ПОЛТАВСЬКИЙ ДЕРЖАВНИЙ АГРАРНИЙ УНІВЕРСИТЕТ
Навчально-науковий інститут економіки, управління, права та інформаційних технологій
Кафедра інформаційних систем та технологій

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти магістр

на тему: **«Локалізація малих мовних моделей на Raspbery Pi5»**

Виконав: здобувач вищої освіти
за освітньою програмою
Інформаційні управляючі системи та технології
спеціальності 126 Інформаційні системи та технології
ступеня вищої освіти магістр
групи 126ІСТ_мд_2023
Чорновіл Вадим Сергійович
Керівник: Слюсар Вадим Іванович
Рецензент: Муравльов Володимир Вячеславович

Полтава – 2024 року

ВСТУП

Актуальність теми кваліфікаційної роботи підтверджується необхідністю інтеграції генеративного штучного інтелекту (Generative Artificial Intelligence, GenAI) та граничних обчислень зменшення затримок, скорочення трафіку, що передається в хмару та ін. Для цього необхідно виконувати локалізацію великих мовних моделей (Large Language Model, LLM). Їх впровадження дозволяє підприємствам працювати в автономному режимі, підвищувати конфіденційність та безпеку даних, досягати економічної ефективності та налаштовувати функції LLM відповідно до конкретних операційних вимог. Однак, LLM досягають успіху в розумінні і генерації тексту та опису зображень за рахунок високих вимог до обчислювальних ресурсів, зокрема, наявності графічних процесорів, обсягів пам'яті. Як наслідок, замість LLM намагаються використовувати малі мовні моделі (SLM) на периферійних пристроях з обмеженими ресурсами. Такий підхід є недостатньо дослідженим, що підкреслює його актуальність для розвитку та впровадження GenAI.

Зв'язок роботи з науковими програмами, темами. Робота відповідає дослідженням в межах науково-дослідної ініціативної тематики «Організаційно-методологічні аспекти впровадження інформаційно-комунікаційних систем і технологій в управлінні діяльністю сучасних організацій та підприємств за умов переходу до цифрової економіки» (ДРН 0123U105060, 2023-2028 рр.), що реалізується на кафедрі інформаційних систем та технологій, тематиці досліджень навчально-дослідної лабораторії інтелектуальних систем, комп'ютерних мереж та інтернет речей кафедри інформаційних систем та технологій Полтавського державного аграрного університету.

Метою кваліфікаційної роботи є підтвердження можливості функціонування мовних моделей у вбудованих комп'ютерних системах.

Завданнями кваліфікаційної роботи є:

– аналіз стратегій локального розгортання моделей штучного інтелекту;

– розробка методики локалізації мовних моделей;

– розробка рекомендацій щодо використання методики локалізації мовних моделей.

Об'єктом дослідження є процеси розгортання та функціонування великих мовних моделей у вбудовуваних комп'ютерних системах.

Предметом дослідження є реалізація та використання мовних моделей на комп'ютерних платформах з обмеженими апаратними ресурсами.

Методами дослідження є аналітичний, інформаційно-пошуковий, методи оптимізації мовних моделей, робота з бібліотекою Ollama.

Інформаційна база кваліфікаційної роботи базується на ресурсах про велики та малі мовні моделі, інструментарій для їх локалізації на периферійних пристроях, а також методи оптимізації їх роботи на вбудованих комп'ютерних системах з обмеженими обчислювальними потужностями.

Елементи наукової новизни роботи полягають в розробці методики локалізації мовних моделей на Raspbery Pi 5.

Практична значущість роботи полягає в розробці рекомендацій щодо локалізації мовних моделей – можуть бути використані для подальших досліджень за даною тематикою та при проектуванні вбудованих комп'ютерних систем.

Апробація результатів відбувалася в рамках VII Міжнародної студентської наукової конференції «Наука сьогодення: від досліджень до стратегічних рішень» (листопад 2024 р., м. Кривий Ріг) та VII Міжнародної студентської наукової конференції «Розвиток суспільства та науки в умовах цифрової трансформації» (листопад 2024 р., м. Тернопіль).

Структура кваліфікаційної роботи логічно пов'язана з завданнями досліджень і містить вступ, три розділи основної частини, висновки, список використаних джерел, додатки. Загальний обсяг пояснювальної записки кваліфікаційної роботи складає 76 сторінок формату А4. Вона містить 19 рисунків і 3 таблиці.

РОЗДІЛ 1
АНАЛІЗ ОСОБЛИВОСТЕЙ РЕАЛІЗАЦІЇ ГЕНЕРАТИВНОГО ШТУЧНОГО ІНТЕЛЕКТУ

1.1 Напрями розвитку генеративного штучного інтелекту

Як відомо [1], генеративний штучний інтелект (Generative Artificial Intelligence, GenAI) – одна з галузей, що швидко розвиваються, яка створює те, чого раніше не існувало. Насамперед це нові форми контенту: текстовий, аудіо та візуальний. Генеративні моделі використовують для навчання набори даних, проте не просто комбінують їх відповідно до запиту, а фактично створюють із нуля. У цьому полягає головна відмінність від дискретного AI, який аналізує різницю між різними типами даних.

Всього за останні півроку GenAI повністю змінив те, як працюють галузі. Від залів засідань до класів – можна побачити вплив AI скрізь. Діловий світ сприйняв GenAI досить позитивно. Згідно опитування [2], 65 % компаній зараз регулярно використовують GenAI, що вдвічі більше, ніж на початку 2024 р. Тому надалі доцільно визначити напрями подальшого розвитку GenAI.

Перший напрям – це формування якісних власних даних. Чим вище ця якість, тим краще працює GenAI [3]. Тобто, компанії бажають, щоб він працював на їхні конкретні потреби. Нажаль, 85 % проєктів GenAI ще не запущені, головним чином тому, що компанії неефективно використовують свої дані [3]. Якщо компанія збирає багато відгуків клієнтів або відстежує ланцюжок поставок або будь-які дані компанії, це гарна основа для реалізації GenAI. Але справжня проблема полягає в тому, щоб перетворити всі ці дані на корисні програми AI. Таким чином, розумне використання даних може забезпечити перетворення всієї інформації компанії на тренувальні дані, які забезпечують роботу GenAI.

Другий напрям – це тонке налаштування моделей, тобто створення індивідуальних моделей. У світі бізнесу справжня сила GenAI полягає в тонкому налаштуванні моделей відповідно до конкретних потреб, формуючи кастомний AI для підприємств. Недостатньо мати багато даних. Ключем до успіху є коригування моделей AI для ефективної роботи з цими даними. Компаніям потрібно перетворювати свої необроблені дані на корисні навчальні набори, які можуть конкретно спрямовувати та покращувати продуктивність AI. В даному випадку, можна використовувати послуги певних ресурсів, наприклад, SuperAnnotate [4]. Компанії створюють набори даних для тонкого налаштування та оцінки великих мовних моделей (Large Language Model, LLM) з урахуванням їхніх конкретних випадків використання, щоб навчати моделі на цих наборах даних.

Третій напрям – це мультимодальний AI. Пройшли ті часи, коли AI міг обробляти лише текст. У 2024 р. досить поширено стали застосовуватись мультимодальні моделі, що можуть писати, розмовляти, бачити і розуміти відео. У 2023 р. розмір ринку мультимодальних моделей оцінювався в 1,2 мільярда доларів США і, як очікується, зростатиме із середньорічним темпом зростання понад 30 % у період з 2024 по 2032 роки (рис. 1.1) [5].

Рисунок 1.1 – Аналіз ринку акселераторів дата-центрів [5]

Чому мультимодальність зараз така актуальна? Все просто: ці моделі ближчі до того, як люди сприймають і обробляють інформацію з кількох джерел одночасно. Мультимодальний AI починає наздоганяти людський багатозадачний мозок. На даний час, існує кілька мультимодальних моделей AI, і очікується, що найближчим часом їх з'явиться багато інших, тобто майбутнє за мультимодальністю.

Четвертий напрям – моделі з відкритим вихідним кодом. У 2024 р. анонсувалось багато нових версій моделей AI з відкритим вихідним кодом, наприклад, Llama від к. Meta, моделі від к. Mistral (Mixtral 8×7B, Mixtral 8×22B), Gemma від к. Google та ін. Вони доступніші за ціною, що полегшує впровадження інновацій для розробників і малого бізнесу. Наприклад, «Llama 3» перетворили на інструмент, який компанія може завантажити та використовувати відразу. Цей крок трансформує компанії будь-якого розміру, допомагаючи їм інтегрувати AI у свою повсякденну діяльність. Цей перехід до відкритого вихідного коду дозволяє кожній компанії стати компанією з AI, що є великим кроком вперед. На рис. 1.2 показано, як актуальна модель з відкритим вихідним кодом, Llama 3.1 від к. Meta, вперше в історії скоротила розрив з моделями з закритим вихідним кодом.

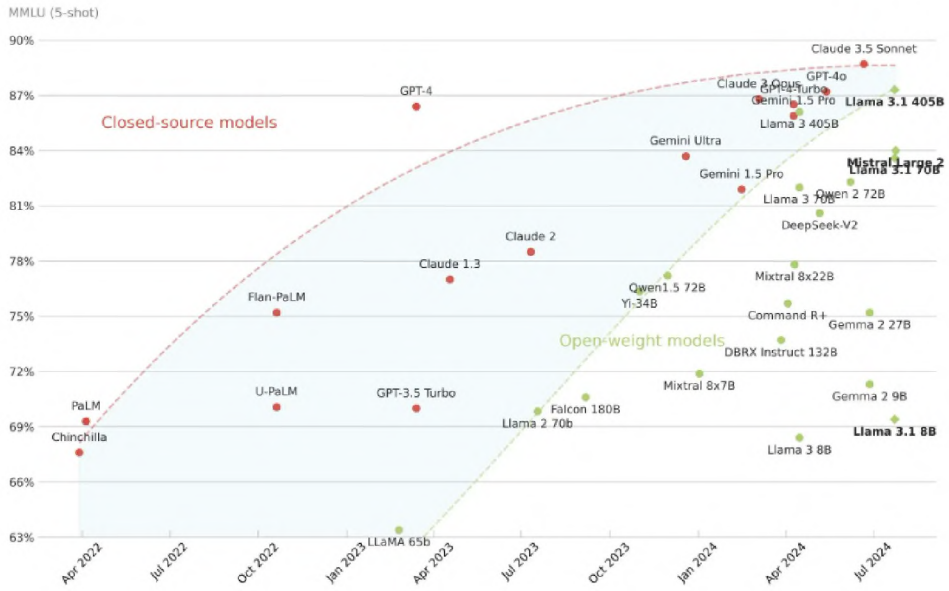


Рисунок 1.2 – Скорочення розрив між закритими моделями і Llama 3.1 [6]

Передбачається, що вони перевершать всіма використовувані моделі з закритим вихідним кодом. Моделі AI з відкритим і закритим вихідним кодом є важливими. Моделі з відкритим вихідним кодом стимулюють широке впровадження інновацій та впровадження, тоді як моделі із закритим вихідним кодом розширюють межі AI.

П'ятий напрям – агентна система. Агентний AI з'явився лише в середині 2024 р., і це вже гаряча тема. Цей новий вид AI може працювати самостійно, приймаючи рішення та діючи, не потребуючи, щоб люди керували ним на кожному кроці. Це значно відрізняється від попереднього AI, який робив лише те, що йому сказали. Такі агенти AI особливо корисні в обслуговуванні клієнтів, де вони можуть ефективніше виконувати завдання, економлячи час і підвищуючи продуктивність, або у фінансах, де вони швидко аналізують дані та надають рекомендації. Особливо цікавими є агенти на основі LLM [7]. Вони справляються зі складними завданнями генерації тексту, надаючи відповіді на складні питання, з якими не можуть впоратися простіші моделі.

Шостий напрям – регулювання та етика AI. Оскільки AI стає все більшою частиною людського життя, важливо обговорювати, як люди забезпечують його безпеку та справедливість. Ця тенденція полягає у встановленні правил, які гарантують, що AI використовується так, щоб це було корисно для всіх. Уряди та організації по всьому світу працюють над цими правилами прямо зараз. Вони зосереджуються на таких напрямах: як AI повинен обробляти особисті дані, щоб переконатися, що AI не поводиться з людьми несправедливо; як тримати рішення щодо AI відкритими, щоб люди могли їх розуміти та ставити під сумнів. Мова йде про відповідальне використання AI та забезпечення того, щоб він став позитивним доповненням до нашого світу.

AI неймовірно швидко розвивається, і здається, що зараз усі стають компаніями зі штучним інтелектом. У міру того, як все більше підприємств вступають у гонку AI, важливо уважно спостерігати за тенденціями та

прогнозувати, що станеться далі. Таким чином, майбутнє GenAI залежить від даних, охоплює мультимодальність, надає пріоритет кастомізації, бачить великий потенціал у відкритому вихідному коді та визнає необхідність етики.

1.2 Однобітові LLM

У міру зростання обсягу даних підприємства стикаються з проблемами в управлінні своїми системами знань. Хоча LLM досягають успіху в розумінні та генерації тексту, вони вимагають значних обчислювальних ресурсів, часто потребуючи сотень гігабайт пам'яті та дорогого обладнання графічного процесора. Це створює значний бар'єр для багатьох організацій, поряд із занепокоєнням щодо конфіденційності даних та операційних витрат [8]. Як наслідок, важко використовувати можливості AI, необхідні для збереження конкурентоспроможності, оскільки поточні LLM часто технічно та фінансово недосяжні.

Новаторським рішенням, яке вирішує ці проблеми, є однобітові LLM [9]. Також вони мають іншу назву – Ternary Large Language Models (TriLM). На відміну від традиційних LLM, які зазвичай вимагають 32- або 16-бітові ваги для зберігання даних параметрів, однобітові LLM використовують висококвантові представлення – лише до 3-ох значень (наприклад, -1, 0 та 1). Це стиснення зменшує потреби моделі в пам'яті приблизно на 97 % і дає можливість запускати складні мовні моделі локально, використовуючи CPU, а не спеціалізовані GPU, таким чином усуваючи потребу в викликах API до зовнішніх серверів.

Щоб зрозуміти значимість цього досягнення, треба врахувати те, що традиційна модель з 7 мільярдами параметрів, зазвичай, вимагає приблизно 26 ГБ дискового простору. Ця ж модель при перетворенні в однобітову LLM вимагає всього 0,815 ГБ – скорочення майже на 97 %. Підприємства можуть запускати потужні моделі AI без необхідності в ЦОД, заповненому GPU,

знижуючи витрати та ризики, пов'язані із зовнішніми залежностями, одночасно підвищуючи конфіденційність даних. Принцип роботи таких однобітових LLM полягає в наступному.

Спрощені обчислення, які керують однобітовими LLM, дозволяють їм ефективно працювати на стандартних CPU, навіть замінюючи GPU в деяких випадках. Як правило, GPU віддають перевагу через їхню здатність швидко обробляти складні множення, що є необхідністю в традиційних архітектурах нейронних мереж. Однак з однобітовими LLM потреба в множенні відпадає.

Бінарні нейронні мережі та квантування ваги. Уявіть, що зберігаєте в бібліотеці лише суть кожної книги, а не кожну деталь. Однобітові LLM використовують двійкові нейронні мережі (BNN) для зберігання лише важливої інформації, що робить модель легшою та швидшою. Подібно до лаконічної бібліотеки, де кожна книга зберігає лише свої ключові моменти, ці моделі економлять місце, гарантуючи, що критична інформація залишається чіткою та доступною у високодеталізованих «книгах».

Однобітові LLM використовують два ключові підходи для ефективного стиснення моделі AI.

По-перше, квантизація після тренування (PTQ) починається з повністю навченої моделі, стискаючи її шляхом перетворення більшості параметрів в 1 біт (+1 або -1), зберігаючи критичні параметри з трохи вищою точністю. Цей метод дозволяє зберегти продуктивність моделі при значному зменшенні розмірів. Він знімає зайву вагу, але підсилює ключові компоненти, щоб забезпечити його стабільність.

Метод Quantization-Aware Training (QAT) тренує модель з нуля з урахуванням однобітових параметрів. Хоча це вимагає більше зусиль, це оптимізує модель для високої ефективності з самого початку.

По-друге, застосовується усунення множення. Уявіть, що рахуєте купу каменів, додаючи або видаляючи їх, а не виконуєте складні обчислення. У однобітової LLM ваги стискаються до 3-ох значень (наприклад, -1, 0 та 1). Ця екстремальна квантизація повністю усуває потребу в множенні, дозволяючи

проводити ефективні обчислення на процесорах, замінюючи складні операції простим додаванням і відніманням. Модель використовує додавання та віднімання для операцій -1 та 1, оскільки додавання та віднімання набагато простіші та швидші, ніж множення, і вона пропускає операції, коли йдеться про 0. Це робить весь процес набагато швидшим та ефективнішим, заощаджуючи час на обчислення та енергію.

Таке квантування відкриває нові апаратні можливості, дозволяючи центральним процесорам ефективно обробляти ці моделі. Цей процес мінімізує споживання енергії та обчислювальні потреби, роблячи такі LLM сумісними з іншим обладнанням, таким як інтегральні схеми для конкретного застосування (ASIC).

Впровадження однобітових LLM вирішує кілька критичних проблем, попутно трансформуючи можливості управління знаннями. Традиційні LLM часто залежать від дорогих сторонніх серверів та API, що викликає увагу до конфіденційності даних, затримки та витрат. Локальне розміщення 1-бітових LLM дозволяє компаніям зберігати конфіденційні дані в безпеці та знижувати операційні витрати, пов'язані з хмарною інфраструктурою. Локальний хостинг дозволяє організаціям впроваджувати безпечні моделі управління знаннями на основі AI, такі як AI Fortune Cookie [10], для обробки документів у режимі реального часу на основі структурованих і неструктурованих даних. Такий підхід підвищує безпеку даних, зменшує затримку та пропонує масштабоване керування документами без значних витрат на обладнання. Підтримуючи аналіз, класифікацію та вилучення інформації за допомогою можливостей запитів природною мовою, ці моделі покращують операційну ефективність, прийняття рішень і конфіденційність даних, мінімізуючи залежність від дорогих зовнішніх API. Незважаючи на ефективне стиснення, однобітові LLM досягають вражаючої швидкості обробки. Вони демонструють прискорення в діапазоні від 1,37× до 5,07× на процесорах ARM і від 2,37× до 6,17× на системах x86. Ці моделі можуть обробляти 5-7 токенів за секунду на стандартному обладнанні процесора. Ці моделі

оптимізовані для швидкості, що дозволяє швидше обробляти завдання на системах на базі процесора. Вони також здатні ефективно працювати з різними розмірами моделі (їх можна адаптувати до різних можливостей процесора), що робить їх універсальними для різних програм. Приріст енергоефективності однобітових LLM є вражаючим. Тести показують зниження енергоспоживання на 55-70 % на процесорах ARM і на 71-82 % економії на процесорах x86. Це безпосередньо призводить до зниження операційних витрат і зменшення вуглецевого сліду, допомагаючи організаціям досягати як фінансових, так і сталих цілей. Це важливо для процесорів у мобільних та периферійних пристроях, оскільки енергоефективність може продовжити час автономної роботи та знизити експлуатаційні витрати.

В цілому, розробка однобітових LLM є не просто технічним досягненням – вона знаменує собою фундаментальну зміну в тому, як підприємства можуть підійти до впровадження AI. Ефективно знижуючи вимоги до ресурсів при збереженні високого рівня продуктивності, ці моделі демократизують доступ до передових можливостей AI. Для підприємств це означає:

- зниження вимог до початкових інвестицій;
- зниження поточних експлуатаційних витрат;
- кращий контроль за безпекою та безпекою даних;
- спрощені процедури розгортання та обслуговування;
- підвищена доступність можливостей AI в різних відділах.

У міру того, як технологія продовжує розвиватися, можна очікувати, що все більше організацій впроваджуватимуть ці ефективні моделі для своїх ініціатив щодо AI. Це широке впровадження сприятиме подальшим інноваціям в ефективності моделей та корпоративних застосувань, створюючи позитивний цикл технологічного прогресу та практичного впровадження.

1.3 Стратегії локального розгортання моделей штучного інтелекту

LLM трансформували обробку природної мови (Natural Language Processing, NLP) і генерацію контенту, продемонструвавши чудові можливості в інтерпретації та створенні тексту. LLM часто розгортаються в інфраструктурах хмарних обчислень, що спричиняють кілька проблем [12]. Наприклад, для моделі з 7 мільярдами параметрів вимоги до пам'яті варіюються від 7 до 28 ГБ, залежно від точності, при цьому тренування вимагають вчетверо більшого обсягу. Така висока потреба в пам'яті в хмарних середовищах може виснажити ресурси, збільшити витрати та спричинити проблеми з масштабованістю та затримкою, оскільки дані повинні переміщатися до хмарних серверів і з них, що призводить до затримок у програмах у реальному часі. Витрати на пропускну здатність можуть бути високими через великі обсяги переданих даних, особливо для програм, які потребують частих оновлень. Проблеми з конфіденційністю також виникають, коли конфіденційні дані надсилаються на хмарні сервери, наражаючи інформацію користувачів на потенційний витік.

Ці проблеми можна вирішити за допомогою периферійних пристроїв, які наближають обробку LLM до джерел даних, забезпечуючи локальну обробку великих обсягів даних у режимі реального часу [13]. В даному контексті варто розглянути кілька підходів.

З'єднання точок (поєднання інтерфейсу Edge AI та інтеграції LLM). Периферійні пристрої обробляють дані локально, зменшуючи затримку, використання пропускної здатності та експлуатаційні витрати, одночасно покращуючи продуктивність. Завдяки розподілу робочих навантажень між кількома периферійними пристроями навантаження на хмарну інфраструктуру зменшується, що сприяє масштабуванню завдань з інтенсивним використанням пам'яті, таких як навчання LLM і вивід, для швидшого та ефективнішого реагування. Розгортання LLM на периферійних

пристроях вимагає вибору менших, оптимізованих моделей, адаптованих до конкретних випадків використання, забезпечуючи безперебійну роботу в умовах обмежених ресурсів. Методи оптимізації моделей підвищують ефективність LLM, зменшуючи обчислювальні вимоги, використання пам'яті та затримку без значного погіршення точності або ефективності периферійних систем.

Квантування – процедура знижує точність моделі, перетворюючи параметри з 32-бітових чисел з плаваючою комою в формати з нижчою точністю, такі як 16-бітові або 8-бітові цілі числа. Це передбачає представлення високоточних значень у меншому діапазоні з коригуванням масштабу та зміщення, що економить пам'ять та прискорює обчислення. Воно економить пам'ять і прискорює обчислення, знижуючи витрати на обладнання та споживання енергії, зберігаючи при цьому продуктивність в реальному часі. Це робить можливим використання LLM для пристроїв з обмеженими ресурсами (мобільні телефони та периферійні платформи). Інструменти AI, такі як TensorFlow [14], PyTorch [15], Intel OpenVINO [16] та NVIDIA TensorRT [17], підтримують квантування для оптимізації моделей для різних фреймворків і потреб. На даний час, можливо виділити кілька методів квантування (рис. 1.3).

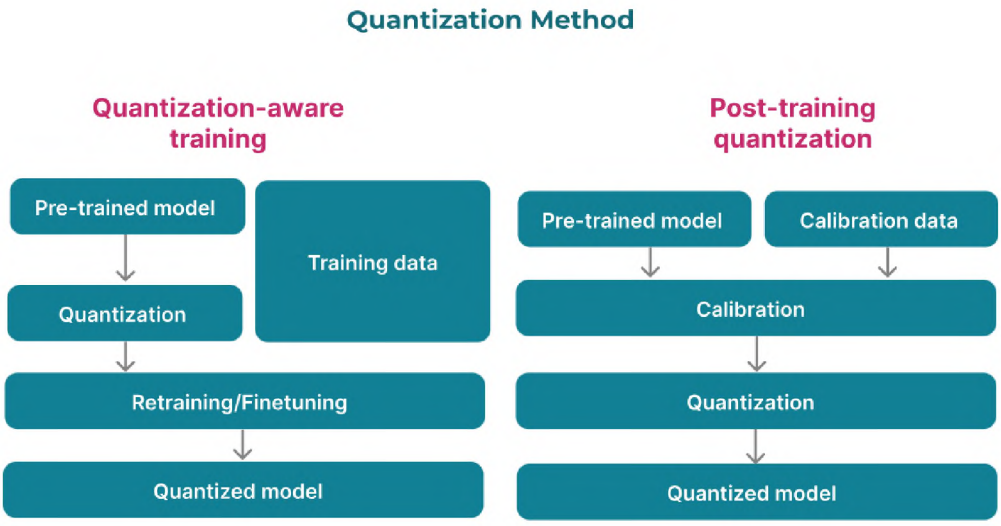


Рисунок 1.3 – Методи квантування

Квантування після тренування (PTQ). Цей варіант зменшує точність вагів у попередньо навченій моделі після тренування, перетворюючи їх на 8-бітові цілі числа або 16-розрядні числа з плаваючою комою.

Навчання з урахуванням квантування (QAT) – інтегрує квантування під час тренування, дозволяючи регулювати вагу для нижчої точності.

Уніфіковане квантування після тренування з нульовим пострілом. Воно застосовує стандартне квантування без додаткового навчання, оцінюючи його вплив на різні моделі.

Квантування лише за вагою – метод фокусується лише на вагах, перетворюючи їх у fp16 під час множення матриць для покращення швидкості висновків та зменшення завантаження даних.

Крім квантування існує також обрізка. Вона зменшує зайві нейрони та зв'язки в моделі AI. При цьому виконується аналіз мережі, використовуючи вагу, величину (припускає, що менші ваги менше впливають на вихід) або методи аналізу чутливості (наскільки змінюється вихідний сигнал моделі при зміні конкретної ваги), щоб визначити, які частини мають мінімальний вплив на кінцеві прогнози. Потім їх або видаляють, або встановлюють їх ваги на нуль. Після обрізки модель може бути точно налаштована для відновлення будь-якої продуктивності, втраченої під час процесу обрізки. Серед основних методів обрізки слід виділити наступні (рис. 1.4).

Структурована обрізка видаляє групи вагів, як-от канали або шари, щоб оптимізувати ефективність моделі на стандартному обладнанні (CPU, GPU). Такі інструменти, як TensorFlow і PyTorch, дозволяють користувачам вказувати частини для обрізки з подальшим тонким налаштуванням для відновлення точності.

Неструктурована обрізка усуває індивідуальні, менш важливі ваги, створюючи розріджену мережу та зменшуючи використання пам'яті за рахунок встановлення незначних вагів на нуль. Для цього використовуються такі інструменти, як PyTorch, і застосовується точне налаштування для відновлення будь-якої втрати продуктивності.

Обрізка допомагає інтегрувати LLM з периферійними пристроями, зменшуючи їх розмір і обчислювальні вимоги, що робить їх придатними для обмежених ресурсів, доступних на периферійних пристроях. Нижчий рівень споживання ресурсів призводить до швидшого часу відгуку та зменшення споживання енергії.

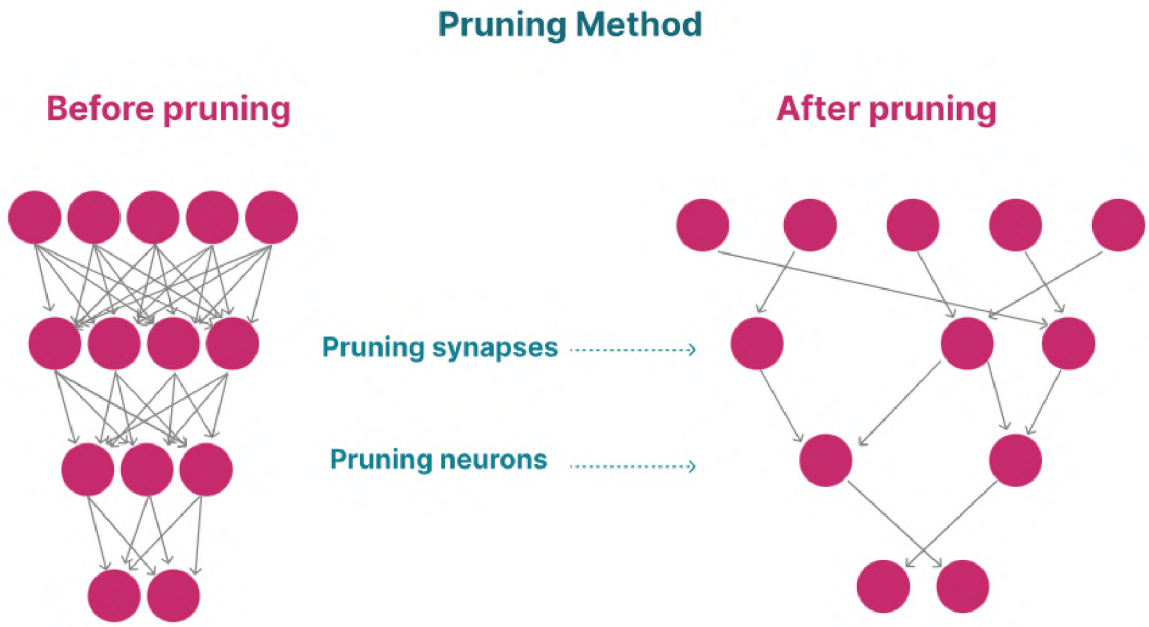


Рисунок 1.4 – Методи обрізки

Дистиляція знань (рис. 1.5). Вона стискає велику модель (вчитель) у меншу, простішу модель (учень), зберігаючи більшу частину продуктивності вчителя при зменшенні вимог до обчислень та пам'яті. Ця техніка дозволяє моделі учня вчитися на результатах вчителя, фіксуючи його знання, не потребуючи такої ж великої архітектури. Модель учня тренується з використанням виходів моделі вчителя замість фактичних міток. Процес дистиляції знань використовує втрати розбіжності для вимірювання різниці між розподілами ймовірностей вчителя та учня для уточнення прогнозів учня. Такі інструменти, як TensorFlow, PyTorch і Hugging Face Transformers [18], надають вбудовані функції для дистиляції знань. Таке зменшення розміру та складності знижує вимоги до пам'яті та обчислень, що робить

його придатним для пристроїв з обмеженими ресурсами. Менша модель споживає менше енергії, ідеально підходить для пристроїв, що живляться від акумулятора, зберігаючи при цьому більшу частину продуктивності оригінальної моделі, що дозволяє розширювати можливості AI на периферійних пристроях.

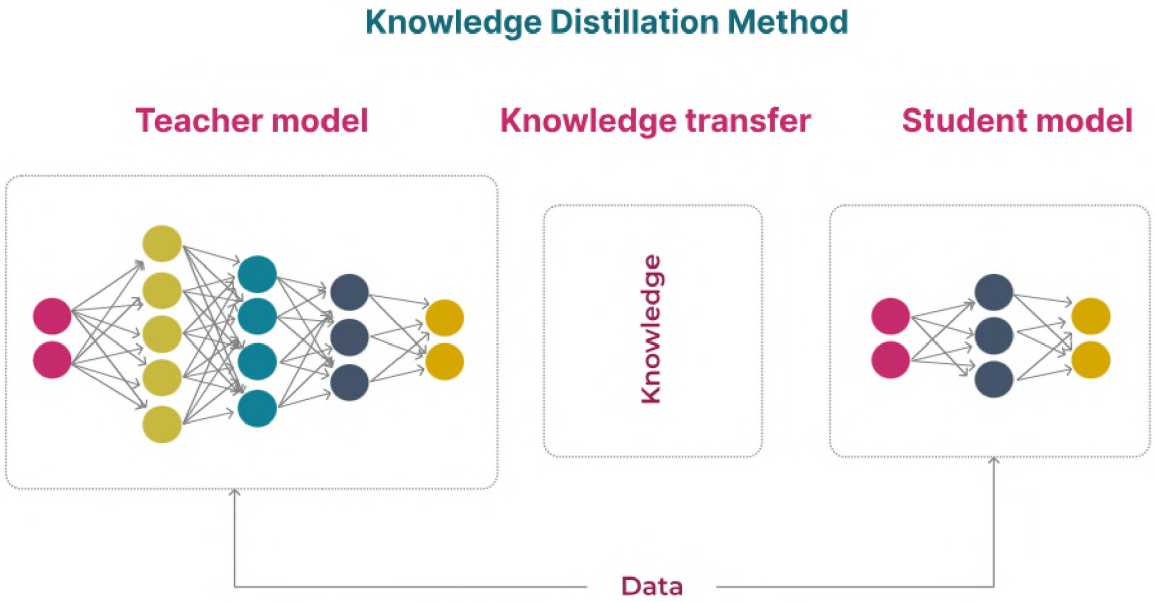


Рисунок 1.5 – Методи дистиляції знань

Сила LLM обумовлена їх величезними розмірами, які часто мають мільярди параметрів. Хоча цей масштаб покращує їхню продуктивність, він також створює проблеми, особливо коли йдеться про адаптацію моделі для конкретних завдань або доменів. Традиційні способи управління LLM, такі як точне налаштування всіх параметрів, є обчислювальними та дорогими, тим самим створюючи значний бар'єр для їх широкого впровадження в реальних додатках. Таким чином, для LLM є проблеми, пов'язані з вимогами до обчислювальних ресурсів та пам'яті для обробки величезної кількості параметрів під час точного налаштування. Налаштування гігантських моделей з відкритим кодом на стандартних системах неможливе без спеціальних методів оптимізації. До таких слід віднести низькорангову адаптацію (LoRA) [11]. Низькорангова адаптація (LoRA) стискає моделі,

розкладаючи вагові матриці на компоненти нижчої розмірності, зменшуючи кількість параметрів, що тренуються, зберігаючи при цьому точність. Це дозволяє проводити ефективне тонке налаштування та адаптацію до конкретного завдання без повної перепідготовки. Інструменти AI інтегрують LLM з LoRA, додаючи матриці низького рангу до архітектури моделі, зменшуючи параметри, що навчаються, і забезпечуючи ефективне тонке налаштування. Такі інструменти, як Loralib [19], спрощують його, роблячи кастомізацію моделі економічно ефективною та ресурсоефективною. Наприклад, LoRA зменшує кількість параметрів, що тренуються, у великих моделях, таких як LLaMA-70B, значно знижуючи використання пам'яті GPU. Це дозволяє LLM ефективно працювати на периферійних пристроях з обмеженими ресурсами, забезпечуючи обробку в режимі реального часу та зменшуючи залежність від хмарної інфраструктури.

Розгортання LLM на периферійних пристроях є значним кроком у створенні передового AI більш доступним і практичним у різних програмах. Проблема полягає в адаптації цих ресурсомістких LLM для роботи в умовах обмеженої обчислювальної потужності, пам'яті та сховища, доступних на периферійному обладнанні. Для досягнення цієї мети потрібні інноваційні методи оптимізації розгортання без шкоди для продуктивності LLM.

Вивід на пристрої. Запуск LLM безпосередньо на периферійних пристроях усуває потребу в передачі даних на віддалені сервери, забезпечуючи негайну реакцію та забезпечуючи функціональність в автономному режимі. Крім того, обробка даних на пристрої зменшує ризик розкриття даних під час передачі, підвищуючи конфіденційність.

В якості прикладів виводу на пристрої слід вказати малі мовні моделі (Small Language Model, SLM), такі як Gemma-2B, Phi-2 і StableLM-3B, успішно працювали на пристроях Android з використанням TensorFlow Lite [20] і MediaPipe [21]. Кількісна оцінка цих моделей зменшила їх розмір і обчислювальні вимоги, що зробило їх придатними для периферійних пристроїв. Після перенесення кількісної моделі на телефон Android і

коригування коду програми, тестування на чіпі Snapdragon 778 показало, що модель Gemma-2B може генерувати відповіді за лічені секунди. Це демонструє, як квантування та вивід на пристрої забезпечують ефективну продуктивність LLM на мобільних пристроях.

Гібридне виведення поєднує периферійні та хмарні ресурси, розподіляючи обчислення моделі для балансу між продуктивністю та обмеженнями ресурсів. Цей підхід дозволяє виконувати ресурсомісткі завдання в хмарі, тоді як завдання, чутливі до затримки, керуються локально на периферійному пристрої.

Розбиття моделі. Цей підхід розділяє LLM на менші сегменти, розподілені між кількома пристроями, підвищуючи ефективність і масштабованість. Це забезпечує розподілені обчислення, балансує навантаження між пристроями та дозволяє проводити незалежну оптимізацію на основі можливостей кожного пристрою. Ця гнучкість підтримує розгортання великих моделей на різних конфігураціях обладнання, навіть на периферійних пристроях з обмеженими ресурсами.

Наприклад, EdgeShard [22] – це фреймворк, який оптимізує розгортання LLM на периферійних пристроях шляхом розподілу сегментів моделей як на периферійні пристрої, так і на хмарні сервери на основі їхніх можливостей. Він використовує адаптивний вибір пристроїв для розподілу шарів відповідно до продуктивності, пам'яті та пропускної здатності. Він включає профілювання в автономному режимі для збору даних під час виконання, оптимізацію планування завдань для мінімізації затримки та кульмінацію спільного висновування, коли шари моделі обробляються паралельно. Тести з моделями Llama2 показали, що EdgeShard зменшує затримку до 50 % і подвоює пропускну здатність, демонструючи свою ефективність і адаптивність при різних умовах мережі і ресурсах.

Підсумовуючи, Edge AI має вирішальне значення для майбутнього LLM, забезпечуючи обробку в режимі реального часу з низькою затримкою, підвищену конфіденційність і ефективну роботу на пристроях з обмеженими

ресурсами. Завдяки інтеграції LLM з периферійними системами зменшується залежність від хмарної інфраструктури, забезпечуючи масштабовані та доступні рішення штучного інтелекту для наступного покоління додатків.

1.4 Локалізація мовних моделей

Запуск LLM, таких як ChatGPT і Claude, зазвичай передбачає надсилання даних на сервери, якими керує OpenAI та інші постачальники моделей AI. Хоча ці служби є безпечними, деякі компанії вважають за краще зберігати свої дані повністю в автономному режимі для більшої конфіденційності.

Як наслідок, локалізація LLM дозволяє отримати низку переваг та застосувань. Локальна версія моделі робить користувача менш залежним від хмарних провайдерів та їх умов використання. Для регулярного використання хмарних LLM потрібні постійні платежі за обчислювальні ресурси. Локальні LLM можна адаптувати для роботи з конкретними даними та вузькоспеціалізованими областями. У віддалених або важкодоступних місцях, де інтернет нестабільний або відсутній, локалізована модель може працювати автономно. Оптимізовані та локалізовані моделі, такі як ті, які можуть працювати на Raspberry Pi, дозволяють використовувати LLM на пристроях з обмеженими обчислювальними ресурсами та низьким енергоспоживанням. автономність та низьке енергоспоживання. При локальній роботі з моделлю скорочуються затримки, які виникають при використанні хмарних сервісів, особливо при високих навантаженнях. Локалізація LLM дозволяє покращити якість обробки малопоширених мов, діалектів або регіональних термінів. Локалізована модель дозволяє краще контролювати і розуміти, як саме вона обробляє дані. аудит та валідацію, а також можливість налаштовувати її так, щоб уникнути упередженості. Локальні LLM дозволяють тестувати та розробляти нові архітектури та

алгоритми NLP. Це особливо актуально для університетів та лабораторій, де можуть бути обмежені ресурси та доступ до хмарних сервісів. Локалізовані LLM можна використовувати як резервний варіант при збої хмарних сервісів.

Локалізація LLM для забезпечення інформаційної безпеки є важливим завданням, особливо в умовах, коли чутливі дані повинні оброблятися всередині локальної інфраструктури. Серед основних аспектів локалізації LLM для безпеки слід виділити наступні. При локальному розгортанні моделі на власних серверах або пристроях, таких як Raspberry Pi, дані не залишають організацію. Це зменшує ризики витоків, що особливо важливо під час роботи з конфіденційною інформацією.

Локальна робота LLM означає, що немає необхідності передавати дані третім сторонам або хмарним провайдерам. Це підвищує довіру користувачів і дозволяє дотримуватись міжнародних стандартів захисту персональних даних, таких як GDPR.

Використання LLM через API від зовнішнього постачальника відкриває можливість атак, таких як перехоплення даних або їх заміна. Локальна модель повністю ізольована від зовнішніх мереж, що підвищує безпеку.

Можливість працювати з LLM без підключення до інтернету робить систему більш стійкою до можливих збоїв та мережних атак. Це важливо для захисту інформації в місцях з обмеженим доступом до Інтернету.

Локалізовані моделі можна адаптувати для вузькоспеціалізованих завдань та конфіденційної інформації. Моделі, розгорнуті на місці, не оновлюватимуться без відома адміністратора, що унеможливлює ненавмисне впровадження вразливостей.

Локалізація моделей дозволяє оптимізувати їх під конкретні завдання та не потребує великих обчислювальних потужностей, особливо якщо модель запускається на енергоефективних пристроях, як Raspberry Pi. Це може бути корисним для невеликих організацій або окремих користувачів, яким потрібні безпечні та доступні рішення для обробки інформації.

Локалізація LLM на пристроях з обмеженими обчислювальними ресурсами (Raspberry Pi), може значно підвищити ефективність використання цих ресурсів. Коли модель локалізована, обробка даних відбувається на пристрої без необхідності надсилати запити на віддалені сервери. Це знижує затримки, особливо в середовищах з обмеженим або нестабільним інтернет-з'єднанням, і дозволяє швидше отримувати відповіді. Локальна обробка знижує залежність від Інтернету, оскільки пристрій не потребує передачі великих обсягів даних на сервер і назад. Це особливо корисно для віддалених або автономних систем, де зв'язок може бути дорогим або обмеженим. Обробка даних на пристрої дозволяє уникнути передачі особистих або конфіденційних даних через мережу, що мінімізує ризики витоку даних та підвищує загальну безпеку системи. Локалізована модель може бути адаптована для енергозбереження та ефективної роботи на пристроях з обмеженими ресурсами. Це дозволяє зменшити загальне споживання енергії, що особливо важливо для портативних та автономних систем. Локалізація моделі дозволяє налаштовувати її під конкретні завдання та умови використання. Наприклад, можна зменшити розмір моделі, адаптувати словник та налаштувати параметри для вирішення конкретних завдань, що зменшить споживання пам'яті та процесорного часу. Локалізовані моделі можуть працювати без підключення до інтернету, що дозволяє використовувати їх в умовах, де немає надійного доступу до мережі, наприклад, у польових умовах для сільського господарства.

Локалізація LLM робить передові технології доступнішими для носіїв менш поширених мов. Більшість сучасних LLM підтримують обмежену кількість мов, часто залишаючи малі та регіональні мови без уваги. Локалізація допомагає подолати цей мовний бар'єр, забезпечуючи рівний доступ до передових інструментів та технологій для всіх груп користувачів. Малі та регіональні мови часто знаходяться під загрозою зникнення, оскільки молодь та організації переходять на більш поширені мови. Локалізація LLM сприяє збереженню цих мов, створюючи попит на їх використання та

підтримку. Це допомагає не лише підтримувати, а й розвивати мови у цифровому просторі, створюючи умови для їхньої актуальності в сучасному суспільстві. Локалізовані моделі здатні краще враховувати культурні та мовні особливості, що підвищує точність та релевантність відповідей. Це дозволяє створювати інтерфейси та системи, які ближче та зрозуміліші користувачам, мінімізуючи ризики непорозуміння чи некоректних інтерпретацій. Безліч проектів, у тому числі освіти, охорони здоров'я, державного управління, потребують підтримки місцевих мов. Локалізовані LLM можуть підвищити якість та доступність послуг для населення, особливо у віддалених чи сільських регіонах, де офіційні мови можуть бути широко поширені. Локалізовані LLM стимулюють дослідження та розробки в галузі малих мов, підвищуючи інтерес до них серед студентів та дослідників. Підтримка локальних мов через сучасні технології також сприяє популяризації цих мов та мотивує нові покоління вивчати їх, що позитивно позначається на їхньому статусі та репутації.

Таким чином, локалізація LLM відкриває доступ до цих технологій та розширює можливості для їх використання, роблячи мовні моделі більш доступними, зручними та адаптованими під реальні завдання, більш незалежними, надійними та енергоефективними, сприяючи їх стійкості та зручності для користувачів у різних умовах.

Висновки до розділу 1

Майбутнє GenAI залежить від даних, охоплює мультимодальність, надає пріоритет кастомізації, бачить великий потенціал у відкритому вихідному коді та визнає необхідність етики. Практичні кейси використання LLM демонструють їх застосування у різних галузях: медицині, освіті, технічній підтримці, маркетингу та ін. У міру зростання обсягу даних підприємства стикаються з проблемами в управлінні своїми системами знань.

багатьом підприємствам важко використовувати можливості AI, необхідні для збереження конкурентоспроможності, оскільки поточні LLM часто технічно та фінансово недосяжні. Новаторським рішенням, яке вирішує ці проблеми, є однобітові LLM (TriLM) використовують високо-квантові представлення – лише до 3-ох значень (наприклад, -1, 0 та 1). Це стиснення зменшує потреби моделі в пам'яті приблизно на 97 %.

Завдяки представленню ваги всього 3-ма значеннями, моделі вимагають набагато менше пам'яті, що дозволяє запускати їх на пристроях з обмеженими ресурсами, наприклад, Raspberry Pi.

Edge AI має вирішальне значення для майбутнього LLM, забезпечуючи обробку в режимі реального часу з низькою затримкою, підвищену конфіденційність і ефективну роботу на пристроях з обмеженими ресурсами. Завдяки інтеграції LLM з периферійними системами зменшується залежність від хмарної інфраструктури, забезпечуючи масштабовані та доступні рішення штучного інтелекту для наступного покоління додатків

При цьому важливу роль відіграє локалізація LLM, яка забезпечує такі переваги: незалежність від зовнішніх сервісів; зниження витрат; налаштування та адаптація під завдання; робота в умовах обмеженого Інтернет; енергоефективні рішення; прискорення реакції та відсутність затримок; підтримка національних мов та діалектів; етичність і прозорість; освітні та наукові дослідження; резервування та відмовостійкість.

РОЗДІЛ 2
РОЗРОБКА МЕТОДИКИ ЛОКАЛІЗАЦІЇ МОВНИХ МОДЕЛЕЙ

2.1 Застосування LoRA до архітектурі Transformer

LLM зайняли унікальну нішу, пропонуючи безпрецедентні можливості за рахунок великих розмірів. Хоча цей масштаб покращує їхню продуктивність, він також створює проблеми, особливо коли йдеться про адаптацію моделі для конкретних завдань або доменів. Традиційні способи управління LLM, такі як точна настройка всіх параметрів, є обчислювально і фінансово витратними, тим самим створюючи значну перешкоду для їх широкого впровадження в реальних додатках. Тому застосовується LoRA (див. п. 1.3). Її суть полягає в тому, щоб підійти до адаптації моделі, не заглиблюючись у тонкощі перенавчання всієї моделі. На відміну від традиційного тонкого налаштування, де кожен параметр можна змінити, LoRA використовує інший шлях. Вона заморожує попередньо навчені ваги моделей та вводить тренувальні матриці розкладання рангів на кожному рівні архітектури Transformer [23]. Такий підхід різко скорочує кількість параметрів для навчання, що дозволяє більш ефективно проводити процес адаптації. При створенні LLM, можна виділити кілька стратегій, які практики використовували протягом багатьох років.

Спочатку основна увага приділялася тонкому налаштуванню заздалегідь навчених моделей – стратегії, яка передбачає всебічну модифікацію параметрів моделі відповідно до конкретного завдання. Однак, у міру зростання розмірів і складності моделей зростали і обчислювальні вимоги цього підходу.

Наступною поширеною стратегією стала тонке налаштування підмножини параметрів – більш стримана версія свого попередника, що певною мірою знижує обчислювальне навантаження. Незважаючи на переваги, тонке налаштування підмножини все одно не встигало за темпами

зростання розмірів LLM. У міру того, як практики наважувалися досліджувати більш ефективні способи, повне тонке налаштування стало жорстким, але корисним підходом.

Ранг матриці дає уявлення про розміри, створювані її стовпцями, які визначаються кількістю унікальних рядків або стовпців, які вона має. Повна матриця рангів – його ранг відповідає меншому числу між рядками або стовпцями. Матриця низького рангу – оскільки ранг значно менший за кількість рядків і стовпців, він охоплює менше ознак.

Тепер LLM охоплюють широке розуміння своєї області. Однак, щоб точно налаштувати їх під конкретні завдання, часто потрібно виділити лише невелику частину цих понять. Саме тут LoRA надає максимальний ефект. Це свідчить про те, що матриця, яка демонструє ці коригування ваги, може бути низькоранговою, що дозволяє охопити менше функцій. LoRA розумно обмежує ранг цієї матриці оновлення, розділяючи її на дві менші рангові матриці. Таким чином, замість зміни всієї вагової матриці змінюється лише її частина, що робить завдання тонкого налаштування більш ефективним.

LoRA допомагає мінімізувати тренувальне навантаження в нейронних мережах за рахунок фокусування на конкретних вагових матрицях (рис. 2.1). В архітектурі Transformer певні вагові матриці пов'язані з механізмом самообслуговування (рис. 2.2), а саме W^q, W^k, W^v і W_0, а також ще дві у модулі Multilayer Perceptron (MLP) [24]. Все починається з попередньо навченої шкали (матриця W_0 розміром $d \times k$).

Замість безпосереднього оновлення всієї матриці W_0, що може бути обчислювально дорогим, метод пропонує підхід декомпозиції низького рангу. Оновлення ΔW до W_0 можна представити як добуток двох матриць: B ($d \times r$) і A ($r \times k$). Ключовим моментом тут є те, що ранг r набагато менший, ніж d і k, що дозволяє отримати більш ефективне з обчислювальної точки зору представлення. У процесі навчання W_0 залишається незмінною. Це називається «заморожуванням» вагів. З іншого боку, A і B – параметри, що піддаються тренуванню.

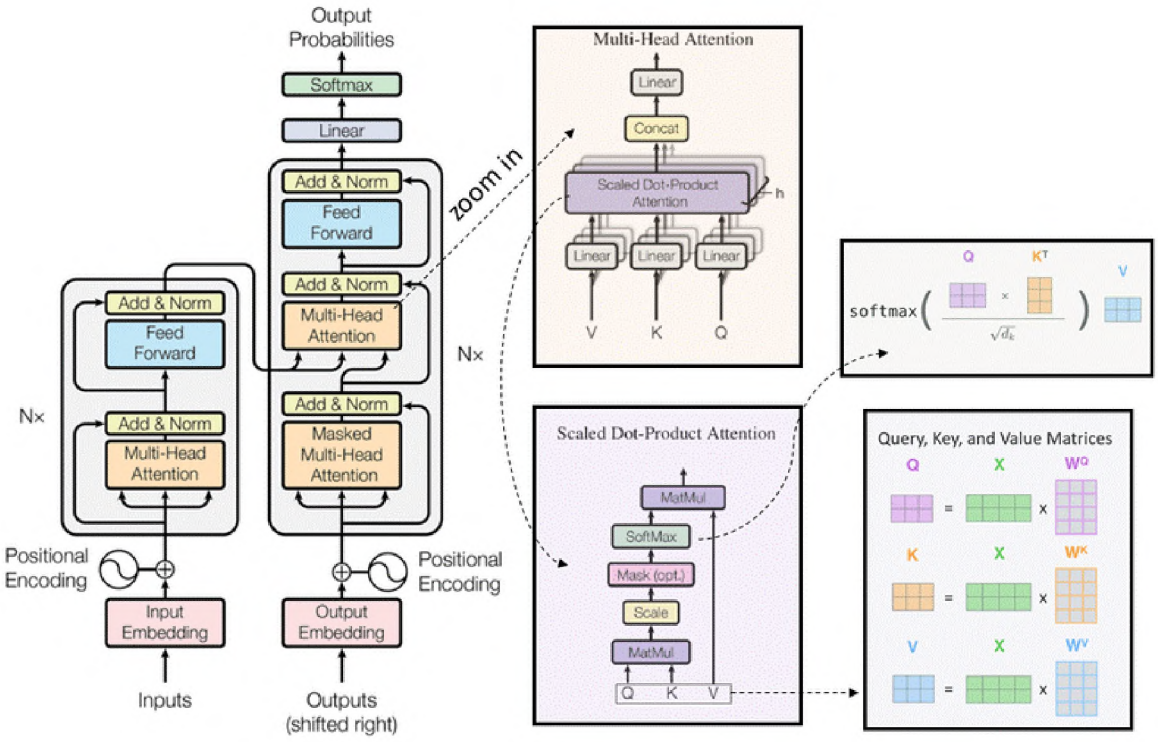


Рисунок 2.1 – Архітектура Transformer

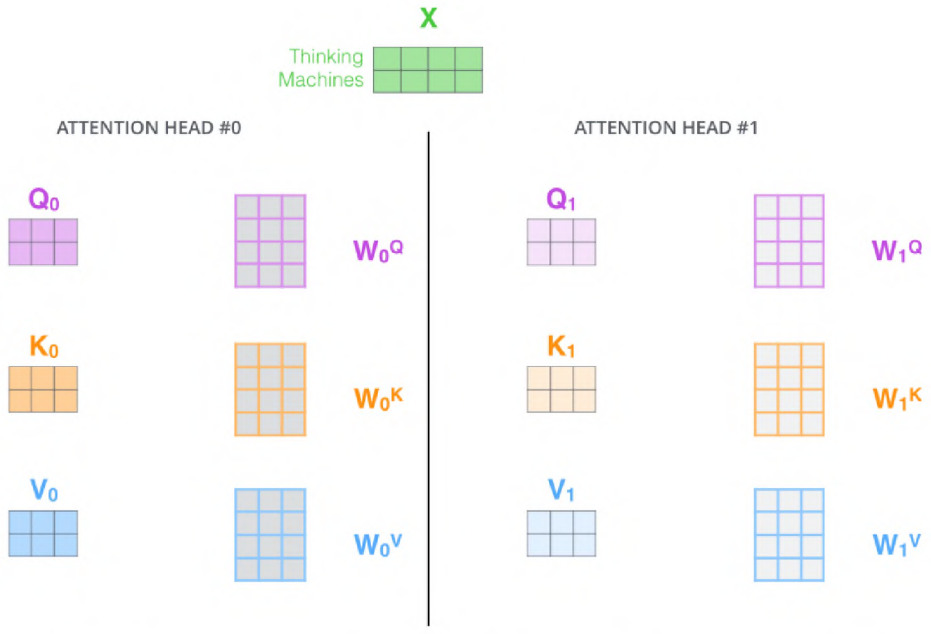


Рисунок 2.2 – Компоненти механізму самообслуговування Transformer

Це означає, що в процесі навчання в матриці A і B вносяться корективи для підвищення продуктивності моделі. Обидва формати W_0 оновлення ΔW (що є добутком B і A) множаться на один і той же вхід (позначається як x). Результати цих множення потім складаються разом:

$$h = W_0 \cdot x + \Delta W \cdot x = W_0 \cdot x + B \cdot A \cdot x . \quad (2.1)$$

В даному випадку, h представляє кінцевий результат після застосування оновлень до вхідних даних x. Цей метод дозволяє більш ефективно оновлювати матрицю великих вагів, представляючи оновлення за допомогою низькорангової декомпозиції, що може бути корисним з точки зору обчислювальної ефективності та використання пам'яті.

Для різних моделей навчання спосіб ініціалізації параметрів може мати значний вплив на ефективність і результативність тренувального процесу. У контексті оновлення вагової матриці за допомогою A та B. Матриця A ініціалується випадковими значеннями Гауса (нормальний розподіл). Сенс його використання полягає в порушенні симетрії: різні нейрони в одному шарі навчаються різним функціям, коли вони мають різну початкову вагу. Матриця B ініціалізується нулями. При цьому оновлення $\Delta W = B \cdot A$ починається з нуля на початку навчання. Це гарантує відсутність різких змін у поведінці моделі на початку, що дозволяє моделі поступово адаптуватися за потреби (матриця B засвоює відповідні цінності під час навчання). Після обчислення оновлення ΔW його вихід масштабується на коефіцієнт $r \cdot \alpha$, в якому α є константою. Масштабування дозволяє контролювати масштаб оновлень. Введення шкали (градація) особливо важливе при зміні рангу r. Наприклад, якщо треба підвищити рейтинг для більшої точності (за допомогою обчислень), масштабування гарантує, що не доведеться налаштовувати багато інших гіперпараметрів у процесі. Це забезпечує певний рівень стійкості моделі. LoRA продемонструвала свій потенціал для ефективної адаптації LLM до конкретних стилів мистецтва за допомогою людей зі спільноти AI. Особливо яскраво це проявилося в адаптації моделі для імітації художнього стилю Грега Рутковскі [25]. Методологія LoRA не тільки втілює значний крок до того, щоб зробити LLM більш доступною, але й підкреслює потенціал для подолання розриву між теоретичними досягненнями та практичними застосуваннями в галузі штучного інтелекту. Зменшуючи обчислювальні перешкоди та сприяючи більш ефективному

процесу адаптації моделі, LoRA має відігравати ключову роль у ширшому впровадженні та розгортанні LLM у реальних сценаріях

Квантована LoRA (Quantized LoRA, QLoRA). Незважаючи на те, що LoRA змінює правила гри стосовно зниження вимог до пам'яті, вона все ще вимагає потужного GPU для завантаження тренувальної моделі. Саме тут на допомогу приходить QLoRA, що поєднує LoRA з квантуванням для більш розумного підходу. Зазвичай, параметри ваги зберігаються в 32-бітовому форматі (fp32), тобто кожен елемент матриці займає 32 біта простору. Далі потрібно уявіти, що можна вмістити ту саму інформацію лише у 8 або навіть 4 біти. Це і є основна ідея QLoRA. Квантування відноситься до процесу перетворення безперервних нескінченних значень у менший набір дискретних скінченних значень.

У контексті LLM це відноситься до процесу перетворення вагів моделей від типів даних з вищою точністю до типів з нижчою точністю. Це відбувається наступним чином. Початкове квантування – LLM квантується до 4 бітів, що значно зменшує обсяг пам'яті. Потім виконується тренування LoRA, але зі стандартною 32-бітовую точністю (fp32). Тепер виникає питання – навіщо повертатися до 32-бітового формату навчання після скорочення до 4-бітового? Для ефективного тренування адаптерів LoRA у fp32 ваги моделі також потрібно повернути до fp32. Це перемикання вперед і назад виконується розумно, крок за кроком, щоб уникнути перевантаження пам'яті GPU. Практична реалізація LoRA виконана в рамках бібліотеки PEFT [26] Для використання QLoRA доступна комбінація бітів і байтів [27], а також PEFT. Крім того, є бібліотека HuggingFace Transformer Gain Learning (TRL) [28], що сприяє контрольованому тонкому налаштуванню завдяки вбудованій підтримці LoRA. Разом, ці три бібліотеки надають необхідний набір інструментів для точного налаштування обраної попередньо навченої моделі, дозволяючи створювати переконливі та послідовні описи продуктів. Після тонкого налаштування за допомогою QLoRA ваги повинні повернутися до високоточного формату, що може призвести до втрати точності та

відсутності оптимізації для прискорення процесу. Запропоноване рішення полягає в тому, щоб згрупувати вагову матрицю на менші сегменти та застосувати квантування та низькорангову адаптацію до кожної групи окремо. Новий метод, названий QA-Laura [29]. Він намагається поєднати переваги квантування та низькорангової адаптації, зберігаючи при цьому ефективність процесу та ефективність моделі для бажаних завдань.

2.2 Інструментарій для локального запуску мовних моделей

Запуск LLM, таких як ChatGPT [30] і Claude [31], передбачає надсилання даних на сервери, якими керує OpenAI та ін. постачальники моделей AI. Хоча ці служби є безпечними, деякі компанії вважають за краще зберігати свої дані повністю в автономному режимі для більшої конфіденційності. Залежно від конкретного випадку використання, можна вибрати кілька off-line моделей LLM. Деякі з цих інструментів абсолютно безкоштовні для особистого та комерційного використання. Іншим треба надіслати запит для використання в бізнесі. Локальний запуск LLM може допомогти розробникам, які хочуть детально зрозуміти їх ефективність і принцип роботи. Локальні LLM можуть запитувати приватні документи та технічні документи, щоб інформація про ці документи не залишала пристрої, які використовуються для запиту до будь-яких хмарних API AI. Локальні LLM корисні в місцях без Інтернету та в місцях, де прийом мережі поганий. Для телемедицини магістратури можуть сортувати документи пацієнтів без необхідності завантажувати їх до будь-якого постачальника API AI через проблеми конфіденційності. Є кілька локальних інструментів LLM для Mac, Windows і Linux.

Першим варіантом є LM Studio [32]. Ця платформа не збирає дані користувачів і не відстежує дії користувачів, коли вони використовують його для запуску локальних LLM. Завдяки цьому всі дані чату клієнта

залишаються на локальному комп'ютері, не передаючи їх серверу AI/ML. Можна надсилати підказки локальним LLM у багатооборотний спосіб, не залишаючи дані підказок на локальному хості. Локальні LLM надають розширені конфігурації для потоків процесора, температури, довжини контексту, налаштувань GPU тощо. В LM Studio забезпечують подібну підтримку та безпеку, як OpenAI або Claude. Вона є безкоштовною для використання та не потребують щомісячної підписки. Для хмарних сервісів, таких як OpenAI, кожен запит API вимагає оплати. Локальні LLM допомагають заощадити гроші, оскільки немає щомісячної підписки. Можна завантажувати та підключатися до LLM у режимі off-line (іноді підключення до хмарної служби може призвести до поганого сигналу та з'єднання). LM Studio може запускати будь-який файл моделі у форматі gguf та підтримує файли gguf від таких постачальників моделей, як Llama 3.1 [33], Phi 3 [34], Mistral [35] і Gemma [36]. Існує також панель пошуку для фільтрації та завантаження конкретних моделей від різних постачальників AI. В цілому, LM Studio надає такі ж функції та можливості, як ChatGPT. Він виконує кілька функцій. Нижче висвітлюються ключові функції LM Studio: налаштування параметрів моделі – дозволяє регулювати температуру, максимальну кількість жетонів, штраф частоти тощо; історія чату – зберігає підказки для подальшого використання; параметри та підказки інтерфейсу користувача – можна навести вказівник миші на інформаційні кнопки, щоб переглянути параметри та умови моделі. LM Studio доступна в операційних системах Linux, Mac і Windows. LM studio перевіряє характеристики комп'ютера, як-от GPU та пам'ять, і звітує про сумісні моделі. Це запобігає завантаженню моделі, яка може не працювати на певній машині. При цьому можна спілкуватися з LLM у форматі багатоходового чату та експериментувати з кількома LLM, завантажуючи їх одночасно. Локальний сервер виводів дозволяє розробникам налаштувати локальний HTTP-сервер, подібний до API OpenAI. Він надає зразки клієнтських запитів Curl і Python. Ця функція допомагає створювати програму AI за допомогою LM Studio для

доступу до конкретної LLM. LM Studio дозволяє розробникам імпортувати бібліотеку OpenAI Python і спрямовувати базову URL-адресу на локальний сервер (localhost). Як наслідок, можна повторно використовувати наявну конфігурацію OpenAI і змінити базову URL-адресу, щоб вона вказувала на локальний хост. Таким чином, LM Studio є безкоштовною для особистого користування та дозволяє розробникам запускати LLM через інтерфейс чату в додатку та ігровий майданчик. Вона забезпечує простий у використанні інтерфейс із фільтрами та підтримує підключення до бібліотеки Python OpenAI без ключа API. Компанії та підприємства можуть використовувати LM Studio за запитом. Однак для цього потрібен M1/M2/M3 Mac або новіший комп'ютер або ПК з Windows із процесором, який підтримує AVX2. Користувачі Intel і AMD можуть використовувати механізм виводів Vulkan у версії 0.2.31.

Наступний варіант – це Jan [37]. По суті, Jan – як версія ChatGPT з відкритим кодом, призначена для роботи в автономному режимі. Він створений спільнотою користувачів із філософією, що належить користувачам. Jan дозволяє запускати такі популярні моделі (Mistral, Llama), на локальному пристрої без підключення до Інтернету, а також можна отримати доступ до віддалених API, таких як OpenAI і Groq [38]. Jan – це електронний додаток із функціями, подібними до LM Studio. Він робить AI відкритим і доступним для всіх, перетворюючи споживчі машини на комп'ютери AI. Оскільки це проект із відкритим вихідним кодом, розробники можуть внести свій внесок у нього та розширити його функціональні можливості: можна запускати бажані моделі AI на пристроях, не підключаючи їх до Інтернет; після завантаження Jan ви отримуєте набір уже встановлених моделей для запуску (є можливість пошуку конкретних моделей); підтримується імпорт моделей із таких джерел, як Hugging Face; безкоштовний, кросплатформний і з відкритим вихідним кодом; можна налаштувати параметри виводу (максимальний маркер, температура, потік, частотний штраф тощо). Усі параметри, використання моделі та

налаштування залишаються локально на вашому комп'ютері. Jan підтримує такі розширення, як TensortRT і Inference Nitro, для налаштування та покращення користувацьких моделей AI. Jan надає зрозумілий і простий інтерфейс для взаємодії з LLM і зберігає всі ваші дані та інформацію про обробку локально. У ньому вже встановлено понад 70 LLM якими можна користуватися. Наявність цих готових до використання моделей полегшує підключення та взаємодію з такими віддаленими API, як OpenAI і Mistral.

Ще один варіант – це Llamafile [39]. Він підтримується Mozilla [40], мета якої – підтримати та зробити AI, з відкритим кодом. Він перетворює LLM у мультиплатформений виконуваний зв'язуваний формат (ELF). Він забезпечує один із найкращих варіантів інтеграції AI, в програми, дозволяючи запускати LLM лише з одним виконуваним файлом. Llamafile призначений для перетворення вагових коефіцієнтів у кілька виконуваних програм, які не потребують інсталяції для роботи на таких архітектурах, як Windows, MacOS, Linux, Intel, ARM, FreeBSD тощо. До ключових характеристик Llamafile відносяться наступні: на відміну від інших інструментів LLM, таких як LM Studio та Jan, Llamafile вимагає лише одного виконуваного файлу для запуску LLM; Llamafile підтримує використання існуючих інструментів моделей, таких як Ollama [41] та LM Studio; можна отримати доступ до популярних LLM із OpenAI, Mistral, Groq тощо. Він також підтримує створення моделей з нуля; можна конвертувати формат файлу багатьох популярних LLM, наприклад, .gguf, у .llamafile за допомогою команди: llamafile-convert mistral-7b.gguf. У порівнянні з іншими локальними програмами LLM, такими як Llama.cpp, Llamafile забезпечує найшвидшу оперативну обробку та кращу продуктивність на ігрових комп'ютерах. Оскільки вона має більшу продуктивність, це чудовий варіант для підсумовування довгого тексту та великих документів. Спільноти ML, такі як Hugging Face, підтримують формат llamafile, що полегшує пошук пов'язаних моделей Llamafile. Він також має чудову спільноту з відкритим кодом, яка розвиває та розширює його далі.

GPT4ALL [42] побудовано на принципах конфіденційності, безпеки та не вимагає доступу до Інтернет. Користувачі можуть встановити його на Mac, Windows і Ubuntu. Порівняно з Jan або LM Studio, GPT4ALL має більше щомісячних завантажень, зірок GitHub та активних користувачів. GPT4All може запускати LLM на основному побутовому апаратному забезпеченні (CPU MAC серії M, GPU AMD і NVIDIA). Його основні характеристики: зберігає конфіденційну інформацію чату та підказки лише на своєму комп'ютері; працює повністю в автономному режимі; дослідження моделей – ця функція дозволяє переглядати та завантажувати різні типи LLM для експериментів (вибір близько 1000 мовних моделей із відкритим вихідним кодом); можна надати локальній LLM доступ до конфіденційних даних за допомогою локальних документів (.pdf і .txt), не залишаючи даних на локальному пристрої та без мережі; надає кілька параметрів налаштування чат-бота (температура, розмір партії, довжина контексту та ін.); у версії Enterprise Edition GPT4ALL надає корпоративний пакет із безпекою, підтримкою та ліцензіями на кожен пристрій, щоб забезпечити локальний штучний інтелект для компаній. За винятком Ollama, GPT4ALL має найбільшу кількість учасників GitHub і близько 250000 активних користувачів щомісяця і порівняно з його конкурентами [43]. Додаток збирає анонімні дані користувачів про аналітику використання та обмін чатами. Однак користувачі можуть вибрати або відмовитися. Використовуючи GPT4ALL, розробники отримують переваги від великої бази користувачів, спільнот GitHub і Discord.

Наступний інструмент Ollama [41]. Він дозволяє легко створювати локальні чат-боти без підключення до API, наприклад OpenAI. Оскільки все працює локально, не потрібно платити за жодну підписку чи виклики API. Ollama дозволяє конвертувати файли моделі .gguf і запускати їх за допомогою команди: ollama run modelname. Ollama має велику колекцію моделей, які можна спробувати на [44], а також підтримує імпорт моделей з PyTorch та декілька платформ даних. Ollama легко інтегрується в вебпрограми та

програми для настільних комп'ютерів, наприклад, Ollama-SwiftUI, HTML UI, Dify.ai тощо. При цьому реалізується мобільна інтеграція. Ollama має понад 200 учасників на GitHub з активними оновленнями. Він має найбільшу кількість учасників і є більш розширюваним серед інших інструментів LLM з відкритим кодом, про які йшлося вище.

LLaMa.cpp [45] – це базова технологія (система логічного виводу), яка підтримує локальні інструменти LLM, такі як Ollama та ін. Llama.cpp підтримує значні вивод LLM з мінімальною конфігурацією та чудовою локальною продуктивністю на різному обладнанні. Він також може працювати в хмарі. LLaMa.cpp має такі основні характеристики: має мінімальне налаштування; дуже добре працює на різному обладнанні локально та в хмарі; підтримує такі популярні та великі LLM, як Mistral 7B, Mixtral MoE, DBRX, Falcon та ін. LLaMa.cpp підтримує інструменти інтерфейсу LLM з відкритим кодом, такі як MindWorkAI/AI-Studio (FSL-1.1-MIT), iohub/collama тощо.

2.3 Реалізація малих мовних моделей

Згідно [46], для локалізації на обмежених за обчислювальною потужністю ресурсах можна використовувати нащадків LLM – малих мовних моделей (Small language model, SLM) [47]. SLM – це моделі GenAI, призначені для обробки та генерації людської мови. Їх називають «маленькими», оскільки вони мають відносно невелику кількість параметрів у порівнянні з LLM, такими як GPT-3 та вище. Це робить їх легшими, ефективнішими та зручнішими. SLM менш здатні обробляти та генерувати текст, оскільки вони мають менше параметрів, на відміну від більших моделей. Це означає, що вони краще справляються з менш складними завданнями, які є більш конкретними, такими як класифікація тексту, аналіз настроїв і базова генерація тексту. Такі моделі гарно підходять для бізнес-

сценаріїв використання, які не потребують складного аналізу. Вони ідеально підходять для кластеризації, тегування або витягування необхідної інформації. Крім розміру SLM оптимально підходять для нішевих, специфічних для конкретної галузі завдань і можуть надати більше експертної, деталізованої інформації. Наприклад, якщо працюєте в такій галузі, як банківська справа, можна «нагодувати» її спеціальною термінологією та перетворити на фінансову модель. LLM, з іншого боку, схожі на універсалів і мають ширший набір даних. Вони оперуватимуть даними з різних дисциплін. Чим більш детальна або галузева потреба потрібна, тим важче може бути отримати точний результат. Будучи експертом у цій галузі, SLM швидше за все, перевершить LLM. Крім того, завдяки компактності, SLM легко та швидко налаштувати не лише на смартфонах і планшетах, але й на периферійних обчислювальних пристроях. Цього не можна сказати про LLM, для розгортання яких потрібні великі обчислювальні ресурси. Нарешті, з точки зору безпеки та конфіденційності, SLM також безпечніші. Оскільки вони працюють локально, користувачі не обмінюються даними із зовнішніми серверами, що знижує ризик витоку конфіденційних даних (LLM більш вразливі до хакерських атак, оскільки вони часто обробляють дані в хмарі).

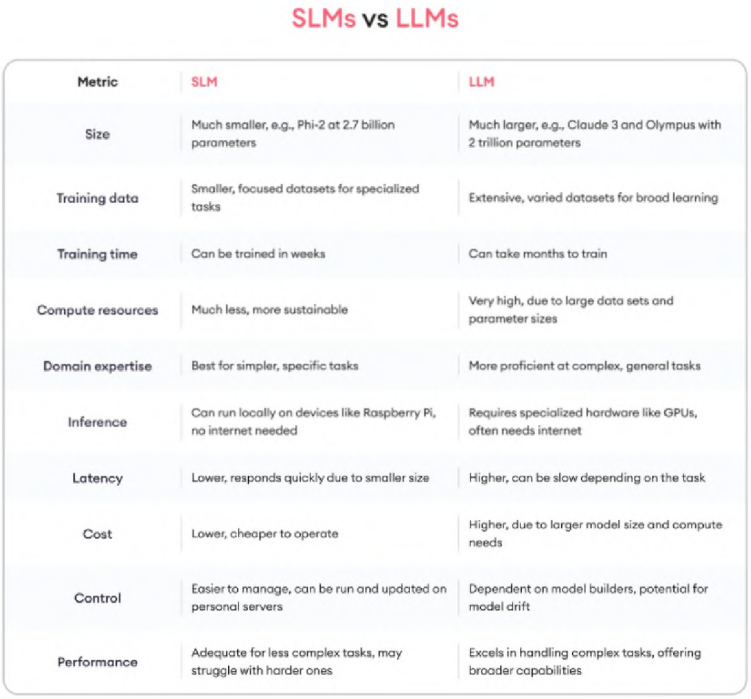
Найкраще в моделях малої мови (SLM) полягає в тому, що вони чудово працюють навіть на простіших апаратних засобах (смартфон або портативний пристрій), а це означає, що можна використовувати їх у багатьох різних налаштуваннях. Вони оптимально підходять, якщо не потрібні всі модні функції LLM. Крім того, варто точно налаштувати SLM, щоб робити саме те, що потрібно, що робить їх дійсно хорошими для конкретних завдань. SLM призначені для розміщення в невеликих приміщеннях, таких як ваш, не жертвуючи занадто великими можливостями розуму. Щоб створити SLM (іншими словами, перетворення LLM на SLM), найбільш часто використовують дві основні техніки тренування: обрізка LLM і дистиляція знань (KD).

Архітектура SLM використовує технологію Transformer. Вона має наступні особливості. Механізми самоуваги [48], які допомагають моделі з'ясувати, які частини даних важливі, а які можна ігнорувати. Це як наявність вбудованого фільтра, який допомагає зосередитися лише на необхідному. Нейронні мережі Feedforward [49] швидко та ефективно обробляють цю відфільтровану інформацію, гарантуючи, що модель не загрузне. Нормалізація шарів [50] використовується для забезпечення безперебійної роботи, забезпечуючи стабільну та надійну роботу виходів моделі. Все стає цікавим, коли навчання SLM закінчується. Виявляється, ці моделі досить гнучкі. SLM передбачає навчання моделі під конкретні завдання. Це може означати трохи більше навчання його на спеціалізованих даних, щоб він міг обробляти конкретні запити, наприклад, розуміти медичні терміни або розпізнавати різні акценти в мові. Тому SLM з тонким налаштуванням [51] заслуговує окремого напряму досліджень. Вивід – це коли модель фактично використовується в роботі, і завдяки всім оптимізованням вона може забезпечити швидкі та надійні результати. Це робить їх оптимальними для програм у реальному часі (переклад мови на ходу або допомога в навігації по меню в новій програмі). Надалі потрібно визначитись коли варто вибирати LLM або SLM? Для цього варто проаналізувати кілька метрик (рис. 2.3).

Розмір (Size). LLM, такі як Claude 3 [31] і Olympus [52], мають близько 2 трильйонів параметрів. Тим часом менші моделі, такі як Phi-2, мають лише 2,7 мільярда параметрів, продемонструвала сильні навички в таких областях, як математика і кодування, іноді навіть перевершуючи моделі, які набагато більші. Phi-2 показала кращі результати, ніж модель Llama-2-70B, у завданнях, які вимагають багатоетапного обґрунтування, показуючи, що менші моделі все ще можуть показувати відмінні результати.

Дані тренінгу (Training data). LLM, такі як GPT-4, потребують широкого спектру даних, від книг до веб-сайтів, для створення детального та деталізованого тексту. З іншого боку, SLM, такі як Phi-2, зосереджуються на

високоякісних, конкретних даних, включаючи 1,4 трильйона токенів як із синтетичних наборів даних, так і з вибраного вебконтенту.



SLMs vs LLMs

Metric	SLM	LLM
Size	Much smaller, e.g., Phi-2 at 2.7 billion parameters	Much larger, e.g., Claude 3 and Olympus with 2 trillion parameters
Training data	Smaller, focused datasets for specialized tasks	Extensive, varied datasets for broad learning
Training time	Can be trained in weeks	Can take months to train
Compute resources	Much less, more sustainable	Very high, due to large data sets and parameter sizes
Domain expertise	Best for simpler, specific tasks	More proficient at complex, general tasks
Inference	Can run locally on devices like Raspberry Pi, no internet needed	Requires specialized hardware like GPUs, often needs internet
Latency	Lower, responds quickly due to smaller size	Higher, can be slow depending on the task
Cost	Lower, cheaper to operate	Higher, due to larger model size and compute needs
Control	Easier to manage, can be run and updated on personal servers	Dependent on model builders, potential for model drift
Performance	Adequate for less complex tasks, may struggle with harder ones	Excels in handling complex tasks, offering broader capabilities

Рисунок 2.3 – Метрики порівняння LLM або SLM [46]

Час навчання (Training time). Навчання великої моделі, такої як GPT-3, може зайняти кілька місяців і вимагає великої обчислювальної потужності, зазвичай із залученням багатьох потужних графічних процесорів. Phi-2, однак, був навчений всього за 14 днів на 96 GPU A100. Це свідчить про те, що SLM можна розробляти набагато швидше, що вигідно для організацій, яким потрібно швидко впроваджувати ітерації своїх моделей.

Обчислювальна потужність та ресурси (Computing power and resources). LLM, такі як GPT-4, вимагають значної потужності та пам'яті комп'ютера, що може зробити їх дорогим у використанні. Однак, SLM можуть ефективно працювати на стандартному обладнанні, що робить їх доступнішими для ширшого спектру застосувань і простішими для вашого бюджету.

Знання (Proficiency). У той час як LLM добре справляються з широким спектром складних завдань – від творчого письма до детального аналізу або перекладу мов – SLM особливо добре справляються з конкретними

завданнями, такими як кодування та міркування. Фактично, Phi-2 досяг найвищого балу 53,7 балів за тестом HumanEval для кодування, перевершивши багато більших моделей.

Адаптації (Adaptation). Адаптація LLM до конкретних потреб може вимагати значних зусиль і часу. Навпаки, SLM можуть бути швидко налаштовані для конкретних завдань (наприклад, аналіз настроїв), що робить їх більш гнучкими та простішими в налаштуванні.

Вивід (Inference). Для роботи великих моделей потрібне потужне обладнання та часто хмарні сервіси, а це означає, що вони покладаються на підключення до Інтернету. Phi-2 досить компактний для роботи на невеликих пристроях, таких як Raspberry Pi або навіть смартфон, пропонуючи більшу гнучкість, оскільки йому не потрібен доступ до Інтернету.

Затримки (Latency). Якщо використовувати велику модель для чогось на кшталт голосового помічника, ви могли помітити затримку. SLM, будучи меншими, можуть обробляти запити набагато швидше, що покращує взаємодію з користувачем у програмах реального часу.

Вартість (Cost). Запуск великих моделей може бути дорогим, оскільки вони потребують багато обчислювальних ресурсів. Оскільки SLM не потребують стільки енергії, вони дешевші в експлуатації, що може бути серйозною перевагою для організацій, які хочуть заощадити на витратах.

Контроль (Control). Використання великих моделей означає покладатися на їх розробників для оновлень, що може призвести до таких проблем, як дрейф моделі. За допомогою Phi-2 та ін. SLM можна запускати їх на власних серверах, точно налаштовувати їх під свої потреби та підтримувати їх стабільним протягом тривалого часу.

Якість даних дійсно має значення, особливо для моделей малої мови (SLM). Оскільки ці моделі не такі великі або складні, як великі, вони значною мірою залежать від якості даних, на яких вони навчені для хорошої роботи. Наприклад, датасет Microsoft TinyStories спеціально розроблений для написання дитячих історій та використовує лише близько 3000 слів. Оскільки

дані настільки сфокусовані та чисті, SLM, навчені на них, насправді можуть писати досить хороші історії, які мають сенс і дотримуються правильної граматики. Це яскравий приклад того, що менше – це більше, коли менше – це дійсно добре. Ця ідея актуальна і в інших спеціалізованих напрямках. Якщо працювати з юридичними текстами, модель, навчена на множині юридичних документів, впорається набагато краще, ніж та, яка вивчалася на випадкових сторінка Інтернет. Те ж саме стосується і охорони здоров'я – моделі, навчені на точній медичній інформації, дійсно можуть допомогти лікарям приймати кращі рішення, оскільки вони отримують пропозиції, які ґрунтуються на надійних даних. Можливість швидко адаптувати ці моделі до нових завдань є однією з їхніх великих переваг. Наприклад, у компанії є SLM, який веде чат служби підтримки клієнтів. Якщо він знадобиться для вирішення питань про новий продукт, вони можуть зробити це відносно легко, якщо модель була навчена на гнучких, високоякісних даних. Таким чином, ті дані, на яких тренуються SLM, можуть їх зробити або зламати. Ось чому будь-хто, хто їх використовує, повинен переконатися, що вони «годують» GenAI хорошими матеріалами – не просто великою кількістю, а високоякісними, добре підібраними даними, які відповідають поставленому завданню. Ось чому користувачі повинні довіряти створення своїх даних надійним експертам. Правильне отримання SLM залежить від якості ваших тренувальних даних. Тонке налаштування насправді полягає в удосконаленні здібностей вашої моделі для конкретних завдань.

2.4 Підвищення продуктивності периферійних пристроїв за допомогою AI-акселераторів

Сучасні моделі AI вимагають значних обчислювальних ресурсів, що тривалий час обмежувало їх застосування переважно потужними серверними рішеннями та хмарними платформами. Однак, завдяки розвитку технологій

периферійних пристроїв та появі спеціалізованих прискорювачів для AI, обробка даних та виконання складних моделей стало можливим навіть на пристроях з обмеженими ресурсами. Це відкриває нові можливості для реалізації AI-рішень на периферії, у тому числі на IoT-пристроях, мобільних платформах, а також промислових та побутових пристроях [53].

AI-акселератор (чіп AI, процесор глибокого навчання або блок нейронної обробки – NPU), – це апаратний прискорювач, створений для прискорення нейронних мереж, глибокого та машинного навчання. У міру того, як технологія AI розширюється, AI-акселератори мають вирішальне значення для обробки великих обсягів даних, необхідних для запуску додатків AI. Наразі сценарії використання AI-акселератори охоплюють смартфони, ПК, робототехніку, автономні транспортні засоби, IoT, периферійні обчислення та ін. Без таких акселераторів як GPU, FPGA і ASIC для прискорення глибокого навчання, прориви в галузі AI, як-от ChatGPT, зайняли б набагато більше часу та були б дорожчими. AI-акселератор широко використовуються деякими з найбільших світових компаній, включаючи Apple, Google, IBM, Intel і Microsoft та ін.

Завдяки своєму унікальному дизайну та спеціалізованому обладнанню AI-акселератори значно підвищують продуктивність обробки AI порівняно зі своїми попередниками. Спеціально створені функції дозволяють вирішувати складні алгоритми AI зі швидкістю, яка значно перевершує чіпи загального призначення. Деякі AI-акселератори призначені для певної мети, тоді як інші мають більш загальний функціонал. Наприклад, NPU – це прискорювачі AI, що створені спеціально для глибокого навчання, тоді як GPU – це прискорювачі зі AI, призначені для обробки відео та зображень.

Основним завданням AI-акселераторів є, насамперед, вирішення передових алгоритмів, що пов'язані з AI. Вони можуть вирішувати швидко і точно багато алгоритмів одночасно. Ці відбувається завдяки унікальному способу розгортання обчислювальних ресурсів, в першу чергу за допомогою паралельної обробки, унікальної архітектури пам'яті та функції, відомої як

знижена точність [54]. Сучасні AI-акселератори призначені для вирішення великих, складних проблем шляхом поділу на менші та одночасного їх вирішення, експоненціально збільшуючи їхню швидкість. Жодна інша функція не підвищує продуктивність AI-акселераторів так, як його здатність виконувати багато обчислень одночасно, завдання, відоме як паралельна обробка. На відміну від інших чіпів, вони можуть виконувати завдання за хвилини, секунди, навіть мілісекунди, на які раніше йшли години і навіть дні. Ця можливість робить їх незамінними для технологій AI, які покладаються на обробку даних у реальному часі, таких як периферійні обчислення. Через велику кількість складних алгоритмів у процесах машинного та глибокого навчання, AI-акселератори мають вирішальне значення для розвитку як технології, так і її застосування. Щоб заощадити енергію, вони можуть використовувати функцію, відому як арифметика зниженої точності. Нейронні мережі все ще мають високу функціональність, використовуючи 16-бітові або навіть 8-бітові числа з плаваючою комою, замість 32 бітів у вагах, які використовують більш універсальні чіпи. Це означає, що вони можуть досягти вищої швидкості обробки при менших витратах енергії без шкоди для точності. Спосіб, яким дані переміщуються з одного місця в інше в AI-акселератори, має вирішальне значення для оптимізації робочих навантажень AI. Прискорювачі використовують інші архітектури пам'яті, ніж чіпи загального призначення, що дозволяє їм досягати менших затримок і кращої пропускної здатності. Ці спеціалізовані конструктивні особливості, включаючи кеш-пам'ять на кристалі та пам'ять з високою пропускною здатністю, що є основою для прискорення обробки великих наборів даних, необхідних для високопродуктивних робочих навантажень AI.

На даний час, AI-акселератори поділяються на дві архітектури залежно від їх функцій: прискорювачі для центрів обробки (ЦОД) даних та прискорювачі для фреймворків периферійних обчислень. AI-акселератори ЦОД вимагають високомасштабованої архітектури та великих чіпів, таких як Wafer-Scale Engine (WSE), що створений Cerebras для систем глибокого

навчання, тоді як AI-акселератори, створені для екосистем периферійних обчислень, більше зосереджені на енергоефективності та здатності забезпечувати результати, близькі до реального часу. Інтеграція в масштабі слотів, або WSI, – це процес створення надзвичайно великих мереж чіпів AI в єдиний, «супер» чіп для зниження вартості та прискорення продуктивності моделей глибокого навчання. Найпопулярнішою інтеграцією в масштабі слотів є мережа чіпів WSE-3 від к. Cerebras і побудована за 5-нм техпроцесом TSMC, який на даний момент є найшвидшим AI-акселератором у світі.

NPU, або нейронні процесори, – це прискорювачі AI для глибокого навчання та нейронних мереж, а також відповідають вимогам щодо обробки даних, які унікальні для цих робочих навантажень. NPU можуть обробляти великі обсяги даних швидше, ніж інші чіпи. Вони можуть виконувати широкий спектр завдань AI, пов'язаних із машинним навчанням, таких як розпізнавання зображень і нейронні мережі, що стоять за популярними моделями AI та ML, такими як ChatGPT.

GPU – електронні схеми, створені для підвищення продуктивності комп'ютерної графіки та обробки зображень, що використовуються в різних пристроях, включаючи відеокарти, материнські плати та мобільні телефони. Однак, завдяки можливостям паралельної обробки, вони також все частіше використовуються в навчанні моделей AI. Одним із популярних методів є підключення багатьох GPU до однієї системи AI, щоб збільшити обчислювальну потужність цієї системи.

FPGA – це пристрої з широкими можливостями налаштування, які залежать від спеціалізованих знань для перепрограмування для конкретної мети. На відміну від інших AI-акселераторів, FPGA мають унікальний дизайн, який підходить для конкретної функції, часто пов'язаною з обробкою даних у режимі реального часу. FPGA можна перепрограмувати на апаратному рівні, що забезпечує набагато вищий рівень налаштування. Поширені програми FPGA включають аерокосмічну промисловість, IoT і бездротові мережі.

Інтегральні схеми для конкретного застосування (ASIC) як AI-акселератори, які були розроблені з певною метою або робочим навантаженням, наприклад, глибоке навчання у випадку з акселератором WSE-3 ASICs (к. Cerebras). На відміну від FPGA, ASIC не можна перепрограмувати, але оскільки вони сконструйовані з єдиною метою, вони, зазвичай, перевершують інші, більш універсальні прискорювачі. Одним із прикладів цього є Tensor Processing Unit (TPU) від Google, розроблений для ML нейронних мереж з використанням власного програмного забезпечення – Google TensorFlow.

В цілому, завдяки швидкості та масштабованості на високому рівні в індустрії швидкоплинних технологій AI, AI-акселератори стали незамінними, допомагаючи компаніям впроваджувати масштабні інновації та швидше виводити на ринок нові моделі AI.

AI-акселератори перевершують свої старші аналоги за 3-ма важливими параметрами: швидкість, ефективність і дизайн. Тобто, вони набагато швидші за традиційні процесори через їх значно нижчу затримку, що є показником затримок у системі. Низька затримка особливо важлива при розробці додатків AI в галузі медицини та автономних транспортних засобів, де затримки в секунди, навіть мілісекунди, є небезпечними. AI-акселератори можуть бути 100…1000 разів ефективнішими, ніж інші, більш стандартні обчислювальні системи. Як AI-акселератори ЦОД, так і менші (в периферійних пристроях), споживають менше енергії та розсіюють меншу кількість тепла, ніж їхні старіші аналоги. AI-акселератори мають так звану гетерогенну архітектуру, що дозволяє кільком процесорам підтримувати окремі завдання – здатність, яка збільшує продуктивність обчислень до рівня, необхідного для додатків AI.

Однак, при орієнтації на використанні AI-акселераторів варто враховувати наступні чинники, які перешкоджатимуть інноваціям. По-перше, 60 % світових напівпровідників і 90 % її передових чіпів (включаючи AI-акселераторів) виробляються на острові Тайвань. Крім того, найбільша у

світі компанія з виробництва апаратного та програмного забезпечення AI – Nvidia, покладається майже виключно на одну компанію – Taiwan Semiconductor Manufacturing Corporation (TSMC) у своїх AI-акселераторів. Моделі AI розвиваються швидше, ніж дизайн AI-акселератора. Сучасні потужні моделі AI вимагають більше обчислювальної потужності, ніж можуть впоратися багато AI-акселератора, а темпи інновацій у дизайні мікросхем не встигають за інноваціями, що відбуваються в моделях AI. Компанії досліджують такі сфери, як обчислення в пам'яті та продуктивність і виробництво на основі алгоритмів штучного інтелекту, щоб підвищити ефективність, але вони не рухаються так швидко, як зростання попиту на обчислення додатків на основі AI. Тобто, AI-акселератори потребують більше енергії, ніж дозволяє їх розмір. При цьому, вони невеликі, більшість з них вимірюються в міліметрах, а найбільший у світі має розмір лише з iPad, що ускладнює спрямування кількості енергії, необхідної для їх живлення, у такий невеликий простір. Це стає дедалі складнішим, оскільки останніми роками вимоги до обчислень через робочі навантаження AI зросли. Найближчим часом необхідно буде досягти прогресу в архітектурах мережі доставки енергії (PDN), що стоять за AI-акселераторами, інакше це почне впливати на їхню продуктивність.

На даний час, слід виділити кілька базових напрямів застосування AI-акселераторів. Вони можуть фіксувати та обробляти дані майже в режимі реального часу, що робить їх критично важливими для розробки безпілотних автомобілів, дронів та інших автономних транспортних засобів. Їхні можливості паралельної обробки не мають собі рівних, що дозволяє їм обробляти та інтерпретувати дані з камер і датчиків, а також обробляти їх, щоб транспортні засоби могли реагувати на навколишнє середовище. Edge AI дозволяє використовувати можливості AI та AI-акселератор для виконання завдань ML на периферії, а не переміщувати дані до ЦОД для обробки. Це зменшує затримку та підвищує енергоефективність у багатьох моделях AI. LLM залежать від AI-акселераторів, які допомагають їм розвивати свою

унікальну здатність розуміти та генерувати природну мову. Паралельна обробка AI-акселераторів допомагає прискорити процеси в нейронних мережах, оптимізуючи продуктивність передових додатків AI, таких як GenAI і чат-боти. AI-акселератор мають вирішальне значення для розвитку галузі робототехніки завдяки своїм можливостям ML та CV. Оскільки робототехніка з AI розробляється для різних завдань, від особистих супутників до хірургічних інструментів, AI-акселератори продовжуватимуть відігравати вирішальну роль у розвитку їхніх здібностей виявляти та реагувати на середовище з такою ж швидкістю та точністю, як й людина.

2.5 Вибір апаратної платформи для локалізації малих мовних моделей

Локалізація SLM на малопотужних пристроях є важливою темою в галузі NLP, особливо в умовах, коли потрібно забезпечити автономну роботу в пристроях з обмеженими обчислювальними ресурсами. У контексті локалізації мовних моделей часто обирають платформу Raspberry Pi [55], яка поєднує в собі доступність, гнучкість та доволі потужні можливості обробки. Це робить її привабливим вибором для дослідників та розробників, які працюють над впровадженням мовних моделей у пристрої IoT, автономні системи та ін. вбудовані додатки. Платформа Raspberry Pi стала популярною через свою доступність, компактність і гнучкість у використанні. Сучасні версії платформи, зокрема Raspberry Pi 5 (рис. 2.4), оснащені процесорами ARM із достатньою обчислювальною потужністю, що дозволяє виконувати ресурсоємні завдання, включно з обробкою нейронних мереж невеликого розміру. Raspberry Pi значно розширила свої можливості AI завдяки співпраці з такими компаніями, як Hailo та Sony, пропонуючи рішення, які інтегрують прискорення AI безпосередньо в їхнє обладнання. Наприклад, для Raspberry

Pi 5 створена нова версія додаткової плати – Raspberry Pi M.2 HAT+. Вона містить з модулем AI-акселератора – Hailo-8 або Hailo-8L, що забезпечує продуктивність виведення 13 або 26 тераопераций в секунду (TOPS). Обидві моделі розроблені для підтримки передових додатків AI, включаючи обробку зображень в реальному часі і прискорення нейронних мереж, що робить їх підходящими для широкого спектру застосувань.

Рисунок 2.4 – Мікрокомп'ютер Raspberry Pi 5

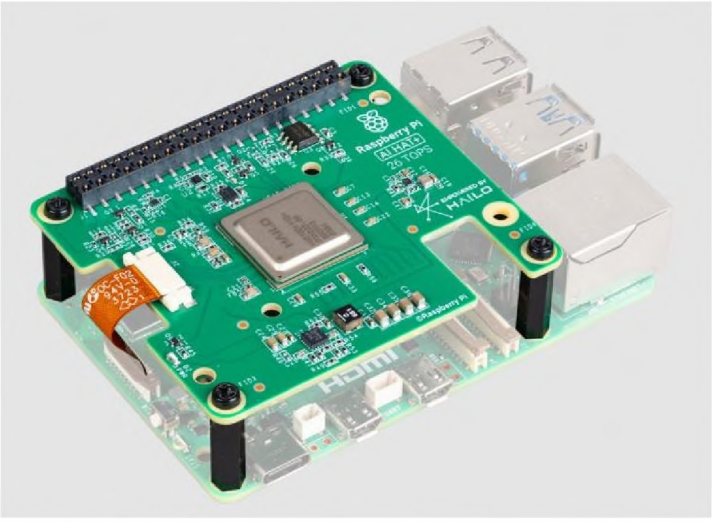


Рисунок 2.5 – Плата Raspberry Pi M.2 HAT+ з AI-акселератором Hailo-8

Hailo-8 – це високопродуктивний AI-акселератор, розроблений для ефективного виконання завдань глибокого навчання на пристроях периферії. До його основні характеристик відносяться такі показники: обчислювальна потужність – 26 TOPS; типове споживання енергії становить 2,5 Вт, що

робить його одним із найефективніших рішень у своєму класі; модуль доступний у форматі M.2 з ключами M, B+M та A+E, що забезпечує гнучкість інтеграції в різні системи; підтримує PCIe Gen-3.0 з 2 або 4 лініями, залежно від моделі, забезпечуючи високу пропускну здатність для передачі даних; сумісний та підтримує фреймворки з TensorFlow, TensorFlow Lite, ONNX, Keras та PyTorch, що спрощує розробку та розгортання AI-моделей; працює під управлінням ОС Linux та Windows, що розширює можливості застосування в різних середовищах; робочий температурний діапазон: від -40° до +85°C, що дозволяє використовувати акселератор в умовах з екстремальними температурами. Тобто, Hailo-8 забезпечує реальну обробку AI-завдань з низькою затримкою та високою ефективністю на пристроях периферії, що робить його оптимальним вибором для широкого спектру застосувань. Крім цього, аксесуара на платформі пропонується відеокамера з вбудованими можливостями AI-обробки – IMX500 (рис. 2.6).

Рисунок 2.6 – AI Camera IMX500

Це дозволяє створювати рішення на основі AI без необхідності використання додаткових прискорювачів або GPU. AI Camera сумісна з усіма одноплатними комп'ютерами Raspberry Pi. Також є пристрів – Picamera3, що підтримує завдання постобробки на основі AI, дозволяючи користувачам відносно легко реалізовувати складні додатки типу «CV+AI» у економічно

ефективному форм-факторі. Крім того, платформа підтримує широкий спектр ОС, включно з легкими версіями Linux, такими як Raspbian, що спрощує налаштування середовища для локалізації SLM.

В цілому, до переваг запропонованого підходу варто віднести наступні положення. Raspberry Pi є недорогим рішенням у порівнянні з іншими спеціалізованими пристроями для локалізації мовних моделей. Це дозволяє впроваджувати SLM у широкому спектрі пристроїв без значних фінансових витрат. Малий розмір Raspberry Pi дозволяє легко інтегрувати її в портативні пристрої, сенсори, а також інші IoT-рішення. Це важливо для створення мобільних автономних систем із підтримкою обробки природної мови. Raspberry Pi підтримує різноманітні інтерфейси для підключення до мережі, включно з Wi-Fi та Bluetooth, що дозволяє обмінюватися даними з іншими пристроями або хмарними сервісами. Сучасні варіанти (Raspberry Pi 5), підтримують апаратне прискорення для обчислювальних операцій, що може бути використане для оптимізації обробки нейронних мереж. Це особливо важливо для зменшення затримок при виконанні обчислень та підвищення ефективності роботи моделі. Таким чином, застосування Raspberry Pi 5 як апаратної платформи для локалізації SLM дозволяє розробникам створювати компактні, енергоефективні та доступні рішення для автономної роботи мовних моделей. Використання цієї платформи особливо перспективне для таких додатків, як голосові асистенти, персональні перекладачі та пристрої моніторингу стану навколишнього середовища, де NLP здійснюється на пристрої, без необхідності звертатися до хмари.

Висновки до розділу 2

Щоб забезпечити конфіденційність та в автономний режим роботи виконується локалізація LLM. Для цього можна використовувати певний інструментарій, наприклад: LM Studio, Jan, Llamafile, GPT4ALL, Ollama,

LLaMa.cpp. LLM мають значний функціонал за рахунок великих розмірів. Однак такий масштаб створює проблеми, особливо коли йдеться про адаптацію моделі для конкретних завдань або наявних обчислювальних ресурсів.

З метою реалізації відповідності параметрів мовної моделі до зазначених ресурсів периферійних пристроїв здійснюється перехід від LLM до SLM. Останні забезпечують аналогічні результати з набагато меншими ресурсами. Незважаючи на обмежені ресурси, поєднання сучасних методів оптимізації та апаратних можливостей дозволяє розробляти ефективні та енергоощадні рішення для обробки природної мови на вбудованих пристроях.

При локалізації мовних моделей доцільно використовувати LoRA, що заморожує попередньо навчені ваги моделі і вводить матриці розкладання рангу на кожен рівень архітектури Transformer. Цей підхід радикально скорочує кількість навчальних параметрів, забезпечуючи ефективніший процес адаптації. На даний час, крім класичної LoRA можна використовувати QLoRA та QA-LoRA.

В ході проведених досліджень зроблений висновок, що перспективною платформою для локалізації SLM на периферійних обчисленнях є Raspberry Pi. Завдяки своїй доступності, гнучкості та достатній обчислювальній потужності Мікрокомп'ютери останньої генерації цієї платформи (Raspberry Pi 5) дозволяють їх використовувати для розгортання SLM. В даному контексті вони забезпечують низьку вартість рішень, компактність і мобільність, розширені можливості підключення, підтримку апаратного прискорення, в тому числі за допомогою AI-акселераторів.

РОЗДІЛ 3
РЕКОМЕНДАЦІЇ ЩОДО ВИКОРИСТАННЯ МЕТОДИКИ ЛОКАЛІЗАЦІЇ МОВНИХ МОДЕЛЕЙ

3.1 Використання платформи Ollama

Згідно п. 2.2, для локалізації SLM будем використовувати Ollama. Дана платформа спрощує процес розгортання та запуску цих моделей локально, полегшуючи розробникам інтеграцію можливостей AI у свої програми. Це є одним з найкращих рішень для запуску локальних мовних моделей на периферійних пристроях на базі Raspberry Pi без необхідності налаштовувати все з нуля, а також надає REST API для виведення моделі, тобто можна залишити його працюючим на Raspberry Pi і викликати його з інших програм та пристроїв [56]. З точки зору технічних вимог, доцільно вибрати Raspberry Pi 5 з ОЗП 8 ГБ, який підходить для моделей 7B, SD-карту ≥ 16 ГБ. Чим більший розмір, тим більше моделей ви можете розмістити. Чи на ньому завантажена відповідна ОС, наприклад Raspbian Bookworm або Ubuntu. По суті, будь-який пристрій потужніший, ніж Raspberry Pi, працює під керуванням ОС Linux і має аналогічний обсяг пам'яті, має бути здатним запускати Ollama. Існує також вебінтерфейс Ollama (Ollama Web UI) [57], який без проблем працює з Ollama для тих, хто побоюється інтерфейсів командного рядка. Він нагадає локальний інтерфейс ChatGPT. Разом, ці дві частини ПЗ open source забезпечують найкращий досвід LLM на локальному хостингу. І Ollama, і Ollama Web UI також підтримують VLM, наприклад, LLaVA, що відкриває ще більше можливостей даного варіанта використання GenAI. Таким чином, для інсталяції даного ПЗ потрібно виконати команду (рис. 3.1): curl https://ollama.ai/install.sh | sh. Як сказано у виведеній на екран інформації, треба перейти до 0.0.0.0:11434 для переконання, що Ollama працює. Повідомлення «УВАГА: графічний процесор NVIDIA не виявлено» є нормальним явищем. Ollama працюватиме в режимі лише CPU», оскільки

використовуємо Raspberry Pi. Для оновлення Ollama до актуальної версії можна скористатись ресурсом репозиторія [58]. На Raspberry Pi об'ємом 8 ГБ моделі розміром більше ніж 7B не підходять, тому в [57] треба вибирати моделі за цим критерієм.

```
admin@raspberrypi:~ $ curl https://ollama.ai/install.sh | sh
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  6881    0  6881    0     0   6938      0 --:--:-- --:--:-- --:--:--  6936>>> Downloading ollama...
100  8354    0  8354    0     0   7924      0 --:--:--  0:00:01 --:--:--  7925
######################################################################## 100.0%##O#-#
######################################################################## 100.0%
>>> Installing ollama to /usr/local/bin...
>>> Creating ollama user...
>>> Adding ollama user to render group...
>>> Adding current user to ollama group...
>>> Creating ollama systemd service...
>>> Enabling and starting ollama service...
Created symlink /etc/systemd/system/default.target.wants/ollama.service → /etc/systemd/system/ollama.ser
vice.
>>> The Ollama API is now available at 0.0.0.0:11434.
>>> Install complete. Run "ollama" from the command line.
WARNING: No NVIDIA GPU detected. Ollama will run in CPU-only mode.
admin@raspberrypi:~ $
```

Рисунок 3.1 – Запуск Ollama на Raspberry Pi 5

Скористаємося SLM Phi-3 обсягом 2.7 B від к. Microsoft, запуск якої виконаємо за допомогою команди: `ollama run phi` (рис. 3.2). Приклад взаємодії з нею наведений на рис. 3.3.

```
admin@raspberrypi:~ $ ollama run phi
pulling manifest
pulling 04778965089b... 100% ▕████████████████████████████████▏ 1.6 GB
pulling 7908abcab772... 100% ▕████████████████████████████████▏ 1.0 KB
pulling 774a15e6f1e5... 100% ▕████████████████████████████████▏   77 B
pulling 3188becd6bae... 100% ▕████████████████████████████████▏  132 B
pulling 0b8127ddf5ee... 100% ▕████████████████████████████████▏   42 B
pulling 4ce4b16d33a3... 100% ▕████████████████████████████████▏  555 B
verifying sha256 digest
writing manifest
removing any unused layers
success
>>> Send a message (/? for help)
```

Рисунок 3.2 – Запуск SLM Phi-3

```
>>> Hello, please tell me what a Raspberry Pi is.
 A Raspberry Pi is a small computer that can be programmed to do various tasks. It was designed
for educational purposes but has many practical applications as well. It runs on an operating
system called Raspbian and is often used in robotics, home automation, and other similar projects.
```

Рисунок 3.3 – Взаємодія з Phi-3

Природно, що чим більше мовна модель, то повільнішим буде результат. На Phi-2 2.7B можна отримати близько 4 токенів за секунду. Але з

Mistral 7B (рис. 3.4) швидкість генерації знижується до ≈ 2 токенів за секунду (токен приблизно еквівалентний одному слову).

```
>>> hello explain what a raspberry pi is
 A Raspberry Pi is a small, affordable, and versatile single-board computer that was initially
created to promote the teaching of basic computer science in schools. It was designed in the
United Kingdom by the Raspberry Pi Foundation in 2012.

The Raspberry Pi board is around the size of a credit card and comes with various input/output
(I/O) interfaces, such as USB ports, Ethernet, HDMI, audio jack, camera interface, and GPIO pins.
These interfaces allow users to connect various peripherals like keyboards, mice, displays,
cameras, and sensors.

The Raspberry Pi can run a full-fledged operating system, including popular distributions such as
Debian, Raspbian, Ubuntu, and Windows 10 IoT Core. Users can write code in various programming
languages like Python, Scratch, C++, and others to interact with the hardware and create projects.

Raspberry Pi is commonly used for educational purposes, prototyping new projects, building home
automation systems, media centers, game consoles, and even as a small server. It's an excellent
platform for makers, hobbyists, and engineers to learn, experiment, and innovate in the field of
electronics, computing, and robotics.
```

Рисунок 3.4 – Взаємодія з Mistral 7B

Встановлення та запуск веб-інтерфейсу Ollama виконується згідно інструкції з офіційного репозиторію Ollama Web UI GitHub для встановлення його без Docker [59]. Рекомендується, щоб Node.js був щонайменше не старіше версії 20.10. Він також рекомендує використовувати Python версії не нижче 3.11. Спершу нам потрібно встановити Node.js. У терміналі треба виконати команди:

```
curl-fsSL https://deb.nodesource.com/setup_20.x | sudo -E bash - &&\
sudo apt-get install -y nodejs
```

Потім запустіть блок коду нижче:

```
git clone https://github.com/ollama-webui/ollama-webui.git
cd ollama-webui/

# Copying required .env file (Копіювання необхідного файлу .env)
cp -RPp example.env .env

# Building Frontend Using Node (Зовн. інтерфейс за доп. Node)
npm i
npm run build

# Serving Frontend with the Backend (Обслуговування зовн.
інтерфейсу за допомогою внутр. інтерфейсу)
cd ./backend
pip install -r requirements.txt --break-system-packages
sh start.sh
```

Це невелика модифікація того, що представлено GitHub. Зверніть увагу: для простоти і стислості ми не слідуємо кращим практикам, таким як використання віртуальних середовищ, і використовуємо прапор-break-system-packages. Якщо виникає помилка, наприклад, «uvicorn не знайдено», потрібно перезапустити сеанс терміналу. В разі успішного запуску можна отримати доступ до вебінтерфейсу Ollama через порт 8080 через http://0.0.0.0:8080 на Raspberry Pi або через http://<локальна адреса Raspberry Pi>:8080/, якщо виконується доступ через інший пристрій у тій самій мережі (рис. 3.5). Після того, як ви створили обліковий запис та увійшли до системи, ви повинні побачити щось схоже на зображення, яке наведено на рис. 3.6. Якщо раніше було завантажено деякі SLM, то повинні побачити їх у меню (рис. 3.7).

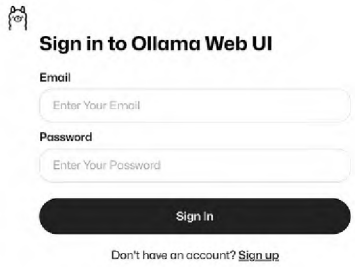


Рисунок 3.5 – Доступ до Ollama Web UI

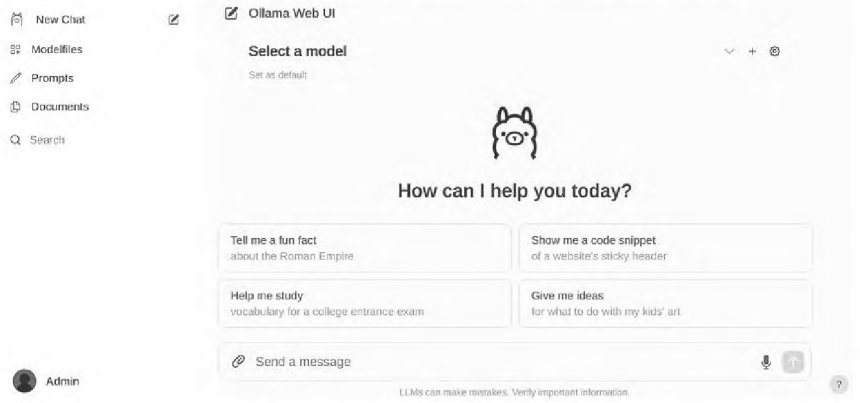


Рисунок 3.6 – Інтерфейс Ollama Web UI

Рисунок 3.7 – Встановлені локально SLM

Весь інтерфейс дуже чистий та інтуїтивно зрозумілий, тому я не буду багато про нього пояснювати. Це дійсно дуже добре зроблений проект із відкритим вихідним кодом. Приклад взаємодії з Mistral 7B через вебінтерфейс Ollama наведено на рис. 3.8.

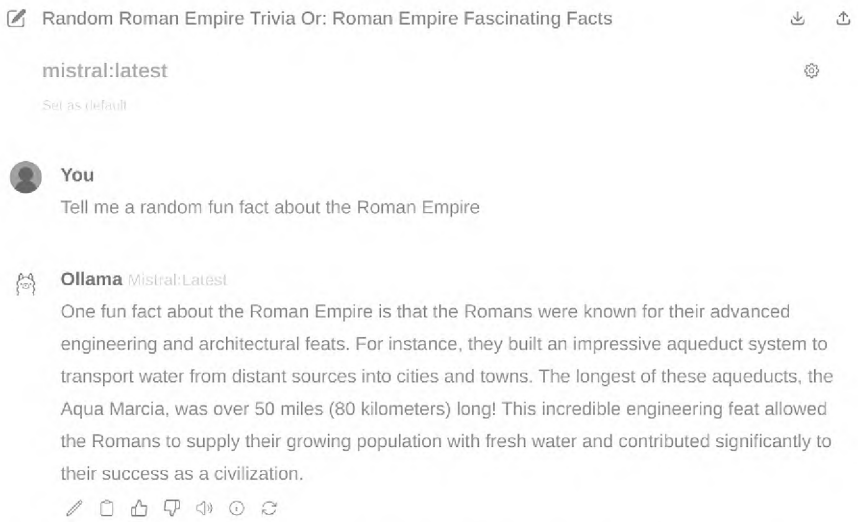


Рисунок 3.8 – Взаємодія з Mistral 7B через Ollama Web UI

Завдяки таким платформам, як Ollama, доступ до цих моделей та розгортання їх стали простішими, ніж будь-коли, що дозволяє розробникам безперешкодно інтегрувати розширені можливості NLP у свої проекти.

3.2 Вибір SLM для локалізації на Raspberry Pi 5

У 2024 р. множина SLM є більш яскравою, ніж будь-коли. Такі моделі, як Gemma2, Lamma 3.1 8B, Mistral Nemo 12B, Qwen2 0.5, 1 і 7B, а також Phi3 пропонують вражаючі можливості, будучи оптимізованими для локального розгортання. Ці моделі розроблені для ефективної роботи на пристроях з обмеженими обчислювальними ресурсами, що робить їх оптимальними для широкого спектру застосувань. Зазвичай мовна модель висуває вимоги до CPU або GPU у вимогах до проекту. Тому для Raspberry Pi 5 потрібно

віддавати перевагу SLM, які підтримують роботу CPU без GPU. Крім того, через обмеження оперативної пам'яті Raspberry Pi 5 треба вибирати моделі з меншим обсягом пам'яті. За звичайних обставин модель вимагає вдвічі більшого обсягу оперативної пам'яті для нормальної роботи. Квантована модель має нижчі вимоги до пам'яті, тому рекомендується використовувати Raspberry Pi 5 на 8 ГБ і квантовану SLM для використання та тестування.

У квітні 2024 р. Microsoft оголосила, що створює сімейство відкритих моделей Phi-3 [60], які вони назвали «найпотужнішими та економічно ефективними SLM». Вона призначена для локального розгортання та використовує передові методи оптимізації для забезпечення високої продуктивності з мінімальними потребами в ресурсах. Phi3 особливо добре підходить для таких завдань, як генерація тексту, переклад та узагальнення. Архітектура дозволяє безперебійно працювати на пристроях з обмеженою обчислювальною потужністю, що робить її чудовим вибором для розробників, які прагнуть інтегрувати можливості AI у свої програми. Першою з цього сімейства є Phi-3-mini (3.8 B). Вона може добре працювати для таких сценаріїв, як:

– узагальнення довгих, специфічних для конкретної галузі документів, таких як нові нормативні акти, та виділення ключових моментів із них;

– налаштування чат-бота, який міг би точно відповідати на запитання клієнтів і занурюватися в записи CRM, щоб запропонувати відповідні оновлення;

– створення маркетингових матеріалів (публікації в соц. мережах, описи продуктів/послуг та ін.).

Моделі Phi-3 охоплюють такі сфери, як конфіденційність, безпека, надійність та інклюзивність (завдяки навчанню на високоякісних, інклюзивних даних).

Llama 3 – це просунута мовна модель від к. Meta, яка набагато потужніша за свого попередника. Набір даних, на якому він був навчений, у сім разів більший, ніж у Llama 2, і містить у 4 рази більше коду. Вона може

розуміти текст довжиною до 8 000 токенів, що вдвічі перевищує можливості її старшого брата, що робить її здатною розуміти та генерувати довші та складніші текстові фрагменти. Ця мовна модель має розширені можливості міркувань і демонструє високу продуктивність за різними галузевими тестами. Не дивно, що їх вважають найкращими моделями з відкритим вихідним кодом у своїй категорії. Meta зробила її доступною для всіх своїх користувачів, маючи намір просувати «наступну хвилю інновацій у сфері AI, що впливає на все, від додатків та інструментів розробників до методів оцінки та оптимізації висновків». Llama 3 використовується в продуктах Meta: вона називається Meta AI. Meta AI доступний через пошук, стрічки та чати на Meta, Instagram, WhatsApp і Messenger. Користувачі можуть отримувати інформацію в режимі реального часу з Інтернет, не перемикаючись між програмами. Її подальшим розвитком стала Lamma 3.1 8B. Вона оптимізована для таких завдань, як відповіді на питання, аналіз настроїв і класифікація текстів. Здатність працювати локально робить її популярним вибором для розробників, яким потрібні надійні можливості NLP, не покладаючись на хмарні служби.

Qwen2 – це сімейство мовних моделей, які бувають різних розмірів для задоволення різних потреб. Модель Qwen2 0.5B призначена для надлегких застосувань, тоді як моделі Qwen2 1B і 7B пропонують підвищений рівень продуктивності. Ці моделі оптимізовані для таких завдань, як генерація тексту, узагальнення та аналіз настроїв. Їхня масштабованість робить їх універсальним вибором для розробників, які прагнуть інтегрувати можливості штучного інтелекту у свої програми.

Gemma 2 – це передова мовна модель від к. Google, призначена для локального розгортання та використовує передові методи стиснення для зменшення розміру без шкоди для продуктивності. Gemma2 підходить для таких завдань, як генерація тексту, переклад та узагальнення.

Mistral 7B – високопродуктивна мовна модель, призначена для локального використання та надає розширені можливості для складних

завдань NLP. Mistral добре підходить для таких завдань, як переклад мови, генерація тексту та діалогові системи. Архітектура дозволяє безперебійно працювати на пристроях із помірними обчислювальними ресурсами.

DeepSeek-Coder-V2 – модель з відкритим вихідним кодом, створена за допомогою техніки машинного навчання Mix-of-Experts (MoE). Вона поставляється з попередньо навченим 6 трильйонами токенів, підтримує 338 мов і має довжину контексту в 128 тисяч токенів. Порівняння показують, що під час виконання завдань кодування вона може досягати показників продуктивності, подібних до GPT4-Turbo. DeepSeek-Coder-V2 має найвищу точність на HumanEval з показником 90,2 %.

В цілому, в ході досліджень визначено основні властивості найбільш поширених SLM.

Deepseek Coder V2. Спеціалізуючись на питаннях, пов'язаних з кодом, ця модель пропонує ефективні можливості генерації коду. Вона розроблена для завдань програмування та надає надійну основу для розробників для автоматизації та вдосконалення процесів кодування.

Phi 3 – модель характеризується своєю універсальністю і здатна впоратися з широким спектром завдань. Він забезпечує баланс між продуктивністю та гнучкістю, що робить його популярним вибором для розробників, які шукають багатоцільовий SLM.

Llama 3.1 відома своїми компактними розмірами та ефективністю. Вона є чудовим варіантом для додатків з обмеженими обчислювальними ресурсами та справляється із завданнями з NLP та особливо вправна у генерації та узагальненні тексту.

Gemma 2 2B – модель виділяється вражаючою швидкістю виконання і точністю, яка добре підходить для додатків у режимі реального часу, таких як чат-боти та віртуальні помічники, де швидкі та точні відповіді мають вирішальне значення.

Qwen – ця модель з відкритим вихідним кодом, яка пропонує велику гнучкість і можливості налаштування. Особливо вона подобається

розробникам, яким потрібна SLM, що налаштовується для конкретних випадків використання.

3.3 Оцінка продуктивності мовних моделей для локальної роботи

Щоб оцінити продуктивність SLM на Raspberry Pi 5, вимірювалась швидкість виконання кожної моделі, а також оцінювалась продуктивність за набором заздалегідь визначених завдань [61]. Крім цього, увага зосереджувалась розмірі моделі, ліцензіях з відкритим вихідним кодом і фреймворках виконання. Ці завдання охоплювали широкий спектр програм, включаючи математичні розрахунки, генерацію коду, узагальнення тексту та взаємодію з чат-ботами. Результати виявили значні відмінності у швидкості виконання серед різних моделей, причому деякі моделі продемонстрували надзвичайну ефективність у конкретних завданнях (табл. 3.1). Розмір моделей також був вирішальним фактором у проведеному аналізі, в ході якого порівнювались вимоги до сховища для кожної моделі та оцінювався їх вплив на загальну продуктивність системи.

Таблиця 3.1 – Продуктивність мовних моделей на платформі Ollama

Модель	Розмір, ГБ	Швидкість, токен/с
Mistral-7B-Q4	4,1	0,97
PHI3 3.8B-Q4	2,2	3,06
PHI3.5-3.8B-Q4	2,2	3,42
Llama3.1-8B-Q4	4,7	1,18
Gemma2-2B-Q4	1,6	2,97
QWEN2-0.5B-Q4	395	20
QWEN2.5-0.5B-Q4	398	19,41
CoderDeepseekV2-7B-Q4	8,9	не може працювати
Llama2-7B-Q4	3,8	не може працювати

Такі моделі, як Qwen і Gemma2 2b, виділялися своїми компактними розмірами, що робить їх оптимальними для додатків з обмеженими можливостями зберігання.Як ітераційна версія Phi3, Phi3.5 має покращену продуктивність, зберігаючи розмір моделі та кількість параметрів незмінними. Qwen2.5 досягає підвищеної точності при незначному збільшенні розміру моделі. Тестування широкого спектру LLM на Raspberry Pi 5 дало цінну інформацію про типи моделей, які реально можуть працювати на цьому компактному пристрої. Для такого пристрою добре підходять моделі з параметрами менше 7 B, пропонуючи хороший баланс між продуктивністю і використанням ресурсів. Однак є винятки, такі як Mistral 7B, який, незважаючи на те, що був більшою моделлю, працював нормально, хоча і трохи повільно. З іншого боку, моделі в діапазоні 2B, 3B і 4B показали винятково хороші результати, продемонструвавши здатність Raspberry Pi справлятися зі складними завданнями AI. Raspberry Pi 5 8 ГБ забезпечила послідовну та надійну платформу для запуску SLM, дозволяючи компактно та економічно ефективно використовувати можливості інтелектуальних обчислень. Однак слід зазначити, що для моделей об'ємом понад 8 ГБ мікрокомп'ютер Raspberry Pi 5 все ж таки має обмеження у навантаженні моделі, виділяючи її обмеження щодо ємності оперативної пам'яті. Аналіз різних SLM на Raspberry Pi 5 дає цінну інформацію про їх продуктивність, можливості та обмеження. Її можна для обґрунтування вибору найбільш підходящої моделі для конкретних вимог.

3.4 Техніко-економічне обґрунтування прийнятих рішень

На даний час, існує кілька мотивів для вибору та локального використання мовних моделей. Наприклад, можна налаштувати модель для виконання спеціалізованого завдання в програмі телемедицини, якщо нема бажання надсилати свій набір даних через Інтернет постачальнику AI API.

Багато локальних інструментів SLM open source з графічним інтерфейсом користувача надають інтуїтивно зрозумілий зовнішній інтерфейс для налаштування та експериментування з SLM без служб на основі підписки, таких як OpenAI або Claude. SLM, такі як PHI-3, Mixtral, Llama 3, DeepSeek-Coder-V2 і MiniCPM-Llama3-V 2.5, покращують різноманітні операції завдяки своїм розширеним можливостям, до яких слід віднести:

– використовуйте можливості пошуку та узагальнення для пошуку відповідної інформації у великих обсягах тексту;

– швидко перетворюйте будь-який текст на необхідні метадані. наприклад, позначте коментарі на своїй веб-сторінці або відгуках про продукт і об'єднайте їх у кластери, щоб отримати статистику;

– будь-який текст на актуальну інформацію для API систем;

– перетворення медіанотатків на правильні виклики API з тегами, що дозволяє вашим системам автоматично збирати та аналізувати інформацію (особливо корисно для того, щоб бути в курсі новин галузі та аналізувати фінансову інформацію);

– позначення та групування всі думок та коментарів про продукти, щоб покращити їх на основі відгуків користувачів;

– аналізувати конкурентів, вивчаючи їхні сторінки, витягуючи всі функції, які вони пропонують, кластеризуючи ці дані та використовуючи інсайти для покращення своїх продуктів.

Які бізнеси можуть отримати найбільшу вигоду від моделі DeepSeek-Coder-V2? Вона найкраще підходить не лише для тих, кому потрібно, щоб його SLM мала аналітичні можливості на найвищому рівні. Вона також оптимальна, коли не можна поділитися кодом через критично важливі системи, якщо вони працюють у хмарі. SLM пропонують значні переваги з точки зору економії коштів, ефективності та універсальності. Вони дешевші в навчанні та розгортанні, ніж LLM, що робить їх доступними для ширшого спектру застосувань. Оскільки вони ефективно використовують обчислювальні ресурси, вони можуть запропонувати хорошу продуктивність і

працювати на різних пристроях, включаючи смартфони та периферійні пристрої. Крім того, оскільки ви можете навчати їх на спеціалізованих даних, вони можуть бути надзвичайно корисними під час вирішення нішевих завдань. Загалом, SLM для конкретної предметної області забезпечують практичне, економічно ефективне рішення для бізнесу без шкоди для продуктивності та точності виведення. В табл. 3.2 наведено положення щодо обґрунтування вибору SLM.

Таблиця 3.2 – Порівняння SLM і LLM

Аспект	SLM	LLM
Визначення	Менша кількість параметрів (від мільйонів до кількох мільярдів)	Величезна кількість параметрів (десятки або сотні мільярдів)
Продуктивність	Добре підходить для простіших завдань	Відмінний у складному розумінні мови та генерації
Навчання та ресурси	Менше обчислювальної потужності пам'яті та сховища	Потрібні значні обчислювальні ресурси, пам'ять і сховище
Час навчання	Коротший і дешевший	Трудомісткий і дорогий
Випадки використання	Вбудовані системи, мобільні додатки, локальна генерація	Просунуті розмовні агенти, створення контенту, складні системи перекладу
Розгортання	Простіше розгортання на різних платформах	Часто потрібне спеціалізоване обладнання та потужні хмарні сервери
Гнучкості	Швидша точна настройка або корекція під конкретні завдання або домени	Більша гнучкість і адаптивність, але вимагає значних зусиль для точного налаштування
Вплив на навколишнє середовище	Низьке споживання енергії та вуглецевий слід	Більш високе енергоспоживання та більший вуглецевий слід
Безпека	Нижчий ризик зараження завдяки меншій площі розгортання	Вищий ризик через більшу поверхню атаки та залежність від хмарної інфраструктури

Для реалізації запропонованих в роботі рішень необхідне апаратне забезпечення (табл. 3.3). Аналіз вартості свідчить про можливість розгортання рішень GenAI на основі локалізації та пристроях, що мають обмеження за обчислювальною потужністю та обсягом ОЗП. Raspberry Pi 5 значно покращила швидкість обробки в порівнянні з Raspberry Pi 4B. Крім

цього, підвищити продуктивність Raspberry Pi 5 можливо за рахунок інтеграції AI-акселераторів. У Додатку А запропонований програмний код для підключення AI-акселератору до Raspberry Pi. Загальна вартість обладнання складає ≈ 15100 грн. Для створення периферійних пристроїв на основі рішень «GenAI + CV» до вказаних компонентів варто додати камеру AI Camera IMX500, в яку інтегровані інтелектуальні функції обробки зображень. На даний час, її вартість складає 4036 грн. Таким чином, цей варіант периферійного пристрою обійдеться у 19134 грн.

Таблиця 3.3 – Вартість обладнання для реалізації локалізованих SLM

№ з/п	Найменування обладнання	Вартість обладнання, грн
1	Мікрокомп'ютер Raspberry Pi 5 8 ГБ	4200
2	AI-акселератор Raspberry Pi AI HAT+ на базі Halo8 (26 TOPS)	9250
3	Блок живлення Raspberry Pi 27W PD USB-C	543
4	Радіатор і вентилятор для Raspberry Pi 5	340
5	Карта microSD MakerDisk на 64 ГБ з ОС Raspberry Pi	700
6	Адаптер USB-microSD для карти	65
Разом витрати на обладнання:		15098

Для додатків LLM, які вимагають більш високої продуктивності, розглядається LattePanda Sigma [62]. При роботі LLaMA2-7B-Q4 його швидкість може досягати 6 токенів/с [63]. LattePanda Sigma забезпечує більш високу продуктивність для задоволення вимог додатків на основі мовних моделей.

Завдяки прогресу в методах стиснення, квантування та оптимізації моделей SLM тепер можуть запропонувати продуктивність, яка конкурує з їхніми більшими аналогами. В цілому, Raspberry Pi 5 значно покращила швидкість обробки в порівнянні з попередником, але все ще є обмеження при роботі з великими LLM через обмеження ємності оперативної пам'яті. Це свідчить про доцільність подальших досліджень за цим напрямом.

Висновки до розділу 3

Вибір SLM для локалізації на Raspberry Pi 5 у 2024 р. надзвичайно широкий, охоплюючи варіанти від компаній Microsoft, Meta, Google та ін.. Моделі, наприклад, Phi-3, Llama 3.1 та Qwen2, оптимізовані для роботи на пристроях з обмеженими ресурсами. Вони здатні виконувати завдання генерації тексту, перекладу та аналізу настроїв з мінімальними вимогами до CPU та ОЗП. Використання квантованих моделей дозволяє ефективно працювати на таких пристроях, як Raspberry Pi 5 з 8 ГБ оперативної пам'яті.

Для локалізації SLM на базі Raspberry Pi 5 обрано платформу Ollama, яка значно спрощує їх розгортання та запуск. Вона надає REST API для взаємодії з моделями, що дозволяє викликати їх з інших пристроїв та додатків. Для роботи Ollama не потрібен GPU, що особливо важливо для Raspberry Pi. Платформ підтримує інтеграцію з вебінтерфейсом, що забезпечує зручний спосіб взаємодії для користувачів.

Оцінка продуктивності SLM на Raspberry Pi 5 показала, що моделі з об'ємом до 7B забезпечують оптимальний баланс між продуктивністю та використанням ресурсів. Такі моделі, як Qwen і Gemma2 2B, продемонстрували високу ефективність завдяки своїм компактним розмірам. Деякі більші моделі, наприклад, Mistral 7B, працюють повільніше, але стабільно. Phi3.5 виявилася ефективнішою версією Phi3, хоча обсяг та параметри залишаються незмінними. Загалом, Raspberry Pi добре підходить для компактного та економного використання рішень GenAI, хоча обмеження обсягу пам'яті не дозволяють працювати з моделями понад 8 ГБ.

В цілому, вибір локальних SLM для спеціалізованих завдань має значні переваги з точки зору безпеки, продуктивності та економії ресурсів. Вони дозволяють уникнути передачі конфіденційних даних в Інтернет, забезпечуючи локальну обробку інформації. SLM, як-от Phi3.5, Mixtral та Llama 3.2, можуть виконувати завдання пошуку, узагальнення, кластеризації даних та аналізу відгуків, що робить їх корисними для широкого кола

застосувань. Вони забезпечують економічну ефективність завдяки меншим вимогам до обчислювальних ресурсів, що дозволяє використовувати їх на периферійних пристроях, як-от Raspberry Pi 5. Загалом, локальні SLM є практичним рішенням для бізнесів, яким потрібні спеціалізовані можливості AI без значних затрат.

ВИСНОВКИ

Подальший розвиток GenAI залежить від доступних даних, поширення мультимодальності, кастомізації моделей, відкритого вихідного коду та етичних стандартів. LLM відкривають широкі можливості для розвитку AI та покращення якості життя людей, забезпечуючи високий рівень безпеки, продуктивності та ефективності в різних сферах діяльності. Приклади практичного застосування LLM охоплюють різні сфери, зокрема медицину, освіту, технічну підтримку, маркетинг.

Зі збільшенням обсягів даних компанії стикаються зі складнощами в управлінні знаннями. Багатьом підприємствам важко впроваджувати AI через високі технічні та фінансові вимоги сучасних LLM. Хоча LLM, такі як GPT-4, досягають успіху в розумінні та генерації тексту, вони вимагають значних обчислювальних ресурсів, часто потребуючи сотень гігабайт пам'яті та дорогого обладнання графічного процесора. Це створює значний бар'єр для багатьох організацій, поряд із занепокоєнням щодо конфіденційності даних та операційних витрат.

Однобітові моделі (TriLM), які використовують тризначні квантові представлення (-1, 0, 1), пропонують перспективне вирішення цих проблем, скорочуючи потреби в пам’яті приблизно на 97 %.

Edge AI стає ключовим для LLM, оскільки забезпечує обробку даних у реальному часі з мінімальною затримкою, підвищує конфіденційність і оптимізує використання ресурсів. Локалізація LLM сприяє незалежності від зовнішніх сервісів, зниженню витрат, адаптації до специфічних задач, а також забезпечує енергоефективність та підтримку національних мов і діалектів. Для забезпечення автономності використовують інструменти, такі як LM Studio, Llamafile, GPT4ALL, LLaMa.cpp. Для інтеграції LLM на пристрої з обмеженими ресурсами переходять до SLM, які надають аналогічні результати при значно меншій витраті ресурсів. Оптимізація моделей за допомогою LoRA або QLoRA суттєво зменшує кількість

параметрів, полегшуючи адаптацію. Встановлено, що платформа Raspberry Pi (особливо Raspberry Pi 5) є ефективною для розгортання SLM, забезпечуючи доступне рішення з можливістю апаратного прискорення. Остання версія Raspberry Pi 5, завдяки своїй доступності, гнучкості та достатній обчислювальній потужності, стає оптимальною платформою для впровадження SLM в IoT-пристрої та автономні системи. Співпраця Raspberry Pi з к. Hailo дозволила інтегрувати AI-акселератори, наприклад, Hailo-8 з продуктивністю 26 TOPS, що підходить для додатків на основі обробки зображень та нейронних мереж.

В свою чергу, платформа Ollama, що підтримує REST API, значно спрощує використання SLM на Raspberry Pi. Оцінка продуктивності SLM на Raspberry Pi 5 показала оптимальне співвідношення між продуктивністю та використанням ресурсів для моделей до 7B параметрів. Вибір локальних SLM для спеціалізованих завдань пропонує значні переваги з погляду безпеки, продуктивності та економії ресурсів.

Загальна вартість обладнання складає для локалізації SLM на базі Raspberry Pi 5 складає ≈ 15100 грн. За необхідності реалізації рішень «GenAI + CV» номенклатура компонентів за рахунок додавання AI-камери IMX500. При цьому, такий варіант периферійного пристрою обійдеться у ≈ 19135 грн.

Таким чином, результатами роботи є методика локалізації мовних моделей на Raspbery Pi5; рекомендації щодо локалізації мовних моделей. Вони можуть бути використані для подальших досліджень за даною тематикою та при проектуванні вбудованих комп'ютерних систем.