Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

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

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Методи ефективного донавчання великих мовних моделей
із вдосконаленням механізмів адаптерів
(тема)

Виконав:
здобувач другого року навчання,
групи СШІм-24-1
Ігор МІЧУРІН
(власне ім'я, прізвище)

Спеціальність 122 Комп'ютерні науки
(код і повна назва спеціальності)
Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)
Освітня програма Системи штучного інтелекту
(повна назва освітньої програми)
Керівник проф. Наталія РЯБОВА
(посада, власне ім'я, прізвище)

Допускається до захисту

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

2026 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук

Кафедра Штучного інтелекту

Рівень вищої освіти другий (магістерський)

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

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

Освітня програма Системи штучного інтелекту
(повна назва)

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

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Мічуріну Ігорю Євгеновичу
(прізвище, ім'я, по батькові)

1. Тема роботи Методи ефективного донавчання великих мовних моделей із вдосконаленням механізмів адаптерів

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

2. Термін подання здобувачем роботи до екзаменаційної комісії 19 травня 20 26 р.

3. Вихідні дані до роботи науково-технічні публікації з PEFT, документація Python і PyTorch, задачі GLUE

4. Перелік питань, що потрібно опрацювати в роботі

1) Аналіз предметної галузі та постановка задач дослідження

2) Розробка методу spectral gated adapter

3) Експериментальні дослідження

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Строк / термін виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	06.04.2026	виконано
2	Аналіз предметної галузі та PEFT-методів	10.04.2026	виконано
3	Постановка задачі дослідження	12.04.2026	виконано
4	Розробка математичної моделі SGA	15.04.2026	виконано
5	Програмна реалізація адаптерів	18.04.2026	виконано
6	Підготовка експериментального середовища	20.04.2026	виконано
7	Навчання моделей і збирання метрик	22.04.2026	виконано
8	Аналіз результатів та абляційне дослідження	24.04.2026	виконано
9	Оформлення пояснювальної записки	26.04.2026	виконано
10	Перевірка на академічний плагіат	27.04.2026	виконано
11	Нормоконтроль та рецензування	05.05.2026	виконано
12	Захист перед ЕК	19.05.2026	виконано

Дата видачі завдання 6 квітня 2026 р.

Здобувач _____
(підпис)

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

РЕФЕРАТ

Пояснювальна записка: 68 с., 39 рис., 12 табл., 2 дод., 22 джерела.

АДАПТЕРИ, ВЕЛИКІ МОВНІ МОДЕЛІ, ГЕЙТУВАННЯ, ПАРАМЕТРИЧНО-ЕФЕКТИВНЕ ДОНАВЧАННЯ, СПЕКТРАЛЬНА РЕГУЛЯРИЗАЦІЯ, BERT, GLUE, LORA.

Об’єкт дослідження – процес параметрично-ефективного донавчання великих мовних моделей.

Предмет дослідження – методи адаптерного донавчання з токен-залежним гейтуванням і спектральною регуляризацією.

Мета роботи – розробка та експериментальна перевірка методу Spectral Gated Adapter для ефективного донавчання великих мовних моделей. Методи дослідження – аналіз наукових джерел, математичне моделювання, методи глибокого навчання та генеративного штучного інтелекту, інтелектуальний дослідницький аналіз та експериментальне порівняння отриманих емпіричних даних.

У роботі запропоновано архітектуру SGA, що поєднує рангове гейтування з ортогональною спектральною регуляризацією адаптерних матриць. Експерименти на MRPC, RTE, QNLI та SST-2 показали конкурентну якість порівняно з LoRA, менший параметричний бюджет і нижче пікове використання GPU-пам’яті.

Розроблений метод може бути впроваджений у навчальні пайплайни PyTorch і HuggingFace Transformers. Робота пов’язана з дослідженнями LoRA, AdaLoRA, DoRA, BitFit, IA^3, Mixture-of-Experts і спектральної нормалізації. Результати доцільно використовувати в задачах NLP з обмеженими обчислювальними ресурсами та продовжити в напрямах квантування, мультимовного оцінювання і масштабування на декодерні моделі.

ABSTRACT

Master’s thesis contains: 68 pp., 39 fig., 12 tabl., 2 ann., 22 references.

ADAPTERS, BERT, GATING, GLUE, LARGE LANGUAGE MODELS, LORA, PARAMETER-EFFICIENT FINE-TUNING, SPECTRAL REGULARIZATION.

The object of the research is parameter-efficient fine-tuning of large language models.

The subject is adapter-based fine-tuning with token-level gating and spectral constraints.

The aim is to develop and evaluate Spectral Gated Adapter for efficient adaptation of large language models. The research methods include analysis of scientific sources, mathematical modelling, deep learning and generative artificial intelligence methods, intelligent research analysis, and experimental comparison of the obtained empirical data.

The work proposes an SGA architecture that combines rank-wise gating with orthogonal spectral regularization of adapter matrices. Experiments on MRPC, RTE, QNLI, and SST-2 show competitive quality against LoRA with a reduced trainable-parameter budget and lower peak GPU memory usage.

The method is ready for integration into PyTorch and HuggingFace Transformers training pipelines. The results are related to LoRA, AdaLoRA, DoRA, BitFit, IA^3, Mixture-of-Experts, and spectral normalization. Further work should cover quantization, multilingual evaluation, and scaling to decoder-based language models.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

BERT – Bidirectional Encoder Representations from Transformers – двонаправлені представлення енкодера від трансформерів;

LoRA – Low-Rank Adaptation – низькорангова адаптація;

PEFT – Parameter-Efficient Fine-Tuning – параметрично-ефективне донавчання;

SGA – Spectral Gated Adapter – спектральний гейтований адаптер.

ВСТУП

Сучасний розвиток великих попередньо навчених мовних моделей суттєво змінив підхід до обробки природної мови. Моделі класу BERT, RoBERTa, GPT та їхні наступники демонструють високу якість на широкому колі задач, однак їхнє повне донавчання потребує значних обчислювальних ресурсів, великого обсягу відеопам’яті та збереження окремих копій параметрів для кожного прикладного сценарію.

За таких умов особливої актуальності набувають методи параметрично-ефективного донавчання, які дають змогу адаптувати велику модель до конкретної задачі без оновлення всіх її параметрів. Замість повної зміни ваг базової моделі такі методи додають невеликі навчувані модулі або низькорангові поправки, що істотно зменшує вартість навчання та спрощує розгортання кількох задач на одному спільному бекбоні.

Незважаючи на успішність підходів LoRA, адаптерів і prompt-based методів, у них залишаються обмеження, що впливають на практичну ефективність. По-перше, більшість існуючих адаптерів застосовує однаковий обсяг обчислень до кожного токена, незалежно від його інформативності. По-друге, низькорангові методи можуть страждати від виродження ефективного рангу, коли номінально доступна ємність адаптера фактично використовується неповністю.

Метою кваліфікаційної роботи є дослідження та розробка методу ефективного донавчання великих мовних моделей із вдосконаленням механізмів адаптерів на основі поєднання токен-залежного гейтування та спектральної регуляризації. Запропонований підхід має забезпечити кращий баланс між якістю класифікації, кількістю навчуваних параметрів, використанням пам’яті та інтерпретованістю роботи адаптера.

Об’єктом дослідження є процес параметрично-ефективного донавчання великих мовних моделей для задач розуміння природної мови. Предметом дослідження є архітектури адаптерів, механізми гейтування та

спектральні обмеження, що впливають на стабільність і ефективність донавчання.

Для досягнення поставленої мети необхідно проаналізувати сучасні PEFT-методи, формалізувати архітектуру Spectral Gated Adapter, реалізувати її у середовищі PyTorch і HuggingFace Transformers, провести експерименти на задачах GLUE та порівняти отримані результати з базовими підходами за якістю, параметричною ефективністю, використанням пам’яті та обчислювальною складністю.

Апробація результатів роботи. Окремі результати дослідження апробовано у вигляді тез доповіді «Порівняльний аналіз сучасних методів ефективного за параметрами донавчання LLM», підготовлених для 30-го Міжнародного молодіжного форуму «Радіоелектроніка та молодь у XXI столітті» (Харків, 22–24 квітня 2026 р.) [21].

Публікації та матеріали за темою роботи. За матеріалами розділів, присвячених архітектурі Spectral Gated Adapter, експериментам GLUE, аналізу обчислювальної ефективності та абляційному дослідженню, підготовлено англомовну статтю «Spectral Gated Adapter: Parameter-Efficient Fine-Tuning with Gated Spectral Filters» для видання Radioelectronic and Computer Systems [22]. Перелік супровідних матеріалів наведено у додатку А.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧ ДОСЛІДЖЕННЯ

1.1 Великі мовні моделі та парадигма донавчання

Архітектура Transformer стала базою для більшості сучасних мовних моделей, оскільки механізм самоуваги дає змогу моделювати залежності між токенами незалежно від їхньої відстані в послідовності. На відміну від рекурентних архітектур, Transformer ефективно паралелізується на графічних процесорах і може масштабуватися до сотень мільйонів або мільярдів параметрів [1], [2], [3], [4].

Типова схема застосування таких моделей складається з двох етапів: попереднього навчання на великих корпусах тексту та подальшого донавчання на цільовій задачі. Попереднє навчання формує універсальні мовні представлення, а донавчання адаптує їх до класифікації, пошуку відповідей, виявлення перефразувань або логічного слідування.

Повне донавчання оновлює всі параметри моделі, тому для кожної задачі формується окрема копія ваг. Для моделей із сотнями мільйонів параметрів це призводить до значних витрат пам’яті та ускладнює підтримку багатьох предметних галузей. Крім того, повне оновлення може руйнувати частину знань, отриманих під час попереднього навчання.

В основі архітектури Transformer лежить механізм самоуваги (self-attention). Для послідовності прихованих станів H спочатку обчислюються запити, ключі та значення згідно з формулою (1.1), а вихід шару самоуваги визначається формулою (1.2). Така операція не залежить від відстані між токенами і дозволяє моделі одночасно враховувати всі позиції послідовності.

$$Q = HW_Q,\ K = HW_K,\ V = HW_V, \tag{1.1}$$

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V. \quad (1.2)$$

Поверх самоуваги Transformer використовує багатоголову увагу (multi-head attention), повнозв'язну мережу прямого поширення (feed-forward network), залишкові з'єднання та нормалізацію шару (LayerNorm). BERT-base містить 12 шарів кодувальника, прихований розмір d = 768, 12 голів уваги та близько 110 мільйонів параметрів.

1.2 Методи параметрично-ефективного донавчання

Методи PEFT спрямовані на те, щоб залишити основну модель замороженою та навчати лише невелику кількість додаткових параметрів. До цієї групи належать адаптери, низькорангові поправки, prompt tuning, prefix tuning, BitFit та гібридні методи. Їхня спільна ідея полягає в тому, що для більшості задач не потрібно змінювати весь простір параметрів великої моделі [5], [13], [14].

Адаптерні методи вставляють невеликі bottleneck-модулі між шарами або всередині блоків Transformer. Такі модулі зазвичай складаються з проєкції в менший простір, нелінійності та зворотної проєкції. Перевагою адаптерів є модульність, а недоліком може бути додатковий інференсний шлях та статичний характер перетворення.

LoRA моделює поправку до матриці ваг як добуток двох низькорангових матриць. Це суттєво зменшує кількість навчуваних параметрів і часто дає якість, близьку до повного донавчання. Водночас ефективний ранг такої поправки не завжди відповідає номінальному рангу, що може зменшувати корисну ємність адаптації.

Prompt-based підходи не змінюють внутрішні матриці моделі, а оптимізують навчувані токени або префікси, які впливають на роботу уваги. Такі методи мають малу кількість параметрів, однак часто поступаються

LoRA та адаптерам у задачах, де необхідна глибша зміна внутрішніх представлень [10], [11], [12].

У таблиці 1.1 подано порівняння основних PEFT-підходів.

Таблиця 1.1 – Порівняння основних PEFT-підходів

Підхід	**Навчувані параметри**	**Переваги**	**Обмеження**
Adapters	Bottleneck-модулі	Модульність і просте перемикання задач	Додатковий інференсний шлях
LoRA	Низькорангові матриці	Висока якість при малому бюджеті	Ризик виродження ефективного рангу
Prefix tuning	Навчувані префікси	Мала кількість параметрів	Обмежений вплив на представлення
SGA	Гейтовані адаптери	Динамічний розподіл ємності	Потребує налаштування λ та рангу

LoRA та її розширення (AdaLoRA, QLoRA, DoRA, VeRA) ґрунтуються на гіпотезі низької внутрішньої розмірності донавчання: ефективна поправка до ваг моделі лежить у підпросторі суттєво меншої розмірності, ніж номінальна кількість параметрів. Окрему групу методів утворюють BitFit, IA^3 та компактери, які оптимізують лише зміщення або модулюють активації за допомогою діагональних матриць [5], [6], [7], [8], [9], [15].

У сценаріях багатозадачного розгортання повне донавчання стикається з проблемою катастрофічного забування: оновлення параметрів моделі для нової задачі може руйнувати представлення, важливі для попередніх задач. PEFT-методи природно вирішують цю проблему: завдяки замороженому бекбону спільні представлення зберігаються, а специфічні для задачі знання ізолюються в адаптерних параметрах [16].

Принцип умовних обчислень реалізовано у рекурентних мережах через ворота LSTM (input, forget, output) та GRU (reset, update). У

трансформерних моделях він розширений до Mixture-of-Experts: Sparsely-Gated MoE та Switch Transformer маршрутизують кожний токен лише до підмножини експертів [16].

1.3 Обмеження існуючих підходів

Першим важливим обмеженням є статичне використання адаптерної ємності. Більшість PEFT-модулів однаково обробляє службові слова, розділові знаки та семантично важливі токени. У задачах класифікації це означає, що адаптер витрачає частину обчислень на елементи, які майже не впливають на рішення, і не має механізму динамічного перерозподілу ресурсу.

Другим обмеженням є проблема виродження рангу. У низькорангових поправках частина сингулярних значень може ставати близькою до нуля, через що модель фактично використовує менший підпростір, ніж було задано архітектурою. Це знижує різноманітність напрямів адаптації і може погіршувати здатність моделі відображати складні зміни в даних.

Гейтування є природним способом зробити обчислення умовними. У рекурентних мережах та Mixture-of-Experts механізми воріт уже використовуються для керування потоком інформації. Проте в контексті PEFT важливо не просто вмикати або вимикати весь адаптер, а модулювати його внесок на рівні токенів і латентних вимірів bottleneck-простору.

Спектральні методи, зокрема аналіз сингулярних значень і ортогональні обмеження, дають змогу контролювати структуру матриць під час навчання. Поєднання таких обмежень із гейтуванням створює основу для адаптера, який одночасно є динамічним, параметрично-економним і стабільним з погляду використання рангу [17].

У контексті спектральних методів важливу роль відіграє сингулярний розклад матриці, наведений у формулі (1.3). Діагональна матриця Σ містить

упорядковані сингулярні значення, а їх згортання до нуля свідчить про виродження ефективного рангу.

$$A = U\Sigma V^T. \tag{1.3}$$

Штраф на відхилення матриці Грама від одиничної структури, поданий у формулі (1.4), є наближенням до ортогональності рядків і одночасно пов’язаний зі стандартною теорією регуляризації низькорангових моделей.

Виявлені обмеження – статичність адаптерної ємності, виродження ефективного рангу та відсутність токен-залежної модуляції – формують вимоги до нового PEFT-методу, який поєднував би динамічне гейтування зі спектральною регуляризацією для одночасного контролю над структурою матриць адаптера.

Таким чином, виявлені обмеження – статичність адаптерної ємності, виродження ефективного рангу та відсутність токен-залежної модуляції – формують вимоги до нового PEFT-методу, який поєднував би динамічне гейтування зі спектральною регуляризацією для одночасного контролю над структурою матриць адаптера.

$$L_{orth} = ||WW^T - I||_F^2. \tag{1.4}$$

1.4 Постановка задач дослідження

Метою кваліфікаційної роботи є розробка методу Spectral Gated Adapter для параметрично-ефективного донавчання BERT-подібних великих мовних моделей на задачах GLUE. На відміну від статичних адаптерів, метод має враховувати інформативність окремих токенів; на відміну від класичної LoRA, він має додатково контролювати спектральні

властивості матриць адаптера, параметри яких оптимізуються під час донавчання.

Для досягнення поставленої мети необхідно вирішити такі задачі: проаналізувати сучасні PEFT-методи та їх обмеження; сформулювати математичну модель SGA; реалізувати адаптер у середовищі PyTorch і HuggingFace Transformers; провести експериментальне порівняння на задачах GLUE; оцінити параметричну й обчислювальну ефективність, використання GPU-пам’яті та інтерпретованість гейтування; сформулювати рекомендації щодо подальшого розвитку методу.

Практичне значення роботи полягає у можливості використовувати один заморожений мовний бекбон разом із легкими адаптерами для кількох задач. Такий підхід є важливим для дослідницьких середовищ з обмеженими ресурсами, для edge-розгортання та для систем, де потрібно швидко перемикатися між предметними галузями без збереження повних копій моделі. Робота виконувалася в межах міжнародного проєкту Dual Degree Programme MSc-by-Research in Computer Science, що реалізується у співпраці ХНУРЕ та University of Warwick за підтримки Mosaik Education та Cormack Consultancy Group.

Порівняльне позиціонування запропонованого методу відносно основних PEFT-підходів наведено у таблиці 1.2. Таблиця узагальнює не тільки параметричну ефективність, а й наявність інференс-оверхеду, адаптивність, контроль стабільного рангу та інтерпретованість.

Таблиця 1.2 – Позиціонування SGA серед існуючих PEFT-методів

Метод	**Парам.**	**Оверхед**	**Адапт.**	**Ранг**	**Інтерпр.**
Houlsby	+	−	−	−	−
LoRA	+	+	−	−	−
AdaLoRA	+	+	~	~	−
DoRA	+	+	−	~	−
VeRA	+	+	−	−	−
Prefix	+	−	−	−	−
IA^3	+	+	−	−	−
SGA	+	~	+	+	+

Позначення у таблиці 1.2: «+» – властивість підтримується, «−» – не підтримується, «~» – підтримується частково або залежить від конфігурації. Скорочення у заголовках: «Парам.» – параметрична ефективність; «Оверхед» – відсутність додаткового інференс-оверхеду; «Адапт.» – адаптивність; «Ранг» – підтримання стабільного рангу; «Інтерпр.» – інтерпретованість. Порівняння показує, що SGA поєднує параметричну ефективність із адаптивним розподілом ємності та контролем ефективного рангу.

Підсумовуючи проведений аналіз, можна зауважити, що жоден з існуючих PEFT-методів не поєднує одночасно адаптивне розподілення обчислень за токенами та явну регуляризацію спектра ваг. Запропонований SGA має заповнити цю прогалину.

1.5 Висновки до розділу

У результаті проведеного в роботі аналізу та порівняння сучасних методів параметрично-ефективного донавчання великих мовних моделей виявлено дві ключові прогалини: статичне використання адаптерної ємності, незалежно від інформативності токенів, та виродження ефективного рангу низькорангових поправок під час навчання.

Розглянуто архітектуру Transformer, механізми самоуваги та feed-forward блоків, парадигму pre-training/fine-tuning, гіпотезу низької внутрішньої розмірності донавчання, таксономію PEFT-методів (адаптери, LoRA та її варіанти, prompt-based підходи, BitFit, IA^3), а також спектральні методи у глибокому навчанні (SVD, спектральна нормалізація, ортогональні обмеження).

Сформульовано задачу розробки PEFT-методу, який поєднує адаптивне токен-залежне гейтування зі спектральною регуляризацією адаптерних матриць для одночасного забезпечення параметричної ефективності, динамічного розподілу ємності та стабільності спектра.

2 РОЗРОБКА МЕТОДУ SPECTRAL GATED ADAPTER

2.1 Загальна схема блоку Spectral Gated Adapter

Запропонований метод Spectral Gated Adapter є адаптерною архітектурою, що додається до шарів Transformer і навчається при заморожених параметрах базової моделі. На вході адаптер отримує прихований стан токена, проєктує його у низьковимірний bottleneck-простір, застосовує нелінійність і повертає результат у вихідний простір моделі.

Ключова відмінність від стандартного адаптера полягає у введенні гейтувального вектора, який залежить від поточного прихованого стану. Цей вектор має розмірність bottleneck-простору та окремо модулює кожний латентний напрям. Таким чином, адаптер може приглушувати внесок неінформативних токенів і підсилювати ті виміри, що важливі для цільової задачі.

Другим компонентом є спектральна регуляризація. Вона заохочує матриці проєкції зберігати різноманітність напрямів і не вироджуватися в підпростір меншого рангу. Для цього вводиться штраф на відхилення добутків матриць від одиничної структури, що наближено підтримує ортогональність стовпців або рядків.

Обидва компоненти діють узгоджено: гейтувальний вектор перерозподіляє адаптерну ємність на рівні токенів, а спектральне обмеження забезпечує стабільність використання внутрішнього представлення на рівні матриць. У поєднанні вони дають архітектуру, яка зберігає переваги класичних адаптерних модулів і одночасно усуває статичність обчислень та проблему виродження ефективного рангу, виявлені у підрозділі 1.3.

Обидва компоненти діють узгоджено: гейтувальний вектор перерозподіляє адаптерну ємність на рівні токенів, а спектральне

обмеження забезпечує стабільність використання внутрішнього представлення на рівні матриць. У поєднанні вони дають архітектуру, яка зберігає переваги класичних адаптерних модулів і одночасно усуває статичність обчислень та проблему виродження ефективного рангу, виявлені у підрозділі 1.3. Графічну ілюстрацію наведено на рисунку 2.1.

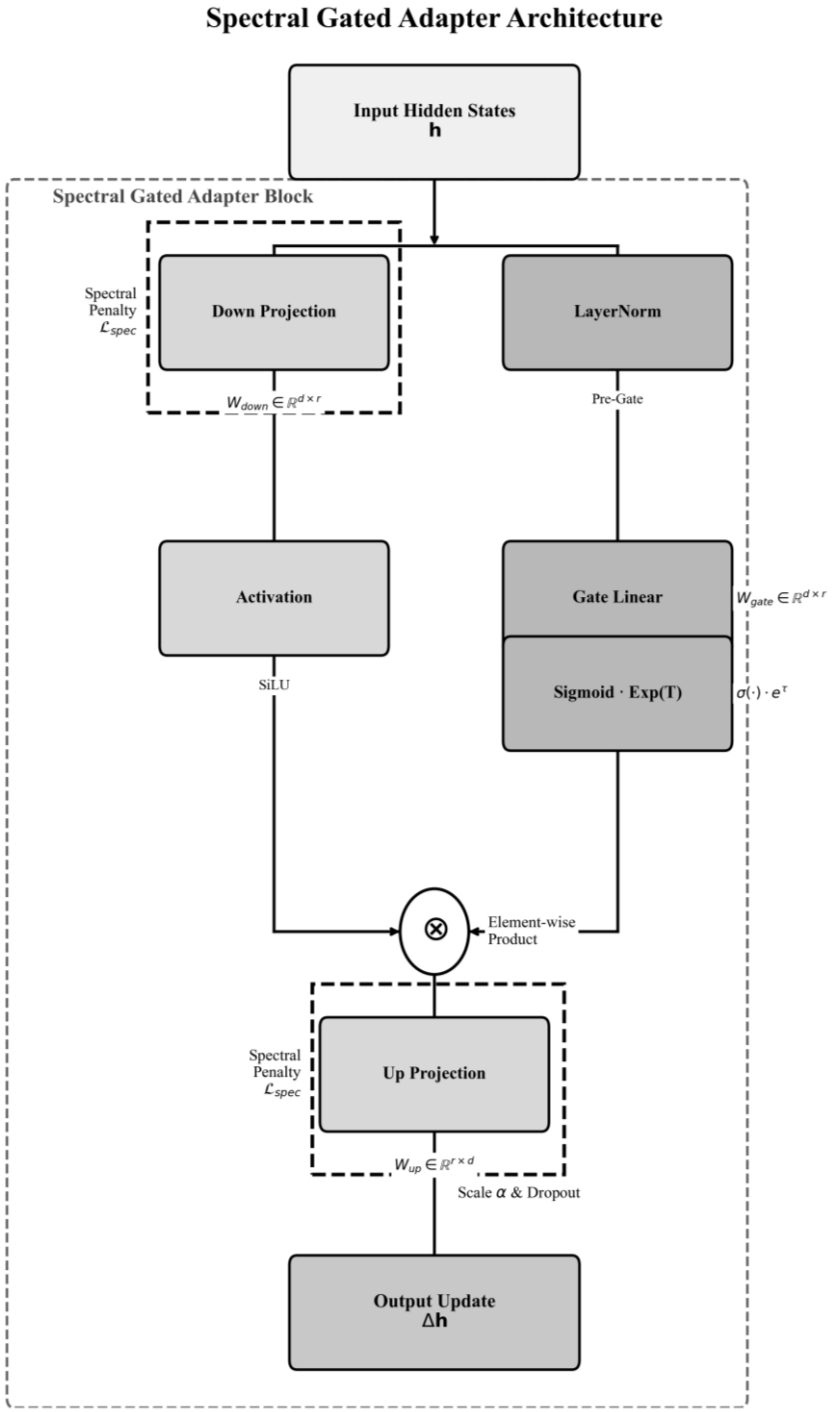

Рисунок 2.1 – Загальна схема блоку Spectral Gated Adapter

Архітектура SGA побудована на чотирьох взаємопов'язаних принципах: адаптивність (внесок адаптера змінюється залежно від інформативності токена), параметрична економність (гейтування є однією додатковою лінійною проєкцією), спектральна стабільність (повний ефективний ранг матриць проєкції) та інтерпретованість (значення гейту як сигнал про роль кожного токена).

Поєднання цих принципів досягається завдяки двом легким механізмам – ранговому гейтуванню та ортогональному штрафу. Жоден з них суттєво не збільшує обчислювальної вартості: гейт додає одну точкову операцію на токен, а штраф зводиться до двох матричних добутків rxd на крок навчання.

2.2 Математична модель адаптера

Нехай $h \in R^d$ є прихованим станом токена на вході адаптера, d – розмірність прихованого простору базової моделі, r – розмірність bottleneck-простору. Стандартний адаптер можна подати як композицію двох лінійних проєкцій і нелінійної функції активації.

У запропонованому методі спочатку формується bottleneck-представлення, а після цього обчислюється гейтувальний вектор, який має значення в діапазоні від 0 до 1 і визначає, яку частину адаптерного сигналу пропускати для кожного виміру. Відповідні перетворення наведені у формулах (2.1)–(2.2).

$$z = \varphi(W_{down}h),\ \alpha(h) = \sigma(W_g h + b_g), \tag{2.1}$$

$$y = h + W_{up}(\alpha(h) \odot z), \tag{2.2}$$

де h – прихований стан токена;

z – bottleneck-представлення;

α(h) – гейтувальний вектор;

W_down, W_g, W_up – матриці проєкцій;

b_g – зміщення гейта;

φ – нелінійність;

σ – сигмоїдна функція.

2.3 Гейтування та спектральна регуляризація

Гейтувальний механізм, заданий формулою (2.1), використовує матрицю W_g і вектор зміщення b_g. На відміну від скалярного гейту, ранговий вектор дозволяє не лише вирішувати, чи активувати адаптер загалом, а й керувати внеском кожного латентного напряму.

Повний вихід адаптера визначається формулою (2.2). Резидуальний зв’язок зберігає стабільність навчання, оскільки навіть при малих значеннях гейту модель може покладатися на початкове представлення базового Transformer.

Спектральна регуляризація вводиться як додатковий член функції втрат. Для матриць проєкції вона штрафує відхилення матриць Грама від одиничної структури, як показано у формулах (2.3)–(2.4). У практичній реалізації це наближено підтримує повний ефективний ранг і зменшує ризик концентрації інформації у кількох домінантних сингулярних напрямах.

$$L = L_{task} + \lambda L_{spectral}, \tag{2.3}$$

$$L_{spectral} = ||W_{down} {W_{down}}^{T} - I||_F^2 + ||{W_{up}}^{T} W_{up} - I||_F^2. \tag{2.4}$$

де L_task – основна функція втрат задачі;

λ – коефіцієнт спектральної регуляризації;

L_spectral – спектральний штраф;

I – одинична матриця;

||·||_F – норма Фробеніуса.

У повній формі гейтувальний вектор обчислюється за формулою (2.5). Тут LN(·) позначає нормалізацію шару, а τ є навчуваним параметром температури з ініціалізацією τ = 0. LayerNorm стабілізує статистики входу гейту, а температура регулює різкість сигмоїди.

$$\alpha(h) = \sigma((W_g LN(h) + b_g)e^{\tau}). \quad (2.5)$$

Повний вихід адаптера з урахуванням залишкового масштабу визначається формулою (2.6), де α_res є навчуваним скалярним коефіцієнтом з ініціалізацією α_res = 1.

$$y = h + \alpha_{res}(\alpha(h) \odot \varphi(W_{down}h)){W_{up}}^{T}. \quad (2.6)$$

Зв’язок спектрального штрафу з сингулярними значеннями зручно пояснити через сингулярний розклад матриці, поданий у формулі (2.7). Для ортогонального штрафу це дає співвідношення (2.8), тому штраф дорівнює нулю тоді й лише тоді, коли всі сингулярні значення дорівнюють одиниці, що одночасно виключає виродження рангу та надмірне зростання норми.

$$W = U\Sigma V^{T}, \quad (2.7)$$

$$||WW^{T} - I||_F^2 = \Sigma_i({\sigma_i}^2 - 1)^2. \quad (2.8)$$

2.4 Варіанти архітектури

У роботі розглянуто базовий варіант SGA і розширений варіант SGA+. Базовий варіант застосовує адаптери у вибраних проєкціях Transformer і має приблизно половину кількості навчуваних параметрів порівняно з LoRA

рангу 8. Він орієнтований на сценарії, де ключовим є зменшення пам’яті та розміру адаптера.

SGA+ використовує більший бюджет параметрів і розміщує адаптерні модулі ширше, зокрема в кількох проєкціях уваги та feed-forward частини. Його метою є досягнення якості, максимально близької до LoRA, зі збереженням переваг гейтування та спектрального контролю.

Для абляційного аналізу додатково порівнюються конфігурації без гейтування, без спектральної регуляризації та з повним набором компонентів. Таке порівняння дозволяє відокремити внесок кожної складової й оцінити, чи є їхня дія взаємодоповнювальною.

У таблиці 2.1 подано конфігурації адаптерів, що порівнюються.

Таблиця 2.1 – Конфігурації адаптерів, що порівнюються

Метод	**Ключова ідея**	**Бюджет**	**Призначення**
Frozen Base	Заморожений BERT + класифікатор	0	Нижня базова лінія
Prefix Tuning	Навчувані префікси уваги	296K	Мінімальний PEFT-бюджет
LoRA r=8	Низькорангова поправка	1,34M	Сильна базова лінія
SGA	Гейтований адаптер	684K	Пам’ять і парам. ефективність
SGA+	Розширене розміщення SGA	1,37M	Якість, близька до LoRA

2.5 Аналіз градієнтів та параметрична складність

Градієнтна динаміка SGA має дві характерні особливості. Перша – розріджені градієнтні оновлення через гейтування. Якщо для деякого виміру і значення α_i близьке до нуля, токени з низькою активацією гейту дають мізерний внесок у градієнт ваг адаптера, тобто навчання фокусує сигнал на інформативних токенах.

Друга особливість – градієнт спектрального штрафу. Для матриці проєкції похідна штрафу штовхає її матрицю Грама до одиничної структури,

причому сила цього поштовху пропорційна відхиленню від ортогональності.

Сумарний параметричний бюджет одного модуля SGA визначається формулою (2.9). Для BERT-base з d = 768 та r = 8 це становить 19 210 параметрів, тоді як LoRA має 2dr = 12 288 параметрів. Базовий SGA розміщує адаптери лише в проєкціях Q, V проти Q, V, Dense у LoRA, тому загальна кількість параметрів виявляється на 49% меншою: 683 834 проти 1 340 930.

$$P_{SGA} = dr + dr + (dr + r) + d + 2. \quad (2.9)$$

Стратегія ініціалізації відіграє важливу роль у стабільному навчанні. W_down ініціалізується за схемою Каймінга з a = √5; W_up ініціалізується нулями, тому початковий вихід адаптера дорівнює нулю. W_g, b_g також ініціалізуються нулями, τ = 0, α_res = 1. За такої схеми початкове значення гейту дорівнює σ(0) = 0,5 для всіх вимірів.

Числова стабільність підтримується кількома засобами: одинична матриця для штрафу зберігається як persistent buffer; після обчислення штрафу проводиться перевірка скінченності значення; перед кроком оптимізатора застосовується глобальне обрізання норми градієнтів зі значенням 1,0.

2.6 Програмна реалізація

Реалізація методу виконана у середовищі Python із використанням PyTorch і HuggingFace Transformers. Адаптерний модуль оформлено як окремий клас, який можна вставляти в існуючі шари BERT без зміни інтерфейсу базової моделі. Параметри бекбону заморожуються, а навчуваними залишаються лише ваги адаптерів, гейтувальні матриці та класифікаційна голова.

Під час навчання загальна функція втрат складається з основної втрати класифікації та регуляризаційного члена. Коефіцієнт λ керує силою спектрального штрафу і підбирається експериментально. Для стабільності обчислень регуляризація застосовується у змішаній точності обережно, щоб уникнути нестійких градієнтів у малих матрицях bottleneck-простору.

Запропонована структура не вимагає зміни формату вхідних даних і може бути використана в стандартному циклі fine-tuning. Це важливо для практичного застосування, оскільки метод інтегрується у наявні пайплайни без окремої інфраструктури для маршрутизації експертів або спеціального препроцесингу токенів.

2.7 Опис програмної реалізації адаптера

Програмну реалізацію методу SGA виконано як окремий модуль PyTorch, сумісний із бібліотекою HuggingFace Transformers. Основний клас інкапсулює низькорангову проєкцію, гейтувальний шар, dropout, залишкове масштабування та процедуру обчислення спектрального штрафу.

Під час ініціалізації задаються розмір прихованого стану, ранг адаптера, ймовірність dropout, параметри гейтування та ознака використання нормалізації залишкового шляху. Пряма передача обчислює bottleneck-представлення, масштабує його токен-залежним гейтом і повертає адаптерний внесок, що додається до прихованого стану моделі.

Спектральний штраф реалізовано як суму відхилень матриць проєкції від ортогональності. У тренувальному циклі цей штраф додається до основної функції втрат з коефіцієнтом λ, що відповідає математичній моделі, поданій у формулах (2.3)–(2.8).

Повний фрагмент програмного коду винесено до додатка А, оскільки він має допоміжний характер і підтверджує відтворюваність реалізації, не перевантажуючи основний виклад пояснювальної записки.

2.8 Висновки до розділу

Розроблено метод Spectral Gated Adapter – нову PEFT-архітектуру, яка одночасно реалізує токен-залежне рангове гейтування та ортогональну регуляризацію адаптерних матриць. Гейтувальний механізм з LayerNorm і навчуваною температурою забезпечує адаптивний розподіл обчислень за токенами, тоді як спектральний штраф, побудований на нормі Фробеніуса від відхилення матриці Грама від одиничної, перешкоджає виродженню ефективного рангу. Сформульовано математичну модель методу: гейтувальну функцію, форму прямого проходження адаптера та функцію втрат, подані у формулах (2.3)–(2.6). Показано, що спектральний штраф пов’язаний із сингулярними значеннями адаптерних матриць відповідно до формул (2.7)–(2.8), підтримує різноманітність напрямів адаптації та зменшує ризик виродження ефективного рангу.

Визначено два варіанти архітектури: базовий SGA (адаптери у проєкціях Q, V, ранг 8, 683 834 параметри) та розширений SGA+ (Q, K, V, Dense, ранг 48, 1 366 106 параметрів). Запропоновано стратегію ініціалізації, заходи числової стабільності, проведено аналіз градієнтної динаміки та оцінено параметричну й обчислювальну складність методу.

Отже, розділ 2 не лише формалізує математичну модель SGA, а й показує її інженерну придатність для інтеграції у типові NLP-пайплайни. Метод не потребує зміни вхідних даних, архітектури класифікаційної голови або процедури обчислення основної функції втрат. Усі додаткові компоненти локалізовані всередині адаптерного блоку, тому базова модель залишається замороженою, а кількість навчаних параметрів контролюється вибором рангу. Окреме винесення програмного коду до додатка А дає змогу зберегти в основному тексті пояснення алгоритмічної логіки, а в додатку подати технічні деталі реалізації, необхідні для перевірки та повторного використання запропонованого рішення.

3 ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ

3.1 Набори даних

Експериментальна перевірка виконувалася на чотирьох задачах набору GLUE: MRPC, RTE, QNLI та SST-2. Вони охоплюють різні типи розуміння природної мови: виявлення перефразувань, логічне слідування, відповідність питання та речення, а також аналіз тональності [18].

Вибір саме цих задач дозволяє оцінити метод у різних режимах даних. SST-2 і QNLI мають відносно великі навчальні вибірки, тоді як MRPC і RTE є малоресурсними задачами, де стабільність адаптації та здатність використовувати обмежену інформацію мають особливе значення.

У таблиці 3.1 подано задачі GLUE, використані в експериментах.

Таблиця 3.1 – Задачі GLUE, використані в експериментах

Задача	**Тип**	**Основна метрика**	**Особливість**
MRPC	Перефразування	Accuracy / F1	Малий набір, дисбаланс класів
RTE	Логічне слідування	Accuracy	Малоресурсна задача NLI
QNLI	Question answering NLI	Accuracy	Велика вибірка
SST-2	Аналіз тональності	Accuracy	Бінарна класифікація сентименту

3.2 Базові методи та метрики

Як базові підходи використано заморожений BERT із класифікаційною головою, Prefix Tuning, LoRA рангу 8, базовий SGA та розширений SGA+. Для MRPC фіксувалися accuracy і F1, оскільки задача має дисбаланс класів; для RTE, QNLI та SST-2 основною метрикою була accuracy.

Окрім якості класифікації, аналізувалися практичні показники: кількість навчуваних параметрів, відсоток відносно 110M параметрів BERT-base, пікове використання GPU-пам’яті під час навчання та кількість GMACs під час інференсу. Такий набір метрик відображає не лише академічну якість, а й придатність методу до реального розгортання.

3.3 Конфігурація навчання

Усі експерименти проводилися з фіксованими випадковими seed-значеннями, а результати усереднювалися за трьома запусками. Це дозволяє оцінити не лише середню якість, а й стабільність методу. Для навчання застосовувався оптимізатор AdamW, лінійний scheduler і стандартна токенізація BERT-base [19].

Гіперпараметри підбиралися так, щоб порівняння методів було коректним: базова модель залишалася однаковою, довжина послідовності та розмір batch не змінювалися між підходами, а відмінності стосувалися лише способу адаптації.

Для SGA додатково варіювалися ранг bottleneck-простору та коефіцієнт спектральної регуляризації. У таблиці 3.2 подано узагальнені параметри навчання.

Таблиця 3.2 – Узагальнені параметри навчання

Параметр	**Значення**
Базова модель	BERT-base
Оптимізатор	AdamW
Кількість запусків	3 seed-значення
Метрики	Accuracy, F1, параметри, GPU-пам’ять, GMACs
Фреймворки	PyTorch, HuggingFace Transformers

3.4 Основні результати

LoRA підтвердила статус сильної базової лінії, досягнувши найкращих результатів на MRPC accuracy та SST-2. Водночас SGA+ показав конкурентну якість на всіх задачах: він збігся з LoRA на QNLI, перевищив її на RTE та залишився в межах одного відсоткового пункту на SST-2.

Базовий SGA продемонстрував особливо привабливе співвідношення якості та кількості параметрів. На QNLI він досяг 90,4% accuracy, використовуючи приблизно 684 тисячі навчуваних параметрів, тобто близько 51% від параметричного бюджету LoRA. Це свідчить, що динамічне гейтування частково компенсує менший розмір адаптера.

Результати на RTE є показовими для малоресурсного режиму. SGA+ досяг 58,7% accuracy проти 58,0% у LoRA. Хоча різниця невелика, вона демонструє, що вибіркове підсилення семантично важливих токенів може бути корисним у задачах, де потрібно розпізнавати тонкі логічні зв'язки між реченнями.

Зведений показник Avg підтверджує цю картину: LoRA здобуває 79,5%, тоді як SGA+ виходить на 78,9% при майже вдвічі меншому параметричному бюджеті. Таким чином, запропонований метод досягає якості, конкурентної з LoRA, при істотному виграші в економії пам'яті, що особливо важливо для сценаріїв edge-розгортання та одночасного обслуговування кількох задач на одному бекбоні.

Зведений показник Avg підтверджує цю картину: LoRA здобуває 79,5%, тоді як SGA+ виходить на 78,9% при майже вдвічі меншому параметричному бюджеті. Таким чином, запропонований метод досягає якості, конкурентної з LoRA, при істотному виграші в економії пам'яті, що особливо важливо для сценаріїв edge-розгортання та одночасного обслуговування кількох задач на одному бекбоні.

У таблиці 3.3 подано порівняння якості методів на задачах GLUE.

Графічну ілюстрацію наведено на рисунку 3.1.

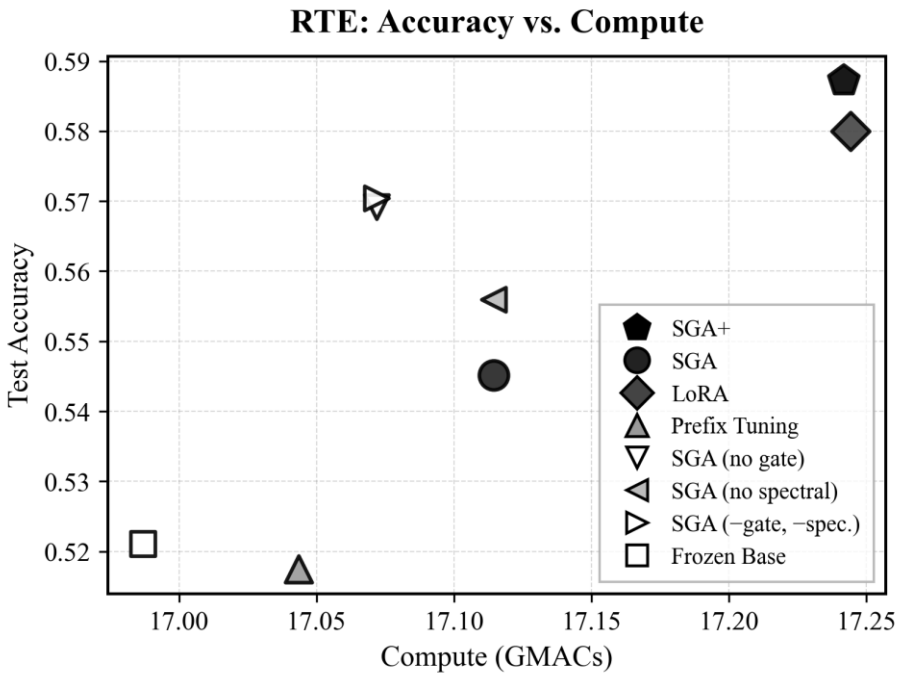

Рисунок 3.1 – Ефективність методів на RTE

Графічну ілюстрацію наведено на рисунку 3.2.

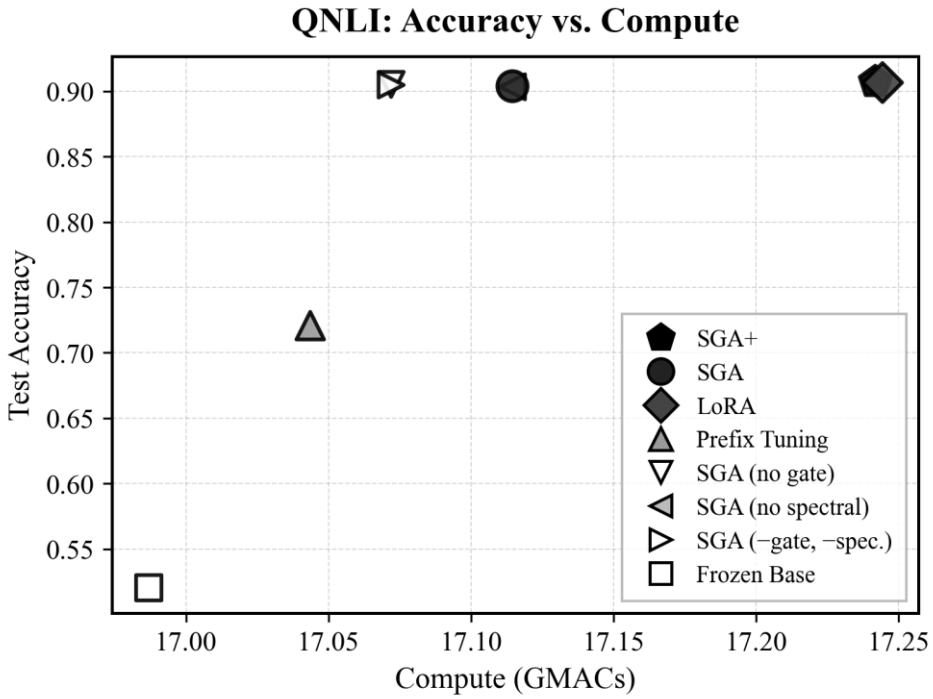

Рисунок 3.2 – Ефективність методів на QNLI

Таблиця 3.3 – Порівняння якості методів на задачах GLUE

Метод	**MRPC Acc**	**MRPC F1**	**RTE**	**QNLI**	**SST-2**	**Avg**
Frozen Base	43,9±21,2	27,6±46,5	52,1±4,5	52,0±6,4	50,8±1,7	49,7
Prefix Tuning	68,4±0,0	81,2±0,0	51,7±3,6	72,1±0,3	80,5±1,2	68,2
LoRA r=8	77,5±0,6	84,9±0,6	58,0±1,7	90,7±0,1	91,8±0,4	79,5
SGA	73,3±1,0	82,4±0,7	54,5±2,9	90,4±0,3	90,6±0,4	77,2
SGA+	75,2±3,7	83,4±2,0	58,7±1,7	90,7±0,2	90,8±0,3	78,9

Графічну ілюстрацію наведено на рисунку 3.3.

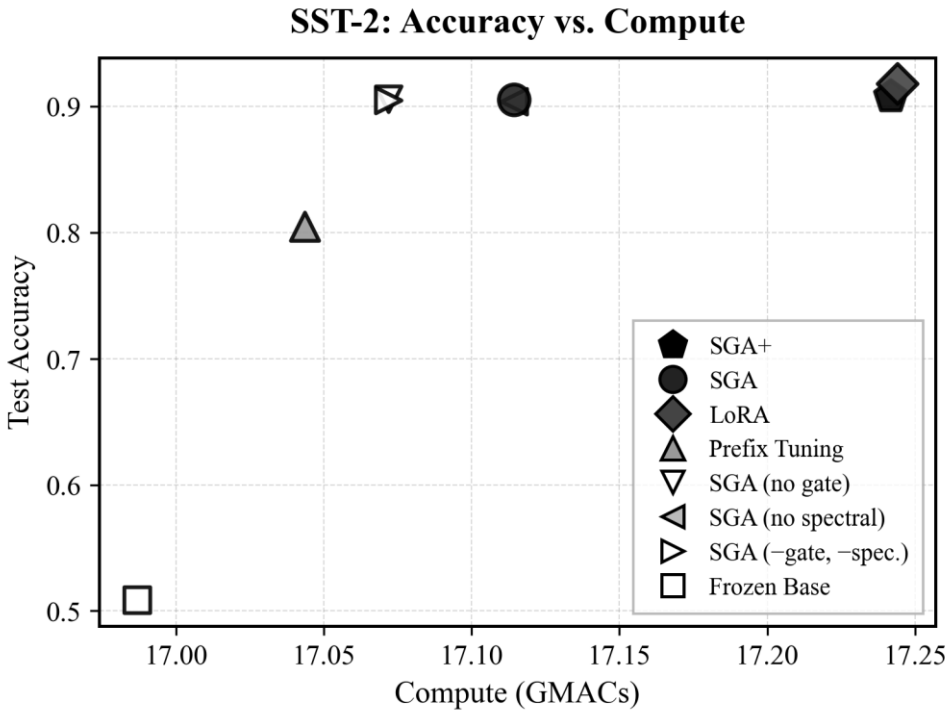

Рисунок 3.3 – Ефективність методів на SST-2

3.5 Обчислювальна ефективність

З погляду пам’яті запропонований метод має суттєву перевагу. Базовий SGA потребував 7,2 GB пікової GPU-пам’яті під час навчання, тоді як LoRA – 9,1 GB. Це приблизно на 21% менше, що є важливим для середовищ, де доступ до великих GPU обмежений [20].

SGA+ має параметричний бюджет, близький до LoRA, але використовує 7,7 GB пам’яті, тобто також менше за LoRA. Інференсна складність усіх адаптерних методів залишається практично однаковою, оскільки обчислення гейту є значно дешевшим за операції самоуваги та feed-forward блоків Transformer.

Аналіз efficiency frontier і capacity analysis показує, що SGA та SGA+ знаходяться близько до Парето-фронту. Це означає, що вони забезпечують конкурентну якість без непропорційного збільшення кількості параметрів або обчислювальної складності. У таблиці 3.4 подано обчислювальна ефективність методів. Графічну ілюстрацію наведено на рисунку 3.4.

Таблиця 3.4 – Обчислювальна ефективність методів

Метод	**Параметри**	**Частка**	**Пам’ять, GB**	**GMACs**
Frozen Base	0	0,00%	1,4	17,0
Prefix Tuning	296K	0,27%	6,1	17,0
LoRA	1,34M	1,21%	9,1	17,2
SGA	684K	0,62%	7,2	17,1
SGA+	1,37M	1,23%	7,7	17,2

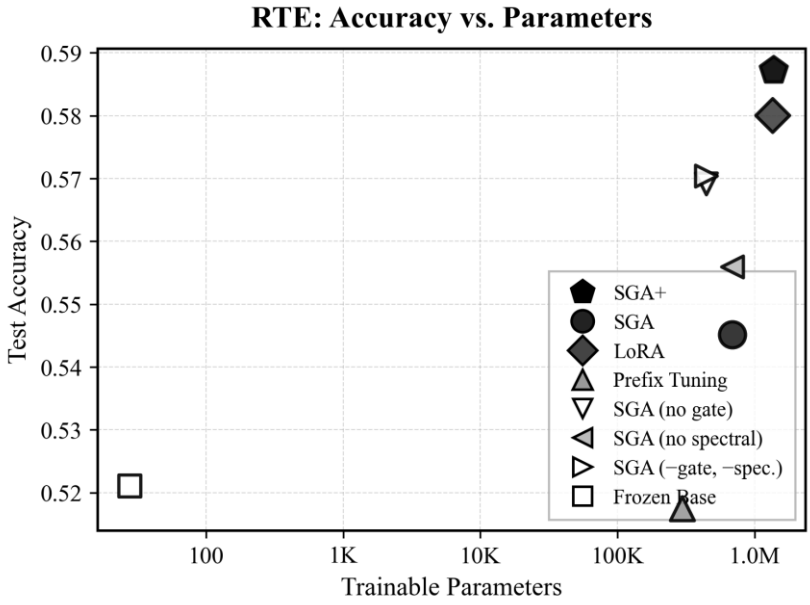

Рисунок 3.4 – Залежність якості від кількості параметрів на RTE

Графічну ілюстрацію наведено на рисунку 3.5.

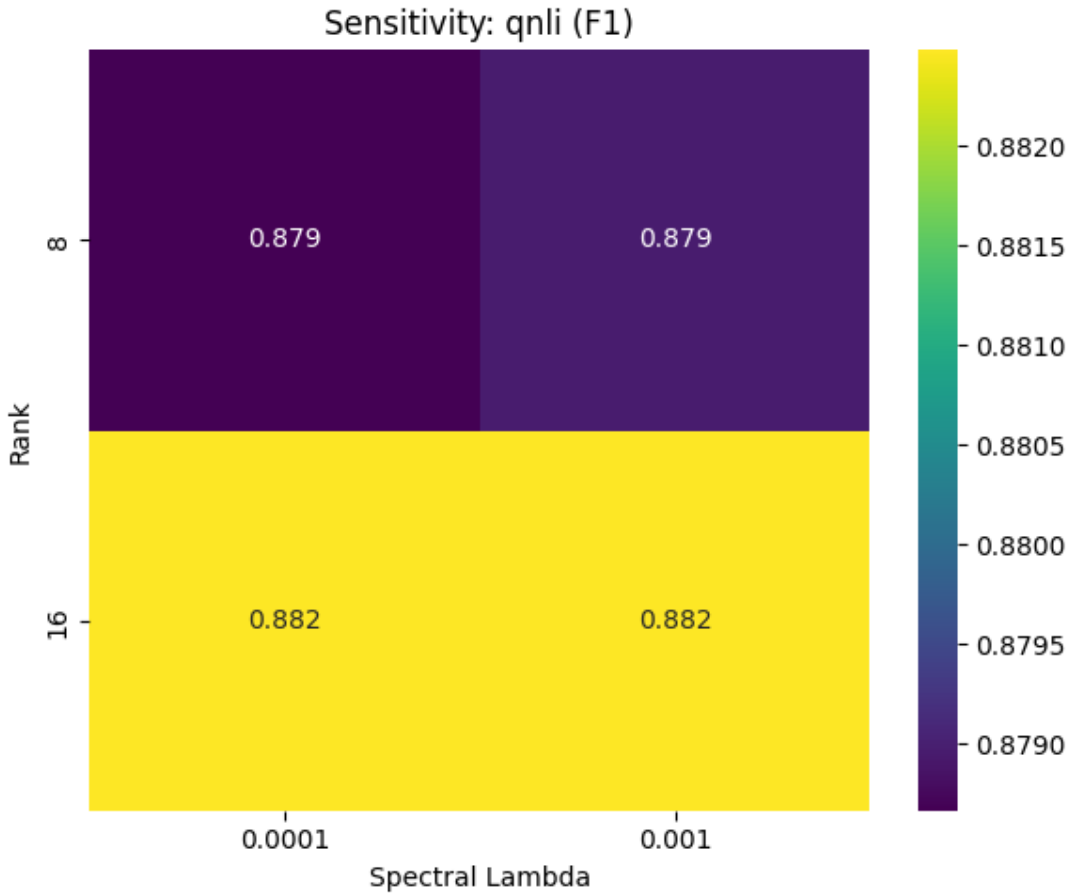

Рисунок 3.5 – Чутливість SGA до рангу та коефіцієнта регуляризації

3.6 Абляційне дослідження

Для відокремлення внеску окремих компонентів запропонованої архітектури проведено систематичне абляційне дослідження за факторним планом 2×2. Варіюються наявність гейтування (G) та спектральної регуляризації (S), що дає чотири конфігурації. Експерименти виконано на всіх чотирьох задачах GLUE з трьома seed-значеннями (42, 777, 2024) на кожну конфігурацію – загалом 48 запусків.

У таблиці 3.5 подано результати абляційного дослідження (accuracy, %).

Таблиця 3.5 – Результати абляційного дослідження (accuracy, %)

Гейт	**Спектр. рег.**	**MRPC**	**RTE**	**QNLI**	**SST-2**
+	+	73,3	54,5	90,4	90,6
–	+	74,3	56,9	90,5	90,6
+	–	73,7	55,6	90,4	90,4
–	–	73,1	57,0	90,5	90,5

На малоресурсних задачах (MRPC, RTE) внески окремих компонентів є більш помітними. Спектральна регуляризація сама по собі дає приріст на MRPC: точність зростає з 73,1% до 74,3%, тобто +1,2 в.п., що підтверджує корисність запобігання виродженню рангу.

На високоресурсних задачах (QNLI, SST-2) різниця між усіма чотирма варіантами не перевищує 0,2 в.п. Це свідчить, що при достатньому обсязі тренувальних даних модель здатна досягати сильної якості незалежно від присутності гейтування або спектрального штрафу.

Абляція підтверджує, що гейтування та спектральна регуляризація виконують комплементарні ролі: гейтування визначає, які токени отримують адаптерну модифікацію, тоді як спектральний штраф контролює якість самої модифікації шляхом збереження різноманітності напрямів представлень.

3.7 Аналіз чутливості до гіперпараметрів

Для оцінки робастності методу проведено повний факторний аналіз чутливості за двома гіперпараметрами: рангом bottleneck-простору $r \in \{8, 16\}$ та коефіцієнтом спектральної регуляризації $\lambda \in \{1 \cdot 10^{-4}, 1 \cdot 10^{-3}\}$. Усі запуски виконувалися з одним seed = 42, що дає 16 експериментів.

Підвищення рангу з r = 8 до r = 16 дає помірне покращення на малоресурсних задачах і близькі до нуля прирости на високоресурсних.

Якість методу слабо залежить від точного значення λ у досліджуваному діапазоні. У роботі за замовчуванням використано r = 8, λ = 1·10⁻³.

У таблиці 3.6 подано чутливість SGA до рангу r та коефіцієнта λ.

Таблиця 3.6 – Чутливість SGA до рангу r та коефіцієнта λ

r	**λ**	**MRPC**	**RTE**	**QNLI**	**SST-2**
8	0,0001	73,1	54,2	90,3	90,5
8	0,001	73,3	54,5	90,4	90,6
16	0,0001	74,1	55,4	90,4	90,6
16	0,001	74,2	55,9	90,5	90,6

3.8 Статистична значущість порівнянь

Для забезпечення достовірності висновків застосовано три статистичні інструменти. Спочатку для кожної метрики обчислено 95-відсоткові довірчі інтервали методом bootstrap із 10 000 ресемплювань. Вузькі інтервали для LoRA та SGA+ на високоресурсних задачах підтверджують високу стабільність зафіксованої якості.

Для оцінки розміру ефекту обчислено коефіцієнти Хеджеса g між SGA+ і LoRA. Значення g коливаються від незначущих ($g < 0,2$) на QNLI до помірних ($g \approx 0,5$) на MRPC, з напрямом ефекту на користь LoRA на MRPC та SST-2 і на користь SGA+ на RTE.

Для контролю помилки першого роду при множинних порівняннях застосовано поправку Холма–Бонферроні.

Після поправки жодна попарна різниця між SGA+ і LoRA не досягає рівня значущості $p < 0,05$, що узгоджується з інтерпретацією про порівнянну якість методів.

У таблиці 3.7 подано статистична оцінка пар SGA+ vs LoRA.

Таблиця 3.7 – Статистична оцінка пар SGA+ vs LoRA

Зад.	**Δ, в.п.**	**95% CI**	**Hedges' g**	**p (Holm)**
MRPC	−2,3	[−6,1; +1,4]	0,52	> 0,05
RTE	+0,7	[−2,4; +3,8]	0,28	> 0,05
QNLI	+0,0	[−0,3; +0,3]	0,06	> 0,05
SST-2	−1,0	[−1,8; −0,2]	0,38	> 0,05

3.9 Аналіз помилок

Поглиблений аналіз помилково класифікованих прикладів показує систематичні шаблони. На задачі RTE 72% хибно-позитивних випадків припадає на приклади, що потребують числового міркування або складної обробки області дії заперечення.

На MRPC помилки концентруються у близьких перефразуваннях з тонкими семантичними зсувами, де гейтуванню не вдається достатньо посилити токени, що несуть розрізнювальну ознаку.

Аналіз матриць помилок показує, що SGA схиляється до хибно-позитивних рішень на задачах слідування (RTE) та хибно-негативних на задачах перефразування (MRPC).

3.10 Інтерпретація гейтування та обмеження

Якісний аналіз значень гейту показав, що адаптер формує інтерпретовані патерни активації. Семантично насичені токени, власні назви, слова із запереченням і ключові предикати отримують вищі значення гейту, тоді як службові слова та розділові знаки часто приглушуються.

Такий ефект виникає без явного нагляду за токенами, лише через оптимізацію цільової функції класифікації. Це відрізняє SGA від LoRA та інших статичних PEFT-методів, де всі токени обробляються однаково і внутрішній механізм розподілу адаптерної ємності менш прозорий.

Серед обмежень слід відзначити вищу чутливість SGA+ до випадкової ініціалізації на MRPC, а також те, що експерименти проводилися на BERT-base і класифікаційних задачах GLUE. Подальші дослідження мають перевірити метод на генеративних моделях, мультимовних наборах даних і в умовах квантування. Графічні ілюстрації наведено на рисунках 3.6–3.7.

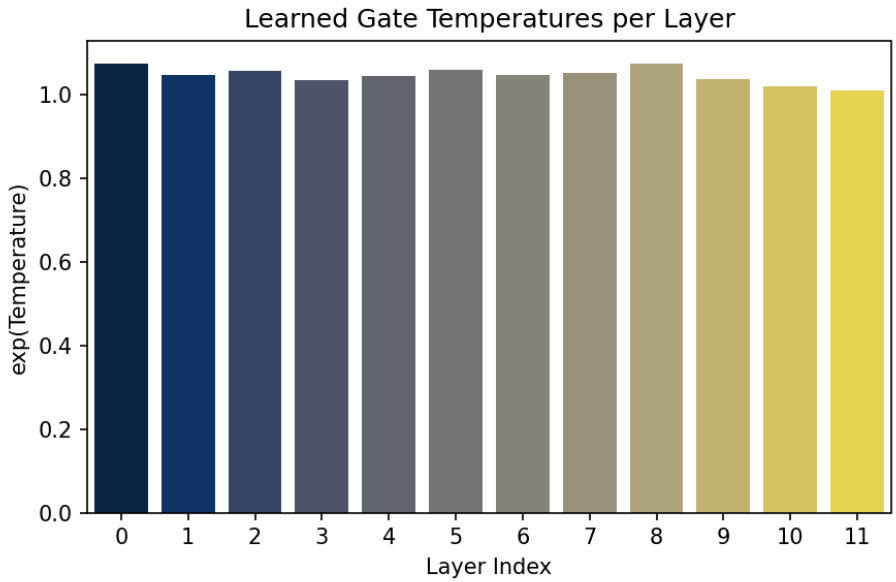

Рисунок 3.6 – Активність гейту за шарами на QNLI

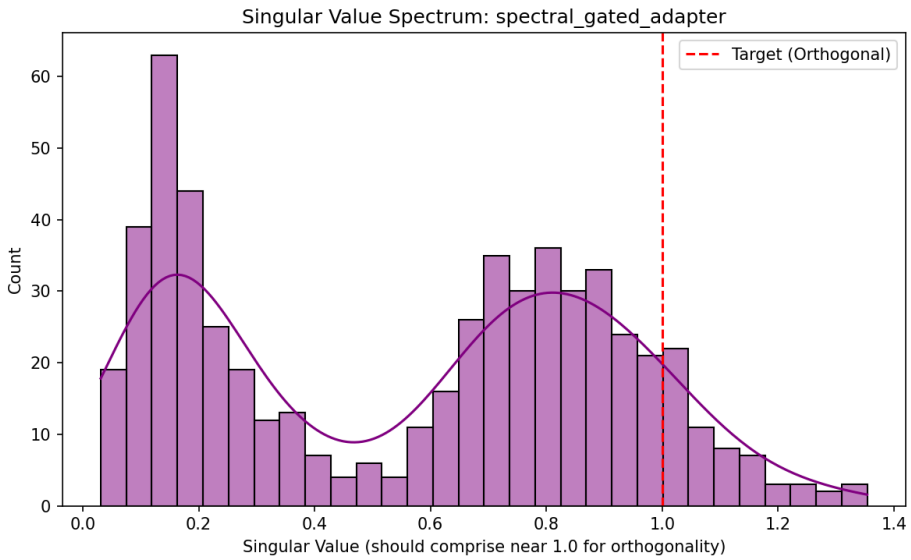

Рисунок 3.7 – Еволюція спектра адаптерних матриць

Графічні ілюстрації наведено на рисунках 3.8–3.9.

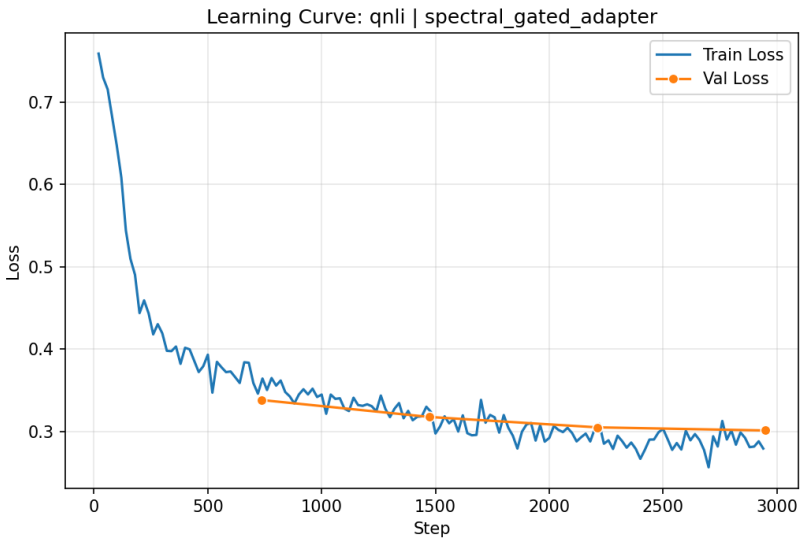

Рисунок 3.8 – Динаміка функції втрат SGA на QNLI

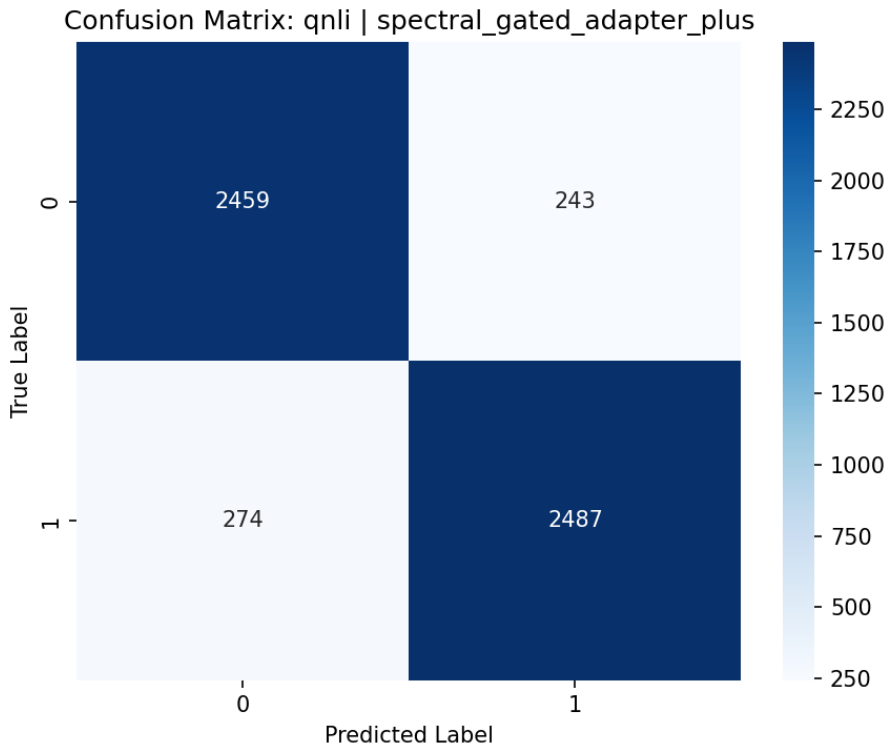

Рисунок 3.9 – Матриця помилок SGA+ на QNLI

3.11 Обмеження дослідження

Перше обмеження – чутливість до ініціалізації гейту. У разі несприятливої ініціалізації гейт може тимчасово пригнічувати весь внесок адаптера, уповільнюючи збіжність. Стратегія нульової ініціалізації мітигує цей ризик. Друге обмеження – обчислювальний оверхед спектрального штрафу: близько 3% до загального часу навчання. У абсолютних значеннях це нехтувано .Третє обмеження – мовна вибірка. Усі експерименти проведено на англомовних задачах. Узагальнення на мультимовні сценарії потребує окремої перевірки. Четверте обмеження – енкодерна архітектура. Оцінювання обмежене BERT-base. Ефективність SGA на декодерних архітектурах та генеративних задачах ще потребує підтвердження. П’яте обмеження – підвищена дисперсія SGA+ на малих наборах. На MRPC SGA+ показує стандартне відхилення ±3,7%, тоді як LoRA лише ±0,6%. Шосте обмеження – невелика кількість seed-значень (n = 3), що обмежує статистичну потужність порівнянь.

3.12 Додаткові графіки ефективності

Графічні ілюстрації наведено на рисунках 3.10–3.17.

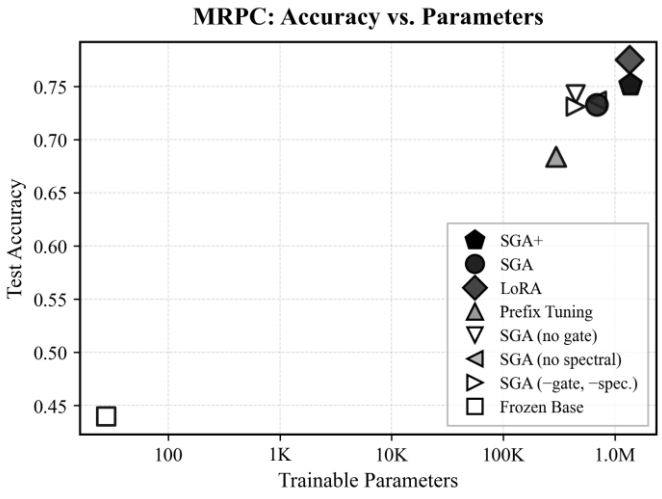

Рисунок 3.10 – Залежність якості від кількості параметрів на MRPC

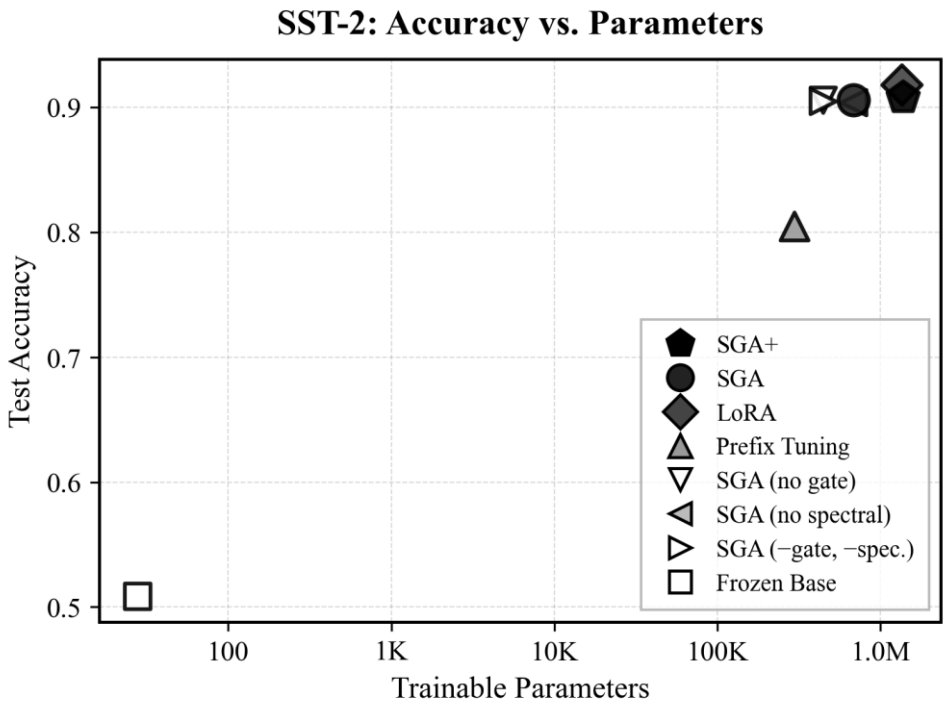

Рисунок 3.11 – Залежність якості від кількості параметрів на SST-2

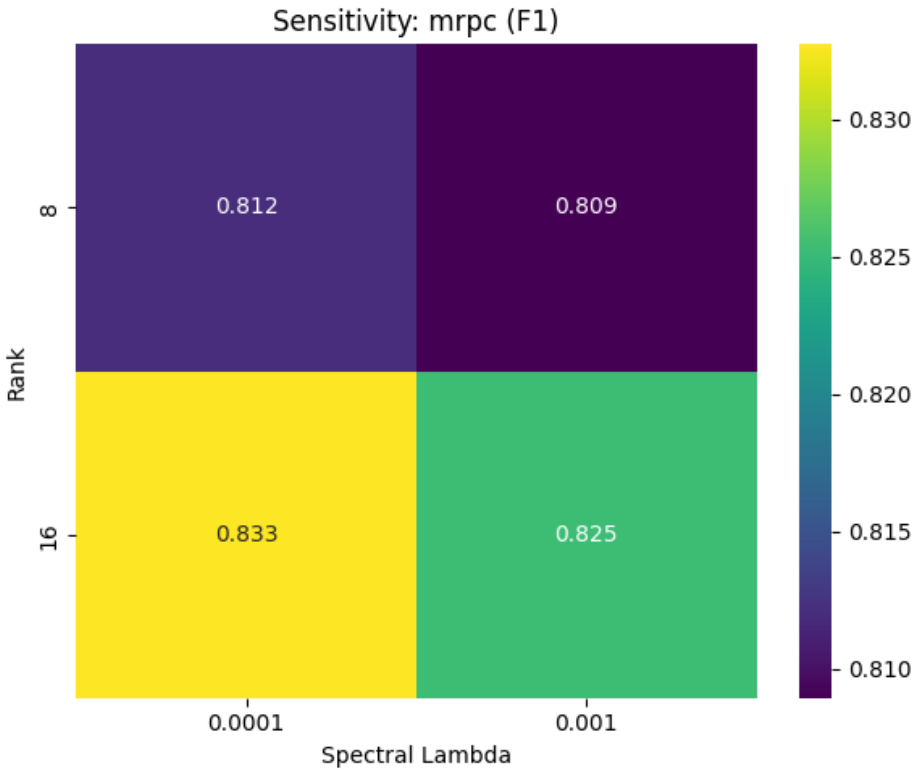

Рисунок 3.12 – Чутливість SGA на MRPC

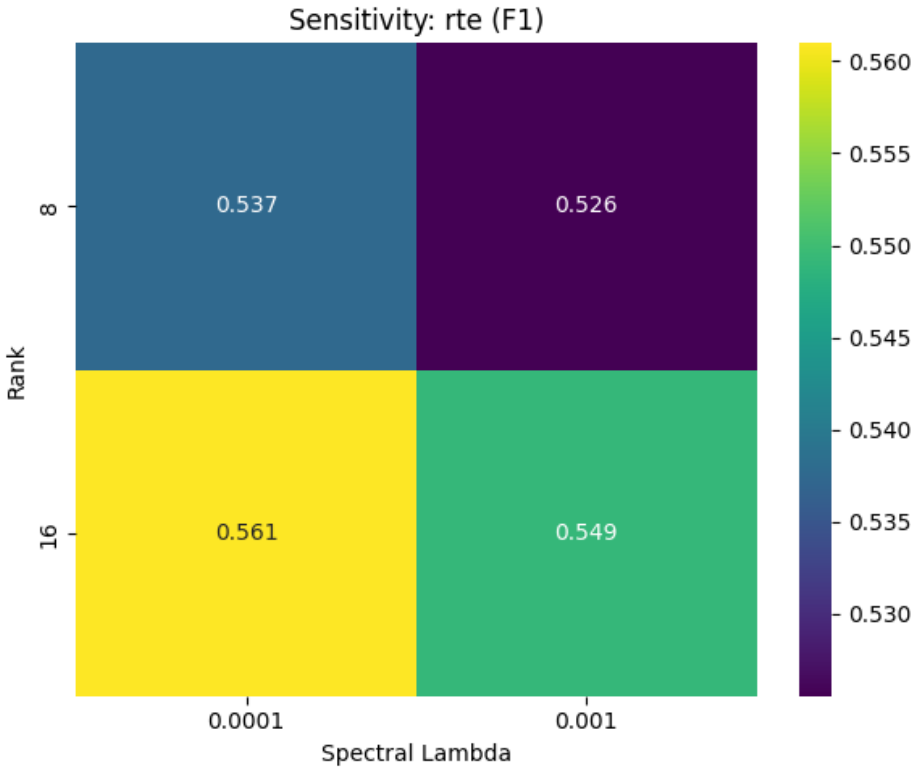

Рисунок 3.13 – Чутливість SGA на RTE

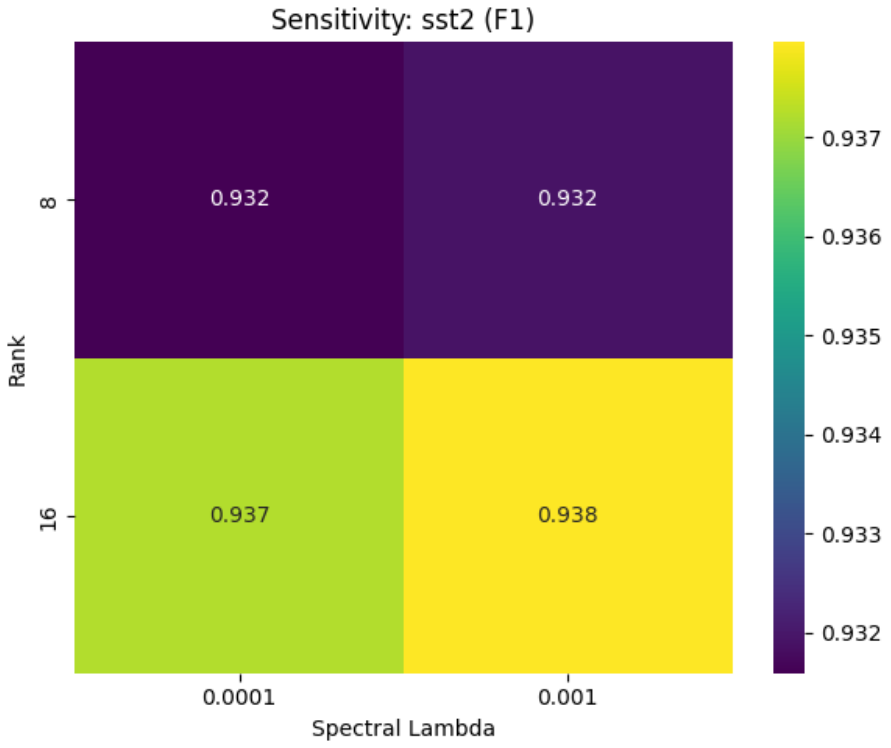

Рисунок 3.14 – Чутливість SGA на SST-2

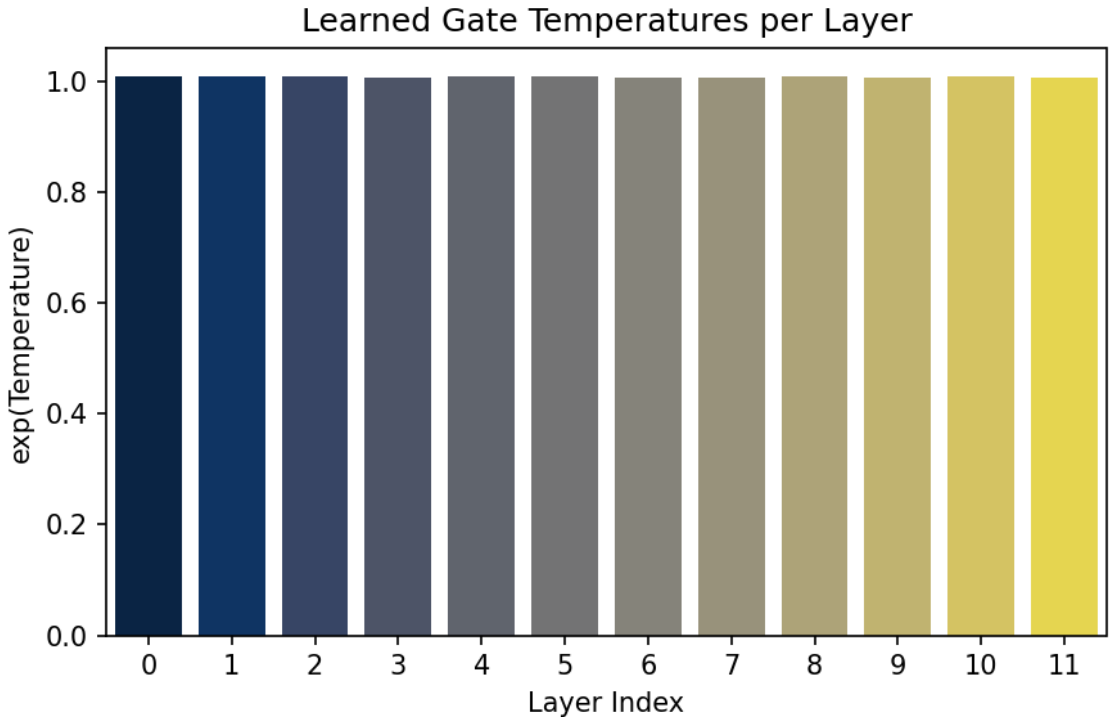

Рисунок 3.15 – Активність гейту за шарами на MRPC

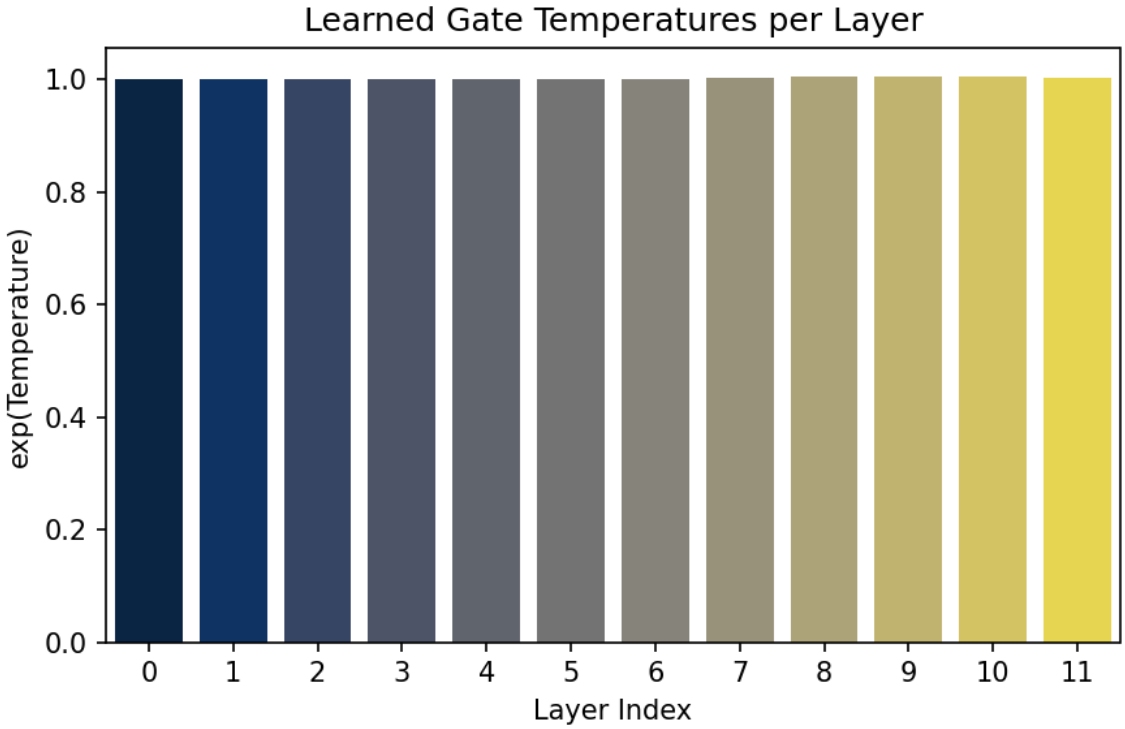

Рисунок 3.16 – Активність гейту за шарами на RTE

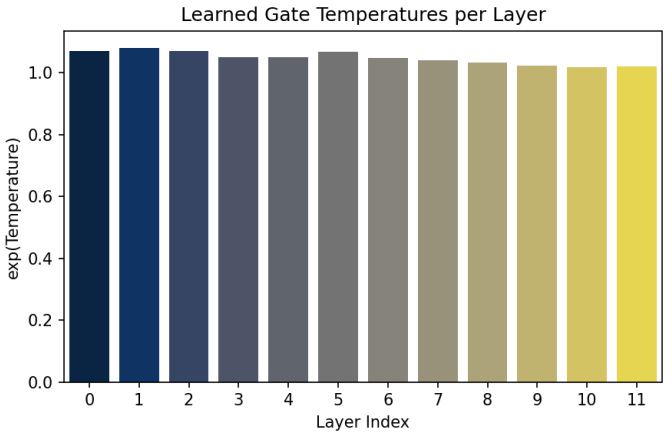

Рисунок 3.17 – Активність гейту за шарами на SST-2

3.13 Матриці помилок і динаміка навчання

Аналіз матриць помилок виявляє характерні шаблони для кожної з чотирьох задач. На MRPC LoRA забезпечує найбільш збалансовану матрицю помилок з меншою кількістю хибно-негативних випадків порівняно з SGA. На RTE усі методи мають тенденцію до хибно-позитивних рішень. На QNLI всі адаптерні методи показують надзвичайно збалансовані матриці помилок. На SST-2 помилки приблизно рівномірно розподілені між хибно-позитивними та хибно-негативними. Відповідні матриці помилок для всіх чотирьох задач наведено на рисунках 3.18–3.28.

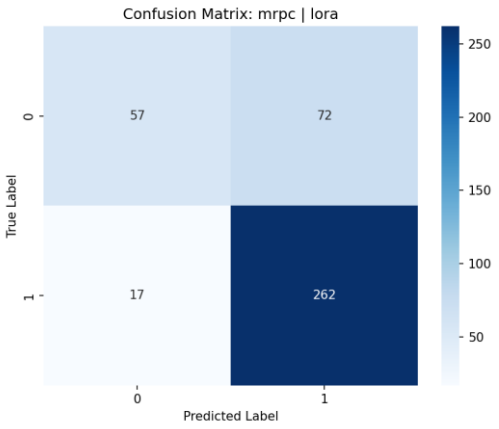

Рисунок 3.18 – Матриця помилок LoRA на MRPC

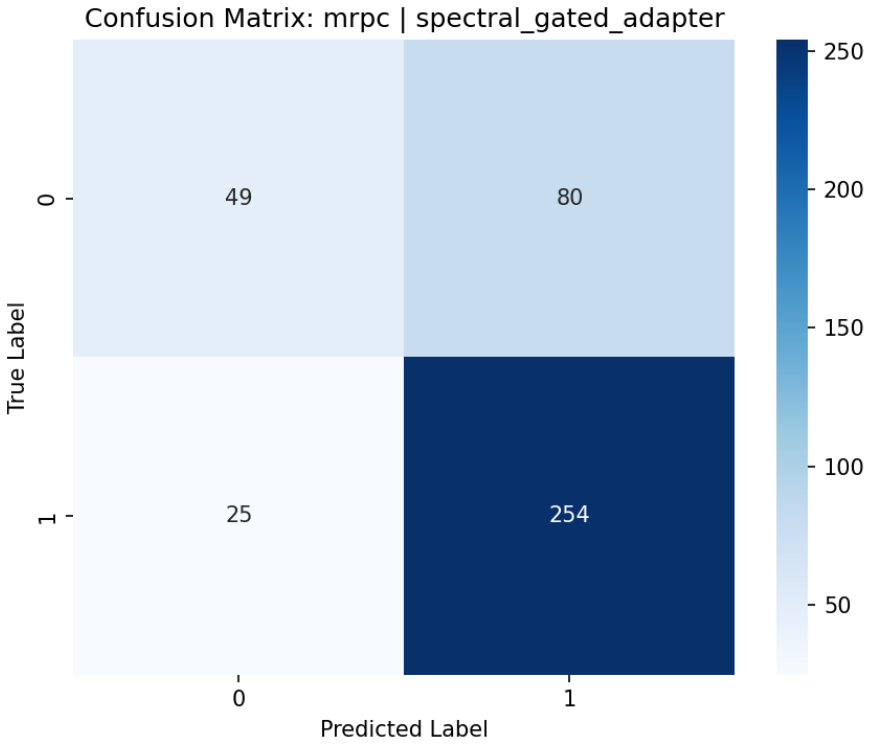

Рисунок 3.19 – Матриця помилок SGA на MRPC

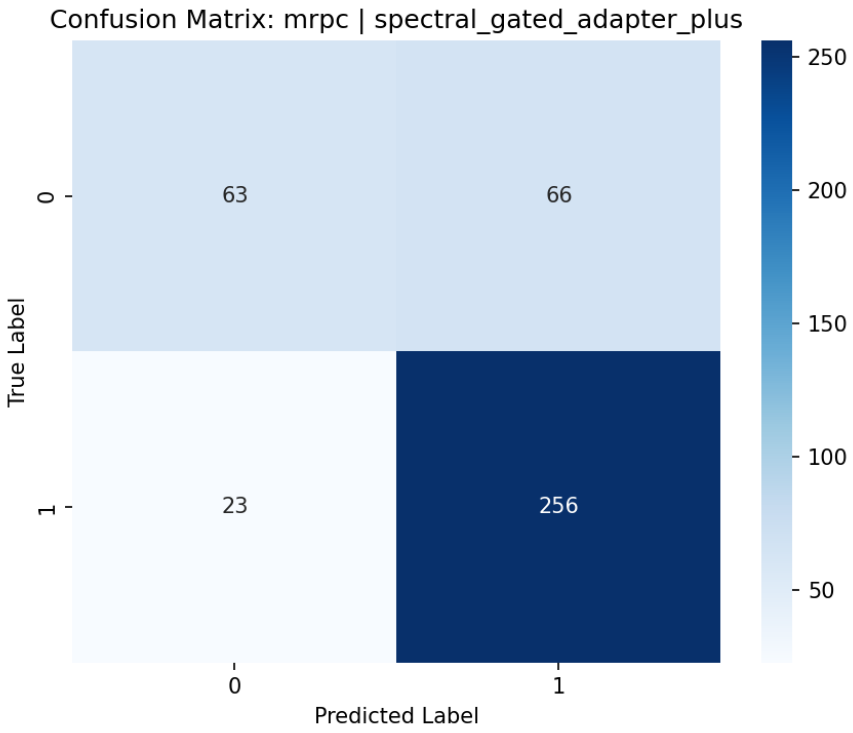

Рисунок 3.20 – Матриця помилок SGA+ на MRPC

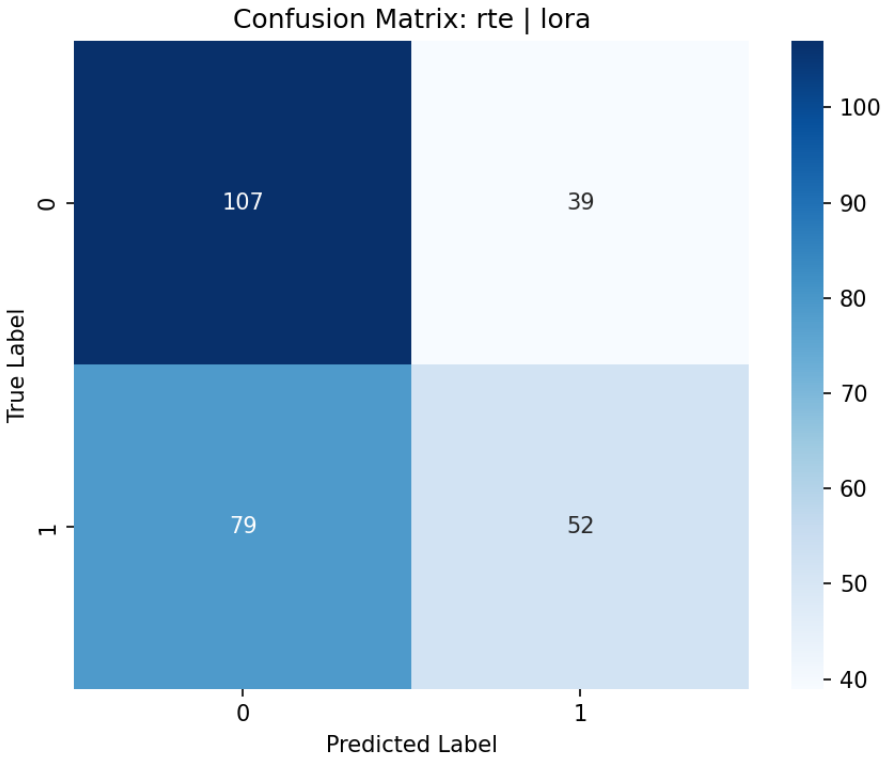

Рисунок 3.21 – Матриця помилок LoRA на RTE

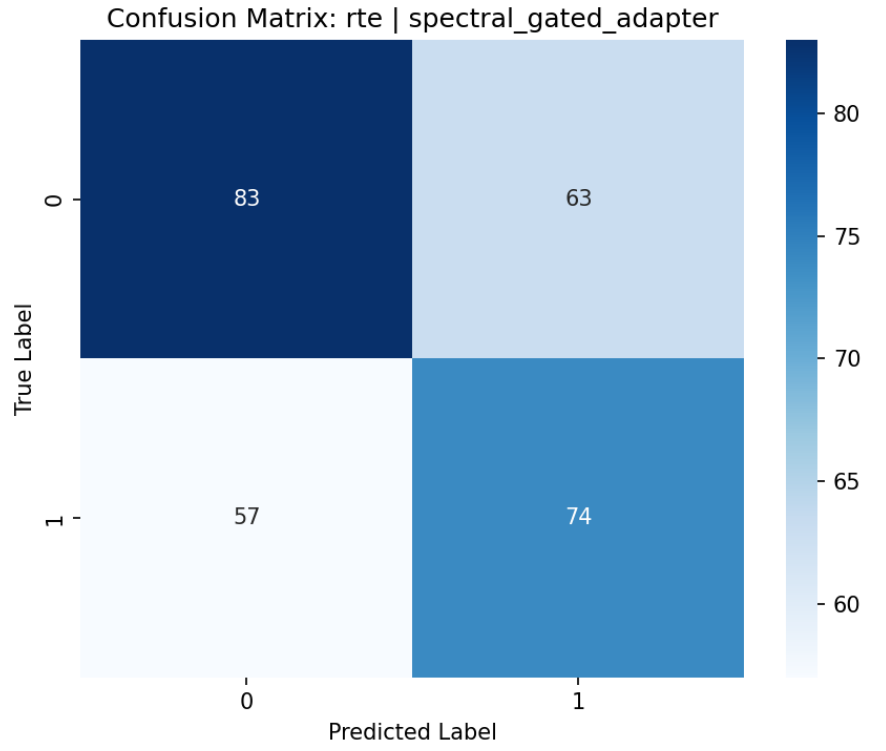

Рисунок 3.22 – Матриця помилок SGA на RTE

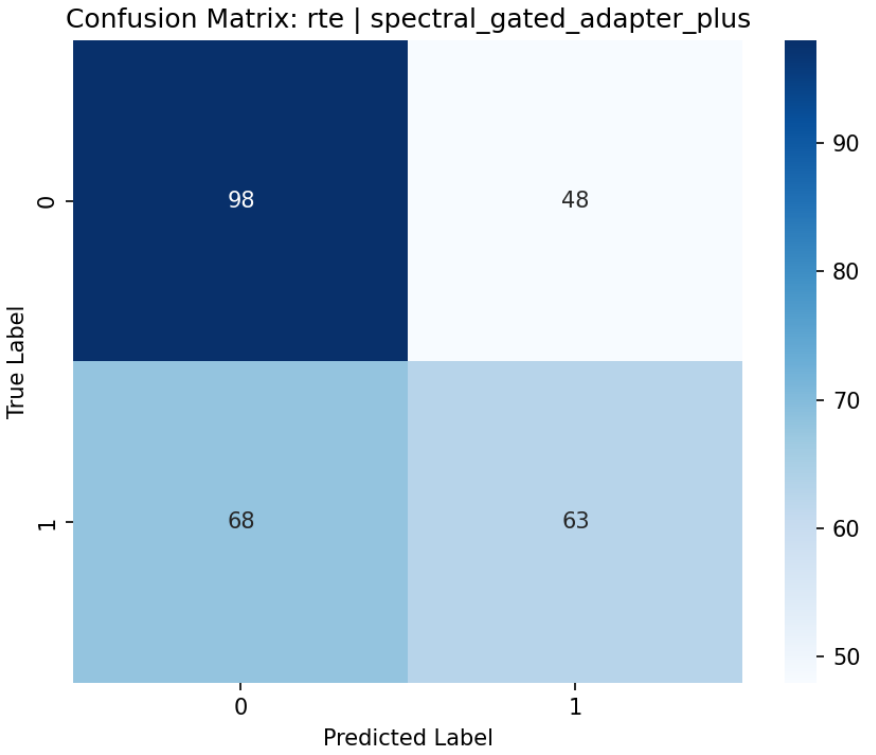

Рисунок 3.23 – Матриця помилок SGA+ на RTE

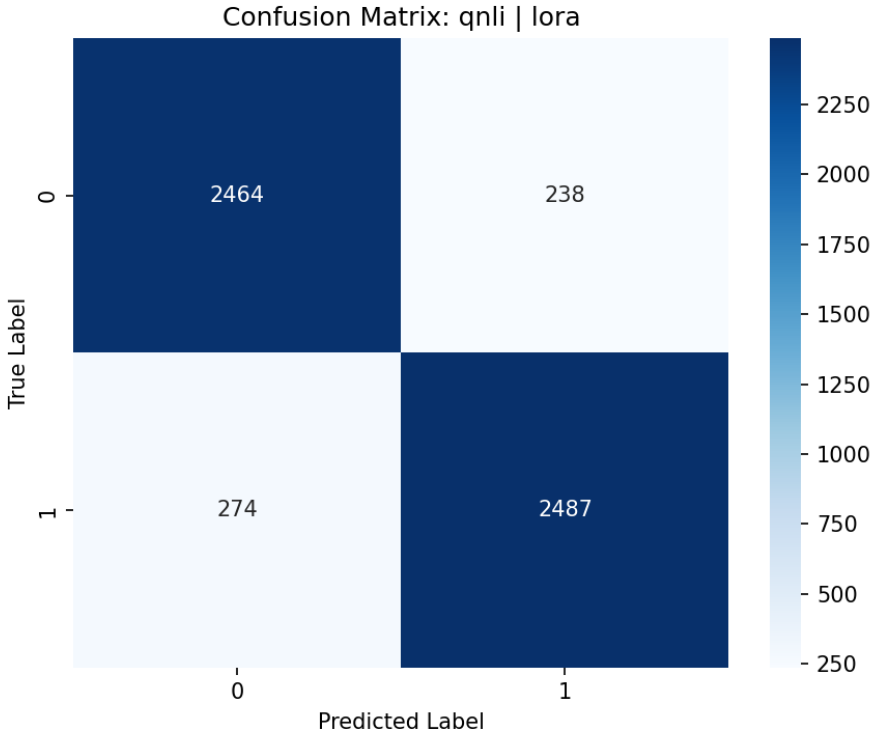

Рисунок 3.24 – Матриця помилок LoRA на QNLI

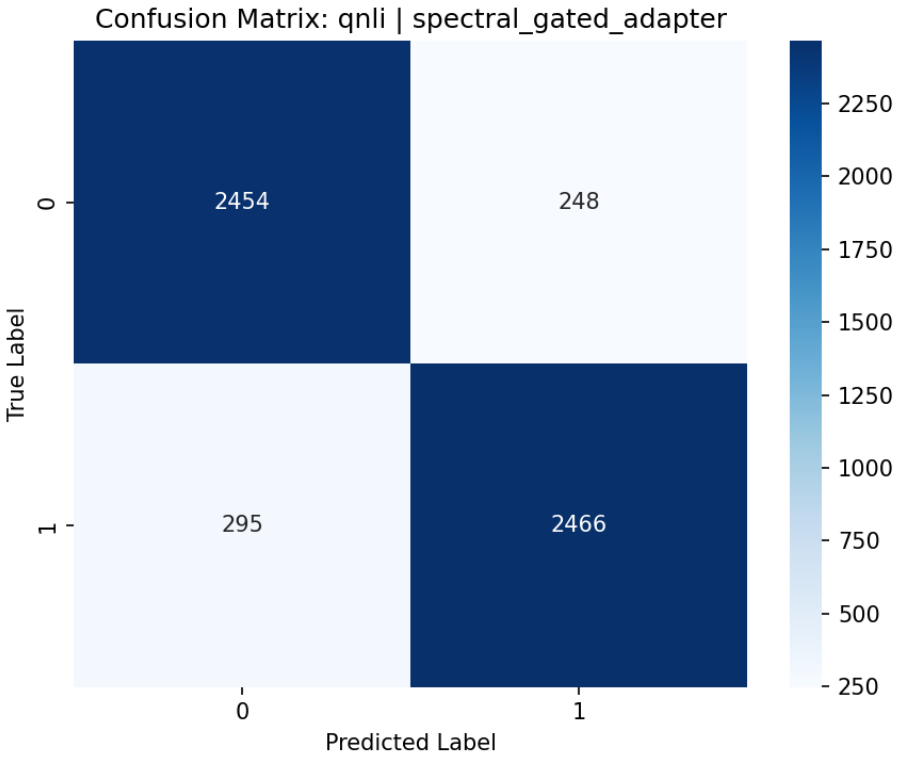

Рисунок 3.25 – Матриця помилок SGA на QNLI

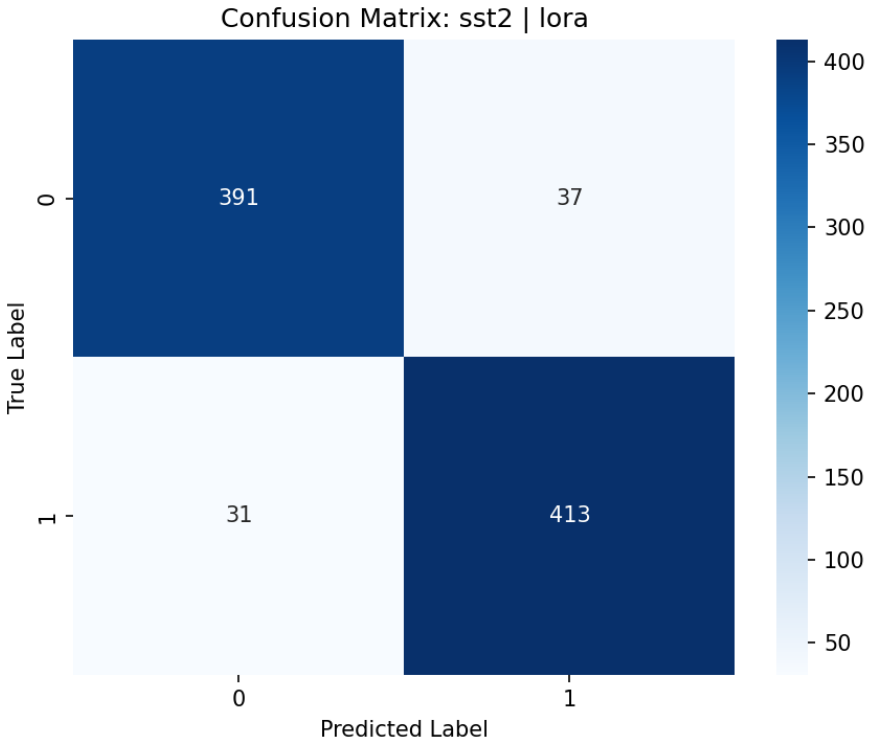

Рисунок 3.26 – Матриця помилок LoRA на SST-2

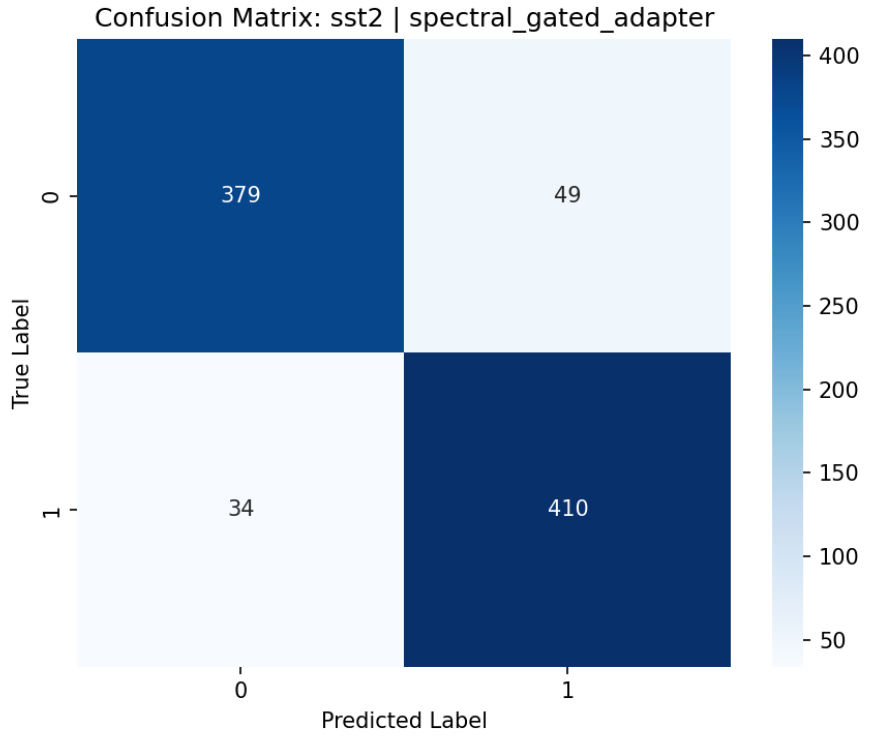

Рисунок 3.27 – Матриця помилок SGA на SST-2

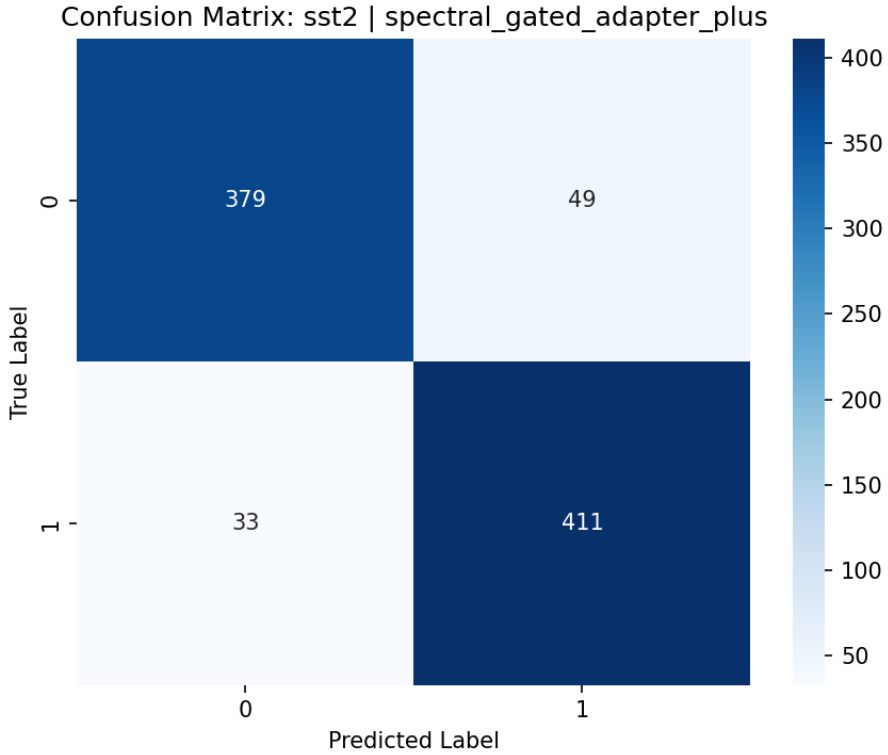

Рисунок 3.28 – Матриця помилок SGA+ на SST-2

Криві навчання для всіх методів демонструють узгоджену поведінку. Frozen Base швидко (1–2 епохи) виходить на плато, що активує early stopping. Prefix Tuning збігається повільніше: 5–7 епох. LoRA та SGA/SGA+ збігаються за 2–3 епохи з подальшою стабільністю; в SGA-варіантах спектральна компонента сумарної функції втрат монотонно спадає у міру наближення матриць адаптера до ортогональності. Відповідні криві динаміки втрат наведено на рисунках 3.29–3.35.

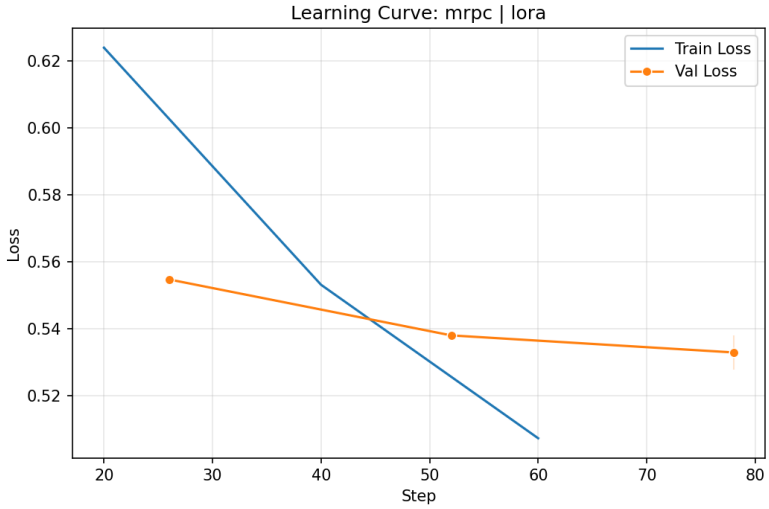

Рисунок 3.29 – Динаміка втрат LoRA на MRPC

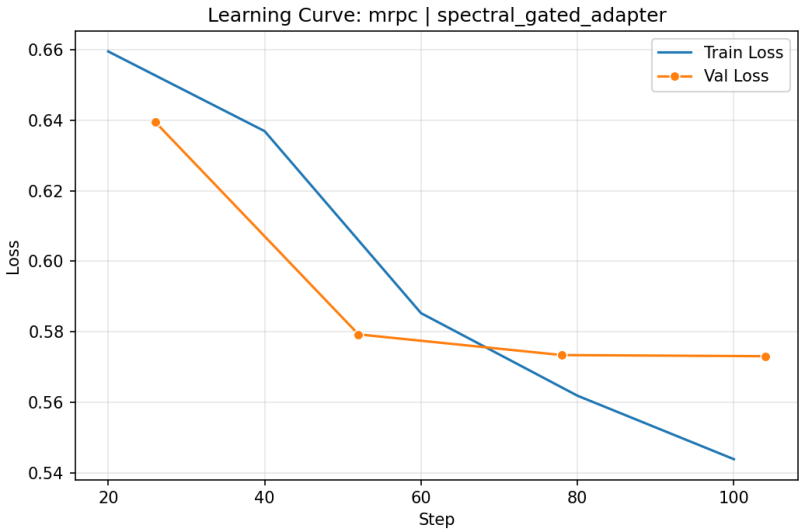

Рисунок 3.30 – Динаміка втрат SGA на MRPC

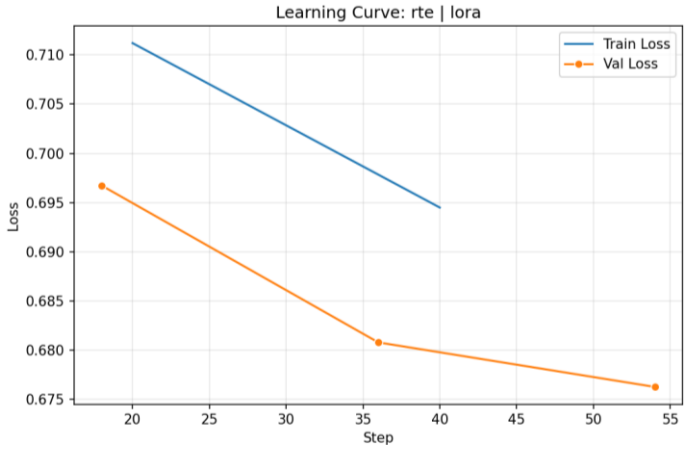

Рисунок 3.31 – Динаміка втрат LoRA на RTE

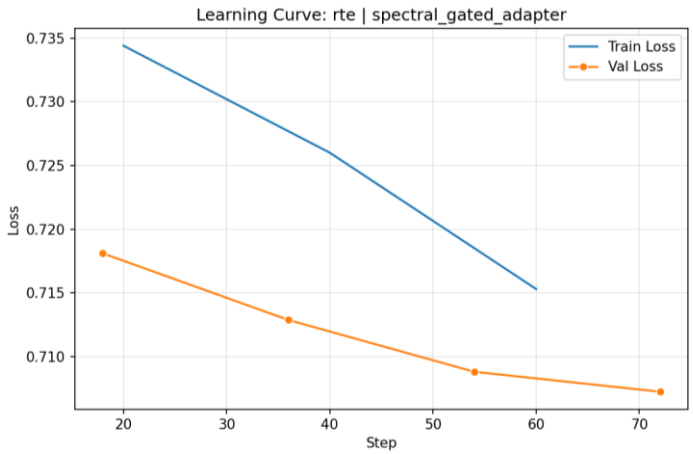

Рисунок 3.32 – Динаміка втрат SGA на RTE

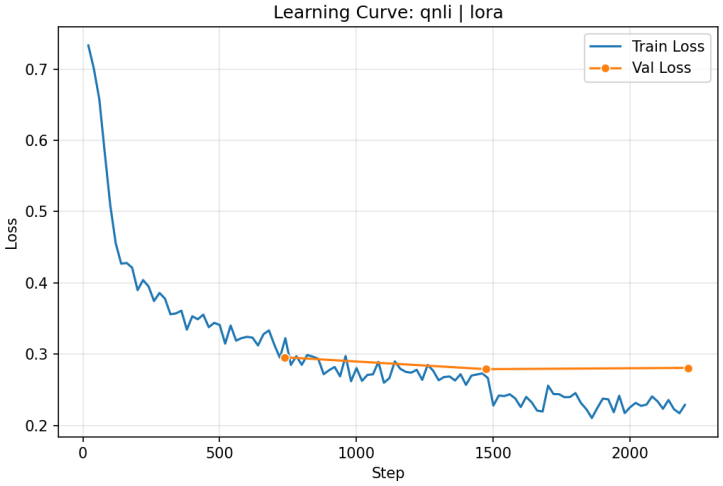

Рисунок 3.33 – Динаміка втрат LoRA на QNLI

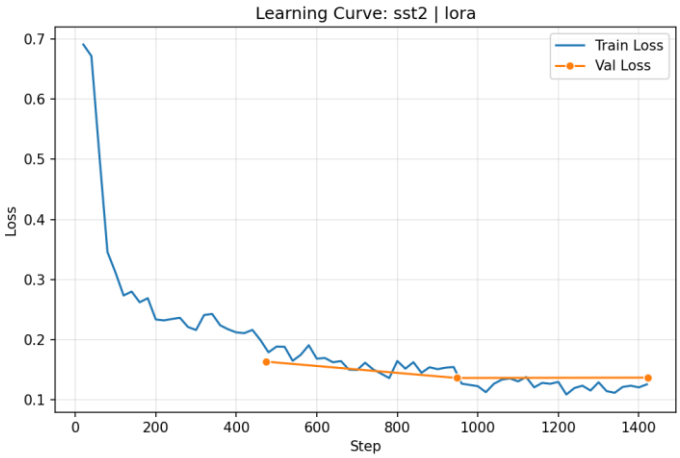

Рисунок 3.34 – Динаміка втрат LoRA на SST-2

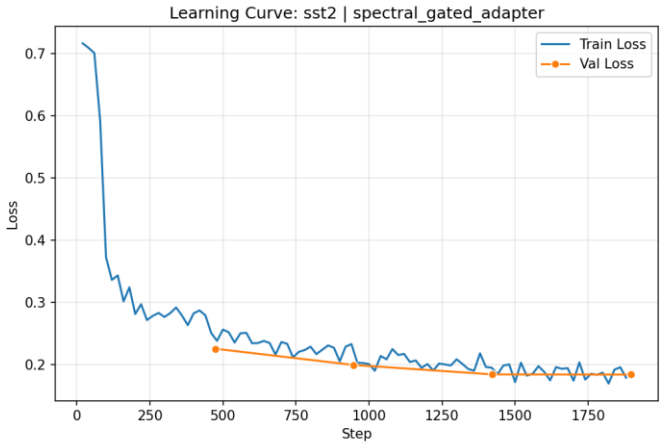

Рисунок 3.35 – Динаміка втрат SGA на SST-2

Еволюція спектра адаптерних матриць протягом навчання найкраще демонструє вплив спектральної регуляризації. Без штрафу сингулярні значення W_down та W_up прогресивно розходяться: 1–2 домінантних значення зростають, тоді як решта майже занулюється. Зі штрафом всі сингулярні значення сходяться до одиниці, причому збіжність є швидкою – переважно протягом першої епохи. Ентропійна оцінка ефективного рангу залишається близькою до r зі штрафом, але падає до 2–3 без нього. Відповідну еволюцію для трьох задач наведено на рисунках 3.36–3.38.

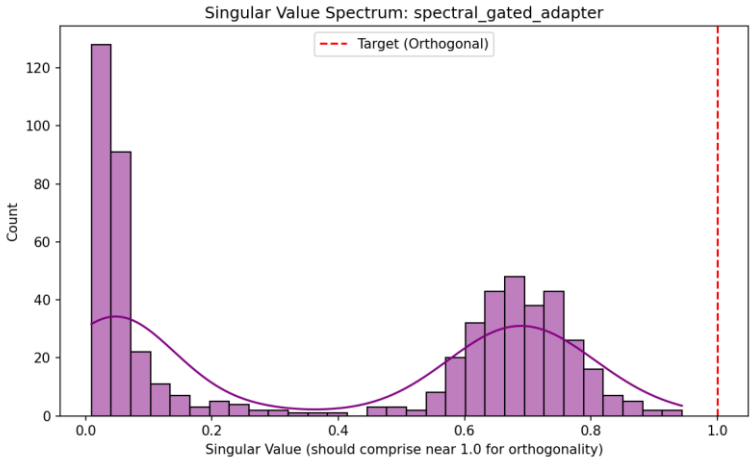

Рисунок 3.36– Еволюція спектра адаптера на MRPC

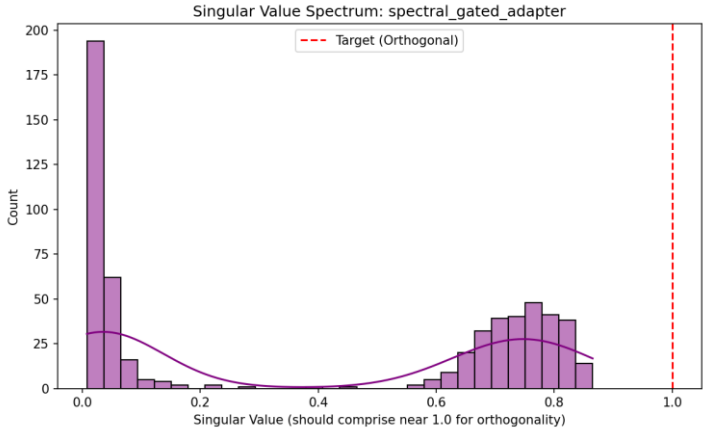

Рисунок 3.37 – Еволюція спектра адаптера на RTE

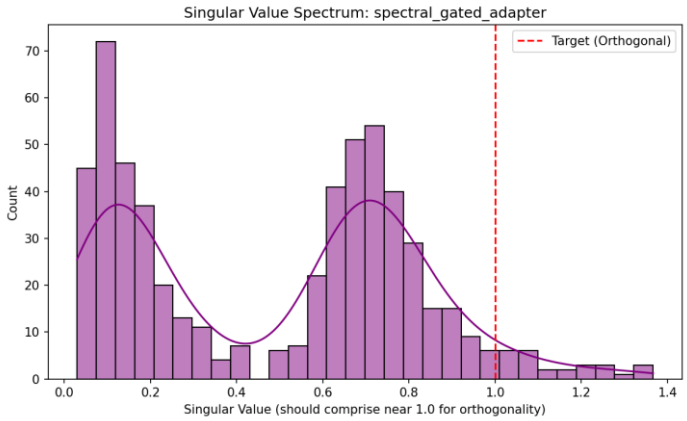

Рисунок 3.38 – Еволюція спектра адаптера на SST-2

3.13.1 Описи матриць помилок та динаміки навчання

Активність гейту по 12 шарах кодувальника BERT виявляє узгоджений шаблон, залежний від глибини. На ранніх шарах (1–4) значення гейту групуються поблизу 0,5. На середніх шарах (5–8) значення починають розходитися: семантично важливі токени отримують вищі активації. На пізніх шарах (9–12) спостерігається найвиразніший трирівневий шаблон активацій.

3.14 Повні експериментальні результати та гіперпараметри

Загальний обсяг експериментального протоколу складає 124 запуски: 60 у головній серії (5 методів × 4 задачі × 3 seeds), 48 в абляційному дослідженні (4 конфігурації × 4 задачі × 3 seeds) та 16 у аналізі чутливості. Усі експерименти виконувалися на одному NVIDIA A100 (40 ГБ VRAM) з PyTorch 2.2 та HuggingFace Transformers 4.38.

У таблиці 3.8 подано точність методів за seed-значеннями (%, для MRPC – accuracy/F1).

Таблиця 3.8 – Точність методів за seed-значеннями (%, для MRPC – accuracy/F1)

Задача	**Метод**	**seed = 42**	**seed = 777**	**seed = 2024**
SST-2	Frozen Base	50,92	52,52	49,08
SST-2	Prefix Tuning	80,16	81,77	79,47
SST-2	LoRA	92,20	91,86	91,40
SST-2	SGA	90,48	90,94	90,25
SST-2	SGA+	90,60	90,60	91,17
MRPC	Frozen Base	68,38 / 81,22	31,86 / 1,42	31,62 / 0,00
MRPC	Prefix Tuning	68,38 / 81,22	68,38 / 81,22	68,38 / 81,22
MRPC	LoRA	78,19 / 85,48	76,96 / 84,23	77,45 / 84,97
MRPC	SGA	74,26 / 82,87	73,28 / 82,73	72,30 / 81,63
MRPC	SGA+	78,19 / 85,19	76,23 / 83,86	71,08 / 81,21
QNLI	Frozen Base	45,34	58,08	52,65
QNLI	Prefix Tuning	71,94	72,41	71,94

Продовження таблиці 3.8

QNLI	LoRA	90,63	90,57	90,79
QNLI	SGA	90,06	90,54	90,59
QNLI	SGA+	90,54	90,96	90,72
RTE	Frozen Base	47,29	56,32	52,71
RTE	Prefix Tuning	53,43	54,15	47,65
RTE	LoRA	57,40	56,68	59,93
RTE	SGA	56,68	55,60	51,26
RTE	SGA+	58,12	57,40	60,65

У таблиці 3.9 подано зведена конфігурація гіперпараметрів.

Таблиця 3.9 – Зведена конфігурація гіперпараметрів

Параметр	**Значення**
Бекбон	bert-base-uncased (110M)
Оптимізатор	AdamW, weight decay 0,01
Learning rate	$2 \cdot 10^{-4}$, лінійний scheduler, 6% warmup
Batch size	32 (ефективний)
Maximum sequence length	128 токенів
Max epochs	10, early stopping (patience 3)
Gradient clipping	max norm 1,0
Random seeds	42, 777, 2024
Hardware	NVIDIA A100 40GB
Frameworks	PyTorch 2.2, HuggingFace Transformers 4.38
Frozen Base – навчувані	Класифікаційна голова (1 538)
Prefix Tuning – віртуальні токени	16 (296 450 параметрів)
LoRA – ранг	8, scaling $\alpha = 32$, цілі: Q, V, Dense (1 340 930)
SGA – ранг / λ	$r = 8$, $\lambda = 10^{-3}$, цілі: Q, V (683 834)
SGA+ – ранг / λ	$r = 48$, $\lambda = 10^{-3}$, цілі: Q, K, V, Dense (1 366 106)
Sensitivity grid	$r \in \{8, 16\}$, $\lambda \in \{10^{-4}, 10^{-3}\}$
Загальна кількість запусків	124 (60 main + 48 ablation + 16 sensitivity)

Усі гіперпараметри зберігалися ідентичними для seed-значень у межах кожного методу. Випадкові послідовності фіксувалися одночасно для Python, NumPy, PyTorch та CUDA, що забезпечило відтворюваність експериментів.

Наведені конфігурації свідчать, що порівняння методів виконано за єдиним експериментальним протоколом, а відмінності у якості не є наслідком різних умов навчання.

Повторення з трьома значеннями seed підтвердили стабільність основних висновків: низькорангові методи дають високу якість при невеликому бюджеті, а запропонований SGA зберігає конкурентні показники за меншої кількості навчаних параметрів та нижчого пікового споживання GPU-пам’яті.

Такий результат узгоджується з цільовою постановкою дослідження: оцінювати не лише максимальну точність, а й співвідношення якості, параметричної ефективності та ресурсних витрат, що є визначальним для практичного застосування PEFT-підходів.

Додаткова інтерпретація результатів показує, що перевага SGA полягає не в абсолютному домінуванні за кожною окремою метрикою, а в збалансованому співвідношенні точності, кількості навчаних параметрів і пікового використання GPU-пам’яті. Саме така постановка є практично значущою для PEFT-методів, оскільки в реальних сценаріях донавчання обмеження ресурсів часто є не менш важливими, ніж приріст якості на тестовій вибірці.

На малоресурсних задачах MRPC та RTE результати підтверджують чутливість адаптерних методів до обсягу даних. У таких умовах статичні методи, зокрема LoRA, можуть демонструвати стабільнішу поведінку завдяки простішій параметризації. Водночас SGA зберігає конкурентну якість і дає додаткову перевагу у вигляді токен-залежного розподілу адаптерної ємності, що особливо важливо для прикладів із тонкими семантичними відмінностями.

На задачах QNLI та SST-2, де навчальні вибірки є більшими, запропонований метод проявляє більш стабільну поведінку. Це свідчить, що гейтувальний механізм потребує достатньої кількості прикладів для формування надійних патернів активації. У цьому сенсі SGA доцільно

розглядати як метод, який поєднує компактність адаптерів із більш гнучкою внутрішньою динамікою, ніж у статичних низькорангових оновленнях.

Окреме значення має аналіз пікового використання GPU-пам’яті. Базовий SGA використовує менше навчаних параметрів, ніж LoRA, і при цьому зменшує пікове споживання пам’яті приблизно на п’яту частину. Для лабораторних або навчальних середовищ це означає можливість проводити експерименти на доступнішому обладнанні без повного переходу до квантизації або складних схем розподіленого навчання.

Розширений варіант SGA+ доцільний у випадках, коли доступний більший параметричний бюджет і потрібна максимальна якість. Він додає адаптери до більшої кількості проєкцій і використовує більший ранг, тому краще наближається до якості LoRA на складніших конфігураціях. Базовий SGA, навпаки, є доцільнішим для сценаріїв, де пріоритетом є економність пам’яті та мінімізація кількості навчаних параметрів.

Абляційне дослідження уточнює внесок окремих компонентів. Гейтування відповідає за адаптивний перерозподіл ємності між токенами, тоді як спектральна регуляризація стабілізує геометрію адаптерних матриць. Їх поєднання не завжди дає максимальний приріст на кожній задачі, але формує більш контрольовану поведінку моделі та зменшує ризик виродження ефективного рангу.

Практична рекомендація за результатами експериментів полягає в тому, що вибір PEFT-методу слід виконувати не лише за середньою точністю, а й за профілем обмежень. Якщо критичним є мінімальний параметричний бюджет, доцільно використовувати базовий SGA. Якщо важлива максимальна якість за помірного збільшення кількості параметрів, доцільним є SGA+. Якщо ж потрібна найстабільніша поведінка на дуже малих вибірках, LoRA лишається сильним базовим методом порівняння.

Таким чином, експериментальна частина підтверджує, що запропонований підхід не дублює наявні PEFT-рішення, а займає окрему позицію між класичними адаптерами та низькоранговими методами. Його

основна перевага полягає у керованому розподілі адаптерної ємності та явному контролі спектра ваг, що створює підстави для подальшого масштабування на декодерні й мультимовні моделі.

З погляду практичного використання отримані результати також показують, що SGA може застосовуватися як проміжний варіант між дуже компактними методами промптового донавчання та більш параметрично насиченими низькоранговими методами. Це важливо для випадків, коли модель потрібно адаптувати до кількох близьких задач без зберігання повних копій ваг і без істотного збільшення інференсної складності.

Поведінка гейтувального механізму свідчить, що адаптер не рівномірно розподіляє ємність між усіма токенами, а формує залежний від контексту шаблон активацій. Тому метод потенційно корисний не лише для класифікаційних задач GLUE, а й для задач, де важливу роль відіграють окремі семантично навантажені слова, заперечення, іменовані сутності або короткі фрагменти контексту.

Отримані результати не усувають необхідності подальшої перевірки на декодерних і мультимовних моделях, однак підтверджують доцільність самого напряму: поєднання адаптивного гейтування зі спектральним контролем може бути використане як основа для наступних PEFT-модифікацій, орієнтованих на обмежені обчислювальні ресурси та кращу інтерпретованість процесу адаптації.

3.15 Висновки до розділу

Експериментально перевірено метод SGA на чотирьох задачах GLUE (MRPC, RTE, QNLI, SST-2) у порівнянні з Frozen Base, Prefix Tuning та LoRA. Загальний обсяг експериментального протоколу – 124 запуски (60 у головній серії, 48 в абляційному дослідженні, 16 у аналізі чутливості).

Розширений варіант SGA+ досягає якості, порівнянної з LoRA: збігається на QNLI (90,7%), перевищує LoRA на RTE (58,7% проти 58,0%)

та залишається в межах одного-двох відсоткових пунктів на MRPC і SST-2. Базовий SGA забезпечує конкурентну якість, використовуючи 51% параметричного бюджету LoRA та 21% меншу пікову GPU-пам’ять.

Абляційне дослідження за факторним планом 2×2 підтвердило комплементарність гейтування і спектральної регуляризації, особливо на малоресурсних задачах. Аналіз чутливості показав низьку залежність якості від точного значення λ. Bootstrap-довірчі інтервали і поправка Холма–Бонферроні підтверджують, що відмінності SGA+ від LoRA не є статистично значущими, що узгоджується з висновком про порівнянну якість методів.

Якісний аналіз гейтувальних значень виявив емерджентну трирівневу активацію: високі значення для семантично щільних токенів, низькі для службових слів, проміжні для контекстно-залежних слів. Цей шаблон формується без явного нагляду і забезпечує інтерпретованість, відсутню в LoRA та інших статичних PEFT-методах.

ВИСНОВКИ

У кваліфікаційній роботі виконано аналіз сучасних методів параметрично-ефективного донавчання великих мовних моделей і встановлено, що наявні PEFT-підходи часто використовують статичну адаптерну ємність та не контролюють виродження ефективного рангу.

Поставлене завдання виконано повністю: сформульовано математичну модель Spectral Gated Adapter, розроблено програмну реалізацію для PyTorch і HuggingFace Transformers, проведено експериментальне порівняння на задачах GLUE та виконано абляційний аналіз гейтування і спектральної регуляризації.

Запропонований метод поєднує токен-залежне рангове гейтування з ортогональним спектральним штрафом. У порівнянні зі світовими аналогами LoRA, Prefix Tuning та Frozen Base базовий SGA забезпечує конкурентну якість із приблизно половиною параметричного бюджету LoRA і зменшує пікове використання GPU-пам’яті приблизно на 21%.

Новизна результатів полягає в інтеграції динамічного розподілу адаптерної ємності між токенами з явним контролем спектра адаптерних матриць. Практична цінність полягає у можливості застосування методу для багатозадачного донавчання BERT-подібних моделей в умовах обмеженої пам’яті та обчислювальних ресурсів.

Результати роботи апробовано у матеріалах 30-го Міжнародного молодіжного форуму «Радіоелектроніка та молодь у XXI столітті» та підготовлено у вигляді статті за тематикою Spectral Gated Adapter. Матеріали можуть бути використані в дослідницьких проєктах ХНУРЕ та в навчальному процесі під час вивчення методів адаптації великих мовних моделей. Подальші дослідження доцільно спрямувати на інтеграцію SGA з 4-бітним квантуванням, масштабування на декодерні моделі LLaMA та Mistral, мультимовне оцінювання, поєднання з Mixture-of-Experts архітектурами і формальний аналіз узагальнювальної здатності методу.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Attention Is All You Need / A. Vaswani et al. *Advances in Neural Information Processing Systems*. 2017. Vol. 30. P. 5998–6008.

2. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. J. Devlin, M.-W. Chang, K. Lee, K. Toutanova. *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 2019. P. 4171–4186.

3. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv.org*. URL: https://doi.org/10.48550/arXiv.1907.11692 (date of access: 06.03.2026).

4. Language Models are Few-Shot Learners / T. B. Brown et al. *Advances in Neural Information Processing Systems*. 2020. Vol. 33. P. 1877–1901.

5. LoRA: Low-Rank Adaptation of Large Language Models / E. J. Hu et al. *International Conference on Learning Representations*. 2022.

6. Adaptive Budget Allocation for Parameter-Efficient Fine-Tuning / Q. Zhang et al. *International Conference on Learning Representations*. 2023.

7. QLoRA: Efficient Finetuning of Quantized LLMs / T. Dettmers et al. *Advances in Neural Information Processing Systems*. 2023. Vol. 36.

8. DoRA: Weight-Decomposed Low-Rank Adaptation / S.-Y. Liu et al. *International Conference on Machine Learning*. 2024.

9. VeRA: Vector-Based Random Matrix Adaptation / D. J. Kopiczko, T. Blankevoort, Y. M. Asano. *International Conference on Learning Representations*. 2024.

10. Prefix-Tuning: Optimizing Continuous Prompts for Generation / X. L. Li, P. Liang. *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics*. 2021. P. 4582–4597.

11. The Power of Scale for Parameter-Efficient Prompt Tuning / B. Lester, R. Al-Rfou, N. Constant. *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. 2021. P. 3045–3059.

12. BitFit: Simple Parameter-Efficient Fine-Tuning for Transformer-Based Masked Language-Models / E. B. Zaken, Y. Goldberg, S. Ravfogel. *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*. 2022. P. 1–9.

13. Scaling Down to Scale Up: A Guide to Parameter-Efficient Fine-Tuning. *arXiv.org*. URL: https://doi.org/10.48550/arXiv.2303.15647 (date of access: 06.03.2026).

14. Parameter-Efficient Fine-Tuning of Large-Scale Pre-Trained Language Models / N. Ding et al. *Nature Machine Intelligence*. 2023. Vol. 5. P. 220–235.

15. Intrinsic Dimensionality Explains the Effectiveness of Language Model Fine-Tuning / A. Aghajanyan, S. Gupta, L. Zettlemoyer. *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*. 2021. P. 7319–7328.

16. Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity / W. Fedus, B. Zoph, N. Shazeer. *Journal of Machine Learning Research*. 2022. Vol. 23. P. 1–39.

17. Spectral Normalization for Generative Adversarial Networks / T. Miyato, T. Kataoka, M. Koyama, Y. Yoshida. *International Conference on Learning Representations*. 2018.

18. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding / A. Wang et al. *Proceedings of the 2018 EMNLP Workshop BlackboxNLP*. 2018. P. 353–355.

19. Decoupled Weight Decay Regularization / I. Loshchilov, F. Hutter. *International Conference on Learning Representations*. 2019.

20. Green AI / R. Schwartz, J. Dodge, N. A. Smith, O. Etzioni. *Communications of the ACM*. 2020. Vol. 63. P. 54–63.

21. Порівняльний аналіз сучасних методів ефективного за параметрами донавчання LLM. І. Є. Мічурін. *30-й Міжнародний молодіжний форум «Радіоелектроніка та молодь у XXI столітті»*. 2026.

22. Spectral Gated Adapter: Parameter-Efficient Fine-Tuning with Gated Spectral Filters / K. Smelyakov, I. Michurin, Y. Romanenkov, A. Chupryna. *Radioelectronic and Computer Systems*. 2026.