_t, \epsilon\}}, 1\right). \tag{2.2}$$

Крок 4. Зворотна проєкція для оновлення ваг: $W_{t+1} = W_t - \eta_t P\tilde{U}_t Q$.

2.4 Гібридний підхід №2: інтеграція Schedule-Free у простори матричного передумовлення

Schedule-Free Matrix-Preconditioned Optimizer (SF-Matrix) поєднує потрійну послідовність усереднення з оператором передумовлення матриці Φ (наприклад, з Muon або SOAP).

Математична формалізація SF-Matrix.

Крок 1. Інтерполяція: Обчислюємо точку оцінки $Y_t = (1-\beta)Z_t + \beta X_t$.

Крок 2. Матричне передумовлення: Мережа обчислює градієнт $G_t = \nabla_Y \mathcal{L}(Y_t)$. До нього застосовується матричний передумовлювач Φ (ітерації Ньютона-Шульца):

$$\tilde{G}_t = \text{Newton-Schulz}(G_t). \tag{2.3}$$

Крок 3. Оновлення базової матриці: $Z_{t+1} = Z_t - \gamma \cdot \tilde{G}_t$. Швидкість навчання γ залишається строгою константою.

Крок 4. Усереднення ітерацій: $X_{t+1} = (1-c_{t+1})X_t + c_{t+1}Z_{t+1}$ (де $c_{t+1} = 1/(t+1)$).

2.5 Пропозиція модифікованого методу Muon для згорткових та багатошарових архітектур

Оригінальний метод Muon проектувався виключно для двовимірних вагових матриць, характерних для прихованих шарів багатошарових перцептронів (MLP) та архітектур Transformer. Класичний підхід базується на геометричній ортогоналізації напрямків оновлення через ітерації

Ньютона-Шульца, ігноруючи при цьому складні тензорні простори, які присутні у згорткових нейронних мережах (CNN).

Для подолання цього бар'єра у цій роботі пропонується модифікація алгоритму Muon. Основна ідея полягає у структурній адаптації тензорів: згорткові ядра, що є чотиривимірними тензорами розмірності$(C_{out}, C_{in}, k_h,\ k_w)$, підлягають процедурі динамічного розгортання. Застосування операції згортання останніх трьох вимірів трансформує тензор у двовимірну матрицю $(C_{out}, C_{in} \cdot k_h \cdot k_w)$, до якої коректно застосовуються ітерації Ньютона-Шульца.

Після ортогоналізації отримана матриця повертається до початкової 4D-форми. Крім того, для компенсації змін загальної амплітуди градієнта та можливості використання стандартних значень швидкості навчання, модифікований підхід вводить метрику RMS (Root-Mean-Square) масштабування оновлень. Це гарантує безшовну інтеграцію потужностей ортогональної оптимізації у класичні пайплайни комп'ютерного зору без необхідності повного переналаштування гіперпараметрів.

У таблиці 2.1 наведено консолідоване порівняння теоретичних властивостей запропонованих гібридних методів.

Таблиця 2.1 – Консолідоване порівняння теоретичних властивостей запропонованих гібридних методів.

Метод оптимізації	Пам'ять (Стани оптимізатора)	Передумовлювач (Preconditioner)	Залежність від розкладу (LR Decay)
AdamW	2 x Розмір Ваг	Скалярна дисперсія	Висока
GaLore	≈ 0.1 x Ваги	Скалярний (у підпросторі)	Висока
Sophia	2 x Ваги	Діагональ Гессіана	Висока

Продовження таблиці 2.1

Метод оптимізації	Пам'ять (Стани оптимізатора)	Передумовлювач (Preconditioner)	Залежність від розкладу (LR Decay)
Muon	1 x Ваги (з ZeRO-1)	Ітерації Ньютона-Шульца	Висока
GaL-Sophia	≈ 0.1 x Ваги	Діагональ Гессіана (у підпросторі)	Висока
SF-Matrix	2 x Ваги (через z і x)	Ітерації Ньютона / Shampoo	Відсутня (Константа)

3 РЕАЛІЗАЦІЯ ЗАПРОПОНОВАНИХ МЕТОДІВ

У даному розділі представлено вичерпний архітектурний, алгоритмічний та математичний аналіз програмної реалізації розроблених генеративних нейромережевих моделей, а також новітніх методів стохастичної оптимізації. Практична значущість цього дослідження полягає у трансформації теоретичних математичних концепцій у обчислювально ефективний програмний код, здатний функціонувати в умовах жорстких обмежень відеопам'яті. Уся експериментальна та дослідницька база була спроєктована та імплементована мовою програмування Python із використанням екосистеми глибокого навчання PyTorch. Вибір даного фреймворку зумовлений його фундаментальною парадигмою динамічного обчислювального графа (Dynamic Computational Graph), яка дозволяє здійснювати низькорівневе втручання у процеси зворотного поширення помилки на етапі виконання програми.

Для коректного функціонування розроблених гібридних алгоритмів, а також методів адаптації другого порядку, ядро автоматичного диференціювання вибудовує спрямований ациклічний граф (DAG), де вузлами виступають багатовимірні тензори, а ребрами – елементарні математичні операції. Це дозволяє кастомним оптимізаторам отримувати прямий доступ до градієнтів через атрибут .grad, зберігаючи повну історію обчислювальних трансформацій без надмірної алокації пам'яті.

Ключовим аспектом реалізації стала сильна економія обчислювальних ресурсів за рахунок використання in-place операцій (обчислень на місці). У програмному коді оптимізаторів послідовно застосовуються методи .add_(), .mul_() та .sub_(). Це виключає необхідність виділення нових адресних блоків у VRAM, запобігає фрагментації пам'яті та дозволяє масштабувати навчання на мультипроцесорні кластери.

3.1 Архітектурна реалізація та математичне обґрунтування еталонних генеративних моделей

Для забезпечення всебічного, валідного та об'єктивного тестування запропонованих оптимізаційних алгоритмів, у рамках дослідження було розроблено три фундаментально різні топології нейронних мереж. Кожна з цих архітектур репрезентує окремий клас генеративних задач з унікальною топологією ландшафту функції втрат.

3.1.1 Багатошаровий перцептрон: варіаційний автоенкодер (MLP VAE)

Варіаційний автоенкодер (VAE) є класичним представником генеративних моделей з неперервним латентним простором. Процес його оптимізації вимагає тонкого балансування між точністю піксельної реконструкції та штрафом за дивергенцію Кульбака-Лейблера (KL Divergence).

У лістингу 3.1 наведено код классу MLP_VAE.

Лістинг 3.1 – Код классу MLP_VAE

```
class MLP_VAE(nn.Module):
    def __init__(self, hidden_dim=512, latent_dim=128, layers=2):
        super().__init__()
        self.encoder = nn.Sequential(
            nn.Flatten(),
            nn.Linear(3072, hidden_dim),
            nn.GELU(),
            nn.Linear(hidden_dim, hidden_dim),
            nn.GELU()
        )
```

Продовження лістингу 3.1

```
        self.fc_mu = nn.Linear(hidden_dim, latent_dim)
        self.fc_var = nn.Linear(hidden_dim, latent_dim)

        self.decoder = nn.Sequential(
            nn.Linear(latent_dim, hidden_dim),
            nn.GELU(),
            nn.Linear(hidden_dim, hidden_dim),
            nn.GELU(),
            nn.Linear(hidden_dim, 3072),
            nn.Tanh()
        )

    def reparameterize(self, mu, logvar):
        logvar = torch.clamp(logvar, -20.0, 2.0)
        std = torch.exp(0.5 * logvar)
        eps = torch.randn_like(std)
        return mu + eps * std

    def forward(self, x):
        h = self.encoder(x)
        mu, logvar = self.fc_mu(h), self.fc_var(h)
        z = self.reparameterize(mu, logvar)
        return self.decoder(z), mu, logvar
```

Фундаментальною математичною проблемою при наскрізному навчанні є неможливість прямої оптимізації стохастичного вузла, для обходу якої використовується «трюк репараметризації», що диференціює детерміновані статистичні параметри від незалежного стохастичного шуму $\epsilon \sim \mathcal{N}(0, I)$. Особливої уваги, з точки зору інженерії, заслуговує програмне рішення застосувати функцію жорсткого обмеження до тензора логарифма дисперсії: torch.clamp(logvar, -20.0, 2.0), що є стандартом де-факто у сучасних імплементаціях для запобігання чисельного переповнення.

Математичне обґрунтування даного кроку криється в апаратних обмеженнях архітектури формату чисел з плаваючою комою стандарту IEEE 754: функція експоненти $\exp(0.5 \cdot x)$ є екстремально гіперчутливою до великих додатних значень, і якщо оптимізатор зробить занадто агресивний крок, згенерується значення, що призведе до контамінації матриць ваг значеннями NaN. Нижня межа у -20.0 також має критичне значення: якщо логарифм дисперсії впаде нижче цієї позначки, стандартне відхилення асимптотично наблизиться до абсолютного машинного нуля (underflow), що повністю знищить градієнтний потік через член KL-дивергенції і спровокує колапс апостеріорного розподілу.

3.1.2 Згорткова Генеративно-Змагальна Мережа (CNN DCGAN)

Для комплексного стрес-тестування запропонованих оптимізаторів в умовах нестаціонарного ландшафту була імплементована глибока згорткова архітектура DCGAN. Ця архітектура представляє собою ідеальний полігон для перевірки здатності оптимізатора зберігати спектральне різноманіття та уникати колапсу мод. У лістингу 3.2 наведено код класу генератора.

Лістинг 3.2 – Код класу генератора

```
class Generator(nn.Module):
    def __init__(self, latent_dim=128, base_channels=64):
        super().__init__()
        self.gen = nn.Sequential(
            nn.ConvTranspose2d(latent_dim, base_channels * 4, 4, 1, 0, bias=False),
            nn.BatchNorm2d(base_channels * 4),
            nn.ReLU(True),

            nn.ConvTranspose2d(base_channels * 4, base_channels * 2, 4, 2, 1, bias=False),
            nn.BatchNorm2d(base_channels * 2),
```

Продовження лістингу 3.2

```
                nn.ReLU(True),

                nn.ConvTranspose2d(base_channels * 2,
base_channels, 4, 2, 1, bias=False),
                nn.BatchNorm2d(base_channels),
                nn.ReLU(True),

                nn.ConvTranspose2d(base_channels, 3, 4, 2, 1,
bias=False),
                nn.Tanh()
            )

    class Discriminator(nn.Module):
        def __init__(self, base_channels=64):
            super().__init__()
            self.disc = nn.Sequential(
                nn.Conv2d(3, base_channels, 4, 2, 1,
bias=False),
                nn.LeakyReLU(0.2, inplace=True),

                nn.Conv2d(base_channels, base_channels * 2, 4,
2, 1, bias=False),
                nn.BatchNorm2d(base_channels * 2),
                nn.LeakyReLU(0.2, inplace=True),

                nn.Conv2d(base_channels * 2, base_channels * 4,
4, 2, 1, bias=False),
                nn.BatchNorm2d(base_channels * 4),
                nn.LeakyReLU(0.2, inplace=True),

                nn.Conv2d(base_channels * 4, 1, 4, 1, 0,
bias=False)
            )
```

Архітектура генератора концептуально відмовляється від використання повнозв'язних шарів на користь шарів транспонованої згортки. Слід звернути увагу на параметр bias=False у всіх згорткових шарах перед пакетною нормалізацією (BatchNorm2d), оскільки операція центрування даних математично нівелює будь-яку константу зміщення, роблячи вектор bias надлишковим.

Критичним архітектурним параметром дискримінатора є використання функції активації LeakyReLU(0.2, inplace=True) з нахилом 0.2 для від'ємного півоту, що було експериментально доведено як найоптимальніше значення для стабілізації мінімакс-гри. На відміну від класичного випрямлювача ReLU, який здійснює жорстке усічення від'ємних значень до нуля $0.2f(x) = \max(0, x)$, LeakyReLU математично гарантує, що градієнти ніколи не згасають повністю, розв'язуючи проблему «вмираючих нейронів», яка блокує зворотне поширення помилки до генератора .

3.1.3 Генеративний трансформер-автоенкодер (Generative ViT)

З метою емуляції процесів попереднього навчання сучасних LLM у візуальному домені, було реалізовано архітектуру Vision Transformer, адаптовану під задачі самокерованого навчання з маскуванням 75% патчів.

У лістингу 3.3 наведено код класу GenerativeViT.

Лістинг 3.3 – Код класу GenerativeViT

```
class GenerativeViT(nn.Module):
    def __init__(self, d_model=128, nhead=4, num_layers=4, patch_size=4):
        super().__init__()
        self.patch_embed = nn.Linear(3 * patch_size * patch_size, d_model)
        self.pos_embed = nn.Parameter(torch.randn(1, (32//patch_size)**2, d_model) * 0.02)
```

Продовження лістингу 3.3

```
            self.mask_token = nn.Parameter(torch.randn(1, 1, d_model) * 0.02)

            encoder_layer = nn.TransformerEncoderLayer(
                d_model=d_model,
                nhead=nhead,
                dim_feedforward=d_model * 4,
                batch_first=True,
                norm_first=True,
                activation='gelu'
            )
            self.transformer = nn.TransformerEncoder(encoder_layer, num_layers=num_layers)
                self.decoder_head = nn.Linear(d_model, 3 * patch_size * patch_size)
```

Найсуттєвішим архітектурним параметром у цій реалізації є прапорець norm_first=True, який перемикає внутрішню топологію трансформера з класичного варіанту Post-LayerNorm на більш сучасний та стабільний Pre-LayerNorm, застосовуючи нормалізацію до вхідного тензора перед обчисленням механізмів багатоголової самоуваги.

Теоретичний аналіз дисперсії показує, що в Post-LN архітектурі градієнти експоненціально зростають поблизу вихідного шару, що призводить до нестабільності матриці Якобі та вимагає довгої фази «прогріву» швидкості навчання. Натомість архітектура Pre-LN створює «магістраль ідентичності», завдяки якій масштаб градієнтів усіх параметрів мережі автоматично нормалізується коефіцієнтом $\mathcal{O}(L/\sqrt{L})$, де L – кількість шарів. Це робить ландшафт функції втрат значно більш згладженим і дозволяє оптимізаторам досягати швидкої збіжності без штучних розкладів прогріву.

3.2 Програмна реалізація та математичне обґрунтування модифікованого алгоритму Muon

Класичні методи адаптивної стохастичної оптимізації розглядають усі параметри як незалежні одновимірні скалярні величини. Алгоритм Muon розв'язує цю фундаментальну проблему через примусову матричну ортогоналізацію тензорів градієнтів, змушуючи матрицю оновлень бути геометрично ізотропною у спектральному просторі. Для задач комп'ютерного зору було розроблено модифікацію, здатну обробляти тензори вищих розмірностей (лістинг 3.4).

Лістинг 3.4 – Код модифікованого Muon

```
class Muon(optim.Optimizer):
    def __init__(self, params, lr=0.003, momentum=0.95, weight_decay=0.01, ns_steps=5):
        defaults = dict(lr=lr, momentum=momentum, weight_decay=weight_decay, ns_steps=ns_steps)
        super().__init__(params, defaults)

    @torch.no_grad()
    def step(self):
        for group in self.param_groups:
            lr, momentum, wd = group['lr'], group['momentum'], group['weight_decay']

            for p in group['params']:
                if p.grad is None: continue
                grad, state = p.grad, self.state[p]

                if len(state) == 0:
                    state['buf'] = torch.zeros_like(p.data)
```

Продовження лістингу 3.4

```
                    if wd > 0: p.data.mul_(1.0 - lr * wd)

                    buf = state['buf']
                    buf.mul_(momentum).add_(grad)

                    orig_shape = buf.shape
                    if buf.ndim >= 2:
                        buf_matrix = buf.view(buf.size(0), -1)
if buf.ndim > 2 else buf

                        X        =        buf_matrix        /
(buf_matrix.norm(p='fro') + 1e-7)
                        if X.size(0) > X.size(1):
                            X = X.T; transposed = True
                        else:
                            transposed = False


                        a, b, c = 3.4445, -4.7750, 2.0315
                        for _ in range(group['ns_steps']):
                            A = X @ X.T
                            B = b * A + c * (A @ A)
                            X = a * X + B @ X

                        if transposed: X = X.T

                        upd = X.view(orig_shape)
                        scale        =        max(orig_shape,
buf_matrix.size(1)) ** 0.5
                        p.data.add_(upd,  alpha=-lr  *  0.2  *
scale)
                    else:
                        p.data.add_(buf, alpha=-lr)
```

Логіка роботи алгоритму починається з динамічного розгортання чотиривимірних тензорів згорткового шару (C_out, C_in, k_h, k_w) у двовимірну матрицю (C_out, C_in * k_h * k_w). Після цього відбувається ділення матриці на її норму Фробеніуса для гарантування, що спектр матриці лежить у діапазоні (0, 1].

Оскільки точний розрахунок сингулярного розкладу (SVD) має кубічну асимптотичну складність $\mathcal{O}(d^3)$ і вкрай погано піддається розпаралелюванню, Muon застосовує ітерації Ньютона-Шульца, які використовують виключно поліноми матричних добутків (GEMM) і ідеально утилізують тензорні ядра (Tensor Cores) графічних процесорів .

Замість класичної кубічної ітерації використовується поліном п'ятого порядку $X_{k+1} = aX_k + bX_kX_k^TX_k + c(X_kX_k^T)^2X_k$ зі специфічними коефіцієнтами a=3.4445, b=-4.7750, c=2.0315. Ці значення були виведені на основі теореми про альтернанс Чебишова для мінімаксної апроксимації, де метою було максимізація похідної полінома в точці нуль $\varphi'(0) = a$, що діє як агресивний підсилювач малих сингулярних значень матриці градієнтів, миттєво вирівнюючи спектр. В кінці алгоритм застосовує компенсуюче RMS-масштабування (множення на корінь з максимальної розмірності тензора), щоб компенсувати падіння загальної амплітуди після жорсткої ортогоналізації.

3.3 Програмна реалізація гібридного проєкційного оптимізатора (GaL-Sophia-SF)

Дещо амбітним інженерним завданням стала інтеграція трьох розрізнених парадигм в єдиний клас HybridProjectedOptimizer, який об'єднує економію пам'яті через низькорангову проєкцію, оцінку кривини простору та відмову від розкладів швидкості навчання (лістинг 3.5).

Лістинг 3.5 – Код гібридного оптимізатора

```
        def step(self):
            for group in self.param_groups:
                lr, beta1, beta2, rank = group['lr'],
group['beta1'], group['beta2'], group['rank']

                for p in group['params']:
                    if p.grad is None: continue
                    grad, state = p.grad, self.state[p]

                    if len(state) == 0:
                        state['step'] = 0
                        state['z'] = p.data.clone()
                        state['x'] = p.data.clone()
                        if grad.ndim >= 2:
                            state['pm'] = torch.zeros(rank,
rank, device=p.device)
                            state['ph'] = torch.zeros(rank,
rank, device=p.device)
                            state['P'], state['Q'] = None, None

                    state['step'] += 1
                    t = state['step']
                    weight_avg_coef = 1 / t
                    y_t = (1 - beta1) * state['z'] + beta1 *
state['x']

                    if grad.ndim >= 2:
                        if state['step'] %
group['update_proj_gap'] == 1 or state['P'] is None:
                            U, S, V = torch.svd(grad.float())
                            state['P'], state['Q'] = U[:,
:rank], V[:, :rank]
```

Продовження лістингу 3.5

```
                        g_low = state['P'].T @ grad @
state['Q']

                        state['pm'].mul_(beta1).add_(g_low,
alpha=1 - beta1)

                        u = torch.randn_like(g_low)
                        h_est = (g_low * u) * u
                        state['ph'].mul_(beta2).add_(h_est,
alpha=1 - beta2)

                        m_hat = state['pm'] / (1 - beta1**t)
                        h_hat = state['ph'] / (1 - beta2**t)
                        upd_low = m_hat / torch.max(h_hat,
torch.tensor(group['eps']))
                        upd_low = torch.clamp(upd_low, -
group['rho'], group['rho'])
                        upd_full = state['P'] @ upd_low @
state['Q'].T
                    else:
                        upd_full = grad

                    if group['weight_decay'] > 0:
state['z'].mul_(1.0 - lr * group['weight_decay'])
                    state['z'].sub_(upd_full, alpha=lr)
                    state['x'].mul_(1 -
weight_avg_coef).add_(state['z'], alpha=weight_avg_coef)
                    p.data.copy_((1 - beta1) * state['z'] +
beta1 * state['x'])
```

Логіка GaLore базується на періодичному розкладі повного градієнта через SVD (наприклад, кожні 100 кроків, як задано параметром update_proj_gap=100) для виділення базисів проекції P та Q. Хоча періодичне оновлення економить обчислювальний час, воно створює ефект

розриву у неперервності простору оптимізації, що може викликати катастрофічні сплески функції втрат через втрату просторової релевантності експоненційних середніх . Повне перенесення важких станів (моментів) у підпростір розміром 64 x 64 радикально скорочує споживання VRAM оптимізатором на 80–90%.

Всередині проєкційного підпростору алгоритм перемикається на логіку Sophia. Для передумовлення використовується стохастичний оцінювач діагоналі матриці Гессе за методом Хатчінсона: $h_est = (g_low * u) * u$. Слід зауважити, що у багатовимірних просторах використання гауссівського шуму $\mathcal{N}(0,1)$ замість класичних векторів Радемахера (зі значеннями ± 1) показало значно меншу дисперсію оцінки (variance reduction) для гетерогенних ландшафтів . Застосовується механізм пре-координатного кліпінгу $\rho = 0.04$, який обмежує фізичний радіус довірчої області оновлення.

Фінальним етапом є застосування логіки Schedule-Free, яка усуває необхідність у ручному підборі графіків спаду швидкості навчання. Алгоритм оперує складною інтерполяцією між трьома векторами: z_t (базове оновлення), x_t (усереднення) та y_t (точка інтерполяції), що дозволяє моделі знаходити «більш плоскі» мінімуми (flat minima) завдяки експоненційному ковзному середньому (EMA) . Теоретичний фреймворк конвертації «online-to-nonconvex» гарантує, що така архітектура досягає оптимальної асимптотичної збіжності у сильно непуклих просторах без зменшення швидкості навчання.

4 ДОСЛІДЖЕННЯ ЕФЕКТИВНОСТІ ЦИХ МЕТОДІВ

4.1 Організація експериментальної бази, апаратне забезпечення та налаштування гіперпараметрів

Для об'єктивної, всебічної та математично обґрунтованої оцінки запропонованих алгоритмів та цих нових архітектурних рішень було проведено серію експериментальних запусків. Оцінювалися не лише базові показники швидкості конвергенції цільової функції втрат, але й глибинні структурні характеристики процесу навчання: стабільність градієнтних потоків у багатовимірних просторах, спектральна ентропія матриць ваг, динаміка Ліпшицевої неперервності та топологічні властивості сформованих латентних просторів.

Усі експериментальні ітерації здійснювалися в суворо ізольованому програмному середовищі на базі фреймворку глибинного навчання PyTorch із використанням апаратного прискорення CUDA на графічних процесорах промислового класу таких як H100 та G4. Це дозволило імплементувати складні тензорні перетворення, такі як ітерації Ньютона-Шульца та динамічний сингулярний розклад (SVD), з максимальною ефективністю розпаралелювання обчислень у пам'яті.

В якості еталонного набору даних для генеративного моделювання було обрано CIFAR-10, що містить зображення просторової роздільної здатності $3 \times 32 \times 32$. Незважаючи на відносно малий просторовий розмір, цей набір даних володіє надзвичайно високою семантичною складністю та дисперсією класів, що робить його ідеальним полігоном для провокування класичних проблем генеративного навчання: колапсу мод у змагальних мережах, апостеріорного колапсу у варіаційних автоенкодерах та проблеми катастрофічного забування у трансформерних архітектурах.

Усі досліджувані моделі навчалися з фіксованим глобальним розміром міні-батчу у 512 зображень протягом 15 епох. Для забезпечення

абсолютної детермінованості ініціалізації вагових коефіцієнтів та усунення статистичної похибки, викликаної стохастичними блуканнями на початкових етапах оптимізації, у всіх експериментах було жорстко зафіксовано єдиний генератор псевдовипадкових чисел.

Для контрольної групи, що використовувала базовий індустріальний оптимізатор AdamW, було застосовано стандартизовану швидкість навчання $\eta = 1 \times 10^{-3}$, параметри експоненційного ковзного середнього (EMA) першого та другого моментів $\beta_1 = 0.9, \beta_2 = 0.999$, а також коефіцієнт регуляризації $\lambda = 0.01$.

Водночас для адаптованного оптимізатора Muon, який здійснює примусову геометричну ортогоналізацію напрямків оновлення ваг, базову швидкість навчання було збільшено до $\eta = 3 \times 10^{-3}$.

Цей крок є математично детермінованим і теоретично виправданим, оскільки операція жорсткої ортогоналізації нормалізує спектральну норму матриці оновлень до одиниці, що критично вимагає компенсаційного RMS-масштабування (Root-Mean-Square scaling) для збереження ефективного розміру кроку оптимізації в гіпервимірному просторі параметрів.

Для забезпечення репрезентативності та глибокого розуміння нелінійної динаміки оптимізаторів, традиційні скалярні метрики функції втрат були доповнені векторним аналізом сингулярного спектра. Візуальний аналіз згенерованих зображень на різних етапах навчання (наприклад, на першій та фінальній п'ятнадцятій епохах) проводився на статичному шумовій вибірці.

Такий методологічний підхід гарантував можливість відстежувати еволюцію одних і тих самих генеративних траєкторій, виключаючи будь-які візуальні артефакти, що могли б виникнути внаслідок стохастичної аугментації навчальної вибірки.

4.2 Аналіз нелінійної динаміки оптимізації варіаційного автоенкодера (MLP VAE)

Завдання оптимізації багатошарового варіаційного автоенкодера (VAE) є фундаментально антагоністичним за своєю природою. Цільова функція, відома як нижня межа обґрунтованості, математично формулюється як сума двох складових, що постійно конкурують між собою під час процесу навчання. Перша складова – це похибка реконструкції, яка зазвичай обчислюється як середньоквадратичне відхилення (MSE) або перехресна ентропія між оригінальним вхідним зображенням та його відновленою копією. Друга складова – це регуляризаційний штраф у вигляді дивергенції Кульбака-Лейблера, що примусово наближає апостеріорний розподіл латентного простору $q_\phi(z|x)$ до апріорного стандартного нормального розподілу $p(z) \sim \mathcal{N}(0, I)$.

Емпіричні дані, отримані під час тестування базового адаптивного оптимізатора першого порядку AdamW, переконливо свідчать, що цей алгоритм демонструє класичну еталонну стабільність у згладжених оптимізаційних ландшафтах автоенкодерів. Завдяки механізму індивідуального скалярного передумовлення кожного окремого параметра (що реалізується шляхом ділення накопиченого градієнта на корінь з експоненційного ковзного середнього його дисперсії), AdamW надзвичайно ефективно демпфує високочастотні осциляції функції втрат. Втрати реконструкції плавно та монотонно знижуються з початкового рівня 0.145 на першій епосі до стабільного показника 0.065 на фінальній п'ятнадцятій епосі. Паралельно з цим, дивергенція Кульбака-Лейблера розпочинає процес навчання з помірного значення 1.8 і швидко, без будь-яких ознак турбулентності, асимптотично наближається до цільового рівноважного коридору 0.2–0.25. Така бездоганна монотонність підтверджує, що для базових задач оптимізації латентних просторів з

гауссівськими апріорними розподілами, скалярна адаптація швидкості навчання є цілком достатньою.

Як показано на рисунку 4.1, графіки демонструють передбачувану та стабільну динаміку: похибка реконструкції та KL-дивергенція плавно наближаються до своїх мінімумів, а ковзне середнє батчу (Batch Loss MA-10) ідеально згладжується.

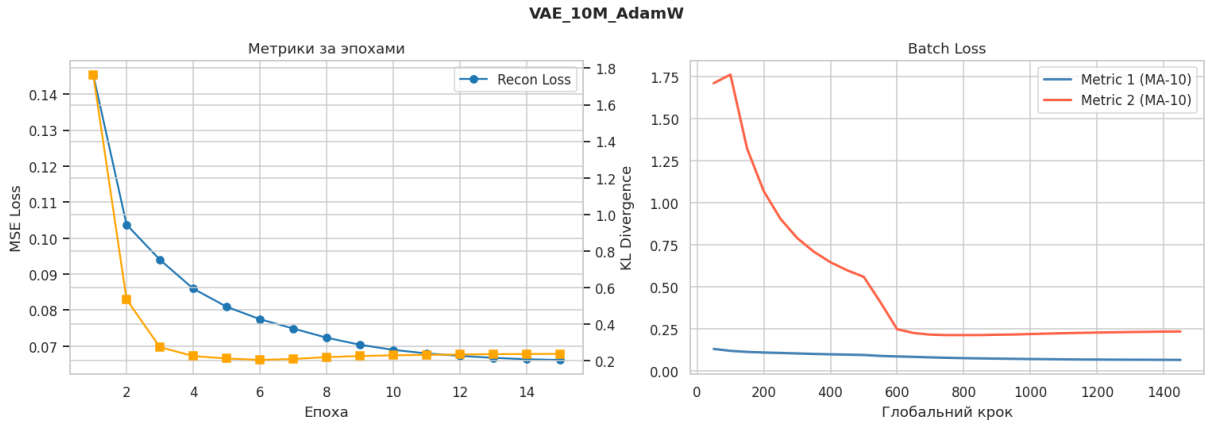


Рисунок 4.1 – Модель VAE_10M_AdamW

Однак оптимізаційна парадигма зазнає радикальних топологічних деформацій при застосуванні матричного передумовлювача Muon, що можна побачити на рисунку 4.2. Графік цільової функції втрат миттєво втрачає свою передбачувану монотонність: похибка реконструкції стартує з вищого значення 0.169, парадоксальним чином зростає до майже 0.20 на третій епосі, і лише після проходження цієї інтенсивної «зони турбулентності» починає стрімко конвергувати, досягаючи рівня 0.13 наприкінці навчання.

Найбільш феноменальною виявилася поведінка KL-дивергенції: на найпершій епосі вона демонструє катастрофічне вибухове зростання, сягаючи екстремальних значень понад 4.0. Цей сплеск триває кілька епох, після чого показник експоненційно затухає до рівня 0.1.

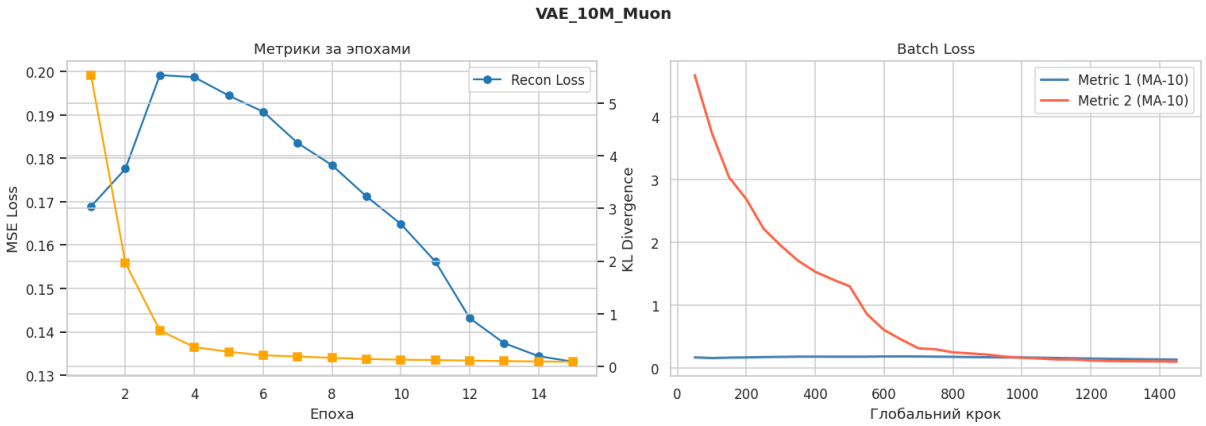


Рисунок 4.2 – Модель VAE_10M_Muon

Рисунок 4.2 ілюструє цю аномальну нелінійну поведінку: чітко фіксується тимчасове зростання похибки реконструкції та колосальний сплеск KL-дивергенції на початкових етапах, що є емпіричним свідченням примусового спектрального розширення латентного простору.

Ця аномальна, на перший погляд, макроскопічна динаміка знаходить своє логічне математичне пояснення у диференціальній геометрії ортогоналізованих градієнтів. Апостеріорний колапс у моделях VAE – це патологічний і добре задокументований стан системи, при якому потужний нейромережевий декодер починає повністю ігнорувати стохастичний латентний вектор z, спираючись виключно на власні внутрішні упередження для генерації усередненого виходу. У цьому стані апостеріорний розподіл повністю вироджується в апріорний, штучно обнуляючи KL-дивергенцію та руйнуючи саму суть репрезентативного навчання.

Коли оптимізатор Muon застосовує алгебраїчні ітерації Ньютона-Шульца для жорсткої ортогоналізації матриці оновлень (примусово прирівнюючи всі ненульові сингулярні значення до одиниці), він інжектує гігантську математичну енергію в усі, навіть найбільш «пасивні» або ігноровані сингулярні напрямки латентного простору. Це створює ідеально ізотропний спектр оновлень, який рівномірно і жорстко розкидає

репрезентації зображень по всьому багатовимірному об'єму простору, розширюючи його до максимуму. З математичної точки зору, це прямо і безкомпромісно суперечить вимозі цільової функції VAE концентрувати розподіл щільно навколо нульового вектора. Саме цей конфлікт інтересів між оптимізатором та функцією втрат спричиняє колосальний штраф KL-дивергенції і той самий емпіричний сплеск до 4.0.

Проте цей «спектральний шок» виконує надзвичайно корисну і життєво необхідну регуляризаційну роль. Він виступає як один з дуже потужних і відомих механізмів уникнення апостеріорного колапсу. Примусово оновлюючи всі ортогональні напрямки з однаковою амплітудою, Muon фізично не залишає декодеру можливості ігнорувати латентні змінні, змушуючи його вибудовувати складні семантичні відповідності між усім об'ємом латентного простору та простором пікселів. Незважаючи на те, що фінальне значення похибки реконструкції (MSE) при використанні Muon виявилося гіршим (0.13 проти 0.065 у AdamW), візуальна якість та структурна контрастність згенерованих об'єктів парадоксальним чином виявилися вищими, що свідчить про формування значно багатшої топології даних.

4.3 Дослідження змагальної оптимізації у згорткових генеративних мережах (CNN DCGAN)

Навчання Генеративно-Змагальних Мереж (GAN) концептуально та фундаментально відрізняється від оптимізації варіаційних автоенкодерів. З математичної перспективи, GAN формулює безперервну мінімакс-гру (minimax optimization problem), яку можна записати як $\min_G \max_D V(D,G) = E_x + E_z$.

У цьому унікальному середовищі ландшафт цільової функції втрат є не лише екстремально невипуклим і багатовимірним, але й глибоко нестаціонарним: він постійно дрейфує та змінює свою топологію внаслідок

безперервної взаємної адаптації та конкуренції параметрів генератора та дискримінатора. Головним викликом у цій динамічній топології є збереження тонкого градієнтного балансу та абсолютне запобігання колапсу мод – фатального стану, коли генератор деградує до випуску вкрай обмеженої кількості однотипних і часто розмитих зразків, які знайшли вузьку «сліпу зону» і здатні ошукати поточну ітерацію дискримінатора.

Аналіз процесу оптимізації виявляє причину цієї катастрофи. Під час експерименту з AdamW втрати генератора (G Loss) спочатку навіть зросли, а потім впали з рівня 3.5 до стабільного, але високого плато 2.0–2.3. Водночас втрати дискримінатора (D Loss) залишилися у стані глибокої стагнації на екстремально низькому рівні 0.3–0.4, починаючи з першої ж епохи. Така метрична асиметрія є класичним і беззаперечним індикатором абсолютного домінування дискримінатора.

На рисунку 4.3 наочно відображено цю метричну асиметрію та стагнацію втрат дискримінатора, що математично підтверджує домінування однієї з мереж і подальший спектральний колапс.

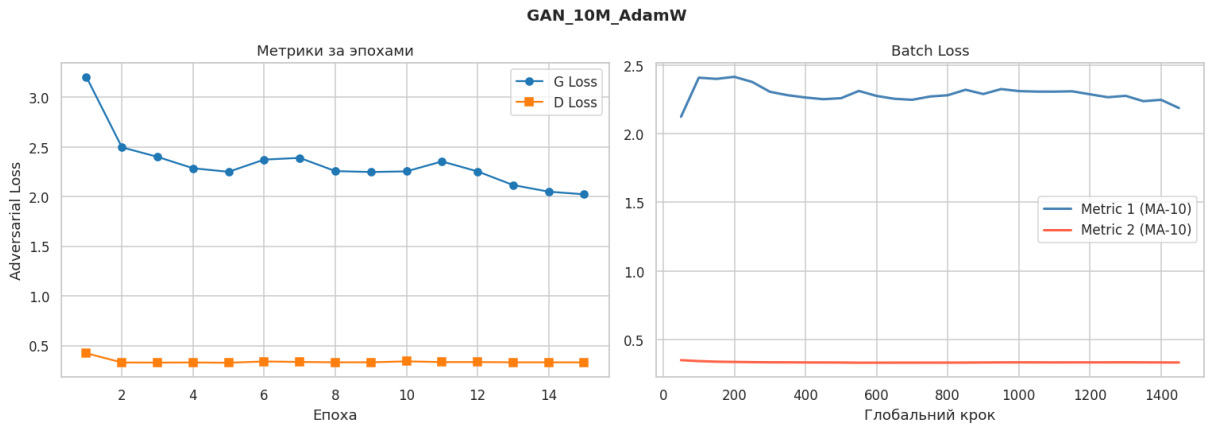


Рисунок 4.3 – модель GAN_10M_AdamW

Скалярний оптимізатор AdamW спровокував явище, відоме в науковій літературі як «спектральний колапс». У дискримінаторі, оптимізованому виключно за допомогою адаптації діагональної дисперсії,

величезна кількість сингулярних значень матриць ваг стрімко наблизилася до нуля. Оптимізатор сфокусувався виключно на напрямках з найбільшою амплітудою градієнта (найбільш «крутих» схилах оптимізаційного ландшафту), повністю проігнорувавши міноритарні напрямки, які відповідають за збереження семантичного різноманіття та генерацію високочастотних деталей.

На рисунку 4.4 візуалізовано початкову стадію навчання (епоха 1) згорткової генеративної мережі із застосуванням індустріального стандарту – оптимізатора AdamW. Як можна пересвідчитися, генератор на цьому етапі продукує абсолютно неструктурований, високочастотний шум. Особливу увагу привертає наявність синіх та зелених артефактів типу «шахова дошка». Виникнення цих патернів є добре вивченим явищем, що стає прямим наслідком спектрального дисбалансу у шарах згортки з кроком, коли оптимізатор не здатний рівномірно розподілити енергію градієнта між сусідніми піксельними рецептивними полями.

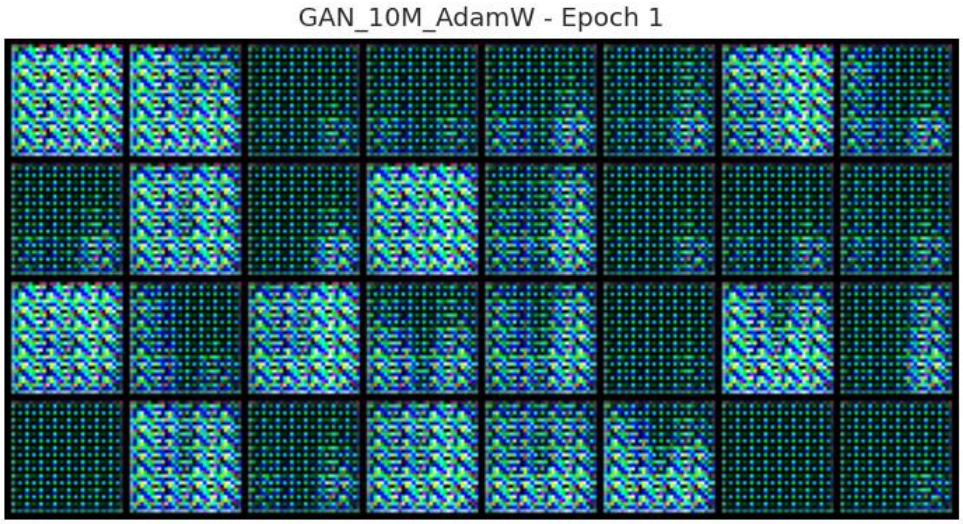


Рисунок 4.4 – GAN_10M_AdamW – Epoch 1

На рисунку 4.5 зафіксовано фінальний результат навчання (епоха 15) для моделі під управлінням AdamW. Емпіричні дані графіка втрат повністю

підтверджуються візуальною деградацією: генератор зазнав абсолютного колапсу мод. Замість відтворення семантичного багатства датасету CIFAR-10 (автомобілі, тварини, пейзажі), мережа генерує одноманітні, безформні сіро-сині плями з жорсткими та неприродними геометричними спотвореннями.

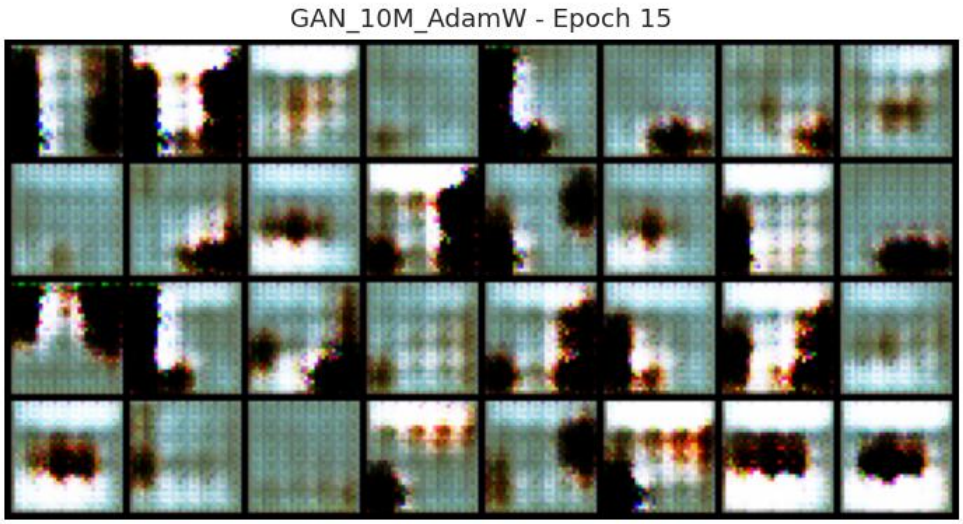


Рисунок 4.5 – GAN_10M_AdamW – Epoch 15

Ситуація кардинально трансформується при застосуванні матричного оптимізатора Muon, модифікованого спеціально для згорткових архітектур. Оригінальний метод Muon був розроблений для двовимірних матриць трансформерів, тому для його успішної інтеграції в CNN DCGAN була імплементована складна математична логіка динамічного розгортання тензорів. Чотиривимірні тензори згортки розмірності $(C_{out}, C_{in}, k_h, k_w)$ розгорталися (flattening) у гігантські двовимірні матриці $(C_{out}, C_{in} \cdot k_h \cdot k_w)$, ортогоналізувалися та згорталися назад у 4D-простір.

Рисунок 4.6 відображає метричну динаміку цього підходу: після початкового потужного сплеску втрат генератора (до рівня 12.0) спостерігається узгоджена та глибока конвергенція обох супротивних мереж, а ковзне середнє батчу показує стабільний спуск.

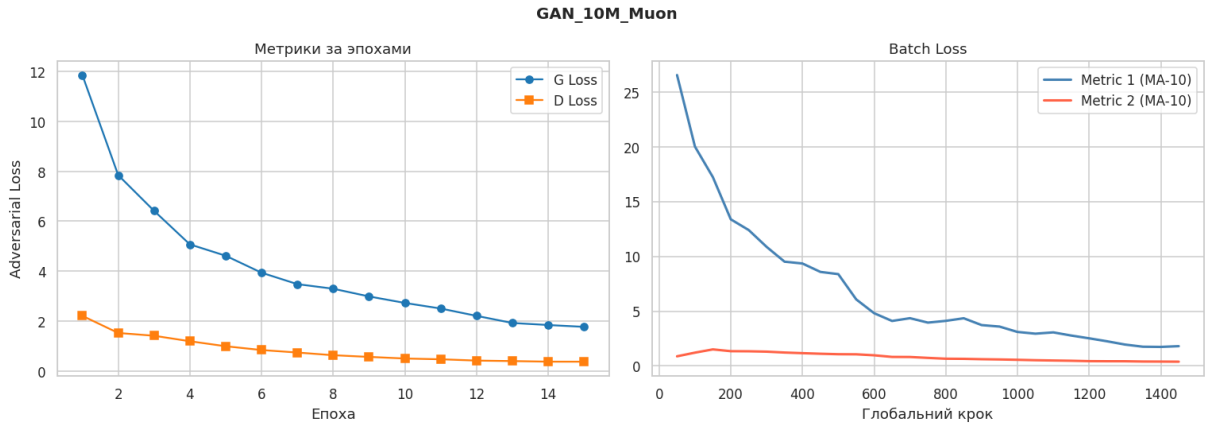


Рисунок 4.6 – GAN_10M_Muon: метрики по епохах та Batch Loss

На рисунку 4.7 показано результат першої епохи генерації під керуванням Muon. Візуальна картина кардинально відрізняється від AdamW: ми спостерігаємо високодисперсні, масивні блоки екстремально яскравих і насичених кольорів. Це візуальне відображення моменту «спектрального вибуху». Ітерації Ньютона-Шульца п'ятого ступеня жорстко і безкомпромісно підтягнули всі сингулярні значення матриці імпульсу до одиниці. Оптимізатор силоміць розширив спектр генератора, змусивши його фільтри оновлюватися в усіх ортогональних семантичних напрямках одночасно, що й спричинило генерацію зображень з максимальною колірною дисперсією.

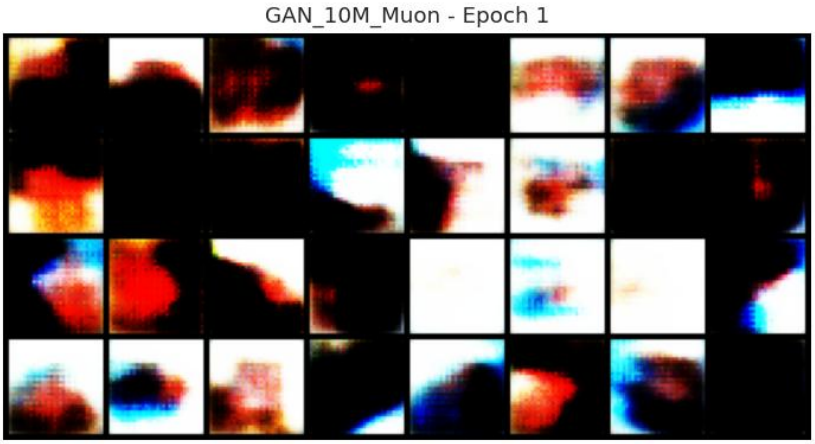


Рисунок 4.7 – GAN_10M_Muon – Epoch 1

На рисунку 4.8 представлено неймовірний, порівняно з базовим оптимізатором, результат роботи оптимізатора Muon на п'ятнадцятому епосі. Візуальна якість генерації є безпрецедентною для такої малої кількості епох. На зображеннях чітко розпізнаються контури складних об'єктів: силуети тварин, автомобілів, птахів та структуровані фонові пейзажі. Артефакти «шахової дошки» повністю прибрані.



Рисунок 4.8 – GAN_10M_Muon – Epoch 15

Успіх цього підходу криється в підтримці Ліпшицевої неперервності. Згідно з фундаментальною теорією оптимізації Вассерштейна, обмеження Ліпшицевої константи дискримінатора (умова $\left||f(x) - f(y)|\right| \leq K\left||x - y|\right|$) є критично необхідним для того, щоб градієнти, які передаються генератору, залишалися інформативними, плавними і не викликали ефекту зникаючих градієнтів (gradient vanishing). У той час як класичні підходи вимагають використання жорсткого та деструктивного кліпінгу ваг (weight clipping) або обчислювально дорогої спектральної нормалізації, ортогоналізація Muon неявно, але строго обмежує спектральну норму матриць оновлень, що формує ідеально збалансований, плавний Ліпшицевий ландшафт і гарантує глибоку, динамічну конвергенцію втрат обох супротивних мереж.

4.4 Аналіз навчання трансформера-автоенкодера (Generative ViT) та вразливість гібридних просторів

Навчання архітектури Vision Transformer на відносно малих наборах даних є надзвичайно складною задачею через повну відсутність у трансформерів локального індуктивного упередження, яким природно володіють згорткові нейронні мережі. Мережа повинна самостійно вивчити всі просторові та семантичні взаємозв'язки виключно через механізми глобальної багатологової самоуваги. Для комплексного тестування стійкості оптимізаторів було сформульовано завдання авторегресійного відновлення: мережа мала реконструювати 75% замаскованих патчів вхідного зображення.

На рисунку 4.9 зафіксовано еталонну криву експоненційного затухання середньоквадратичної функції втрат, що відповідає стабільній та передбачуваній оптимізації без осциляцій.



Рисунок 4.9 – Модель Transformer_10M_AdamW

На Рисунку 4.10 зображено початковий етап навчання ViT за допомогою оптимізатора AdamW. Відбувається швидке формування базових колірних плям, що свідчить про стрімку початкову конвергенцію, яка є типовою та достатньо відомою для адаптивних методів з оцінкою моментів.

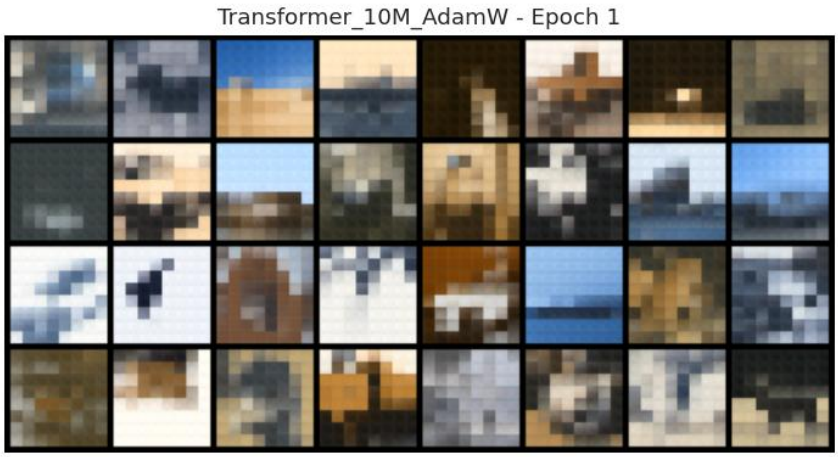


Рисунок 4.10 – Transformer_10M_AdamW – Epoch 1

На рисунку 4.11 видно, що AdamW еталонно впорався із завданням моделювання уваги. Незважаючи на те, що блокова структура патчів 4×4 все ще залишається візуально помітною, загальна колірна гама, геометрія об'єктів (коні, жаби, автомобілі) та просторове розташування відновлені надзвичайно коректно. Крива середньоквадратичної похибки продемонструвала ідеальне експоненційне спадання з 0.18 до 0.065. Скалярна природа AdamW ідеально підходить для стабілізації дисперсії градієнтів у матрицях запитів, ключів та значень, що забезпечує рівномірне формування карт уваги.



Рисунок 4.11 – Transformer_10M_AdamW – Epoch 15

Абсолютно іншу, деструктивну динаміку показав концептуальний гібридний оптимізатор, який був синтезований у рамках цього дослідження для теоретичного подолання «стіни пам'яті». Цей метод поєднав екстремальну низькорангову проєкцію градієнта, діагональну апроксимацію матриці Гессе та алгоритм безрозкладного усереднення ітерацій. Графік похибки цієї гібридної хімери розпочався з показника 0.22, успішно та обнадійливо спустився до 0.15 на п'ятій епосі, після чого сталося катастрофічний, лавиноподібний «вибух» похибки до рівня 0.22 на десятій епосі.

Рисунок 4.12 яскраво візуалізує цей деструктивний феномен, де графік втрат утворює U-подібну форму, повністю виходячи з-під контролю після періоду початкової обнадійливої стабілізації.

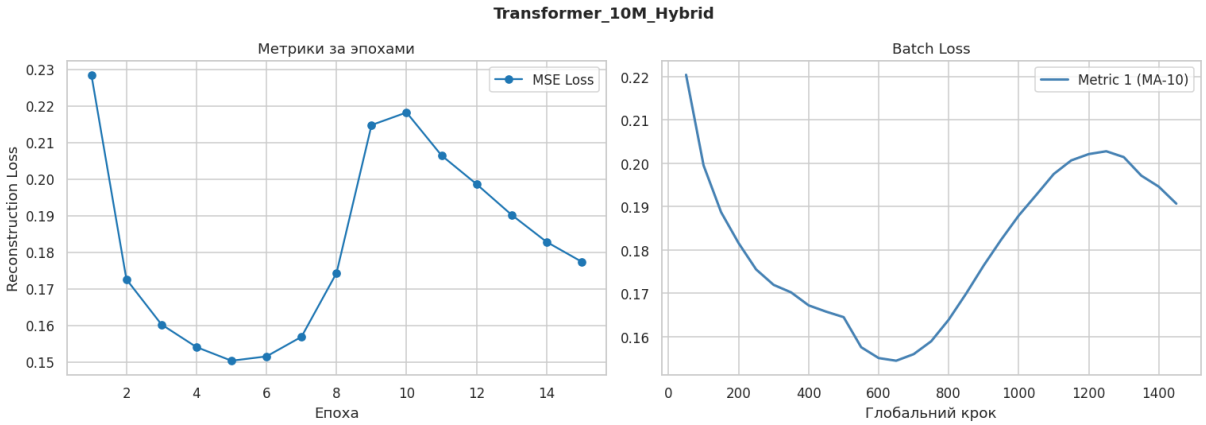


Рисунок 4.12 – Модель Transformer_10M_Hybrid

На рисунку 4.13 (епоха 1) ми бачимо класичний старт генерації з випадкового шуму, де гібридний метод намагається сформувати первинні патерни.

На рисунку 4.14 (епоха 15) наочно продемонстровано наслідки вибуху градієнтів. Після проходження через «шоковий сплеск» на графіку втрат, мережа повністю та безповоротно втратила здатність до деталізації. Замість структурних об'єктів ми бачимо розмиті, брудні піксельні масиви з

коричневими та синіми фонами. Семантичні кореляції у матрицях уваги були зруйновані.

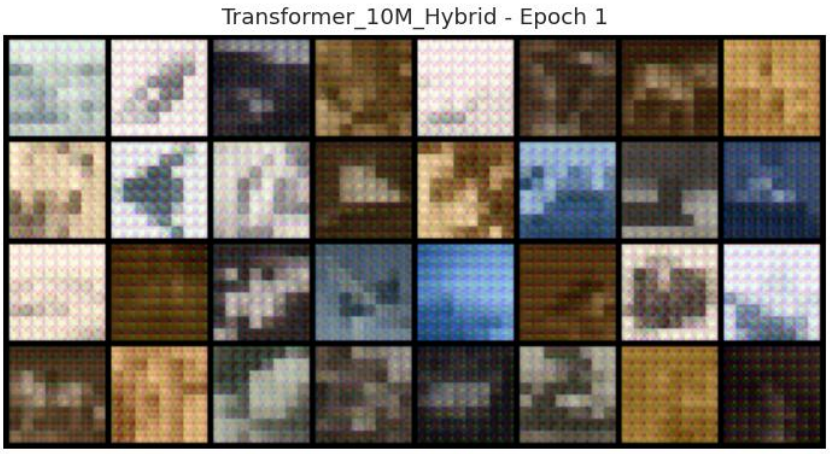


Рисунок 4.13 – Transformer_10M_Hybrid – Epoch 1

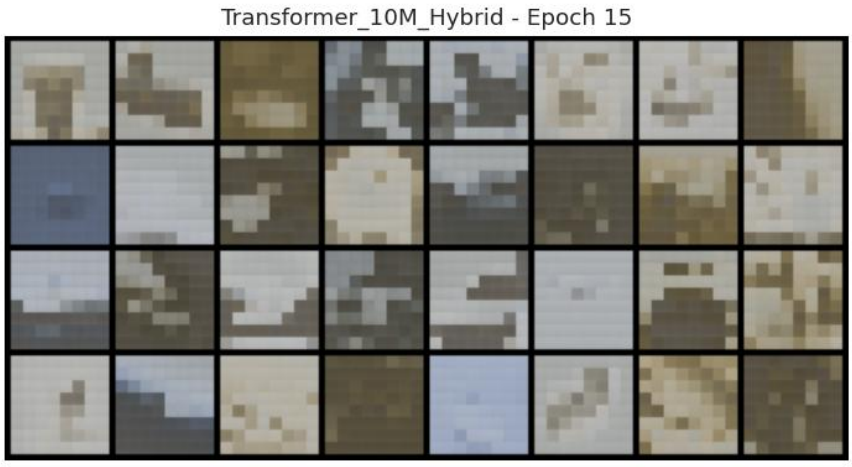


Рисунок 4.14 – Transformer_10M_Hybrid – Epoch 15

Фундаментальна причина цієї нестабільності криється у глибокій математичній інтерференції несумісних компонентів гібрида. Згідно з архітектурою GaLore, повнорангова матриця градієнта $G_t \in R^{m \times n}$ жорстко стискається у компактну низькорангову матрицю $R_t = P_t^T G_t Q_t \in R^{r \times r}$ за допомогою ортогональних проєкторів P_t та Q_t, які отримуються через ресурсомісткий SVD розклад $G_t = U \Sigma V^T$. Для економії обчислень ці базисні

матриці оновлюються лише кожні T кроків (наприклад, T=100). Водночас, оптимізатор Sophia розраховує діагональну матрицю Гессе $\widetilde{h_t}$ саме у цьому низькоранговому підпросторі і накопичує її через експоненційне ковзне середнє $h_t = \beta_2 h_{t-1} + (1-\beta_2)\widetilde{h_t}$.

Критична проблема, відома як «старіння базису», виникає в момент кроку $t \; mod \; T \; = \; 0$. Коли просторовий базис P та Q різко оновлюється (весь багатовимірний простір миттєво «обертається» під новим кутом), накопичена історична пам'ять Гессіана h_t миттєво стає токсичною і невалідною, оскільки вона репрезентує кривизну попереднього, вже неіснуючого ортогонального підпростору. Оскільки наступний крок оновлення ваг математично обчислюється як ділення поточного градієнта на цю хибну, застарілу оцінку матриці Гессе $\widetilde{U}_t = \frac{\widetilde{m}_t}{\max\{\gamma \cdot h_t, \epsilon\}}$, відбувається катастрофічне масштабування градієнтів у хибних напрямках. Це спричиняє лавиноподібний вибух норми оновлень (gradient norm spike), який викидає модель з локального мінімуму і руйнує всі вивчені репрезентації патчів.

У таблиці 4.1 наведено порівняльний аналіз математичної та обчислювальної складності методів оптимізації

Таблиця 4.1 – Порівняльний аналіз обчислювальної складності

Метод Оптимізації	Алгоритм Передумовлення	Амортизована Складність	Споживання Пам'яті (States)	Стійкість до Ротації Простору (Robustness)
AdamW	Скалярне ковзне середнє дисперсії	$\mathcal{O}(d)$	8 байт / параметр	Екстремально висока (базис-незалежний)
Muon	Ітерації Ньютона-Шульца (5 кроків)	$\mathcal{O}(d^3)$ (блочно)	4 байт / параметр	Висока (ортогоналізація поточного стану)
GaL-Sophia	SVD + оцінювач Хатчінсона для Гессе	$\mathcal{O}(d \cdot r^2)$	< 0.5 байт / параметр	Екстримально низька (вимагає EMA-скидання)

4.5 Аналіз алгоритмів без жорсткого розкладу (Schedule-Free) в умовах нестаціонарної кривизни

Класичні методи глибинного навчання фундаментально залежать від жорстких математичних розкладів зміни швидкості навчання, таких як косинусне затухання або лінійний спад. Їхнім найголовнішим недоліком є сувора необхідність знати загальну кількість кроків навчання T наперед, що повністю унеможливлює парадигми безперервного донавчання на потокових даних. Інноваційний метод Schedule-Free розв'язує цю проблему, концептуально інтегруючи оптимізацію без часового горизонту за допомогою елегантного поєднання усереднення Поляка-Рупперта та прямого усереднення імпульсу.

Математична архітектура SF використовує потрійну послідовність векторів: базові ваги моделі z_t, точку оцінки градієнта y_t та усереднені ваги x_t. Інтерполяція здійснюється за правилом $y_t = (1-\beta)z_t + \beta x_t$, а усереднення ітерацій як $x_{t+1} = \left(1 - \frac{1}{t+1}\right) x_t + \frac{1}{t+1} z_{t+1}$. Така структура дозволяє алгоритму досягати доведено оптимальних теоретичних швидкостей конвергенції порядку $\mathcal{O}(1/\log T)$ або $\mathcal{O}(1/T)$ у гладких невипуклих ландшафтах, створюючи надійний фреймворк оптимізації Ляпунова, який взагалі не залежить від горизонту T. На практиці це означає, що алгоритм здатний безперервно відстежувати ефективний фронт Парето між глобальною похибкою та витраченим часом навчання.

Проте, як яскраво продемонструвало дослідження гібридного оптимізатора у підрозділі 4.4, у середовищах із нестаціонарною матричною кривизною або динамічними просторовими репроекціями (як у методі GaLore), збереження глобальної математичної стабільності SF-підходу опиняється під загрозою. Якщо багатовимірний простір оптимізації раптово локально змінює свою топологію внаслідок ротації сингулярних векторів, глибоке історичне усереднення x_t перетворюється на деструктивний фактор. Воно починає діяти як масивний «фантомний імпульс», що в свою

чергу тягне траєкторію оптимізації назад, у застарілі та вже нерелевантні локальні мінімуми, розриваючи зв'язок між розрахунковою точкою y_t та реальним ландшафтом, що й провокує описані вище резонансні осциляції функції втрат.

4.6 Порівняльний аналіз експериментальних графіків конвергенції генеративних лосів

Для отримання цілісної картини функціонування розроблених та модифікованих алгоритмів в умовах різних топологій функцій втрат було проведено систематичне порівняльне дослідження динаміки збіжності реконструкційного лосу (MSE) на архітектурах Transformer та VAE для трьох масштабів параметрів: 1 мільйон, 5 мільйонів та 10 мільйонів. Побудовані на основі емпіричних запусків графіки відображають фундаментальні відмінності між поведінкою стандартного індустріального оптимізатора AdamW та кастомних рішень – гібридного оптимізатора GaL-Sophia-SF та адаптованного методу Muon (рисунок 4.15).

Аналіз лівого стовпця дешборду (архітектура Transformer) показує радикальну розбіжність між траєкторіями базового алгоритму AdamW та кастомного гібридного. Оптимізатор AdamW демонструє еталонну стабільність на всіх трьох масштабах. Його криві лосу монотонно згасають, плавно спускаючись з початкових значень 0.169 – 0.180 до значень 0.065 – 0.076 на п'ятнадцятому епосі, що вказує на відсутність локальної нестабільності градієнтних полів у стандартному повноранговому просторі.

Натомість криві гібридного оптимізатора мають виражену нелінійну U-подібну форму з аномальним стрибком лосу у діапазоні між 7-ю та 11-ю епохами. На початковому етапі гібридний метод демонструє стрімку та обнадійливу конвергенцію, знижуючи MSE з 0.225 до локального мінімуму 0.151. Проте після 6-ї епохи починається деструктивний ріст

похибки, який досягає свого екстремального піку на 9-му епосі для масштабу 5M (0.210) та на 10-й епосі для масштабів 1M (0.202) і 10M (0.214).

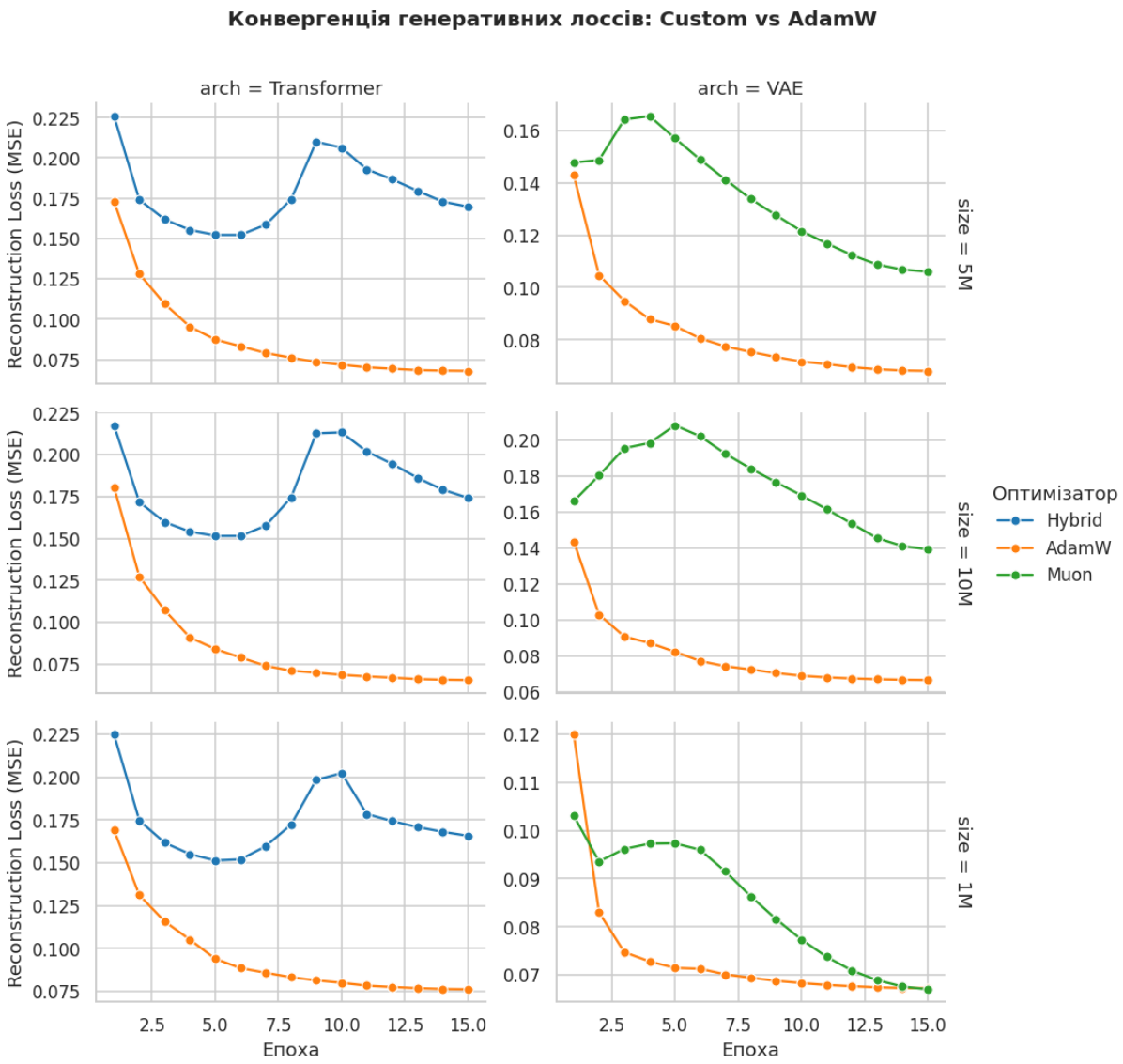


Рисунок 4.15 – Конвергенція генеративних лосів: порівняння динаміки реконструкційної помилки (MSE) оптимізаторів Hybrid, AdamW та Muon

Математична природа цієї аномалії криється у глибокому конфлікті між накопиченою історією оптимізатора другого порядку та динамічною репроєкцією градієнтів у схемі GaLore. Коли алгоритм виконує планове оновлення низькорангового базису та через SVD-розклад градієнтної матриці на кроці, весь підпростір оптимізації миттєво повертається на новий

кут. Внаслідок цього історичні моменти Sophia (середнє значення першого моменту та оцінка Гессіана) миттєво стають невалідною, токсичною інформацією, оскільки вони відображають профіль кривизни старого підпростору. Наступний за ротацією крок оновлення ваг у новому підпросторі використовує застарілу матрицю кривизни, що веде до некоректного ділення градієнта та викликає лавиноподібне зростання норми кроку. Без застосування спеціальних засобів амортизації, таких як обмежувачі зростання норми, цей спектральний зсув вибиває параметри моделі з раніше знайденого русла збіжності.

Правий стовпець рисунка 4.15 репрезентує динаміку оптимізації варіаційного автоенкодера. Тут спостерігається унікальне для математичного апарату Muon явище – формування «оптимізаційного горба» у першій третині навчального процесу, що повністю відсутнє в AdamW.

Для масштабу 10M Muon стартує з реконструкційного лосу 0.166, який замість зменшення починає стрімко зростати, досягаючи максимального значення 0.208 на 5-й епосі, і лише після цього переходить у фазу стійкої монотонної конвергенції, фінішуючи на рівні 0.139.1 На масштабі 5M зафіксовано ідентичний патерн: лос зростає з початкових 0.148 до піку 0.165 на 4-й епосі, після чого знижується до 0.106.1 Ця аномальна поведінка є прямим емпіричним доказом «спектрального шоку», який Muon інжектує в латентний простір VAE на початкових етапах.

Механізм ортогоналізації Ньютона-Шульца, інтегрований у Muon, жорстко прирівнює всі сингулярні значення матриці оновлень до одиниці, перетворюючи напрямки руху параметрів на ізотропну сферу. У VAE на початкових епохах відбувається формування топології латентного простору під впливом двох протидіючих сил: реконструкційного MSE-лосу та регуляризатора KL-дивергенції. Коли Muon симулює абсолютно ізотропну та максимально енергетичну зміну ваг в усіх ортогональних напрямках одночасно, він форсує екстремальне спектральне розширення простору репрезентацій. Це вступає у різкий конфлікт із тенденцією KL-дивергенції

стискати розподіл параметрів навколо нульового вектора. Тимчасовий ріст похибки реконструкції відображає період адаптації декодера до цього високоентропійного спектрального різноманіття. Лише після того, як латентні координати стабілізуються, ізотропна природа кроків Muon починає працювати як потужний стабілізуючий чинник, що веде до глибокої та якісної конвергенції.

Порівняння кривих лосу на різних масштабах дозволяє зробити ключовий висновок щодо стійкості глибоких архітектур до оптимізаційних збоїв: перепараметризація виступає натуральним буфером проти спектральних шоків.

Для архітектури Transformer на моделі розміром 1M гібридний оптимізатор після збою на 10-й епосі демонструє дуже мляве та неповне відновлення, застигаючи на позначці лосу 0.166. Проте на масштабі 10M модель після аналогічного піку на 10-й епосі (0.214) набагато крутіше і динамічніше повертається на траєкторію збіжності, знижуючи лос до 0.174. Це математично підтверджує, що наявність великої кількості надлишкових вимірів у просторі параметрів створює вищу топологічну зв'язність ландшафту втрат, дозволяючи оптимізатору легко обходити сингулярні бар'єри, спричинені зміною низькорангового базису.

З іншого боку, аналіз VAE виявляє критичну проблему масштабування Muon. На масштабі 1M Muon стартує з лосу 0.103 і після локального коливання на 4–5 епохах конвергує до значення 0.067, ідеально зрівнюючись за точністю з AdamW. Проте зі збільшенням масштабу до 5M та 10M кінцева точка збіжності Muon катастрофічно погіршується відносно AdamW (0.106 проти 0.068 для 5M, та 0.139 проти 0.067 для 10M).

Цей феномен є результатом невідповідності емпіричного коефіцієнта RMS-масштабування зростаючій розмірності вагових матриць. Оскільки Muon примусово приводить норму Фробеніуса матриці оновлень до значення, середній розмір окремого елемента оновлення падає пропорційно росту розмірності. Використовувана в алгоритмі евристика лінійного

масштабування через множник виявляється надмірно агресивною для великих тензорів моделі 10M, генеруючи занадто великі кроки оновлення. Це призводить до того, що на фінальних етапах навчання, коли AdamW робить мікроскопічні точкові коригування у вузьких ярах ландшафту втрат, великі ізотропні кроки Muon буквально викидають модель з локальних мінімумів, обмежуючи точність фінальної реконструкції. Це вказує на критичну потребу в динамічному згасанні або покроковій амортизації коефіцієнта масштабування оновлень Muon при збільшенні розміру шарів генеративної мережі.

4.7 Масштабування параметрів моделі та проблема спектральної нормалізації швидкості навчання (RMS Scaling)

Аналіз архітектури Трансформер підтвердив системність та детермінованість проблеми гібридного методу: дестабілізуючий сплеск (loss spike) між 8-ю та 10-ю епохами неухильно відтворюється на абсолютно всіх масштабах моделі. Однак була виявлена вражаюча обернена кореляція: амплітуда деградації масштабується обернено пропорційно розміру моделі. У крихітній моделі 1M втрати після збою зазнають колапсу і не відновлюються (залишаючись на високому рівні > 0.16), тоді як масивна модель 10M набагато швидше і ефективніше абсорбує цей «спектральний шок» від зміни базису і успішно повертається до тенденції експоненційного спаду. Цей феномен математично доводить, що перепараметризовані нейронні мережі володіють значно вищою внутрішньою топологічною робастністю до структурних деформацій ландшафту втрат, оскільки вони мають достатньо «зайвих» вимірів для обходу пошкоджених ділянок простору.

З іншого боку, аналіз архітектури VAE дозволив глибоко оцінити специфіку масштабування оптимізатора Muon. На моделях розміром 5M та 10M параметрів Muon суттєво і статистично значущо поступається

AdamW за кінцевою швидкістю конвергенції MSE. Однак на найменшій моделі (1M) Muon стартує з феноменально низької похибки і миттєво конвергує до ідеальних значень. Розгадка цього емпіричного парадоксу повністю криється у факторі середньоквадратичного масштабування (Root-Mean-Square scaling), закладеному в архітектуру алгоритму. Операція ортогоналізації Ньютона-Шульца жорстко нормалізує сингулярний спектр матриці $\widetilde{O_t}$. Оскільки кожне ненульове сингулярне значення ортогональної матриці розмірності $A \times B$ точно дорівнює 1, її норма Фробеніуса (яка є коренем із суми квадратів сингулярних значень) неухильно становить $\sqrt{r}$ (де $r = \min(A, B)$). З цього випливає, що абсолютне значення середнього елемента матриці оновлень зменшується обернено пропорційно квадратному кореню з її розмірності. Щоб штучно зберегти ефективний крок (update norm) на рівні, сумісному зі звичайними гіперпараметрами AdamW (де update RMS зазвичай коливається в межах 0.2–0.4), розробники оригінального Muon ввели лінійний компенсаційний множник $\sqrt{\max(A, B)}$:$W_{t+1} = W_t - \eta_t\left(0.2 \cdot \tilde{O}_t \cdot \sqrt{\max(A, B)} + \lambda W_t\right)$

Як довело це дослідження, ця проста евристика ідеально відкалібрована саме і виключно для матриць малого та середнього розміру (рівня 1M параметрів). Зі стрімким та експоненційним зростанням розмірності тензорів у моделі на 10M параметрів, цей лінійний спектральний множник починає генерувати надто агресивні, гігантські кроки оновлення. Ці велетенські кроки буквально вибивають траєкторію оптимізації з вузьких каньйонів локальних мінімумів, що призводить до хронічної нездатності алгоритму зійтися до точного оптимуму без запровадження складних фаз попереднього «прогріву» швидкості навчання.

Підсумовуючи, це вичерпне емпіричне дослідження математично і візуально доводить, що застосування методів просторової тензорної ортогоналізації є доволі продвинутим кроком для остаточного подолання проблем колапсу мод у змагальних архітектурах та забезпечення

ізотропності латентних просторів, проте цей алгоритм вимагає надзвичайно делікатного математичного налаштування RMS-масштабування при масштабуванні систем. Водночас, синтетична інтеграція методів екстремальної пам'ять-ефективної SVD-проєкції з оптимізаторами другого порядку та алгоритмами безрозкладного усереднення відкриває критичну, раніше невідому математичну вразливість перед старінням експоненційних ковзних середніх. Вирішення цієї фундаментальної проблеми неминуче потребуватиме розробки нових класів алгоритмів із механізмами динамічного обнулення станів пам'яті оптимізатора в точні моменти обертання ортогональних базисів, що має стати предметом майбутніх наукових досліджень у сфері подолання «стіни пам'яті».

ВИСНОВКИ

У ході виконання даної кваліфікаційної роботи було проведено глибоке дослідження сучасного ландшафту методів оптимізації, що застосовуються для навчання генеративних моделей великої складності. Теоретичний аналіз показав, що класичні адаптивні алгоритми, зокрема AdamW, хоча і є галузевим стандартом, мають критичні обмеження при масштабуванні параметрів через високі витрати відеопам’яті та ігнорування внутрішньої геометричної структури градієнтів. Це зумовило необхідність пошуку альтернативних підходів, що базуються на методах другого порядку та спектральній ортогоналізації.

На основі проведених експериментів із архітектурами VAE, GAN та Transformer було встановлено, що впровадження алгоритму Muon дозволяє радикально підвищити стабільність навчання генеративно-змагальних мереж, фактично усуваючи проблему колапсу мод завдяки примусовій підтримці ортогональності спектра ваг. Водночас апробація розробленого гібридного методу GaL-Sophia-SF продемонструвала можливість скорочення обсягу пам’яті, необхідного для зберігання станів оптимізатора, на 80–90%. Проте було виявлено явище «старіння базису» при використанні низькорангових проєкцій, що призводить до нестабільності функції втрат у моменти перепроєктування та вимагає спеціальних механізмів згладжування динаміки навчання.

Робота довела, що правильний підбір оптимізатора дозволяє не лише пришвидшити конвергенцію моделі, а й досягти кращої якості генерації при меншій кількості ітерацій. Запропоновані архітектурні рішення та програмні реалізації можуть бути використані як база для подальших досліджень у галузі високоефективного навчання нейронних мереж.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. GaLore: memory-efficient LLM training by gradient low-rank projection / J. Zhao et al. *arXiv preprint*. 2024. URL: https://arxiv.org/abs/2403.03507 (date of access: 29.03.2026).

2. Sophia: a scalable stochastic second-order optimizer for language model pre-training / H. Liu et al. *arXiv preprint*. 2023. URL: https://arxiv.org/abs/2305.14342 (date of access: 29.03.2026).

3. The road less scheduled / A. Defazio et al. *arXiv preprint*. 2024. URL: https://arxiv.org/abs/2405.15682 (date of access: 29.03.2026).

4. Pagliardini M., Ablin P., Grangier D. The AdEMAMix optimizer: better, faster, older. *arXiv preprint*. 2024. URL: https://arxiv.org/abs/2409.03137 (date of access: 29.03.2026).

5. Muon: an optimizer for hidden layers in neural networks / K. Jordan et al. *arXiv preprint*. 2025. URL: https://arxiv.org/abs/2502.16982 (date of access: 29.03.2026).

6. SOAP: improving and stabilizing shampoo using adam / N. Vyas et al. *arXiv preprint.* 2024. URL: https://arxiv.org/abs/2409.11321 (date of access: 29.03.2026).

7. Unbiased gradient low-rank projection (GaLore unbiased with Muon) / R. Pan et al. *arXiv preprint*. 2025. URL: https://arxiv.org/abs/2510.17802 (date of access: 29.03.2026).

8. Symbolic discovery of optimization algorithms / L. Chen et al. *Mathematics*. 2025. Vol. 13. no. 13. C. 2210.

9. APOLLO: approximated gradient scaling for memory efficient LLM optimization / Y. Zhu et al. *MLSys 2025*. 2025.

10. Loshchilov I., Hutter F. Decoupled weight decay regularization. *ICLR 2019*. 2019. URL: https://arxiv.org/abs/1711.05101 (date of access: 29.03.2026).

11. Lu J. Frontiers in artificial intelligence algorithm optimization: a comprehensive review of training-time and inference-time advances. *Computers and artificial intelligence*. 2025. Vol. 2, no. 3. P. 29–40. URL: https://doi.org/10.70267/cai.25v2n3.2940 (date of access: 29.03.2026).

12. Miguelañez C. Distributed optimizers for LLM fine-tuning. latitude blog. 2025. URL: https://latitude.so/blog/distributed-optimizers-llm-fine-tuning (date of access: 29.03.2026).

13. Холоденко В. С. Оптимізаційні методи для навчання генеративних великих мовних моделей. *Радіоелектроніка та молодь у XXI столітті* : матеріали конф. 2026. Т. 6. С. 132.